# 微服务平台 TSF

# 操作指南



腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

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。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。
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。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

# 文档目录

- 操作指南
  - 应用管理
    - 业务应用管理（容器）
      - 容器应用管理概述
      - 资源管理
        - 集群管理
        - 命名空间管理
      - 应用部署（K8s）
        - 镜像管理
        - 容器集群部署应用
        - K8S 原生部署
        - 健康检查
      - 应用升级与回滚
        - 应用升级与回滚概述
        - 分批发布
        - 蓝绿发布
        - 全链路灰度发布
          - 概述
          - 泳道配置
          - 灰度发布
        - 查看发布详情
        - 发布回滚
      - 应用运维
        - 应用管理
        - 弹性伸缩
        - 服务预热
      - 脚手架
    - 业务应用管理（虚拟机）
      - 虚拟机应用管理概述
      - 资源管理
        - 集群管理
        - 命名空间管理
      - 应用部署（虚拟机）
        - 程序包管理
        - 虚拟机集群部署应用
        - 健康检查
      - 应用升级与回滚
        - 应用升级与回滚概述
        - 分批发布
        - 蓝绿发布
        - 全链路灰度发布
          - 概述
          - 泳道配置
          - 灰度发布
        - 查看发布详情
        - 发布回滚

应用运维
应用管理
弹性伸缩
服务预热
脚手架
中间件应用管理
Java 网关概述
Java 网关跨命名空间访问
部署 Java 网关
配置分组与 API
API 管理
配置鉴权
设置 API 超时时间
设置 API 限流规则
路径重写配置
网关监控
插件管理
配置 JWT 插件
配置 OAuth 插件
配置 Tag 插件
配置小程序登录插件
持续集成
使用 Python 脚本部署应用
使用 Jenkins 创建持续集成
使用 Coding 创建持续集成
使用插件部署应用
安装配置 IntelliJ IDEA 插件
通过 Tencent Cloud Toolkit 部署应用
平台管理
权限管理
概述
角色管理
数据集管理
授权管理
标签管理
注册配置治理
云原生 API 网关
云原生 API 网关概述
快速入门
使用云原生 API 网关访问 HTTP 服务
使用云原生 API 网关访问 TKE 服务
使用云原生 API 网关访问 Nacos 服务
使用云原生 API 网关访问 TSF 服务
使用云原生 API 网关访问 PolarisMesh
使用 Kong Ingress 实现容器服务访问
使用云原生 API 网关访问私有域名服务
使用云原生 API 网关实现 WebSocket 服务转发
操作指南
网关管理

新建网关
查看网关详情
升级网关版本
升级网关规格
集群分组管理
网关节点管理
共享带宽包管理
删除网关
服务路由
服务
路由
服务来源
灰度发布
配置限流策略
健康检查
全局配置
证书管理
域名管理
Ingress
Kong Ingress介绍
使用 Kong Ingress Controller
使用云原生 API 网关管理多容器集群
灰度配置 Ingress
安全防护
使用 Web 安全防火墙
Web 安全防护概述
域名防护
对象防护
使用 DDos 防护
结合 WAF 和 DDos实现安全防护
监控告警
查看默认监控
使用 Prometheus 监控
链路追踪
日志
查看默认日志
日志投递至 CLS
上报日志到 Kafka
插件上报日志到 CLS
查看日志大盘
Terraform 管理
参数配置
插件管理
弹性伸缩
事件中心
故障演练
迁移指南
从自建 Kong 网关迁移到腾讯云
从 Spring Cloud Gateway 迁移到云原生 API 网关

版权所有：腾讯云计算（北京）有限责任公司

实践教程
认证鉴权
开启 admin-api 并配置安全认证
使用 IP 访问控制
使用 Key Auth 密钥认证
使用 ACL 访问控制
使用 HMAC Auth 认证访问
使用 Bot Detection 插件拒绝爬虫请求
配置 JWT 认证鉴权
配置自定义鉴权
流量控制
使用缓存加速访问
使用熔断保护后端
使用流量镜像插件
请求路由
进行 URL 重写
使用重写插件改写请求或响应
可观测
观测中心
存量用户
升级指引（内测中）
资源概览
业务应用监控
服务依赖拓扑
调用链查询
日志服务
概述
快速入门
日志格式说明
日志配置项
日志投递
日志告警
查看日志
日志检索
事件中心
新用户
资源概览
业务应用监控
调用查询
日志服务
事件中心
业务系统
TCT 任务迁移 XXL-JOB 通用指引

# 操作指南
# 应用管理
# 业务应用管理（容器）
# 容器应用管理概述

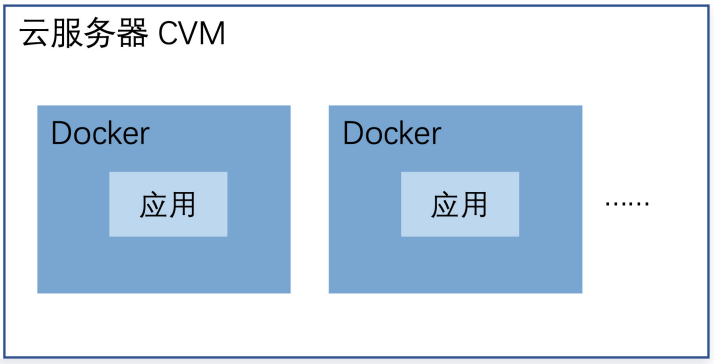
最近更新时间：2025-03-10 18:08:32

TSF 支持以容器的形式托管应用到 TSF 容器集群，您无需理解容器服务底层细节，便可以低门槛拥抱容器技术，最大化地利用资源将应用部署到 TSF，轻松享受 TSF 的服务注册发现、服务治理、可观测性等一系列能力。
本文介绍容器托管应用方式的相关应用场景，支持的应用类型和部署流程。

## 应用场景

- 应用托管方式：容器部署方式通过镜像将应用部署在 Docker 容器中，Docker 应用部署时，将在云服务器上创建多个 Docker 容器实例。
- 应用场景：适用于对容器运行环境需要定制和希望提升资源利用率的场景。
- 部署方式：支持 JAR 包、WAR 包以及镜像部署，不同的部署方式需选择的应用运行环境也会有所不同。
- 应用举例：Spring Boot、Dubbo、MySQL、WordPress。

云服务器 CVM、Docker 容器和应用的关系如下。




## 支持的应用类型

TSF 支持以容器的形式托管 Java 应用、Go应用以及其他多语言（例如 PHP，C++ 等）应用，支持 JAR 包、WAR 包以及镜像部署，不同的部署方式需选择的应用运行环境也会有所不同。

- Java：支持 SpringCloud、Dubbo 和其他框架接入。
- Go：支持 Go-GRPC 和其他框架接入。
- 多语言：支持 PHP，C++ 等应用类型。

## 部署流程

1. 前往资源管理-容器集群
2. 将云主机导入集群
3. 创建应用
4. 上传程序包或镜像
5. 部署应用

## 常见问题

- TSF 与 TKE 的关系是什么?
- 应用和服务之间是什么关系?

# 资源管理
# 集群管理

最近更新时间：2025-03-10 18:08:33

## 操作场景

集群是指云资源管理的集合，包含了运行应用的云主机等资源。TSF 中的集群分为虚拟机集群和容器集群。使用虚拟机或者容器计算资源时，用户需要提前先将云主机导入集群中，才能进行应用的部署。本文介绍在腾讯微服务平台控制台创建容器集群并对集群进行管理的操作方法。

## 创建容器集群

### 操作流程

您可以将容器服务（TKE）中的容器集群导入到腾讯微服务平台控制台进行应用部署，也可以在腾讯微服务平台控制台上直接创建容器集群。
整体操作流程如下：

方式一：导入容器集群服务

1. 在容器服务控制台创建或注册集群。
2. 在腾讯微服务平台控制台导入容器集群服务。
3. 在腾讯微服务平台控制台部署应用到容器集群。

方式二：直接创建容器集群

1. 腾讯微服务平台控制台直接创建容器集群。
2. 在容器集群中导入云主机。
3. 在腾讯微服务平台控制台部署应用到容器集群。

> **说明：**
> 默认情况下每个账号在一个地域最多能创建5个容器集群，如果需要扩大额度，请 提交工单 给**容器服务**产品。

## 方式一：导入容器集群服务

### 适用场景

如果您在容器服务（TKE）上已经创建好了容器集群，可以直接导入到 TSF 集群列表。这种方式适用于更灵活的容器集群配置场景（例如：制定 Kubernetes 版本、独立部署模式集群）。

### 步骤1：在容器服务控制台创建或注册集群。

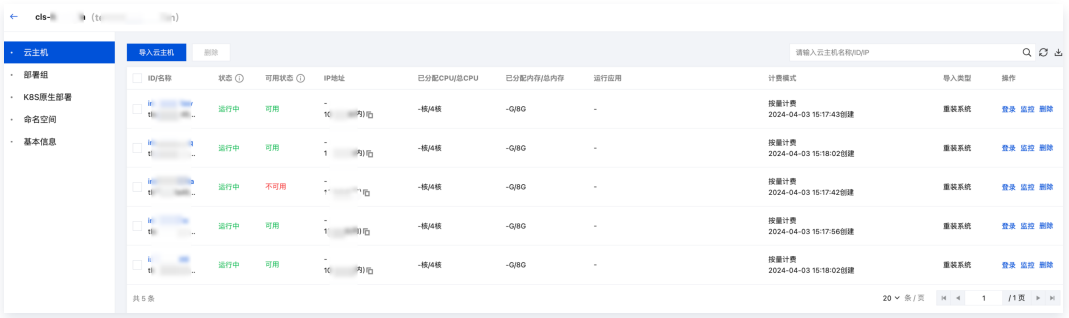
在容器服务控制台创建集群的操作步骤详情可参见 创建集群 。

### 步骤2：在腾讯微服务平台控制台导入容器集群服务。

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中选择**资源管理 > 容器集群**，单击**导入容器集群**。
3. 设置集群的基本信息。
   - 导入类型：选择**选择已有集群**。
   - 所在可用区：必选。选择要导入的容器服务集群所在的可用区。

- 选择集群：必选。选择需要导入的TKE集群。
- 标签：非必选。用于分类管理资源。标签管理方法详情请参见 标签 。
- 备注：非必填。集群的描述，不超过200个字符。
- 数据集：非必选。将新建的集群添加到已有数据集中。数据集使用，数据集管理方法详情请参见 数据集管理 。

4. 单击**确定**，在弹窗中选择**导入云主机**，进入导入云主机页面。如果选择**取消**，您后续可以在集群列表中，单击操作列的**导入云主机**重新导入云主机。
5. 从集群所在 VPC 的云主机列表中，选择需要添加到集群的云主机。
6. 单击**下一步**，进行云主机配置。
   - 导入方式：容器集群只支持**重装系统**部署方式，该方式会使 root 用户安装 agent。
   - 操作系统：支持 CentOS，TencentOS 和 Ubuntu 三种类型。
   - 登录方式：提供三种对应登录方式。
     - 设置密码：请根据提示设置对应密码。
     - 立即关联密钥：密钥对是通过一种算法生成的一对参数，是一种比常规密码更安全的登录云主机的方式。详情参阅 SSH 密钥 。
     - 自动生成密码：自动生成的密码将通过站内信发送给您，也可登录CVM控制台重置密码。
   - 数据卷挂载：勾选后，自动格式化数据盘成ext4，仅对拥有一块数据盘的节点生效。
   - 容器目录：设置容器和镜像存储目录，建议存储到数据盘。
   - 安全组：安全组具有防火墙的功能，用于设置云主机 CVM 的网络访问控制。设置详情参见 安全组设置 。
7. 单击**提交**，等待云主机导入完成，云主机的可用状态将变为“可用”。




> **说明**
>
> 1. 当集群下存在超级节点，可支持资源灵活弹性调度，单击**查看超级节点详情**可跳转到超级节点页面。关于超级节点的详情请参见 超级节点概述 。
> 2. TSF暂不支持使用原生节点的 TKE集群。

### 步骤3：在腾讯微服务平台控制台部署应用到容器集群。

导入容器集群后，您可以使用 TSF 的 K8S 原生部署能力将您的应用快速接入 TSF，详情请参见 K8S 原生部署 。您也可以在腾讯微服务平台控制台直接创建并部署应用，详情请参见 容器托管应用 。

## 方式二：直接创建容器集群

### 步骤1：在腾讯微服务平台控制台直接创建容器集群。

您可以通过腾讯微服务平台控制台直接创建容器集群，调用容器服务 TKE 新建集群 API 创建容器集群。操作简单，适用于未购买容器集群的场景。

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中选择**资源管理 > 容器集群**，单击**导入容器集群**。
3. 设置集群的基本信息。
   - 集群名：集群的名称。不能为空。最长60个字符，只能包含小写字母、数字及分隔符（“-”），且不能以分隔符开头或结尾。
   - Kubernetes 版本：选择容器集群的 Kubernetes 版本。

- 所在可用区：选择容器集群所在的可用区
- 集群网络：为集群内主机分配在主机网络地址范围内的 IP 地址。参见 集群及容器网络设置 。
- 容器网络：为集群内容器分配在容器网络地址范围内的 IP 地址。参见 集群及容器网络设置 。
- 标签：用于分类管理资源，可不选。详情参见 标签 。
- 备注：集群的描述，不超过200个字符。
- 数据集：将新建的集群添加到已有数据集中。非必填字段。数据集使用，请参见 数据集管理 。

4. 单击**提交**，查看集群初始化进度。

### 步骤2：在容器集群中导入云主机。

1. 当集群初始化完成，进入导入云主机页面。如果选择**取消**，您后续可以在集群列表中，单击操作列的**导入云主机**重新导入云主机。
2. 从集群所在 VPC 的云主机列表中，选择需要添加到集群的云主机。
3. 单击**下一步**，进行云主机配置。
   - 导入方式：容器集群只支持**重装系统**部署方式，该方式会使 root 用户安装 agent。
   - 操作系统：支持 CentOS，TencentOS 和 Ubuntu 三种类型。
   - 登录方式：提供三种对应登录方式。
     - 设置密码：请根据提示设置对应密码。
     - 立即关联密钥：密钥对是通过一种算法生成的一对参数，是一种比常规密码更安全的登录云主机的方式。详情参阅 SSH 密钥 。
     - 自动生成密码：自动生成的密码将通过站内信发送给您，也可登录CVM控制台重置密码。
   - 数据卷挂载：勾选后，自动格式化数据盘成ext4，仅对拥有一块数据盘的节点生效。
   - 容器目录：设置容器和镜像存储目录，建议存储到数据盘。
   - 安全组：安全组具有防火墙的功能，用于设置云主机 CVM 的网络访问控制。详情参见 安全组设置 。
4. 单击**提交**，等待云主机导入完成，云主机的可用状态将变为“可用”。

### 步骤3：在腾讯微服务平台控制台部署应用到容器集群。

导入云主机后，您可以在容器集群服务中创建并部署应用，参见 创建并部署应用 。

## 管理容器集群

### 移出云主机

> ⚠ **注意：**
> 只有当节点上没有运行应用时，才可以将节点从集群中移出。停止应用需要到部署组的实例列表页面中操作。

1. 登录 腾讯微服务平台 ，选择**资源管理** > **容器集群**。
2. 在容器集群列表页中，单击目标集群的**ID/集群名**，进入集群的云主机列表页面。
3. 在云主机节点列表页面，选择需要移出的云主机，单击操作列的**删除**。
4. 在弹出的提示框中，单击**确定**将云主机移出节点。

### 删除集群

> ⚠ **注意：**
> - 集群内存在云主机时，无法删除集群，您需要先移除云主机。
> - 集群在删除期间，无法对外提供服务，请提前做好准备，以免造成影响。

1. 登录 腾讯微服务平台 ，选择**资源管理** > **容器集群**。
2. 在容器集群列表页中，单击目标集群操作列的**更多** > **删除**。
3. 在弹出的提示框中，单击**确定**删除集群。

## 集群及容器网络设置

集群网络与容器网络是集群的基本属性。通过设置集群网络和容器网络可以规划集群的网络划分。

- **集群网络**：将为集群内主机分配在节点网络地址范围内的 IP 地址，您可以选择私有网络中的子网用于集群的节点网络，更多私有网络介绍请参见 私有网络和子网 。
- **容器网络**：将为集群内容器分配在容器网络地址范围内的 IP 地址，您可以自定义三大私有网段作为容器网络， 根据您选择的集群内服务数量的上限，自动分配适当大小的 CIDR 段用于 kubernetes service，同时容器网络自动为集群内每台云主机分配一个24位的网段用于该主机分配 Pod 的 IP 地址。

### 集群网络与容器网络的关系

- 集群网络和容器网络网段不能冲突。
- 同一 VPC 内，不同集群的容器网络网段不能冲突。
- 容器网络和 VPC 路由冲突时，优先在容器网络内转发。

### 集群网络与腾讯云其他资源通信

- 集群内容器与容器之间互通。
- 集群内容器与节点直接互通。
- 集群内容器与 腾讯云数据库 TencentDB 、云数据库 Redis 、云数据库 Memcached 等资源同一 VPC 下内网互通。

## 安全组设置

用户将容器集群导入云主机时需要设置安全组，设置方法请参见 容器服务安全组设置 。
此外，如果从 API 网关访问容器集群中的微服务，则需要在安全组中**增加服务的监听端口**。例如，provider－demo 的监听端口是18081，需要新增入站规则如下：

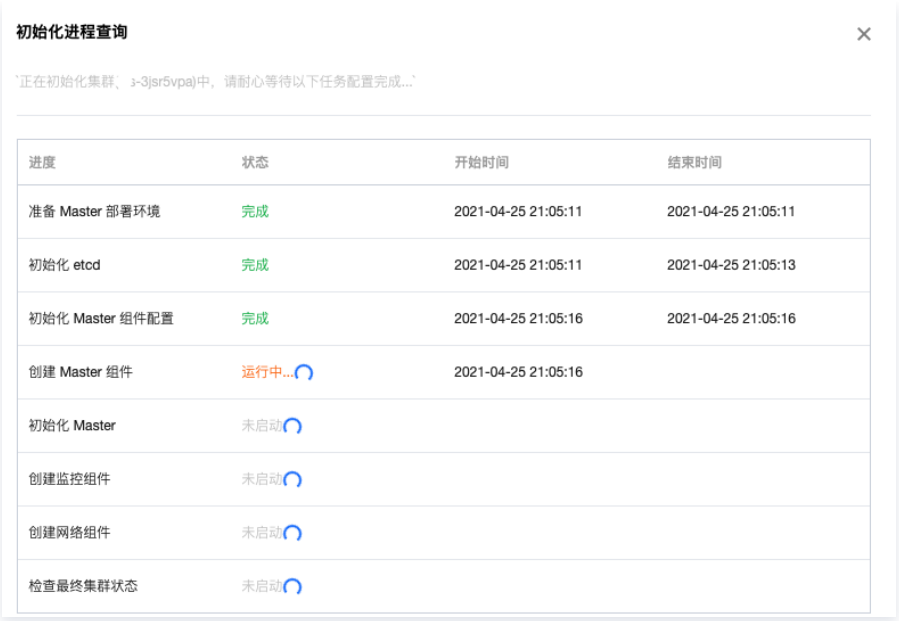
| 协议 | 端口号 | 网段 | 是否允许 |
| --- | --- | --- | --- |
| TCP | 18081 | 0.0.0.0/0 | 允许 |

## 查看容器集群创建事件

登录 腾讯微服务平台控制台 ，在集群页面新建或导入容器集群后，在容器集群列表的集群状态栏可以查看容器集群状态。




- 闲置中：容器集群内没有云主机，处于完全不可用状态，可以通过导入云主机方式唤醒集群或者删除集群。
- 创建中：集群正在创建，正在申请云资源，单击集群状态栏的**查看进度**可查看创建进度。




- 运行中：集群正常运行。
- 异常：集群中存在异常，例如节点网络不可达等，可单击**查看异常原因**。

## 常见问题

- 云服务器加入容器集群后，项目属性为何变为"默认项目"?
- 如何手动恢复 tsf-agent 进程?
- TSF 容器集群数达到上限如何处理?
- TSF 不同集群内的服务是否支持相互调用?
- TSF 如何设置开发环境和测试环境?
- CVM 集群应用的日志清理机制是什么?

版权所有：腾讯云计算（北京）有限责任公司

# 命名空间管理

最近更新时间：2025-03-10 18:08:33

命名空间（Namespace）是对 TSF 一组资源和对象的抽象集合，用于进行 TSF 资源的隔离。

## 命名空间分类

命名空间有三种类型：

| 类型 | 个数限制 | 说明 |
| --- | --- | --- |
| 系统命名空间 | 创建每个集群时会自动创建一个 | 命名规则是 `<cluster-name>_default`，不支持绑定到其他集群。 |
| 非全局命名空间 | 每个用户可以创建多个 | 不同非全局命名空间内的服务之间不能相互调用。 |
| 全局命名空间 | 每个用户在每个地域下只能创建一个 | • 非全局命名空间内的服务可以调用全局命名空间内服务。<br>• 全局命名空间内服务支持相互调用，不支持调用非全局命名空间内的服务。 |

**注意：**

- 全局命名空间特性仅适用于1.18.0版本之后的 SDK。
- 原生应用不支持全局命名空间特性。

## 与 K8s 命名空间、服务注册配置治理中心命名空间的关系

| 使用场景 | 说明 | 功能说明 |
| --- | --- | --- |
| K8s 命名空间 | 容器资源空间 | 用于容器资源相互隔离 |
| 应用管理的命名空间 | TSF 资源空间 | 用于 TSF 资源（集群、应用等）相互隔离，不同命名空间下的资源相互隔离。 |
| 注册配置治理中心的命名空间 | 服务空间 | 用于服务调用隔离，相同服务空间的服务可以互相调用和推送配置，不同服务空间的服务、配置相同隔离，不能进行服务调用和配置推送。 |

**说明：**

- 当注册配置治理中心为共享实例（TSF-Consul）时，服务空间默认创建，与 TSF 资源空间同名。
- 当注册配置治理中心为独占实例（北极星）时，服务空间与 TSF 资源空间需分别创建，独立使用，您可以在应用部署时选择该应用关联的服务空间。

## 应用场景

### 使用命名空间划分开发环境和测试环境

**场景：**用户希望部署一套开发环境和一套测试环境，两套环境支持部署同一个应用不同版本的程序包。两套环境可以共享一个集群作为底层部署资源。

**解决方案：**

1. 创建两个非全局命名空间分别为 `dev-env` 和 `test-env` 作为开发环境和测试环境。
2. 创建一个虚拟机集群，并绑定 `dev-env` 和 `test-env` 命名空间。
3. 创建应用（如 `promotion`），并创建 2 个部署组（`dev-promotion` 和 `test-promotion`），分别选择 `dev-env` 和 `test-env` 命名空间。

4. 部署组 `dev-promotion` 部署程序包版本 v1.1，部署组 `test-promotion` 部署程序包版本 v1.0。

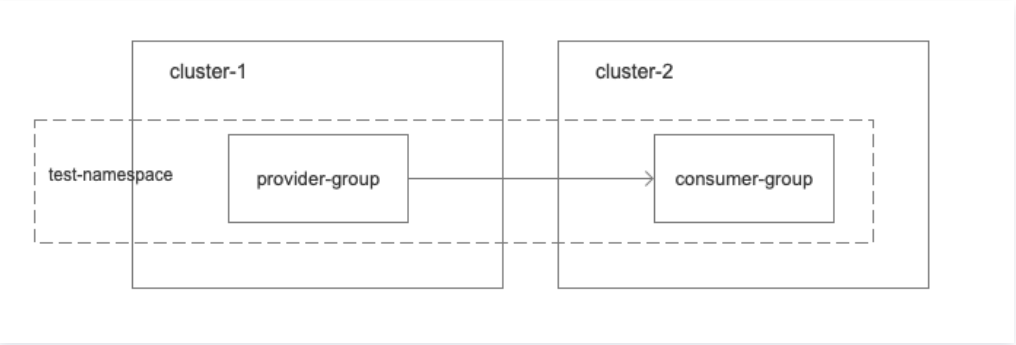
> **注意：**
> 如果用户希望开发和测试环境的部署资源是隔离的，可以使用两个集群来划分开发和测试环境。

## 实现服务跨集群访问

在网络连通性的前提下，同一命名空间内的服务可以相互发现和相互调用。如果要实现**跨集群的服务访问**，有两个前提：

1. 集群内实例网络互通
2. 集群关联相同的命名空间。**多个集群通过命名空间名称（而不是命名空间 ID）作为关联的 key**。用户可以在命名空间一级页面中看到关联的集群信息。

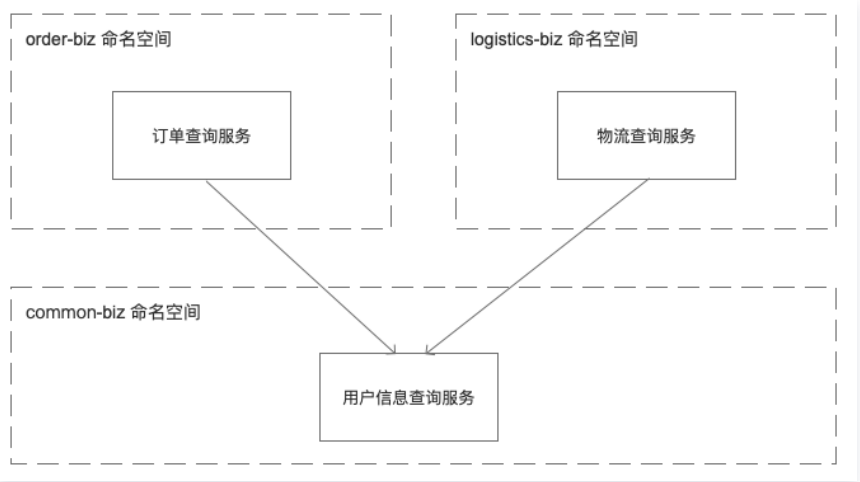
例如：有2个集群分别是 cluster-1 和 cluster-2，两个集群内实例网络互通（例如在相同 VPC 内），并且都关联了命名空间 test-namespace。要实现跨集群访问，需要确保在创建部署组 provider-group 和 consumer-group 时，使用相同的命名空间 test-namespace。




## 使用全局命名空间来部署公共服务

**场景：**在电商场景中，订单业务和物流业务都希望访问用户信息服务。此时可以使用不同命名空间来划分业务领域，将相对独立的业务（例如订单和物流业务）部署在非全局命名空间，将公共服务（例如用户信息服务）部署在全局命名空间中。
**解决方案：**

1. 创建两个非全局命名空间 *order-biz* 和 *logistics-biz* 作为订单和物流的业务领域。
2. 创建全局命名空间 *common-biz* 作为公共服务的业务领域。
3. 将订单业务的服务部署在 *order-biz* 命名空间内，将物流的服务部署在 *logistics-biz* 内，将用户信息服务部署在 *common-biz* 内。
4. *order-biz* 和 *logistics-biz* 命名空间内的服务可以访问 *common-biz* 内的用户信息服务。




# 创建命名空间并关联集群

## 创建命名空间

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中单击**资源管理**，选择好地域后，在**命名空间**标签页单击**新建命名空间**。
3. 填写命名空间信息。
   - 名称：不能为空。最长60个字符，只能包含字母、数字及分隔符（ “-” ），且不能以分隔符开头或结尾。
   - 勾选是否为全局命名空间：每个租户限制创建一个全局命名空间。
   - 标签：非必选。用于分类管理资源。管理方法详情请参见 标签 。
   - 备注：非必填。集群的描述，不超过200个字符。
   - 数据集：非必选。将新建的集群添加到已有数据集中。数据集使用，管理方法详情请参见 数据集管理 。
4. 单击**确定**，完成创建。




## 集群关联命名空间

1. 进入 腾讯微服务平台 ，选择**资源管理 > 容器集群**。
2. 选择目标集群 ID，进入集群详情页，选择**命名空间**。
3. 在命名空间列表左上方，单击**关联命名空间**。
4. 选择要关联的命名空间，并单击**确定**。

> **注意：**
> 集群仅支持关联自定义命名空间。

## 删除命名空间

**前提条件**：删除命名空间前，需要先解除命名空间与集群的绑定关系（解除命名空间绑定时，需要先删除具有该命名空间属性的全部部署组）。不支持删除系统命名空间。

### 解除关联集群

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏中选择**资源管理 > 容器集群**。
2. 选择目标集群，进入集群详情页，单击**命名空间**标签页。
3. 在命名空间列表的操作列，单击**解除绑定**。

## 删除命名空间

1. 登录 腾讯微服务平台控制台，在左侧导航栏中选择**资源管理 > 容器集群**。
2. 选择目标集群，进入集群详情页，单击**命名空间**标签页。
3. 在命名空间列表的操作列，单击**删除**。
4. 单击**确认**，即可删除命名空间。

版权所有：腾讯云计算（北京）有限责任公司

# 应用部署（K8s）
# 镜像管理

最近更新时间：2025-03-10 18:08:33

## 操作场景

镜像仓库用于存放用户上传的镜像，该任务指导您熟悉镜像仓库的常用操作。

## 前提条件

- 安装 Docker。
- 使用 `sudo` 允许系统管理员让普通用户执行 `docker` 命令。

## 操作步骤

### 初始化镜像仓库

首次使用镜像仓库时，需要进行初始化操作，设置登录仓库的密码。




TSF 会针对每个容器应用创建一个名为 `tsf_<账号ID>/<应用名>` 的镜像仓库。




### 获取镜像仓库使用指引

单击**制品管理 > 镜像仓库管理**，选择目标镜像仓库，单击镜像仓库名称，进入镜像仓库详情。单击 使用指引，查看使用指引。




## 将镜像推送到镜像仓库

1. 制作容器镜像，详细制作方式可参见 制作容器镜像 。
2. 复制**使用指引**中登录腾讯云 docker registry 的命令并执行。

```
sudo docker login --username=<账号 ID>  ccr.ccs.tencentyun.com
```

> **说明：**
> 用户需要输入两次密码，首次为 sudo 密码，第二次为镜像仓库登录密码。
> 命令行工具显示 `Login Succeeded` 即表示登录成功。

3. 登录成功后，复制**使用指引**中给镜像打 tag 的命令并执行。

```
sudo docker tag [ImageId] ccr.ccs.tencentyun.com/tsf_<账号ID>/<应用名>:[tag]
```

其中 [ImageId] 和 [tag] 是在制作镜像时获取。

> **说明：**
> 若此时需要输入密码，请输入sudo 密码。

4. 复制**使用指引**中推送镜像到仓库的命令并执行，其中 [tag] 和步骤4相同。

```
sudo docker push ccr.ccs.tencentyun.com/tsf_<账号ID>/<应用名>:[tag]
```

```
[redacted] % sudo docker push ccr.ccs.tencentyun.com/tsf_[redacted]/provider_lm:lm
The push refers to repository [ccr.ccs.tencentyun.com/tsf_[redacted]/provider_lm]
5f70bf18a086: Layer already exists
80716faaa03b: Layer already exists
1fe7549efc8d: Layer already exists
46f49101b7e7: Layer already exists
c5719313253c: Layer already exists
02de69f968ee: Layer already exists
174f56854903: Layer already exists
lm: digest: sha256:bc8dc54211f7591530d44ff885e531dbd647e4995973[redacted] size: 1781
```

## 拉取镜像

1. 复制**使用指引**中登录腾讯云 docker registry 的命令并执行。

```
sudo docker login --username=<账号 ID>  ccr.ccs.tencentyun.com
```

> **说明：**
> 用户需要输入两次密码，首次为 sudo 密码，第二次为镜像仓库登录密码。
> 命令行工具显示 `Login Succeeded` 即表示登录成功。

2. 复制**使用指引**中拉取镜像的命令并执行。

```
sudo docker pull [NAME]:[tag]
```

其中 `[NAME]` 表示镜像名称，`[tag]` 表示镜像标签，例如：

```
sudo docker pull ccr.ccs.tencentyun.com/tsf_123456/tsf_demo:v1.0
```

## 重置密码

用户可以通过腾讯微服务平台控制台重置镜像仓库密码。

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中，单击制品管理，进入 镜像仓库管理 。
3. 单击**重置密码**。




4. 根据页面提示填写密码，单击**确定**完成重置。




## 常见问题

- 协作者为何无法推送镜像到主账号的镜像仓库?
- 默认容器是 UTC 时区，和宿主机的时区不一致如何解决?
- 协作者使用镜像仓库时，为何显示未开通仓库?

版权所有：腾讯云计算（北京）有限责任公司

# 容器集群部署应用

最近更新时间：2025-03-10 18:08:33

## 操作场景

在容器服务集群中部署应用，您需要先在 TSF 控制台中创建好容器服务集群和命名空间，再使用应用程序包或者镜像来部署应用。本文介绍在容器服务集群中部署应用的操作方法。

## 前提条件

- 已创建好 集群 和 命名空间 。
- 已准备好容器镜像或者程序包。

## 操作步骤

### 步骤1：创建应用

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**应用管理**。
2. 在业务应用的应用列表，单击**新建**。设置应用信息后，单击**提交**完成创建。
   - **应用名**：填写应用名称，最长60个字符，只能包含小写字母、数字及分隔符（“-”、“_”），且不能以分隔符开头或结尾。
   - **部署方式**：选择容器部署。
   - **开发语言**：选择您的开发语言。

     > **说明：**
     > 关于开发语言、开发框架和应用类型的选型原则和建议，请参见 技术栈选型 。

   - **开发框架**：选择您的开发框架。
   - **标签**：用于分类管理资源，可不选。管理方法详情请参见 标签 。
   - 备注：选填，可留空。
   - **数据集**：选择“无”。用户可以通过数据集管理配置不同的子账号和协作者使用不同资源的权限，管理方法详情请参见 数据集管理 。
3. 单击**下一步**，进入注册配置治理的配置。

### 步骤2：上传镜像或程序包

您可以选择自己制作镜像并推送到镜像仓库，也可以直接上传程序包，控制台将自动为您制作镜像并上传。

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**应用管理**，在业务应用列表中，单击刚刚创建好的应用 **ID/名称**，进入应用详情。
2. 在应用详情页，查看**制品**卡片。
   - **个人仓库**提供基础的云上镜像托管、分发服务，限额使用，仅面向个人使用或企业客户临时测试使用。个人版服务为云上共享服务，即所有个人版用户将共享服务后端及数据存储，且镜像托管及上传下载具有配额限制。容器镜像服务个人版不收取费用，可直接开通使用。
   - **企业仓库**提供企业级的独享镜像安全托管服务，面向需要在生产业务中使用云原生制品托管服务的个人及企业客户。当前容器镜像服务企业版支持包年包月或按量计费购买。

> **说明：**
> 您可以单击以下页签查看**个人仓库**和**企业仓库**上传镜像的方法。

个人仓库

1. 仓库类型选择**个人仓库**，单击**上传程序包/镜像**。
2. 您可以选择自己制作镜像并推送到镜像仓库，也可以直接上传程序包，控制台将自动制作镜像并上传。




- 镜像：根据使用指引制作镜像并上传。详细操作参见 镜像仓库 。
- JAR包部署：直接上传程序包，无需制作镜像。
  - JDK 版本：支持 KNOA JDK8，KNOA JDK11，OPEN JDK8 和 OPEN JDK17。
  - 上传程序包：单击**选择文件**，选择编译为 fatjar 格式的程序包，程序包格式说明参见 程序包格式说明 。
  - 程序包版本：填写版本号。
- WAR包部署：直接上传程序包，无需制作镜像。
  - 上传程序包：单击**选择文件**，选择编译为 war 格式的程序包，程序包格式说明参见 程序包格式说明 。
  - 程序包版本：填写版本号。

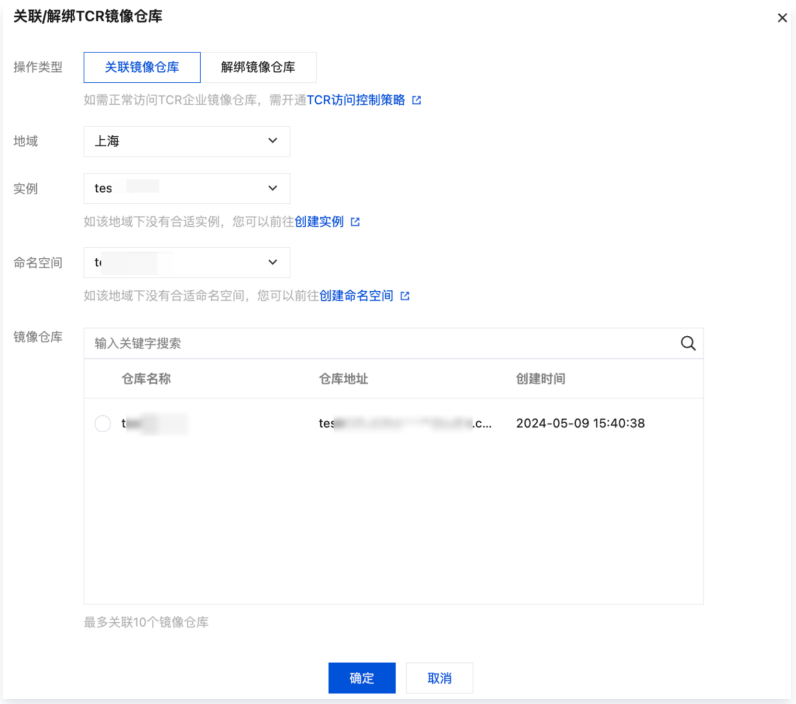
3. 单击**上传程序包并制作镜像**，右上角将出现任务进行的状态。
4. 单击**任务进行中**将看到任务进行的详情，若任务失败，可查看任务失败的原因。

### 企业仓库

> **说明：**
> 在 TCR 镜像场景下，在 JAR包部署 / WAR包部署需在 TCR 镜像仓库开通公网访问控制策略，开通 `0.0.0.0/0` 白名单。

使用企业仓库时，您需要先关联企业仓库再上传镜像。

1. 仓库类型选择**企业仓库**，单击**关联/解绑 TCR 镜像仓库**。
2. 在弹窗中选择需要绑定的 TCR 镜像仓库。




- 操作类型：选择**关联镜像仓库**。

  > **说明：**
  > 如需正常访问 TCR 企业镜像仓库，需开通 TCR 访问控制策略 。

- 地域：选择镜像仓库地域。
- 实例：选择容器镜像服务 TCR 企业版实例。
- 命名空间：选择命名空间
- 镜像仓库：选择要绑定的 TCR 镜像仓库。

3. 单击**确定**，完成 TCR 镜像仓库绑定。
4. 在页面上方单击**上传程序包/镜像**。

> **说明：**
> 您可以选择自己制作镜像并推送到镜像仓库，也可以直接上传程序包，控制台将自动制作镜像并上传。




- 镜像：单击下方**生成指令**，根据生成的使用指引制作镜像并上传。详细操作参见 镜像仓库。
- JAR包部署：直接上传程序包，无需制作镜像。
  - 镜像仓库：建议选择与容器集群相相同地域的企业镜像服务仓库，访问不同地域的实例将受公网网络出入带宽影响。
  - JDK 版本：支持 KNOA JDK8，KNOA JDK11，OPEN JDK11，OPEN JDK8。
  - 上传程序包：单击**选择文件**，选择编译为 fatjar 格式的程序包，程序包格式说明参见 程序包格式说明。
  - 程序包版本：填写版本号。

- WAR包部署：直接上传程序包，无需制作镜像。
  - 镜像仓库：建议选择与容器集群相相同地域的企业镜像服务仓库，访问不同地域的实例将受公网网络出入带宽影响。
  - 上传程序包：单击**选择文件**，选择编译为 war 格式的程序包，程序包格式说明参见 程序包格式说明。
  - 程序包版本：填写版本号。

5. 单击**上传程序包并制作镜像**，右上角将出现任务进行的状态。
6. 单击**任务进行中**将看到任务进行的详情，若任务失败，可查看任务失败的原因。

### 步骤3：部署应用

1. 在左侧导航栏选择**应用管理**，单击刚刚创建好的应用 ID/名称。
2. 在应用部署菜单，单击**应用部署**，将会创建一个部署组进行应用部署。
3. 填写部署组信息。
   - **集群**：选择应用要部署所在的集群
   - **命名空间**：选择应用要部署所在的命名空间。
   - **部署组名称**：填写部署组名称，不超过60个字符，只能包含字母、数字及分隔符(“-”)，且必须以字母开头，数字或字母结尾。
   - **标签**：非必选。用于分类管理资源。详情参见 标签 。
   - **备注**：选填，可留空。
4. 单击**下一步**，进行部署配置。
   - **容器类型**：选择 Deployment。
   - **agent容器**：负责日志、JVM 监控、调用链数据的采集，如果不部署 agent 容器将影响这些功能的使用。agent 容器的内存 limit 和日志量有关，通常使用默认值 400MiB 即可，如果出现 OOM 可适当增加 limit 值。应用容器的资源配置参见 容器部署组资源限制 。
     - request：用于预分配资源，当集群中的节点没有 request 所要求的资源数量时，会导致无法创建容器。
     - limit：用于设置容器使用资源的最大上限，避免异常情况下节点资源消耗过多。
   - **实例数量**：选择实例数量。实例数和实例资源限制的乘积不能超过集群剩余的可用资源。单击检测是否有足够资源可以检测集群中是否有足够剩余的可用资源。检测时间较久，请您耐心等待。由于 K8S 集群资源进行实时调度，真实资源情况以您实际部署时为准，您可以通过部署组状态判断部署是否成功，通过部署事件、日志分析部署失败原因。
5. 进行容器配置。TSF支持在一个部署组添加多个容器，部署组内所有容器将共享生命周期。
   - **镜像仓库类型**：选择个人仓库或者企业仓库。
   - **选择镜像**：选择需要进行应用部署的镜像。
   - **资源配置**：设置CPU和内存配置。
   - **环境变量**：设置应用容器中的变量。
     - 自定义：您可以自定义环境变量
     - Field：容器路径，包含 metadata.name、metadata.namespace、spec.nodeName、spec.serviceAccountName、 status.hostIP、status.podIP。
     - ResourceField：容器资源，通常支持资源限制和请求，例如 limits.cpu、limits.memory、limits.ephemeral-storage、requests.cpu、requests.memory 和 requests.ephemeral-storage。
   - **JAVA_OPT（选填）**： 设置 Java 应用的启动参数。参数会通过 `JAVA_OPTS` 环境变量带到容器运行环境中，参见 制作镜像 中的使用方式。在 Java 启动环境变量中应用容器参数应采用 $(var_name) 的格式。
   - **容器高级配置**：设置容器的高级配置，支持对容器进行精细化管理。
     - 特权容器：默认关闭，开启后，容器将拥有Host宿主机中的root权限，运行特权容器存在安全性风险。
     - 运行命令：支持自定义配置 Entrypoint 和 CMD 默认命令。
     - 运行参数：单击**查看容器运行和参数示例**，查看对应的运行参数示例。
     - 生命周期钩子：监听容器生命周期的特定事件，并在事件发生时执行已注册的回调函数，运行用户指定的程序代码。支持PostStart和PreStop两个阶段进行自定义。
     - 容器健康检查：支持配置存活检查和就绪检查。关于健康检查的作用和配置详情参见 健康检查 。

- 存活检查：用于判断何时重启实例。
- 就绪检查：用于判断 Pod 何时变为 Ready 状态，会影响滚动更新。默认检查无须用户设置检查规则，会根据服务实例是否注册到注册中心来决定 Pod 变为 ready 状态。

6. 设置数据卷配置信息（选填）。数据卷为容器提供存储，目前支持临时路径、主机路径、云硬盘数据卷、文件存储NFS、配置文件、PVC，还需挂载到容器的指定路径中。

| 数据卷类型 | 描述 |
| --- | --- |
| 使用临时路径 | / |
| 使用主机路径 | 将容器所在宿主机的文件目录挂载到容器的指定路径中（即对应 Kubernetes 的 HostPath）。您可以根据业务需求，不设置源路径（即对应 Kubernetes 的 EmptyDir）。如果不设置源路径，系统将分配主机的临时目录挂载到容器的挂载点。指定源路径的本地硬盘数据卷适用于将数据持久化存储到容器所在宿主机，EmptyDir 适用于容器的临时存储。 |
| 使用 NFS 盘 | 只需填写 NFS 路径。使用 NFS 数据卷适用于多读多写的持久化存储，也适用于大数据分析、媒体处理、内容管理等场景。 |
| 使用已有 PersistentVolumeClaim | 使用已有 PersistentVolumeClaim 声明工作负载的存储，自动分配或新建 PersistentVolume 挂载到对应的 Pod 下。主要适用于 StatefulSet 创建的有状态应用。 |
| 使用 ConfigMap | ConfigMap 以文件系统的形式挂载到 Pod 上，支持自定义 ConfigMap 条目挂载到特定的路径。 |
| 使用 Secret | Secret 以文件系统的形式挂载到 Pod 上，支持自定义 Secret 条目挂载到特定的路径。 |

7. 设置访问配置信息。容器部署组支持添加多个 Service 作为请求入口。

| 参数 | 说明 |
| --- | --- |
| 名称 | Service 的名称标识。 |
| 访问方式 | 参见 网络访问方式 了解不同访问方式的区别。 |
| 端口组 | 定义服务端口、容器端口，协议，名称。 |
| Session Affinity | • Client：基于来源IP做会话保持，开启后设置最大会话保持时间。<br>• None：关闭会话保持。 |

> **说明：**
> 每次部署应用的时候，会自动向容器中添加3个 Label：platform，qcloud-app，qcloud-app-name。

8. 设置调度策略（选填）

| 调度策略类型 | 说明 |
| --- | --- |
| 节点调度策略 | 基于节点设置Pod与节点间的调度策略，满足工作负载与节点的亲和/反亲和调度。 |
| 可用区打散策略 | 根据打散策略将工作负载的Pod平均打散部署在各个可用区，提高应用的高可用性，至少需要两个可用区。 |
| Pod调度策略 | 通过设置多条Pod互相之间的调度策略，满足Pod与Pod之间亲和/反亲和调度。 |
| 容忍调度策略 | Pod是否容忍污点。 |

9. 设置重启策略。
10. 单击**下一步**，设置发布策略。

| 参数 | 说明 |
| --- | --- |
| 发布方式 | • 滚动更新（推荐）：对实例进行逐个更新，这种方式可以让您不中断业务实现对服务的更新。<br>• 立即部署：直接关闭所有实例，启动相同数量的新实例。 |
| 滚动间隔 | 滚动更新的更新时间间隔。 |
| 滚动策略 | 滚动更新方式可以设置更新策略：<br>• 启动新的 Pod，停止旧的 Pod（默认）：需确认集群有足够的 CPU 和内存用于启动新的 Pod，否则可能导致集群崩溃。<br>• 停止旧的 Pod，启动新的 Pod ： 适用于需要同时部署多个容器部署组，且集群内剩余资源不够时。<br>• 自定义：需要设置允许超出所需规模的最大 Pod 数量和允许最大不可用的 Pod 数量 |
| 策略配置 | 可以设置批量启动或停止的 Pod 数量，可以设置百分比或正整数。详情请参见 Kubernetes Deployment 滚动更新策略 和 容器服务实例优雅下线 实践教程进行配置。 |

11. 单击**下一步**，设置服务注册配置治理信息。

> **注意：**
> 如果您使用了独占实例，请将集群所属 VPC 添加到独占配置中心访问地址中，否则会网络连接不通，参考文档 北极星引擎管理 。

- 是否开启服务预热：设置服务预热策略。当更新方式选择滚动更新时，建议开启服务预热，关于服务预热的作用和配置详情参见 服务预热 。
- 接入服务Agent：选择是否开启通过字节码注入服务Agent。
- Istio_proxy：负责 Mesh 服务注册、流量转发等任务，通常使用默认的资源限制即可。

> **说明：**
> 仅通过容器部署的 Mesh 应用涉及。

12. 设置可观测信息。
    - **日志配置项**：将设定业务日志的采集路径和解析规则。
    - **日志投递**：日志投递规则将设定日志的转储方式，将规则指定路径中的日志内容投递到指定的接收端。
    - **Agent配置**：若使用 Spring Cloud 2020，请勾选**可观测 Java Agent 配置**选项开启调依赖等高级监控能力。

> **注意：**
> 1. 仅支持业务应用，不支持中间件应用。
> 2. Agent版本需高于1.29.8，升级Agent版本请参考文档 使用Agent进行无侵入应用开发 。

    - 服务 Agent：包含注册发现、治理与配置能力。如您使用 Spring Cloud 2020，且未依赖 TSF SDK，请勾选本选项。
    - 可观测 Agent：如您使用 Spring Cloud 2020，请勾选本选项。相关开发文档，请参见 使用 Agent 进行无侵入应用开发 。

13. 单击**发布**，等待部署完成。

## 访问方式

容器部署组的访问方式定义访问后端 Pod 的访问方式，并提供固定的虚拟访问 IP。您可以在 Service 中通过设置来访问后端的 Pod，不同访问方式的服务可提供不同网络能力。TSF 目前提供以下四种服务访问方式：

| 访问方式 | 说明 |
| --- | --- |
| 提供公网访问 | 使用 Service 的 Loadbalance 模式，公网 IP 可直接访问到后端的 Pod，适用于 Web 前台类的服务。创建完成后的服务在集群外可通过**负载均衡域名或 IP + 服务端口**访问服务，集群内可通过**部署组名称 + 服务端口**访问服务。 |
| 仅在集群内访问 | Headless Service：不创建用于集群内访问的ClusterIP，访问Service名称时返回后端Pods IP地址，用于适配自有的服务发现机制。使用 Service 的 ClusterIP 模式，自动分配 Service 网段中的 IP，用于集群内访问。数据库类 |

| | |
|---|---|
| | 等服务如 MySQL 可以选择集群内访问，以保证服务网络隔离。创建完成后的服务，要在同一个命名空间的容器内，通过**部署组名称 + 服务端口**访问服务。 |
| VPC 内网访问 | 使用 Service 的 Loadbalance 模式，指定 `annotations:service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx` ，即可通过内网 IP 直接访问到后端的 Pod。创建完成后的服务在集群外可通过**负载均衡域名或 IP + 服务端口**访问服务，集群内可通过**部署组名 + 服务端口**访问服务。 |
| 主机端口(NodePort)访问 | 提供一个主机端口映射到容器的访问方式，支持 TCP、UDP、Ingress。可用于业务定制上层 LB 转发到 Node。创建完成后的服务可以通过**云服务器 IP + 主机端口**或**部署组名称 + 服务端口**访问服务。 |

**注意：**
一个账号在单地域创建的公网负载均衡 LB 实例有数量限制，参见负载均衡 使用约束 。如需要扩大负载均衡实例限额，请 提交工单 处理。

## 容器部署组资源限制

### 实例资源限制说明

新建容器部署组时可以设置实例数量和每个实例资源限制，实例的资源限制包括两个指标：CPU 和内存大小。
**Request**：容器使用的最小资源需求，作为容器调度时资源分配的判断依赖。只有当节点上可分配资源量 >= 容器资源请求数时才允许将容器调度到该节点。但 Request 参数不限制容器的最大可使用资源值。
**Limit**： 容器能使用的资源最大值。

**说明：**
更多 **Limit** 和 **Request** 参数介绍，单击 查看详情 。

当 `实例数量 * Request值 > 集群剩余的资源` 时，会提示 "资源不足，请导入节点" 的提示语。此时用户需要去集群页面导入云服务器以扩充资源。例如实例数量为2，实例资源Request是 `CPU=0.5核，内存=1GB` ，而集群剩余资源为 `CPU=0.8核，内存=2GB` 时， 由于 CPU 核数资源不够，会提示资源不足的错误。

### Java 应用的最大堆内存和容器内存大小关系

Java 应用通常需要设置 JVM 启动参数，包括最大堆内存（-Xmx）的设置。建议 JVM 最大堆内存和容器实例内存资源大小的关系符合以下条件，避免容器 OOM（Out of memory）：

```
容器内存 Request >= 1.25 * JVM 最大堆内存 ；

容器内存 Limit >= 2 * JVM 最大堆内存;
```

例如，启动参数中设置 `-Xmx` 为 1024m，应用容器实例内存的 Request 至少为 1280MiB，Limit 至少为 2560MiB。
关于如何使用启动参数，请参见 制作镜像 > Spring Cloud 应用构建材料 中 Dockerfile 启动命令中的 `${JAVA_OPTS}` 。

## 常见问题

- TSF 是否支持在同一台服务器上安装多个应用?
- 如何查看实例的 Agent 版本?
- 如何将实例 Agent 升级至最新版本?
- TSF 应用实例状态为什么显示 Agent 异常?
- 重启服务器后 TSF Agent 是否能自动重启?
- 如何排查应用是否部署失败?

版权所有：腾讯云计算（北京）有限责任公司

- 容器部署组执行部署操作时提示内存（或 CPU）不足?
- 创建应用时，提示命名空间数达到配额如何解决?
- 将一台已在 CVM 创建的实例添加到 TSF 的时候，是否需要重装 Agent?

# K8S 原生部署

最近更新时间：2025-03-10 18:08:33

## 操作场景

TSF 支持 K8S 原生部署方案，当您通过 K8S 部署业务后，使用该功能，您可以实现仅通过页面简单配置或 K8S yaml 的少量修改，将您的应用快速接入 TSF，保留原有 K8S 部署发布全流程，并且您可以使用 TSF 的服务注册发现、服务治理、可观测性等一系列能力。

> **说明：**
> 1. 使用此功能需要保证 K8S 版本不低于 1.16。
> 2. 当前能力仅支持注册配置治理中心为 TSF-Consul 的情景。

## 操作步骤

### 步骤1：在容器服务控制台创建集群

参见 创建集群 和 Deployment 管理 在容器服务控制台创建集群和 Deployment 并正常运行。

### 步骤2：导入容器服务集群到 TSF

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中，选择**资源管理**，选择**容器集群**，进入集群列表 。
3. 在集群列表页的左上方，单击**导入容器集群**。
4. 设置集群的基本信息。
   - **导入类型**：选择**选择已有集群**。
   - **所在可用区**：选择容器服务集群所在的可用区。
   - **选择集群**：选择要导入的容器集群服务。
   - **标签**：用于分类管理资源，可不选。详情参见 标签 。
   - **备注**：集群的描述，不超过200个字符。
   - **数据集**：将新建的集群 添加到已有数据集中。非必填字段。数据集使用，请参见 数据集管理 。

版权所有：腾讯云计算（北京）有限责任公司




5. 单击**提交**，在弹窗中选择**导入云主机**前往导入云主机页面。
6. 从集群所在 VPC 的云主机列表中，选择需要添加到集群的云主机。
7. 选择导入方式，并根据页面提示完成操作。
   - 重装系统：该方式会使用 root 用户安装 agent。
   - 安装 Agent（仅适用于虚拟机集群）： 该方式支持使用 root 用户和非 root 用户安装 agent。

## 步骤3：应用改造

参见开发指南对您的应用进行改造：

- 应用使用 TSF SDK接入
- 原生 Spring Cloud 应用
- Service Mesh 应用接入

## 步骤4：部署服务到 TSF

您可以在腾讯微服务平台控制台上导入 Deployment 到 TSF 来完成部署相关操作，也可以通过 K8S YAML 部署 TSF 服务。

### 自动安装组件

> **说明：**
> 单击开启后，我们将对您的集群进行组件安装，期间不影响集群正常运行，安装组件最少需要64MB内存 和0.1核 CPU。
> 当您单击导入 TSF 时，我们将**为您重启 Deployment**，请选择业务低谷期进行操作。

1. 进入 腾讯微服务平台控制台，选择**资源管理** > **容器集群**，在集群列表页面，单击集群的“ID”，进入集群详情页面。
2. 选择 **K8S 原生部署**页签，单击方式一中的**自动安装**。




3. 在 Deployment 列表页面，选择好K8s命名空间，单击您要导入的 Deployment 操作列的**导入 TSF**。
4. 在弹窗中填写导入的应用信息。
   - 导入后应用名称：填写导入后的应用名称，不超过60字符。
   - 开发语言：选择开发语言。
   - 开发框架：选择对应的开发框架。
   - 注册配置治理：默认开启。
   - 实例类别：选择独享实例或共享实例。
   - 接入方式：如实例类型为共享实例，选择接入注册配置中心的方式，支持SDK/框架接入、Mesh接入和无侵入接入。
   - TSF Agent（采集器）：我们将为您在容器集（Pod）中安装 TSF Agent 以使　平台提供的服务治理、配置管理、可观测性等能　。将在您　作负载重启后　效。
   - Agent 占用资源：
     - request 用于预分配资源，当集群中的节点没有 request 所要求的资源数量时，会导致无法创建容器。
     - limit 用于设置容器使用资源的最大上限，避免异常情况下节点资源消耗过多。




5. 单击**确定**，等待导入完成。

> **说明：**
> 导入完成后，会自动给用户创建好应用和部署组，重启应用后应用会注册到对应的注册中心。
> 通过 K8S 原生部署导入 Deployment 后， 不支持在 TSF 自动创建的部署组页面进行增删改操作，建议继续在TKE 页面进行操作。
> 对于在 TKE 页面创建的命名空间，开启 K8S 原生部署后，也不支持在该命名空间下新建部署组。
> 通过原生部署创建的部署组，所关联 K8S Deployment 被删除后，会同步删除 TSF 部署组，但会保留创建的应用与命名空间。

## 通过 YAML 配置手动安装

接下来将演示如何通过 K8S YAML 来部署 TSF 服务，例如 kubectl。
1. 在集群列表页面，单击的集群的“ID”，进入云主机列表页面。

2. 在页面上方选择 K8S 原生部署页签，单击方式二中的生成配置。




3. 给需要部署 TSF 应用的命名空间加上 label `tsf.tencentcloud.com/inject` 。

```
# ns.yaml
apiVersion: v1
kind: Namespace
metadata:
  labels:
    # new
    tsf.tencentcloud.com/inject: allowlist
  name: default
```

`allowlist` 指白名单，默认不开启，只有在 Deployment 上加上相应 label 才会开启。提交 YAML 到 K8S（后续 yaml 的 kubectl 提交过程将省略）。

```
$ kubectl apply -f ns.yaml
```

4. 创建一个简单的非 TSF Deployment。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: test-light-provider
  name: test-light-provider
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: test-light-provider
  template:
    metadata:
      labels:
        k8s-app: test-light-provider
    spec:
      containers:
      - name: app
        # TODO: 修改为真实的 TSF 应用的镜像
        image: ccr.ccs.tencentyun.com/xxx
      imagePullSecrets:
      - name: qcloudregistrykey
```

此时并不会有相应的部署组和应用创建出来，Deployment 和 Pod 也没有注入 TSF 的信息。

5. 修改创建好的 Deployment，加上一个 label `tsf.tencentcloud.com/inject: enable` 和一个 annotation `tsf.tencentcloud.com/application.type: "N"`。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: test-light-provider
    # new
    tsf.tencentcloud.com/inject: enable
  annotations:
    # new
    tsf.tencentcloud.com/application.type: "N"
    # 应用名，默认是 Deployment name
    tsf.tencentcloud.com/application.name: "test-light-provider"
  name: test-light-provider
...
```

6. 提交 YAML 到 K8S，等待 Deployment 更新成功。

## Label 和 Annotation 的使用方法

K8S 原生部署通过 label 和 annotation 来控制部署的行为，具体规则如下。

### Labels

| key | 范围 | 值和效果 |
| --- | --- | --- |
| `tsf.tencentcloud.com/inject` | Namespace | • `allowlist`：此 namespace 下采用白名单机制，Deployment 默认不导入 TSF。<br>• `enable`：此 namespace 下采用黑名单机制，默认导入。 |
| `tsf.tencentcloud.com/inject` | Deployment | • `enable`：当 namespace 是白名单时，表示导入。<br>• `disable`：当 namespace 是黑名单时，表示不导入。 |

### Annotation

> **说明：**
> Namespace 的 annotation 表示该 namespace 下所有 Deployment 的默认值，如果 Deployment 没有设置某个选项，会使用 namespace 上设置的。如果 Namespace 和 Deployment 都没有设置，如果该 annotation 是选填，会使用默认值；但如果 annotation 是必填，则会报错。

| key | 范围 | 是否必填 | 值和效果 |
| --- | --- | --- | --- |
| `tsf.tencentcloud.com/application.type` | Namespace/Deployment | 必填 | 应用类型。<br>• N：普通应用。<br>• M：service mesh 应用。<br>• G：网关应用。<br>• NATIVE：原生应用。 |

| | | | |
|---|---|---|---|
| `tsf.tencentcloud.com/application.name` | Deployment | 选填。默认是 Deployment name | 应用名称。 |
| `tsf.tencentcloud.com/agent` | Namespace/Deployment | 选填。默认是 `enable` | • `enable`：部署 agent 容器；<br>• `disable`：不部署 agent 容器。 |
| `tsf.tencentcloud.com/agent.cpu-request` | Namespace/Deployment | 选填。默认是 TSF 设定的默认值，当前为 100m。 | 例如 `100m` 。 |
| `tsf.tencentcloud.com/agent.cpu-limit` | Namespace/Deployment | 选填。默认是 TSF 设定的默认值，当前为 200m。 | 例如 `200m` 。 |
| `tsf.tencentcloud.com/agent.mem-request` | Namespace/Deployment | 选填。默认是 TSF 设定的默认值，当前为 125Mi。 | 例如 `125Mi` 。 |
| `tsf.tencentcloud.com/agent.mem-limit` | Namespace/Deployment | 选填。默认是 TSF 设定的默认值，当前为 250Mi。 | 例如 `250Mi` 。 |

版权所有：腾讯云计算（北京）有限责任公司

# 健康检查

最近更新时间：2025-03-10 18:08:33

## 概述

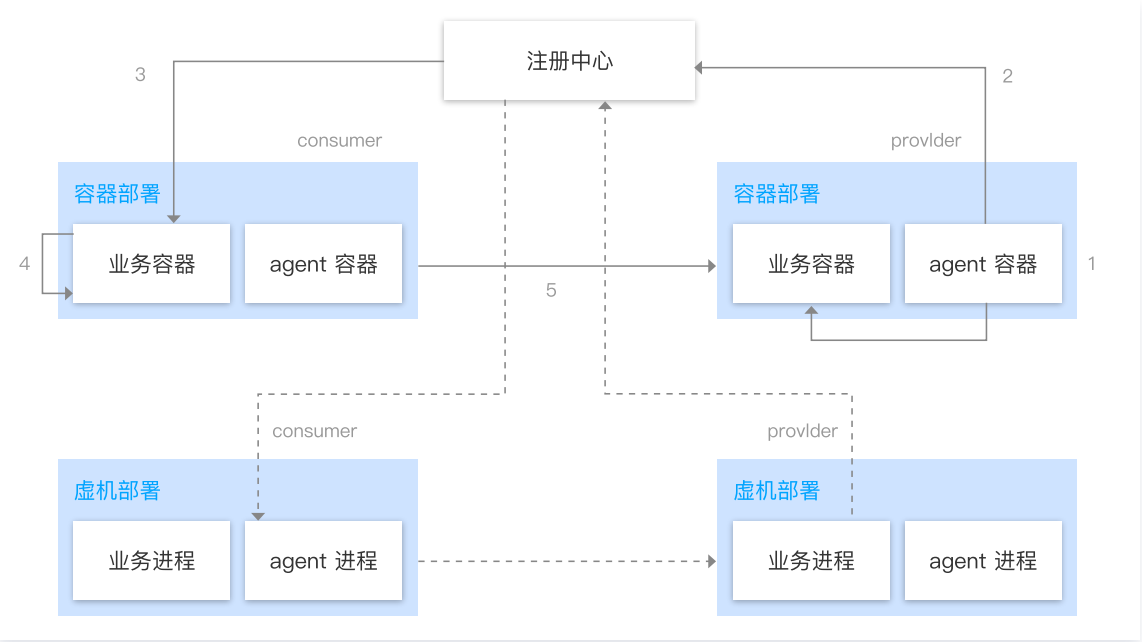
TSF 健康检查分为存活检查和就绪检查：

- **存活检查**：主要作用是确定进程存活状态，判断是否需要进行实例重启。例如存活检查可以捕捉到死锁（应用程序进程还存在，但是无法响应），重启进程或者容器可以让应用程序恢复可用。
- **就绪检查**：主要作用是确定服务实例能否支持对外服务，将健康检查结果与注册中心状态联动实现滚动更新及无损发布。当一个实例没有就绪，实例会在注册中心中被屏蔽（其他服务不会发送请求到微服务实例上）。

## 健康检查支持的产品能力

- 存活检查。
- 就绪检查。
- 无损发布。
- 检查顺序：HTTP 检查、TCP 端口、执行命令检查。
- 支持虚拟机部署组和容器部署组。

## 健康检查与注册中心联动流程




1. 就绪检查，检查实例状态是否 ready。
2. 如果就绪检查 ready 则更新实例注册状态为 passing，反之则检查状态为 Critical。
3. 监听注册中心服务提供方实例状态变更。
4. 存在状态变更更新缓存及本地文件。
5. 发起服务调用。

## 配置存活检查

1. 登录 腾讯微服务平台控制台，在应用管理选择对应的应用，进入应用详情，在左侧菜单栏中，单击**应用部署**。
2. 在部署配置，选择对应的容器，展开容器高级设置，在容器健康检查中开启存活检查。

版权所有：腾讯云计算（北京）有限责任公司




3. 检查方式：
   - HTTP 请求检查：任何大于200小于400的返回码都会认定是成功的返回码。其他返回码都会被认为是失败的返回码。HTTP 检查需要设置端口和请求路径。
   - TCP 端口检查：如果可以建立连接被认为是成功的。该检查方式需要设置检查端口。
   - 执行命令检查：如果命令执行成功并且返回值为 0，认为是成功；其他返回值认为是失败。该检查方式需要填写执行命令。
4. 启动延时、超时时间、检测周期、健康阈值、不健康阈值使用默认值即可。

## 配置就绪检查

**场景：**服务 A 调用服务 B。服务 B 使用 `/health` 接口是否返回 `200` 状态码判断是否健康，当就绪检查失败时，服务 B 在注册中心中被屏蔽，服务 A 通过注册中心监听到服务 B 状态变化后更新本地路由表；当就绪检查成功时，服务 B 在注册中心恢复健康状态。

1. 登录 腾讯微服务平台控制台，在应用管理选择对应的应用，进入应用详情，在左侧菜单栏中，单击**应用部署**。
2. 在部署配置，选择对应的容器，展开容器高级设置，在容器健康检查中开启就绪检查。




   - **优雅滚动重启：**无须用户设置检查规则，会根据服务实例是否注册到注册中心来决定 Pod 变为 ready 状态。
   - **自定义：**自定义容器就绪检查通过的方式。
3. 检查方式：
   - HTTP 请求检查：任何大于200小于400的返回码都会认定是成功的返回码。其他返回码都会被认为是失败的返回码。HTTP 检查需要设置端口和请求路径。
   - TCP 端口检查：如果可以建立连接被认为是成功的。该检查方式需要设置检查端口。
   - 执行命令检查：如果命令执行成功并且返回值为 0，认为是成功；其他返回值认为是失败。该检查方式需要填写执行命令。
4. 启动延时、超时时间、检测周期、健康阈值、不健康阈值使用默认值即可。

## 就绪检查和滚动更新

部署组支持立即更新和滚动更新两种更新方式：立即更新会先停止**所有**应用实例，然后使用新的程序包或者镜像版本部署；滚动更新会根据更新策略，分批更新部署组内的实例。
如果部署组使用滚动更新并且开启就绪检查，当就绪检查失败时滚动更新会被阻塞。

## TSF 健康检查和 Kubernetes 健康检查的关系

对于容器部署组，存活检查和就绪检查和 Kubernetes 的 Liveness 和 Readiness Probe 对应。除此之外，TSF 就绪检查还会和注册中心进行关联，当检查失败时会将实例从注册中心屏蔽，避免流量打到异常实例上。

> **说明：**
> 关于 Kubernetes 健康检查详情参见 Kubernetes 配置存活和就绪检查 。

# 应用升级与回滚
# 应用升级与回滚概述

最近更新时间：2025-03-10 18:08:33

## 概述

应用部署到 TSF 后，还会不断进行升级迭代，TSF 单应用发布支持单批发布（直接升级）、多部署组批次发布、蓝绿发布、全链路灰度发布等不同的发布方式，用户可以根据业务场景需求不同进行选择。当发现升级后的应用出现异常，可以将应用回滚到历史版本，待问题修复后再进行升级发布。
本文介绍 TSF 的发布类型的适用场景及其升级方式和发布回滚功能。

> **说明：**
> 发布计划功能仅专业版、铂金版支持。

## 升级应用

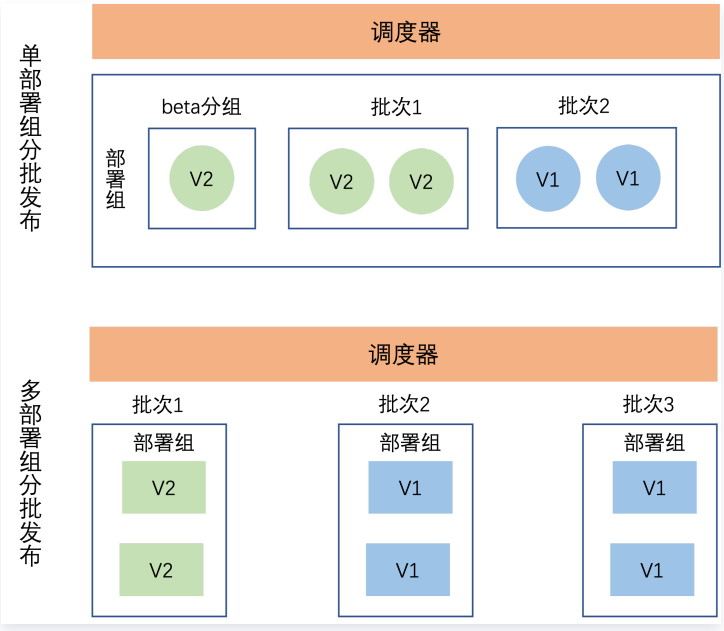
### 单批发布（直接升级）

- 适用场景：应用规模较小或服务架构比较简单，当完成开发测试后，可直接进行升级以便验证。
- 升级方式：应用部署发布策略。

### 多批次发布

- 适用场景：应用规模较大或服务架构比较复杂，可以按批次进行应用部署，分批发布过程中如果出现故障，可以及时终止并回滚，待问题修复后重新发布，降低因版本发布导致业务中断的风险。

  > **说明：**
  > 下图所示，V1（蓝色）代表存量版本，V2（绿色）代表更新版本。




- 升级方式：发布计划。
- 相关文档：分批发布 。

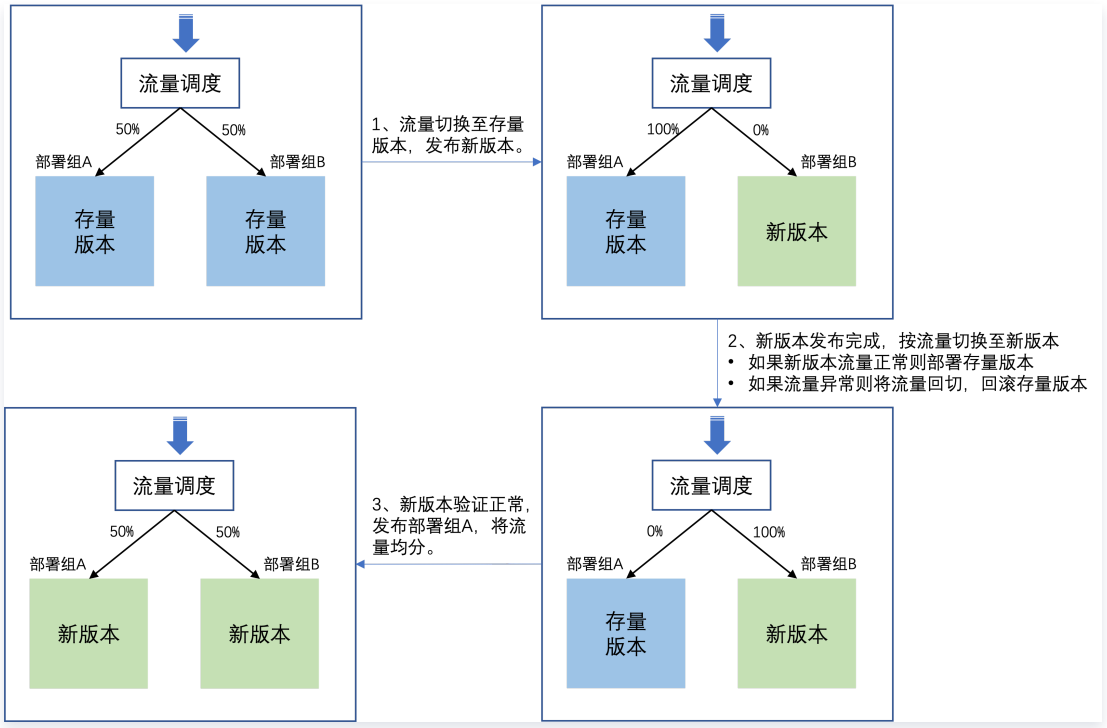
### 蓝绿发布

- 适用场景：蓝绿发布的含义是分别蓝、绿两个分组部署服务不同版本，蓝组为存量版本，绿组为更新版本。发布步骤如下：

  1.1 发布前先将流量切换至蓝组（存量版本）。

  1.2 发布绿组，发布成功后将流量切换至绿组。

  - 如果流量异常则即时切换至蓝组，避免业务异常。
  - 如果流量正常则更新蓝组版本或下线蓝组版本。




- 升级方式：发布计划。
- 相关文档：蓝绿发布 。

### 全链路灰度发布

- 适用场景：当有新功能开发完成准备上线时，可以通过全链路灰度发布先选择一小部分用户流量导入到包含新功能的新版本应用及相关联的全链路服务中进行小规模验证，验证后，再将应用全量升级到新版本。
- 发布方式：结合注册配置治理中心。
- 相关文档：
  - 概述 。
  - 泳道配置 。
  - 灰度发布 。

## 发布回滚

若在发布过程中或者发布完成后发现新版本应用异常，可以使用回滚功能回滚应用，待问题解决后再次发布新版本。

- 发布中回滚：对于正在运行中的发布单，若发现异常，需要先终止发布再进行回滚，终止发布未执行的任务节点不再进行发布操作。
- 发布后回滚：对于已发布的发布单，若发现异常，直接回滚历史版本即可。

发布回滚的具体操作，请参见 发布回滚 。

# 分批发布

最近更新时间：2025-03-10 18:08:33

## 操作场景

应用规模较大或者服务架构比较复杂时，可以进行分批次发布，降低因版本发布导致业务中断的风险。
本文介绍如何在腾讯微服务平台控制台分批次发布应用。

## 操作步骤

### 步骤1：新建发布单 – 基本信息

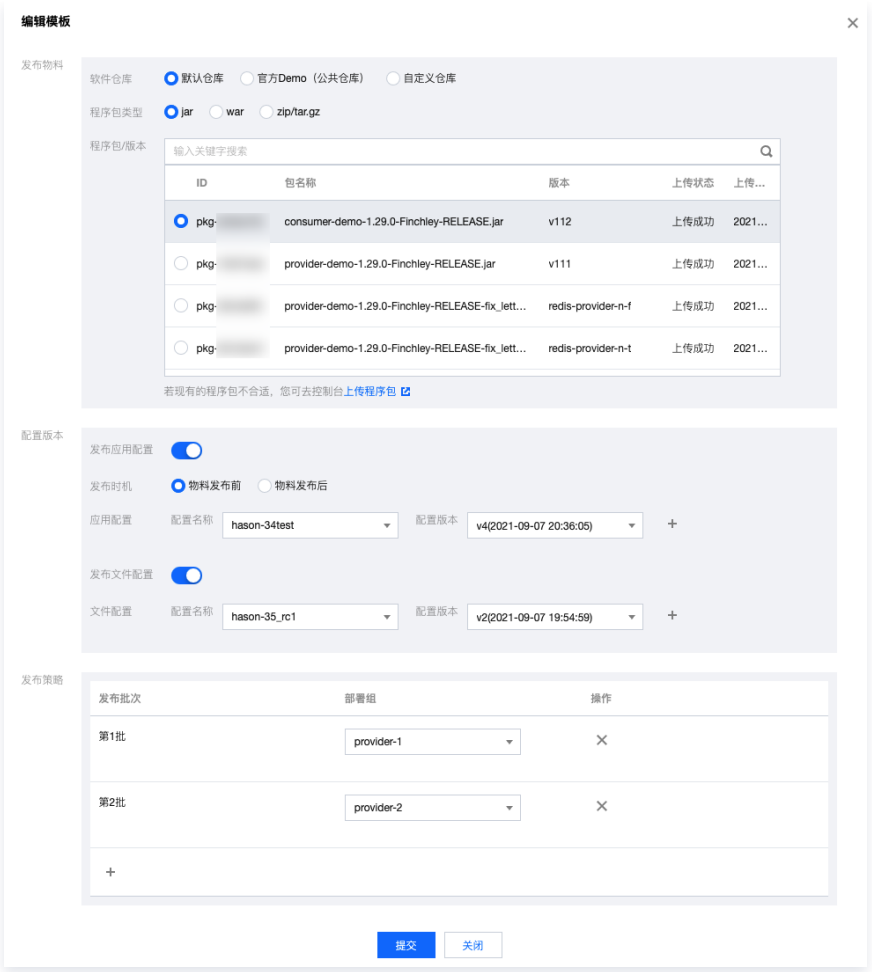
1. 登录 腾讯微服务平台控制台，选择**应用管理**，单击应用 ID，进入应用详情。
2. 在左侧菜单栏选择**应用部署**，选择**发布计划**，单击**新建发布单**。
3. 填写发布单基本信息。




   - 名称：填写发布单名称，最长60个字符，支持英文大小写数字、横杠（-）、下划线(_)、小数点（.）。
   - 触发类型：
     - 手动触发：需要手动触发发布单。
     - 定时触发：按照定时时间自动触发发布计划。
   - 发布类型：选择**单应用发布**。
   - 数据集：选择**数据集**，关于数据集请参见 数据集管理 。
   - 启用状态：启用后，发布单才能生效。
4. 单击**下一步**。

### 步骤2：新建发布单 – 发布类型

1. 在发布类型页面，选择应用后，发布类型选择**分批发布**。
2. 单击**创建模板**，在弹窗中填写发布模板信息。

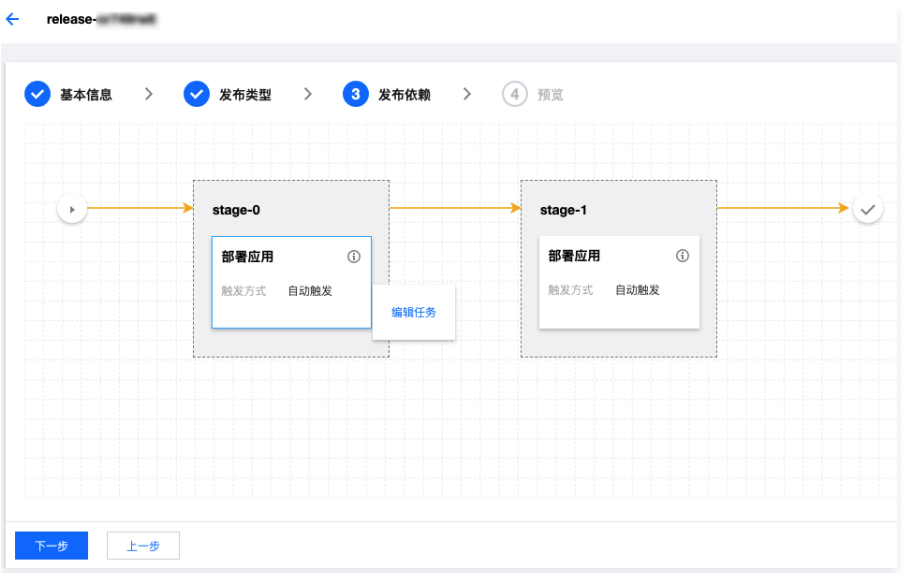



- 发布物料：选择要发布的程序包或者镜像（基于应用部署方式选择，例如应用部署时选择了 JAR 包部署，则只能选择 JAR 包）
- 配置版本：选择所要发布的应用配置和文件配置。关于配置详情参见 应用配置 和 文件配置 。
- 发布策略：在列表中选择分批发布的目标部署组。

3. 单击**提交**，完成模板配置，返回发布类型页面。
4. 单击**下一步**。

## 步骤3：新建发布单 – 发布依赖

在发布依赖页面，会按照模板生成发布依赖拓扑，支持针对部署应用阶段进一步确定发布参数配置，如果不选即为默认或原有部署组的发布参数配置。




1. 单击**部署应用**卡片，选择**编辑任务**，在弹窗中设置发布参数配置。




- 触发方式：支持自动触发和手动触发。
  - 自动触发：按照**任务时长**自动发布剩余批次的发布。
  - 手动触发：需要手动触发下一个批次的发布。

> **说明：**
> 虚机部署场景下批量更新不允许设置为手动触发模式。

- 任务时长：设置任务执行超时时间，设置手动触发但超过24小时未执行则判定为超时。
- 发布物料：由模板确定不可编辑。
- 配置版本：由模板确定不可编辑。
- 发布参数：

- 启动参数：填写应用启动参数。
- 更新方式：
  - 立即更新：会先停止所有运行实例，然后使用新的程序包版本部署，会造成发布期间短暂停服。
  - 滚动更新：当部署组内有多个实例时，可以选择滚动更新方式进行分批发布。
- 健康检查：详细参见 健康检查 。

2. 单击**提交**，完成模板配置，返回发布依赖页面。
3. 单击**下一步**。

## 步骤4：预览

在预览页面，会展示发布单的基本信息和发布任务的依赖拓扑图。




确认无误后单击**完成**即完成发布单创建，在发布计划列表中可看到刚刚创建的发布单。

## 步骤5：执行发布单

> **说明：**
> - 若发布单处于**未初始化**状态时，发布单无法执行，需要编辑完成发布单信息才能被初始化。
> - 发布单必须被启用后才能执行，若未启用，在发布单列表的启用状态栏单击**启用**按钮即可。

针对发布单的触发类型不同，发布单的执行方式也有所不同。

- 自动触发：对于自动触发的发布单，无需手动操作，满足条件后发布单会自动执行。
- 手动触发：对于手动触发的发布单，需要单击发布单操作列的**运行**手动运行发布单。




## 结果验证

### 查看变更记录

1. 单击发布计划的ID，进入发布计划基本信息页面。
2. 在页面上方选择**发布记录**页签，通过发布记录查看此次批次发布的结果是否为**成功**。
   - 如果为**成功**，说明批次发布成功。
   - 如果为**失败**，说明批次发布失败。




### 查看应用详情

单击应用“ID”，进入部署组列表页面，查看目标部署组的状态是否为**正常**，已部署的程序包是否为需要发布的最新版本。

# 蓝绿发布

最近更新时间：2025-03-10 18:08:33

## 操作场景

在蓝绿发布过程中，有两套生产环境。蓝色是当前版本并拥有实时流量，绿色是包含更新代码的环境。无论任何时候，只有一套环境有实时流量。蓝绿发布的整体流程如下：

1. 发布前先将流量切换至蓝组（存量版本）。
2. 发布绿组，发布成功后将流量切换至绿组。
   - 如果流量异常则及时切换至蓝组，避免业务异常。
   - 如果流量正常则更新蓝组版本或下线蓝组版本。

本文介绍如何在通过 TSF 控制台上使用蓝绿发布方式发布应用。

## 操作步骤

### 步骤1：新建发布单 – 基本信息

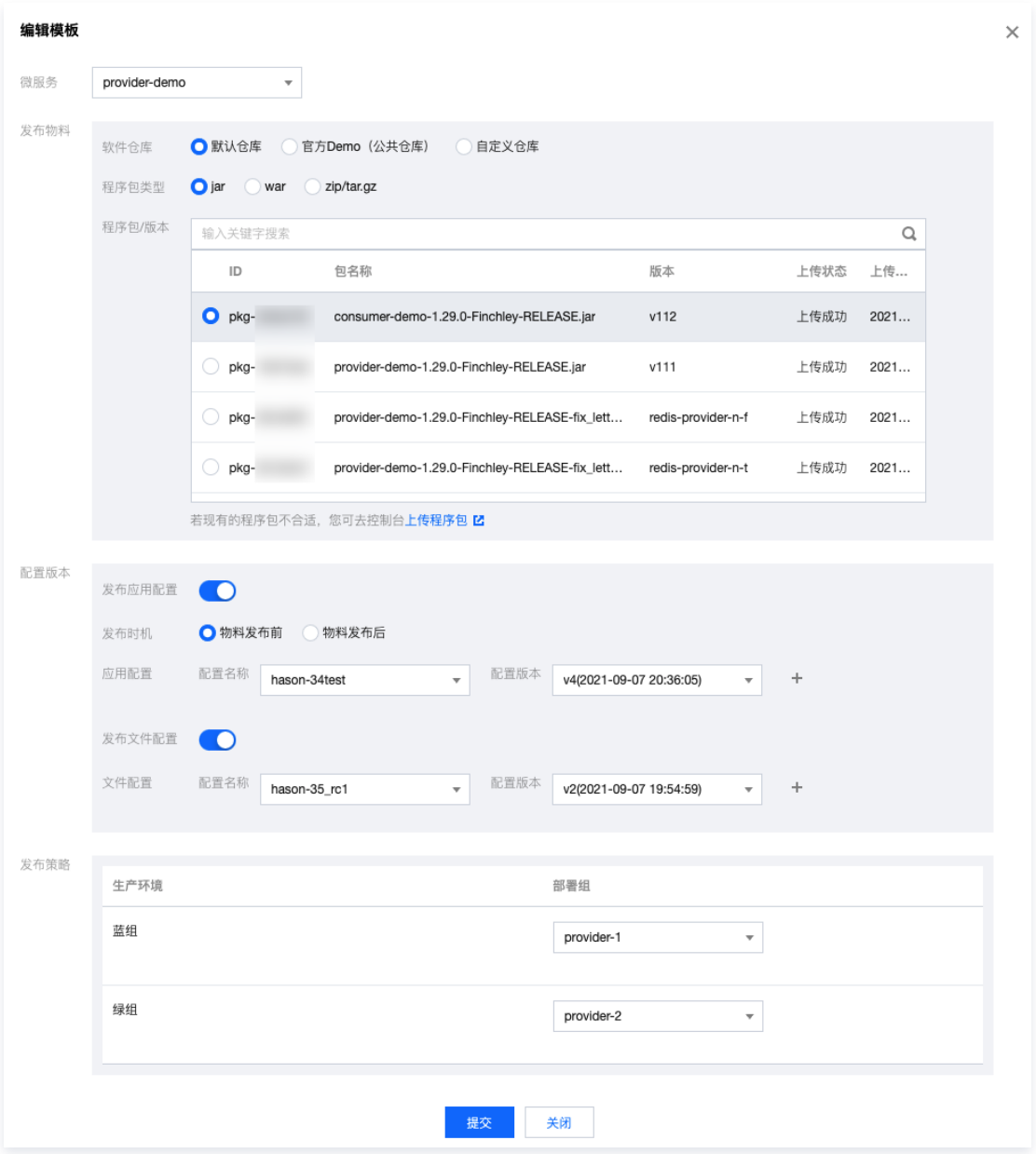
1. 登录 腾讯微服务平台控制台，进入**应用管理**，单击应用 ID，进入应用详情。
2. 在左侧菜单栏选择**应用部署**，选择**发布计划**，单击**新建发布单**。
3. 填写发布单基本信息。




   - 名称：填写发布单名称，最长60个字符，支持英文大小写数字、横杠（-）、下划线（_）、小数点（.）。
   - 触发类型：
     - 手动触发：需要手动触发发布单。
     - 定时触发：按照定时时间自动触发发布计划。
   - 发布类型：单应用发布。
   - 数据集：选择**数据集**，关于数据集请参见 数据集管理 。
   - 启用状态：启用后，发布单才能生效。
4. 单击**下一步**。

### 步骤2：新建发布单 – 发布类型

1. 在发布类型页面，选择应用后，发布类型选择**蓝绿发布**。
2. 单击**创建模板**，在弹窗中填写发布模板信息。

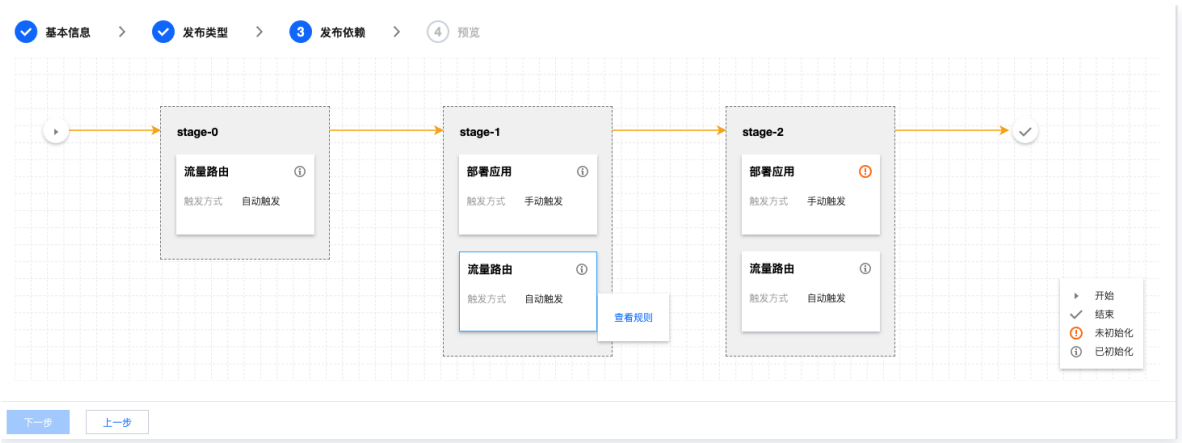



- 发布物料：选择要发布的程序包或者镜像（基于应用部署方式选择，例如应用部署时选择了 JAR 包部署，则只能选择 JAR 包）。
- 配置版本：选择所要发布的应用配置和文件配置。关于配置详情参见 应用配置 和 文件配置 。
- 发布策略：在列表中选择蓝组和绿组的目标部署组。

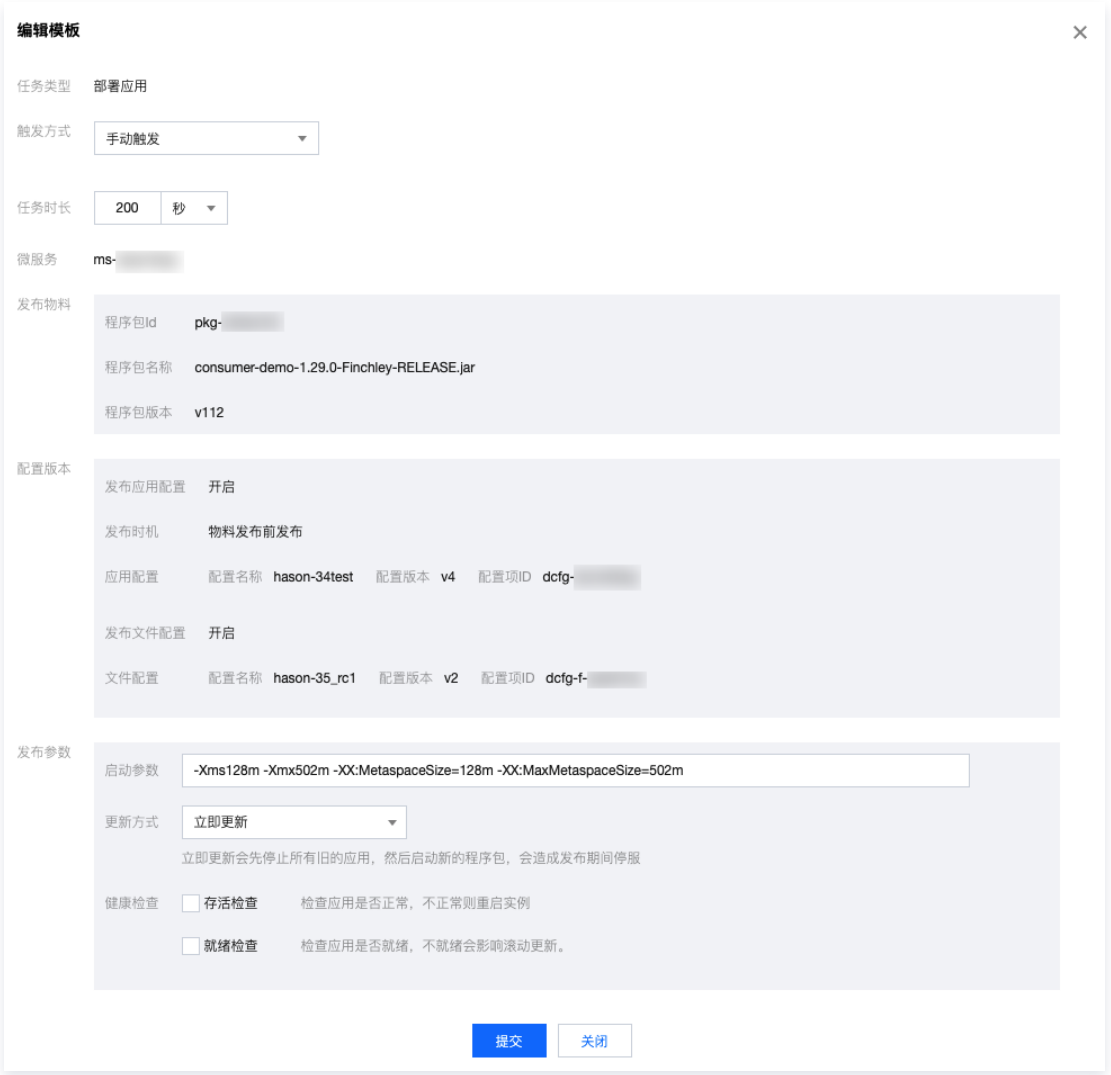
3. 单击**提交**，完成模板配置，返回发布类型页面。
4. 单击**下一步**。

## 步骤3：新建发布单 – 发布依赖

在发布依赖页面，会按照模板生成发布依赖拓扑，支持针对部署应用阶段进一步确定发布参数配置，如果不选即为默认或原有部署组的发布参数配置。




1. 单击**部署应用**卡片，选择**编辑任务**，在弹窗中设置发布参数配置。




- 触发方式：支持自动触发和手动触发。
  - 自动触发：按照任务流程自动发布剩余批次的发布。
  - 手动触发：需要手动触发下一个批次的发布。

  > **说明：**
  > 虚机部署场景下批量更新不允许设置为手动触发模式。

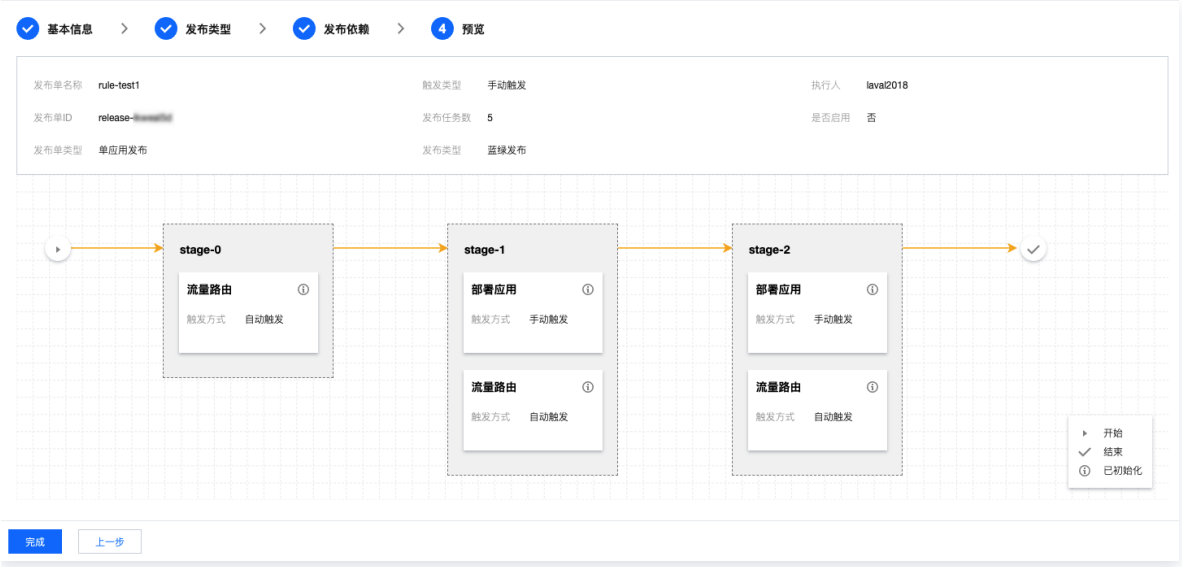
- 任务时长：设置任务执行超时时间，设置手动触发但超过24小时未执行则判定为超时。

- 发布物料：由模板确定不可编辑。
- 配置版本：由模板确定不可编辑。
- 发布参数：
  - 启动参数：填写应用启动参数。
  - 更新方式：
    - 立即更新：会先停止所有运行实例，然后使用新的程序包版本部署，会造成发布期间短暂停服。
    - 滚动更新：当部署组内有多个实例时，可以选择滚动更新方式进行分批发布。
  - 健康检查：详细参见 健康检查 。

2. 单击**提交**，完成模板配置，返回发布依赖页面。
3. 单击**下一步**。

## 步骤4：预览

在预览页面，会展示发布单的基本信息和发布任务的依赖拓扑图。




确认无误后单击**完成**即完成发布单创建，在发布计划列表中可看到刚刚创建的发布单。

## 步骤5：执行发布单

> **说明：**
> - 若发布单处于**未初始化**状态时，发布单无法执行，需要编辑完成发布单信息才能被初始化。
> - 发布单必须被启用后才能执行，若未启用，在发布单列表的启用状态栏单击**启用**即可。

针对发布单的触发类型不同，发布单的执行方式也有所不同。

- 自动触发：对于自动触发的发布单，无需手动操作，满足条件后发布单会自动执行。
- 手动触发：对于手动触发的发布单，需要单击发布单操作列的**运行**手动运行发布单。




# 结果验证

## 查看变更记录

1. 单击发布单的 ID，进入发布计划基本信息页面。

2. 在页面上方选择**发布记录**页签，通过发布记录查看此蓝绿发布的结果是否为**成功**。
   - 如果为**成功**，说明蓝绿发布成功。
   - 如果为**失败**，说明蓝绿发布失败。




### 查看应用详情

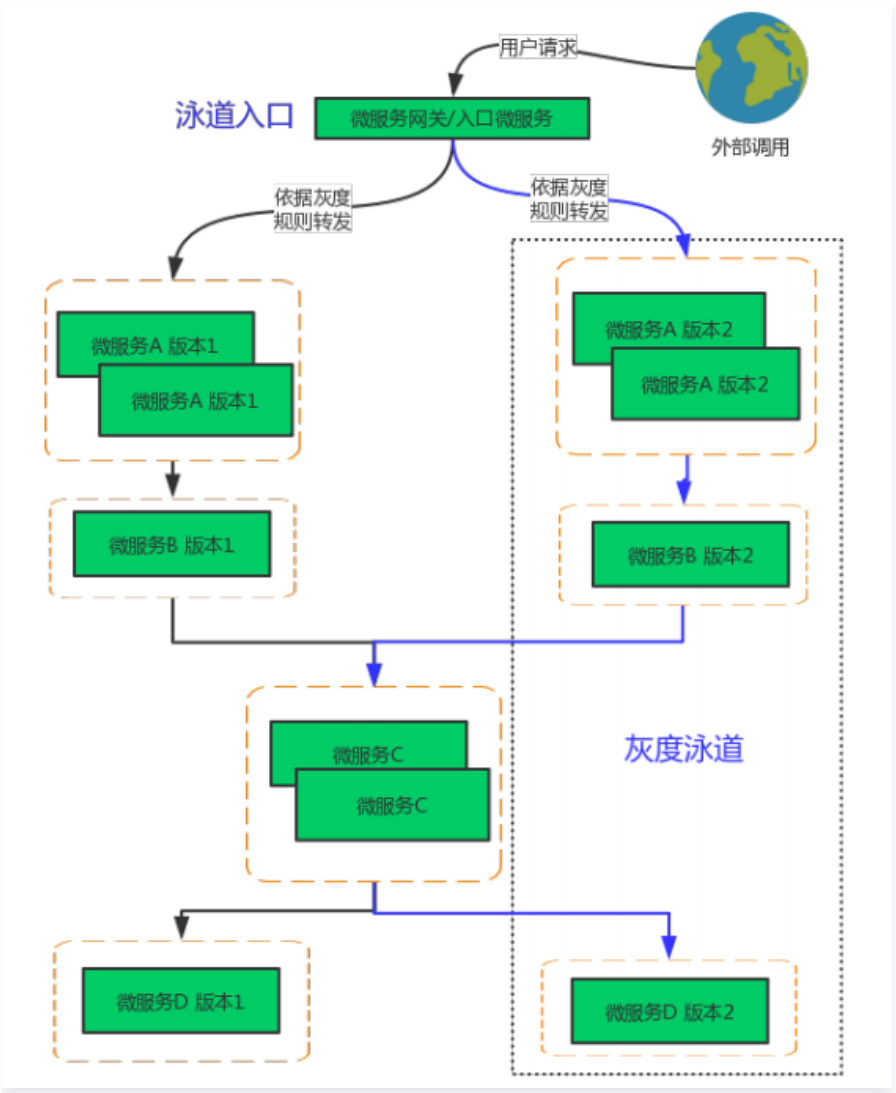
单击应用“ID”，进入部署组列表页面，查看目标部署组的状态是否为**正常**，已部署的程序包/镜像是否为需要发布的最新版本。

# 全链路灰度发布

## 概述

最近更新时间：2025-03-10 18:08:33

灰度发布是软件上线过程中常见的上线方式，是指在发布过程中，将具有一定特征或者比例的流量分配到需要被验证的版本中，用来观察新的验证版本的线上运行状态。
相比全量上线，灰度发布是更加谨慎的发布形式。当线上调用链路较为复杂时，全链路灰度发布可以将线上的各个服务隔离出一个单独的运行环境。
全链路灰度发布的常见场景如下：




如上图所示，我们划分出一个灰度环境，针对线上版本2的所有服务，我们希望都能配置一定规则。当满足一定规则后，所有请求都会流入灰度环境中，经过微服务 A 版本2的请求，一定会同样落入微服务 D 的版本2中。

### 名词解释

- **泳道**：泳道是一组部署组的集合，是灰度发布规则的目的地。泳道中的部署组属于不同的应用。可以认为用户通过划分泳道划分出了灰度环境。
- **灰度规则**：用户在灰度规则上配置请求需要满足的条件。当请求满足一定条件后，可以通过灰度规则将流量路由到某一个泳道中。

版权所有：腾讯云计算（北京）有限责任公司

# 泳道配置

最近更新时间：2025-03-10 18:08:33

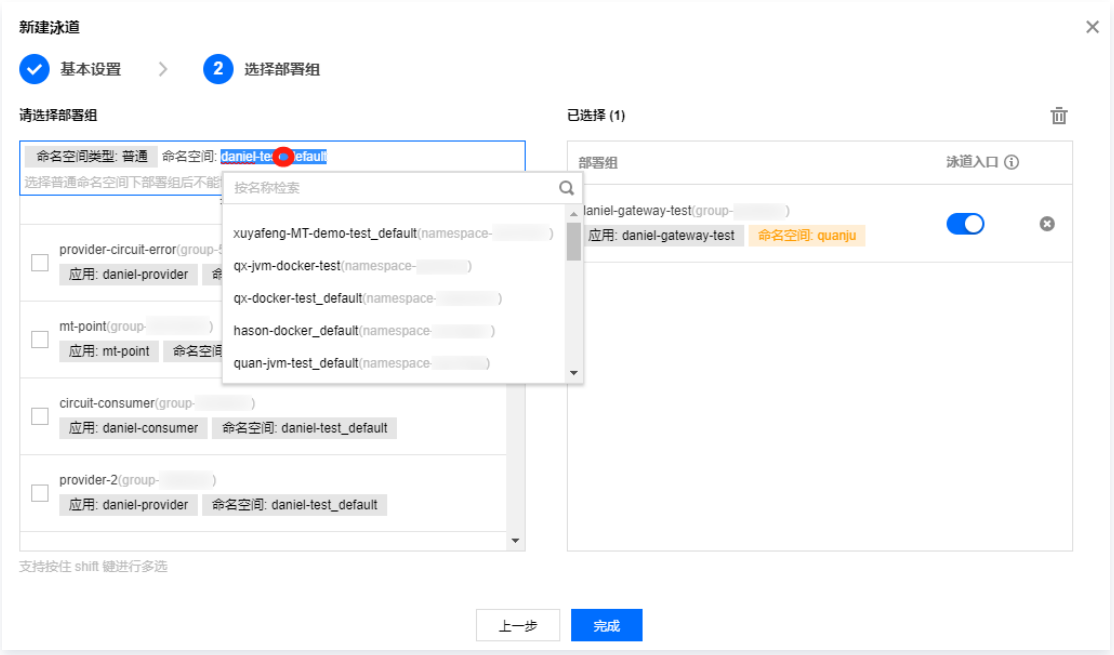
## 操作场景

使用全链路灰度发布之前，需要先配置泳道。一条泳道相当于一个灰度环境，环境中包含应用中需要进行灰度测试的部署组。在 全链路发布概述 中的示例图中，我们希望为不同微服务的版本2创建一条泳道，将微服务 A、B、D 部署版本2的部署组都放在一个泳道中。
假设当前访问链路为 gateway > consumer > provider。其中 consumer、provider 都有两个版本。希望通过请求参数 test = 1时配置流量通过网关访问到 consumer、provider 的版本1中。

> **说明：**
> 该操作文档仅对注册配置治理中心使用 TSF-Consul 的存量用户有效，如注册配置治理中心使用北极星，请参考北极星全链路灰度使用指引。

## 新建泳道

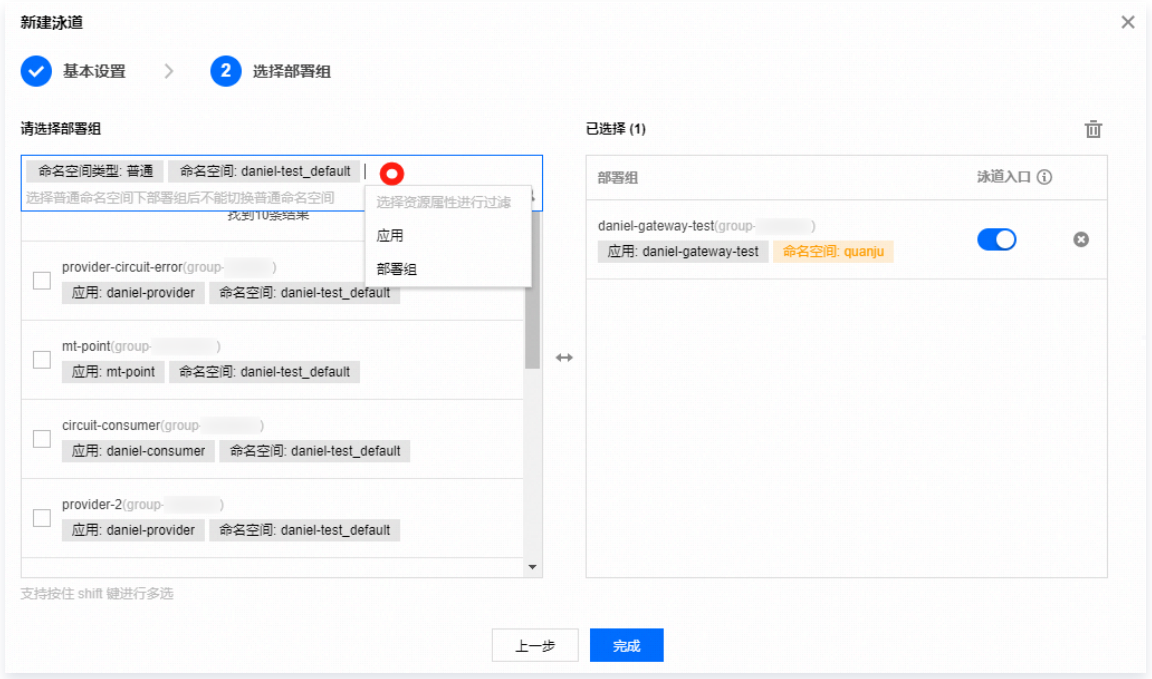
1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏单击 **TSF-Consul**，在页面上方选择 **全链路灰度发布** 页签，进入灰度发布页面。
3. 选择**泳道配置**页签，单击**新建泳道**，填写泳道基本信息。
   - 泳道名称：必填。最长60个字符，只能包括小写字母、数字及分隔符。
   - 标签：用于分类管理资源，可不选。详情参见 标签 。
   - 备注：选填。最长200字符。
   - 数据集：用于细粒度管理子账号权限。数据集使用，请参阅 数据集管理 。
4. 单击**下一步**，勾选需要导入泳道的部署组，单击**完成**。
   此时，我们需要创建一个泳道，将 gateway、consumer 的版本1部署组、provider 的版本1部署组添加到这个泳道中。设置 gateway 为泳道的入口。




### 过滤条件

- 用户可以通过选择命名空间类型、命名空间名称、应用名称和部署组名称进行筛选过滤部署组。默认提供了命名空间类型和命名空间名称作为过滤条件。
- 当用户希望切换命名空间时，单击当前已经选中的命名空间的名称，在弹出的下拉列表中切换。

- 当用户希望通过应用名称筛选部署组时，单击命名空间右侧的空白位置，便可弹出应用和部署组的下拉选项，如图所示。




### 泳道入口

- 全链路灰度发布规则配置后，会在泳道的入口部署组上对请求规则进行校验，以此来判断请求是否应该进入某一个泳道中。通常情况下，泳道入口是一个网关。
- 同一个泳道中支持多个入口，在请求经过每一个入口部署组时，都会判断请求是否应该进入泳道中。
- 配置方法：在选择部署组页面勾选需要被导入到泳道中部署组后，在穿梭框的右侧通过泳道入口开关 配置。

### 泳道与命名空间

- 当前 TSF 支持两种命名空间：全局命名空间和普通命名空间。
- 泳道中可以包含全局命名空间中的部署组和普通命名空间中的部署组。

> **说明**
> 当前接入方式为服务网格，TSF-Mesh 的应用不支持部署在全局命名空间中，不支持泳道中同时出现全局命名空间中的部署组和 Mesh 应用的部署组。

## 删除泳道

> **注意**
> 当泳道上已经绑定了全链路灰度发布的规则，则不能删除。

在泳道配置列表页面，选择目标泳道，单击操作列的**删除**。删除后，全链路灰度规则在该泳道上将不可用。

## 编辑泳道

1. 在泳道配置列表页面，单击目标泳道的“ID”，进入泳道详情页。
2. 在基本信息模块中，单击右上角的**编辑**，可以修改泳道名称和备注。
3. 在部署组管理模块中，可以向泳道添加或移除部署组，也可以设置某个部署组为泳道入口。

## 查看泳道监控信息

1. 在泳道配置列表页面，单击目标泳道的“ID”，进入泳道详情页。
2. 选择**泳道监控**页签，设置好时间范围后，即可查看该泳道的请求量、耗时和错误率信息。

# 灰度发布

最近更新时间：2025-03-10 18:08:33

## 操作场景

创建好泳道后，我们需要配置灰度发布规则。延续泳道配置中的场景：假设当前访问链路为 gateway > consumer > provider。其中 consumer、provider 都有两个版本。希望通过请求参数 test = 1 时配置流量通过网关访问到 consumer、provider 的版本1中。
当前，gateway 以及 consumer、provider 的版本1 已经放在了泳道中，我们需要配置一条规则，当请求参数 test = 1时，流量到版本1的泳道中。

## 前提条件

- 已 创建服务泳道 。
- 不支持创建子线程进行调用后的灰度发布。

## 新建灰度规则

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏单击 **TSF-Consul**，在页面上方选择 **全链路灰度发布** 页签，进入灰度发布页面。
3. 单击**新建灰度规则**，填写规则名称和备注。
   - 规则名称：必填。支持中英文字符，不超过60字符。
   - 标签：用于分类管理资源，可不选。详情参见 标签 。
   - 备注：选填。不超过200字符。
4. 单击**下一步**，填写请求参数规则和规则生效关系。




- **标签名**：是 TSF 中的自定义标签，自定义标签是用户自定义的业务参数转换的 TSF 中使用的标签。 参数名最多32字节，参数值最多128字符。此处，我们配置标签名为 test，设置标签值为1。
  自定义标签的来源有两种：
  - 方式一：通过微服务网关将请求的参数转换为标签：如请求的 path、query、header 中包含的业务参数，通过微服务网关的 tag 插件转换为标签。详细使用说明可以参见 微服务网关 Tag 插件 。
  - 方式二：已在代码中配置标签，详细使用说明可以参见 参数传递 。
- **逻辑关系**：支持等于、不等于、包含、不包含、正则表达式五种逻辑关系。
  - 选择等于和不等于时，标签值只能填写为单个值。
  - 选择包含和不包含时，标签值可以填写多个，使用英文半角逗号分隔。
- **规则生效关系**：上述条件的生效逻辑关系，或（满足任一规则）、与（满足全部规则）。每条全链路灰度规则最多包含10条子规则。

版权所有：腾讯云计算（北京）有限责任公司

5. 单击**下一步**，配置流量目的地，选择已经创建好的泳道。




6. 单击**完成**。

## 切换灰度规则生效状态

在灰度发布页面，通过生效状态栏的按钮 调整规则是否生效。支持生效多条规则。

## 调整灰度规则优先级

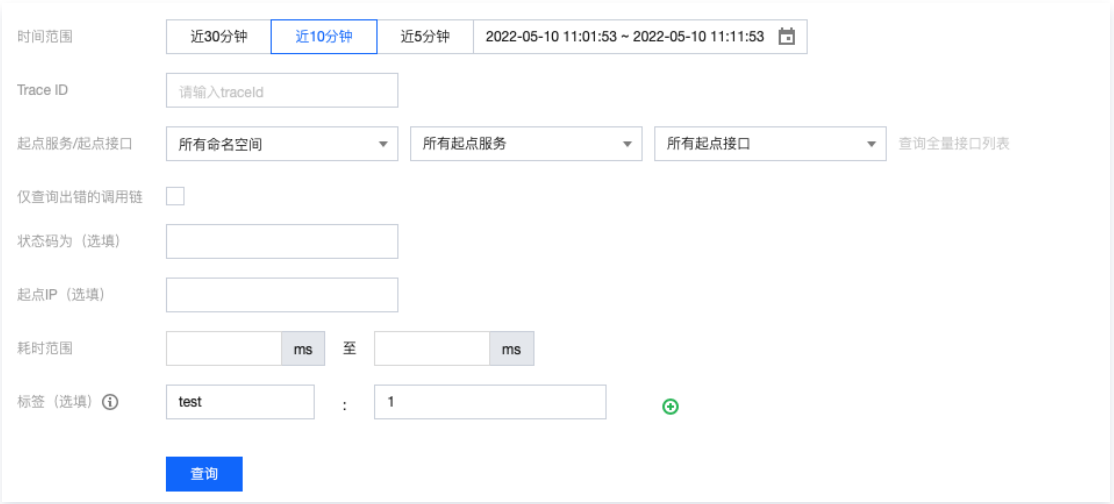
全链路灰度发布支持同时生效多条规则，并支持为规则配置优先级。当同一条请求同时满足多条规则时，会优先匹配高优先级的规则。

在灰度发布页面，单击**调整规则优先级**，拖动规则名左侧的拖拽图标，进行优先级配置，并单击**保存**。实际生效的优先级是生效状态的规则从页面上方向下的顺序进行排列的。

## 全链路灰度结果展示

在 腾讯微服务平台控制台的**依赖分析 > 调用链查询**中，用户可以查询对应的请求是否按照预期走到了泳道中。

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏单击**观测中心 > 调用查询**，进入调用链查询页面。
3. 选择好时间、调用的命名空间和微服务，或通过其他条件筛选需要查询的调用链。如果需要对标签和标签值进行过滤，可以输入标签和标签值，进行过滤。



4. 单击**查询**，结果如下。




5. 单击非泳道入口的微服务的调用链详情或者蓝色的耗时条，可以查看调用链详情。




## 全链路灰度流量流向规则与说明

- 当请求流量没有命中任何灰度规则，流量将走到没有被添加到泳道的部署组中。
- 当某一个微服务下的部署组没有被加入任何泳道中，请求将在该微服务下所有部署组的所有实例中轮询。经过该微服务后，请求将继续按照规则配置流入对应泳道。
  举例：A > B > C 微服务调用，A 和 C 微服务下有版本1和版本2，B只有一个版本。配置灰度规则test = 1 的流量进入版本2对应的泳道后，流量将从 A 的版本2进入，经过 B 微服务，流入到 C 微服务的版本2中。
- 一个部署组可以属于多个泳道。
- 在泳道中，服务路由规则不生效。
- 泳道是一个隔离环境。当泳道上所有规则关闭后，流量不会进入泳道中。如希望恢复建立泳道前的流量分配方式，请将泳道删除。

# 查看发布详情

最近更新时间：2025-03-10 18:08:33

## 操作场景

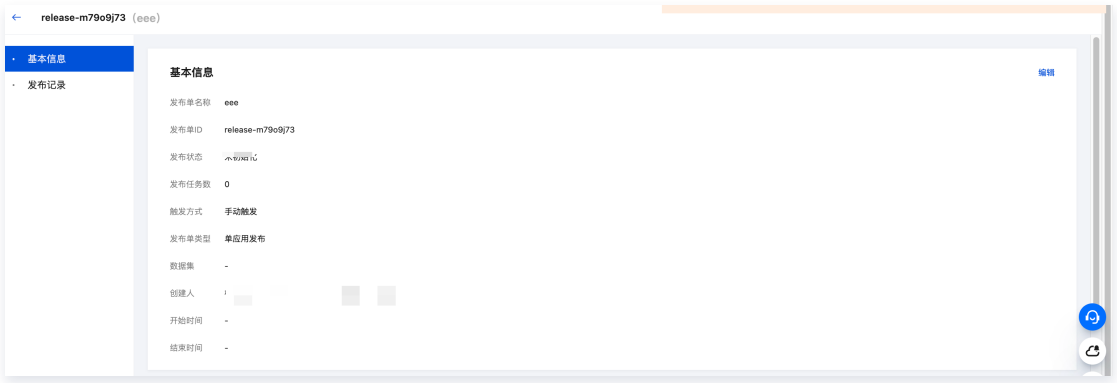
发布单执行后，可以在发布记录中查看发布详情，针对发布成功的发布记录可执行发布回滚操作，用户确认发布回滚操作后按照发布流程依次进行回滚操作。
关于回滚相关操作请参见 发布回滚 。本文介绍在发布单执行后如何查看发布详情。

## 前提条件

发布单已开始执行。

## 操作步骤

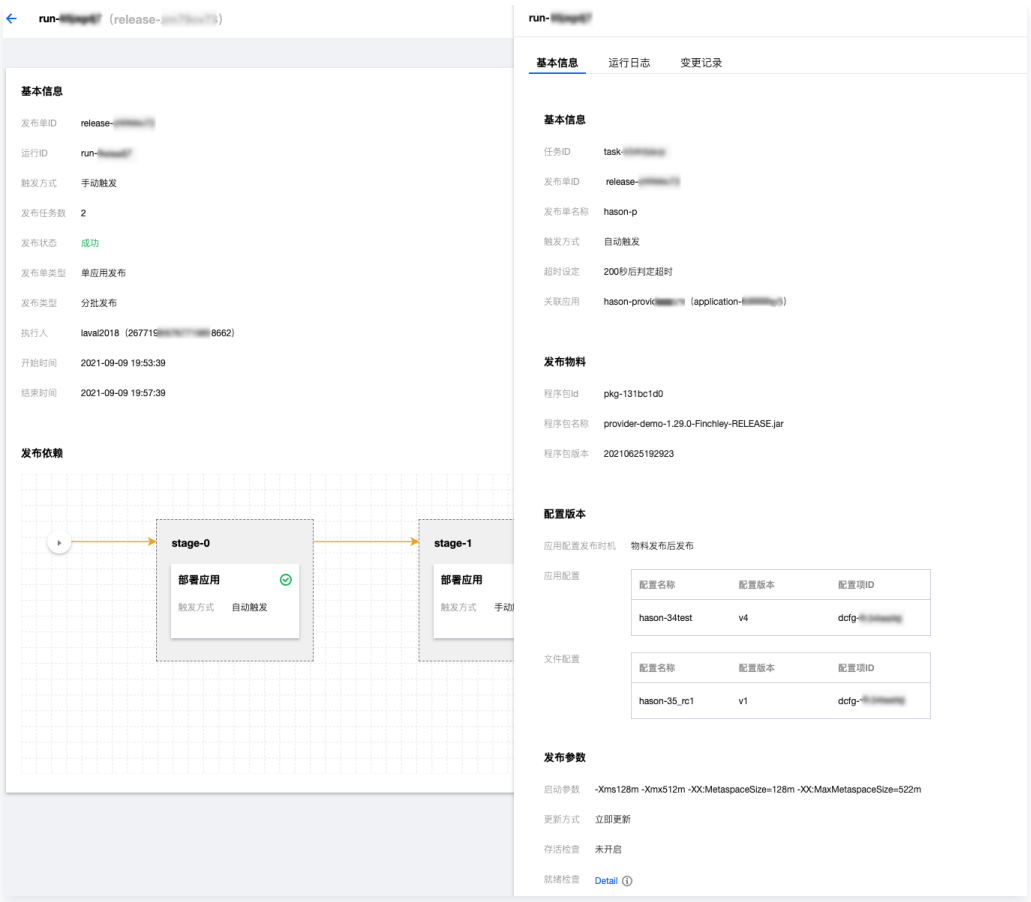
1. 登录 腾讯微服务平台控制台 ，选择**应用管理**，单击应用ID，进入应用详情。
2. 在左侧菜单栏选择**应用部署**，选择**发布计划**，但单击**发布单ID**。
3. 在发布单基本信息页面，选择左侧菜单栏的**发布记录**，可查看执行的发布记录列表。
4. 单击“运行 ID”，可查看发布记录详情，包含发布记录基本信息和发布依赖拓扑。



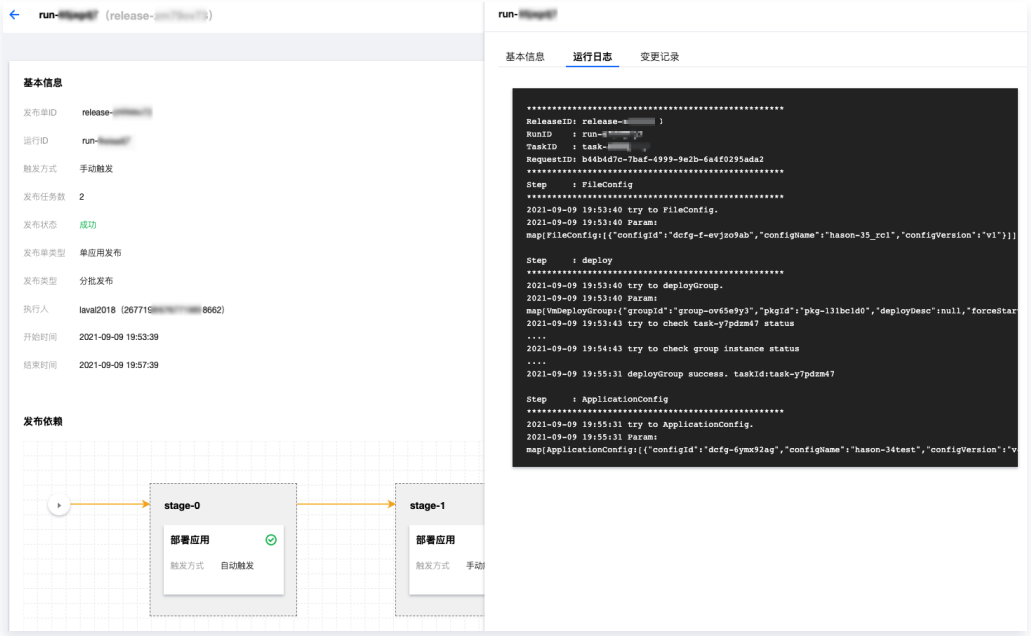

5. 单击**部署应用**或**流量路由**卡片，选择**查看详情**，可查看详细信息。

### 部署应用节点详情

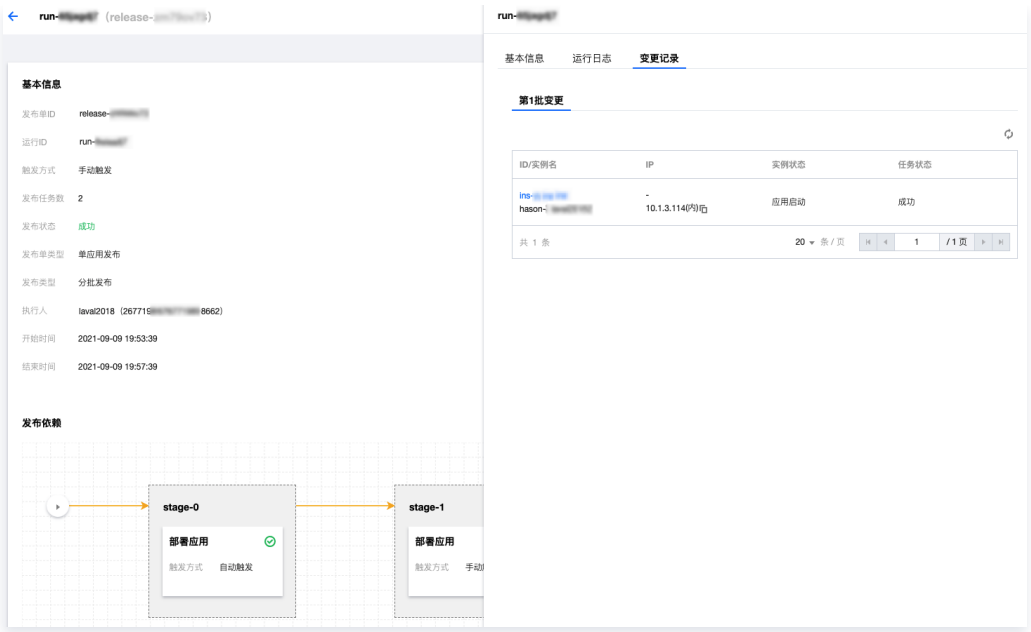
- 基本信息




- 任务节点执行日志



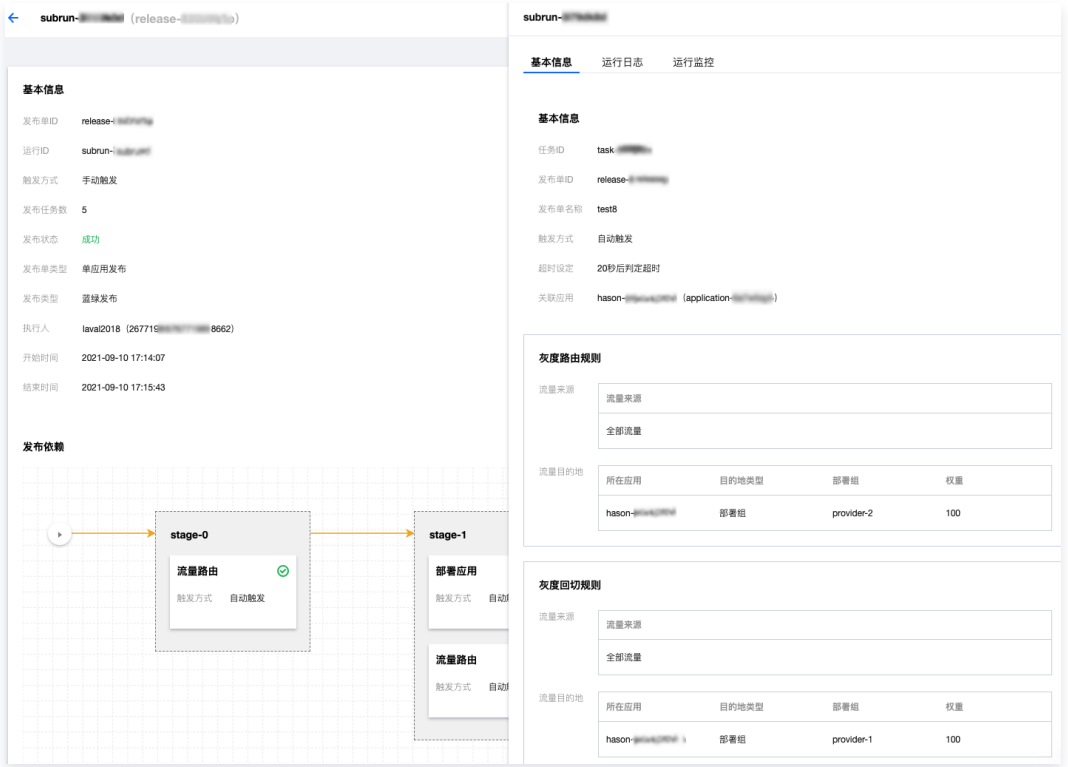

- 变更记录




## 流量路由节点详情

> **说明**
> 只有蓝绿发布才会有流量路由节点。

- 基本信息



- 任务节点执行


基本信息
发布单ID
运行ID
触发方式
手动触发
发布任务数
5
基本信息
运行日志
运行监控
ReleaseID:
RunID :
TaskID :
RequestID:
a52dfb0ae3b9

- 流量


基本信息
发布单ID
运行ID
触发方式
手动触发
发布任务数
5
发布状态
成功
发布单类型
单应用发布
发布类型
蓝绿发布
执行人
laval2018
开始时间
2021-09-10 17:14:07
基本信息
运行日志
运行监控
流量详情
任务执行时
近10分钟
近30分钟
近24小时
2021-09-22 20:03:38 ~ 2021-09-23 20:03:38
暂无数据

# 发布回滚

最近更新时间：2024-11-13 16:14:53

## 操作场景

使用批次发布或者蓝绿发布新版本应用时，若在发布过程中或者发布完成后发现新版本应用异常，可以使用回滚功能回滚应用，待问题解决后再次发布新版本。

## 操作步骤

> **注意：**
> 运行开始超过7天的发布无法回滚。

### 发布中回滚

对于正在运行中的发布单，若发现异常，需要先终止发布再进行回滚，终止发布未执行的任务节点不再进行发布操作。

1. 单击发布计划的 ID，进入发布单基本信息页面。
2. 在页面上方选择**发布记录**页面，单击操作栏的**终止**，终止发布。




3. 发布终止后，单击操作栏的**回滚**。
4. 在回滚信息页面单击任务节点操作栏**详情**，可查看任务节点详情。




5. 在回滚信息页面选择**回滚**，弹窗中单击**确认**。




## 发布后回滚

对于已发布的发布单，若发现异常，直接回滚即可。

1. 单击发布计划的 ID，进入发布单基本信息页面。
2. 在页面上方选择**发布记录**页面，单击操作栏的**回滚**。
3. 在回滚信息弹窗中选择**回滚**，完成应用回滚。



# 应用运维
# 应用管理

最近更新时间：2025-03-10 18:08:33

## 操作场景

将容器服务集群中的应用托管至 TSF 后，可以在 TSF 控制台上对应用进行管理，包括部署新版本的应用、扩缩容、停止、重启和查看配置等，此外也可以通过编辑 YAML 更新部署组。

## 操作步骤

> **说明：**
> 当 Consul 故障时，禁止用户重新部署发布相关操作，减少故障的扩大范围，待故障修复后，可以重新进行操作。具体受限制的操作包含：应用扩容、启动应用、停止应用、重启应用、下线实例。

### 创建应用

登录 腾讯微服务平台控制台，在**应用管理**页面，选择**业务应用**，单击**新建**创建应用，具体操作请参见 容器集群部署应用 。

### 部署应用

在 腾讯微服务平台控制台 创建一个未部署的应用后，您可以单击应用 ID进入**应用详情**页，单击**应用部署**对应用进行部署。具体操作请参见 容器集群部署应用 。
已部署过的应用，可以在应用部署的部署组列表菜单，选择**更新部署**来升级应用版本。

### 删除应用

> **注意**
> - 删除应用会同时删除应用关联的程序包。
> - 当应用下有部署组时，无法执行删除操作，需要先删除所有部署组后才能删除应用。

1. 登录 腾讯微服务平台控制台，在**应用管理**页面的**操作**列，单击**删除**。
2. 在确认弹框中，单击**确认**完成删除。

### 删除部署组

> **注意：**
> 部署组删除后将不再对外提供服务。

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的“ID”，进入应用详情页，左侧导航栏选择**应用部署**，进入部署组列表页。
2. 在部署组列表页面，单击目标部署组操作列的**更多** > **删除**。
3. 弹出提示页面，单击**确定**删除部署组。

### 应用扩缩

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击目标部署组操作列的**更多** > **应用扩缩**。
3. 选择扩缩的实例数量后，检测是否有足够的资源。若资源充足，则单击提交。若集群内主机资源不足以创建实例，则请先在集群内导入云主机，详情参见 集群管理 。

> **注意：**

> 该检测仅适用不包含超级节点的TKE集群，若 TKE 集群中添加了超级节点，可通过弹性集群进行灵活伸缩。

## 停止应用

应用停止后，将不再对外提供服务。

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击目标部署组操作列的**更多 > 停止应用**。
3. 弹出提示页面，单击**确定**停止应用。

## 启动应用

应用停止后，您可以启动应用，应用将重新对外提供服务。

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击目标部署组操作列的**更多 > 启动应用**。
3. 弹出提示页面，单击**确定**启动应用。

## 重启应用

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击目标部署组操作列的**更多 > 重启应用**。
3. 修改启动参数后，单击**保存并重启**后重启应用。

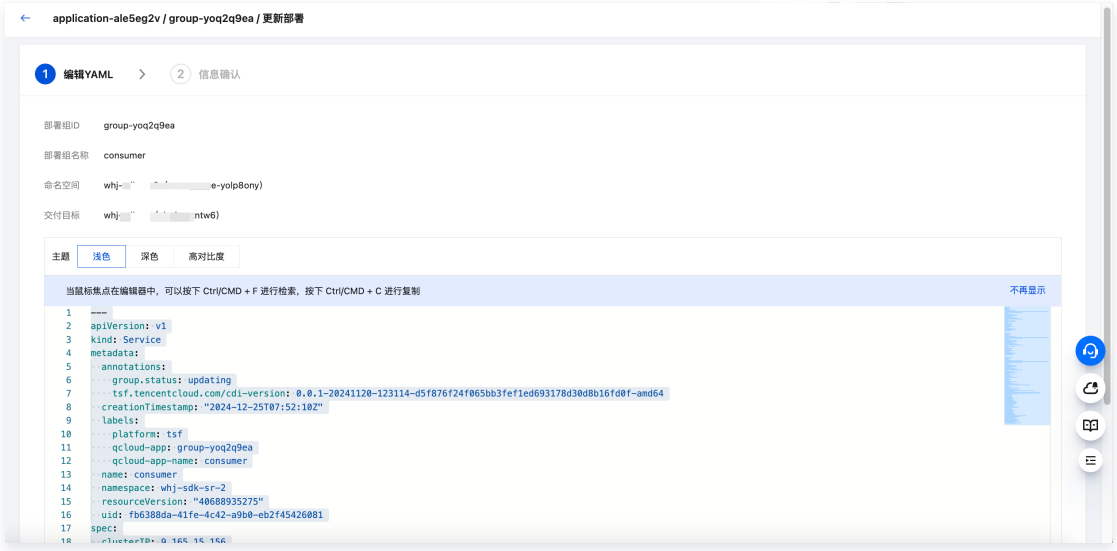
## 通过直接编辑 YAML 更新部署组

容器部署组对应的 Kubernetes 里面的 `Deployment` 和 `Service` 对象，如果您希望直接修改 YAML 实现更能灵活地配置，可以按照下述步骤操作：

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击部署组 ID 进入详情页。
3. 单击右上角查看YAML，显示 `Deployment` 和 `Service` 的 YAML。



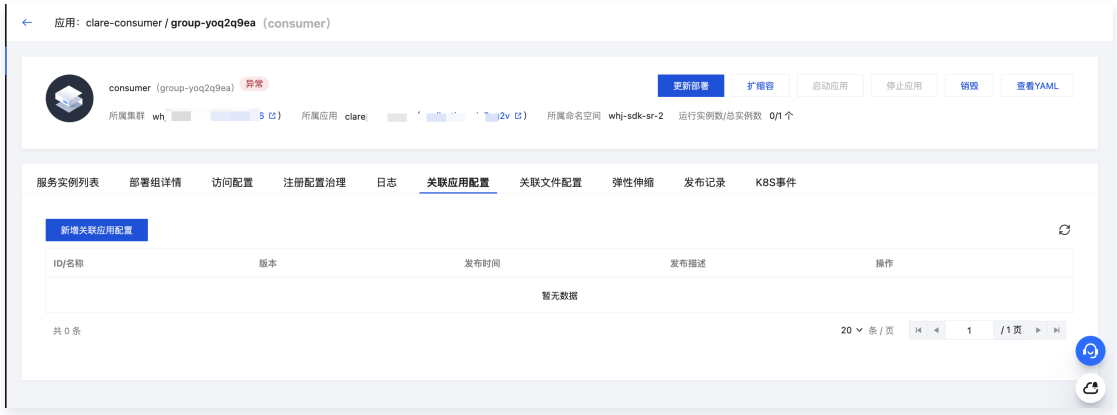

4. 单击**编辑**按钮，进入 YAML 编辑页面，编辑 YAML。




5. 单击**提交**按钮，容器部署组会以更新后的 YAML 进行重新部署。

## 关联应用配置

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击部署组 ID 进入部署组详情页。
3. 在菜单栏选择**关联应用配置**，单击**新增关联应用配置**，填写配置信息。




- 配置项：选择配置项目，如果现有的文件配置项不合适，您可以单击**新建应用配置项**重新创建。
- 版本：选择配置项版本。
- 配置内容：按照 YAML 格式。YAML 格式规范参见 YAML 格式介绍 。
- 发布描述：选填，200字符内。

4. 单击**提交**，完成新增**关联应用配置**。

## 关联文件配置

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击部署组 ID 进入部署组详情页。
3. 在菜单栏选择**关联文件配置**，单击**新增关联文件配置**，填写配置信息。
   - 配置项：选择配置项目，如果现有的文件配置项不合适，您可以单击**新建文件配置项**重新创建。
   - 版本：选择配置项版本。
   - 配置内容：按照 YAML 格式。YAML 格式规范参见 YAML 格式介绍 。
   - 发布描述：选填，200字符内。

4. 单击**提交**，完成新增**关联文件配置**。

## 通过 Webshell 登录容器

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击部署组 ID 进入部署组详情页。
3. 选择服务实例列表，选择需要登录的实例。
4. 单击操作栏的**登录**通过 Webshell 登录容器。
   登录后界面如下：



# 弹性伸缩

最近更新时间：2025-03-10 18:08:33

在分布式应用管理中，弹性伸缩是十分重要的运维能力。它能够根据预先设定的弹性伸缩规则自动增加或减少部署组实例数量，即扩容或缩容，从而提高资源利用率、降低资源成本。

## 概述

TSF 弹性伸缩支持指标规则和定时规则两种类型的弹性伸缩策略。

- **指标规则**：适用于突发流量和典型周期性流量的应用场景。由规则名、扩容活动、缩容活动、冷却时间等参数构成的规则，用来描述弹性扩缩容的触发条件、实例数量变化和限制。

  **弹性伸缩触发指标：**

| 指标 | 说明 |
| --- | --- |
| CPU 利用率 | 在指定时间范围内，部署组内所有实例 CPU 利用率的平均值。 |
| 内存利用率 | 在指定时间范围内，部署组内所有实例内存利用率的平均值。 |
| 请求 QPS | 在指定时间范围内，部署组内所有实例请求 QPS 的平均值。 |
| 响应时间 | 在指定时间范围内，部署组内所有实例响应时间的平均值。 |

- **定时规则**：适用于资源使用率有周期性规律的应用场景。达到设定的周期时间点后，TSF 将依据设定的策略自动调整部署组实例数量。

## 使用限制

- 请勿同时开启容器平台的弹性伸缩，可能会导致 TSF 的弹性伸缩规则被容器平台的弹性伸缩规则覆盖，导致 TSF 弹性伸缩规则生效后实际结果不符合预期。
- 集群中存在超级节点时，当用户需要配置被动触发弹性时，可以使用 TSF 页面的弹性伸缩能力。当用户希望进行定时扩缩容的时候，建议通过 HPC/HPA 进行配置。

## 新建规则

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航单击**应用管理**，选择**弹性伸缩**，选择好地域后单击**新建规则**，填写弹性伸缩规则内容。

**指标规则**

| 参数 | | 说明 |
| --- | --- | --- |
| 扩容活动 | 触发条件 | 由指标（CPU 利用率、内存利用率、请求 QPS 和响应时间），阈值和持续时间构成。**多条触发条件为逻辑或（OR）的关系，满足任一条件，将触发扩容操作。** |
| | 增加实例数 | 每次部署组的指标达到了触发条件后，增加的实例数量。 |
| | 最大实例数 | 部署组的实例数量上限。 |
| 缩容活动 | 触发条件 | 由指标（CPU 利用率、内存利用率、请求 QPS 和响应时间），阈值和持续时间构成。**多条触发条件为逻辑与 AND 的关系，必须同时满足才能触发缩容操作。** |
| | 减少实例数 | 每次部署组的指标达到了触发条件后，减少的实例数量。 |
| | 最小实例数 | 部署组的实例数下限。 |
| 冷却时间 | | 冷却时间是指在同一个伸缩组内，一个伸缩活动（添加或移出实例）执行完成后的一段锁定时间，在这段时间内，该伸缩组不执行伸缩活动。建议设置冷却时间大于持续时间，如 |

| | |
|---|---|
| | 持续时间设置1分钟，冷却时间设置5分钟。 |

**定时规则**

| 参数 | 说明 |
|---|---|
| 周期 | 执行定时弹性伸缩策略的周期。<br>每天：每天指定时间执行定时策略。<br>每周：每周指定天数的指定时间执行定时策略。<br>每月：每月指定日期的指定时间执行定时策略。 |
| 触发时间与实例数 | 每日弹性扩缩容的生效时间与生效后的实例数，触发时间为部署地域时区（UTC+8） |

**说明：**
- 在一条弹性伸缩规则中定时规则最多设置10条。
- 如果定时规则中多个策略产生冲突，将按序执行，您可拖动调整顺序；若触发时间早于当前时间，则从下一个时间周期开始生效。

3. 单击**提交**，完成规则创建。

## 关联部署组

创建弹性伸缩规则后，需要将规则关联到部署组上并开启规则才能生效。

1. 在 弹性伸缩 列表页面，单击目标规则操作栏的**关联部署组**。
2. 在关联部署组页面，选择应用后勾选要关联的部署组。
3. 在关联部署组页面左下方，选择是否立刻开启规则。如果选择开启，则规则会在部署组上立刻生效，否则将不生效。用户可以后续在规则详情页的**关联部署组** tab 页中修改启用状态。

## 验证弹性伸缩策略

设置弹性伸缩规则后，TSF 会自动监控部署组实例状态，当满足触发条件时，TSF 会依据弹性策略进行部署组实例扩容或缩容。您可以通过以下步骤查看弹性伸缩策略是否生效。

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏依次选择**观测中心** > **事件中心** > **事件概览**，进入事件概览页面。
3. 在顶部菜单栏选择地域并在页面上方选择要查看的应用，设置好时间范围（支持近6小时、近3天、近7天和自定义时间范围）。
4. 查看事件类型为弹性伸缩的事件及详情。若符合预期，说明弹性策略已生效。
5. （可选）您也可以在应用列表页面，查看运行实例数量。如果运行实例数与弹性策略设置的实例数一致，说明弹性策略已生效。

## 解除规则和部署组的关联

1. 在 弹性伸缩 列表页面，单击目标规则的“ID”进入详情页，单击**关联部署组**页签。
2. 在关联部署组列表右侧操作栏，单击**删除**，**确认**解除规则和部署组的关联。

## 删除规则

删除弹性伸缩规则前需要解除规则和部署组的关联，否则无法删除。
在 弹性伸缩 列表页面，单击目标规则操作栏的**删除**，**确认**后即可删除规则。

## 编辑规则

在 弹性伸缩 列表页面，单击目标规则操作栏的**编辑规则**，修改规则后单击**提交**即可完成规则的修改。

# 服务预热

最近更新时间：2025-03-10 18:08:33

当一个线上应用正在运行时，用户需要进行重新部署、扩容或者更新的时候，新注册的节点需要进行一些懒加载、连接池初始化等动作，无法接受一些大流量的请求，此时新上线的服务可能会有短时间内无法处理突增流量导致进程异常的风险。
针对应用启动阶段，TSF 提供了服务预热能力进行保护，对刚刚上线的节点仅分配较小流量，以方便新增节点完成初始化。避免在大流量场景下，由于初始化不彻底出现大量请求报错情况。

## 应用场景

当用户开启了滚动部署后，可以配置服务预热策略，在如下情况下服务预热策略会生效：

1. 新增节点：例如新部署组上线，新增节点等。
2. 更新节点：例如部署组进行更新重启，应用扩容等。

## 使用限制

仅 Spring Cloud 普通应用支持服务预热能力，对应支持的版本如下：

| 限制项 | 支持版本 |
| --- | --- |
| Spring Cloud | Spring Cloud Finchley、Spring Cloud Greenwich、Spring Cloud Hoxton、Spring Cloud 2020，且SDK 版本为1.40及之后的版本 |

## 配置服务预热

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中，单击**应用管理 > 业务应用**，选择需要配置的应用ID，进入应用详情。
3. 选择**应用部署 > 部署组**，单击**应用部署**按钮，或找到目标部署组，单击**操作**列的**更新部署**。
4. 在发布策略，选择发布方式为**滚动部署**，并设置好更新策略。
5. 在**服务注册配置&可观测**步骤，打开**是否开启服务预热**开关，在展开的预热配置中根据您的业务需求配置服务预热策略。

> **说明：**
> 仅说明注册配置治理中心为TSF-Consul下的配置，注册配置治理中心为北极星时，请参考北极星配置使用说明。




| 参数 | 描述 |
| --- | --- |
| 预热间隔 | 应用实例启动的预热时间，默认预热时长为 60 秒。服务预热时长设置范围为 0~600 秒。建议该值小于分批等待时间，避免出现多批节点同时在预热、更新状态，从而降低更新中服务的流量承载能力。 |
| 预热流量系数 | 基于已配置的预热间隔，被预热的应用流量权重会根据配置的预热流量系数呈指数型增长。在指定预热间隔内，预热流量系数越大则应用刚启动时分配的权重越小，以满足需要较长时间进行预热的复杂应用的预热需求。预热流量系数默认值为 2，表示在预热间隔周期内服务提供者的流量接收曲线形状呈 2 次曲线形状。预热流量系数设置范围为 0~5。相同预热间隔，预热流量系数越大，表示预热开始时接收的流量越小，临近预热结束时接收的流量增幅越大。 |
| 预热终止保护 | 当超过50%节点处于预热阶段，则预热终止，避免流量聚集在少量节点导致过载。预热终止开关默认打开。 |

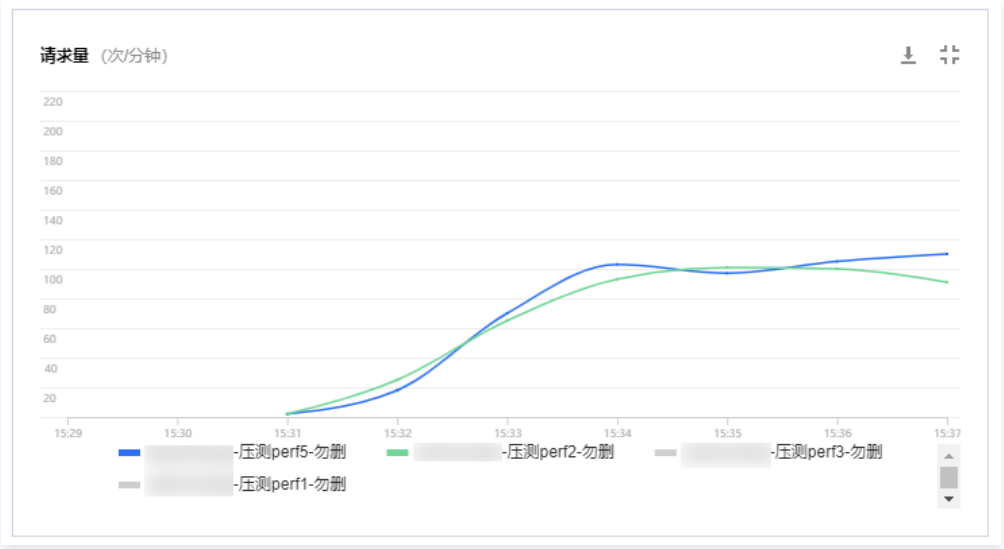
6. 完成设置后单击**提交**。

## 结果验证

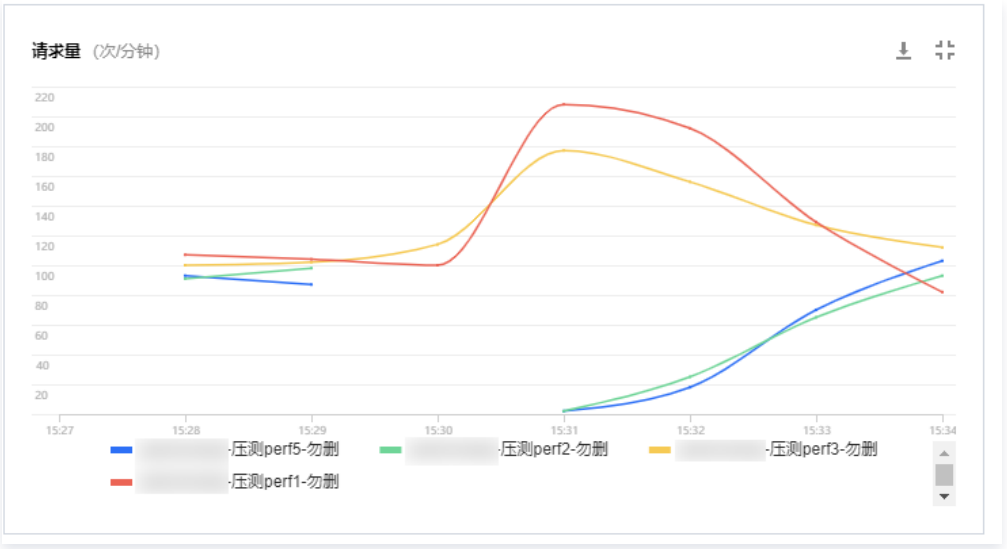
在应用部署后，可以查看预热效果。

1. 在左侧菜单栏选择 TSF-Consul，在**服务监控**页签选择所属命名空间。
2. 选择对应的服务实例后，在监控页面可以查看预热效果。
   - 服务预热效果。

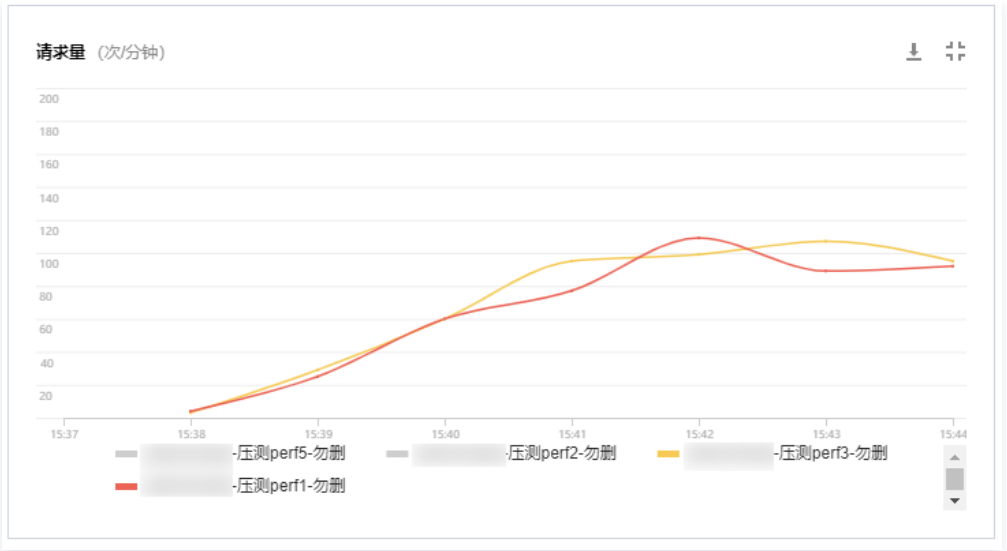
2.1.1 分批重启，第一批重启两个节点，流量系数为 2，预热间隔为 180 秒。可以看到流量缓慢增长，三分钟后达到顶峰。



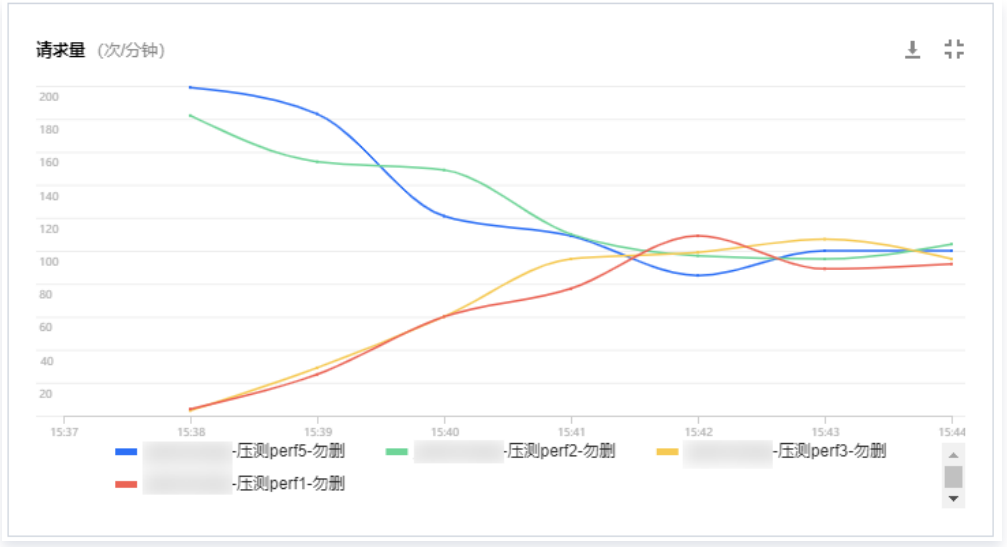

2.1.2 在第一批重启时，未重启的节点，承担了全部流量。



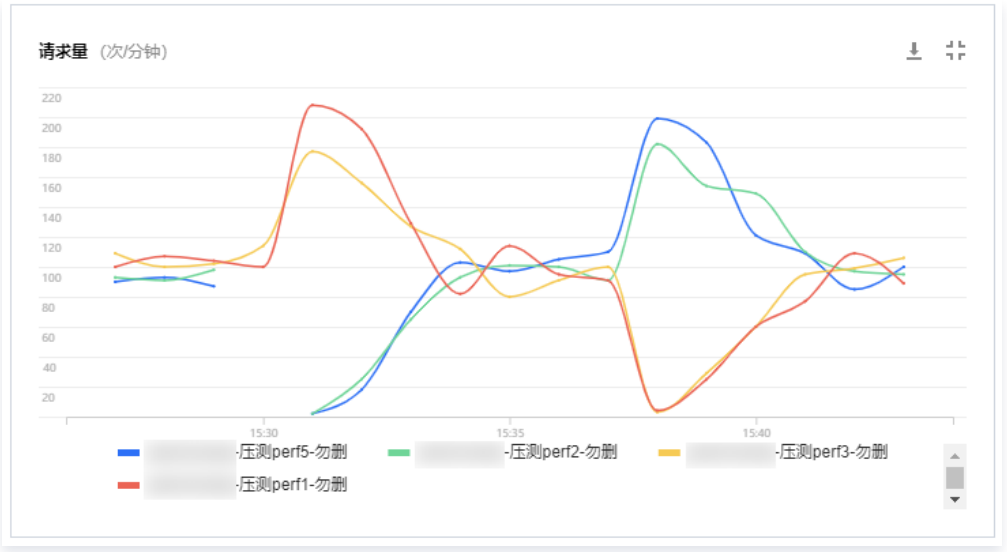

2.1.3 第二批重启时的流量接收表现。



2.1.4 第二批重启时，第一批重启的节点承担了大部分流量。

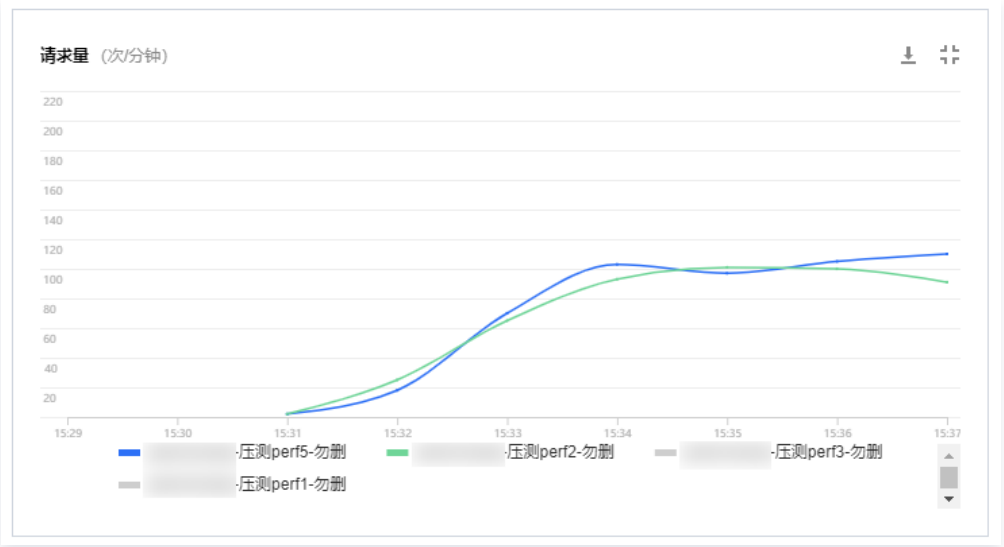



2.1.5 整体预热过程的表现

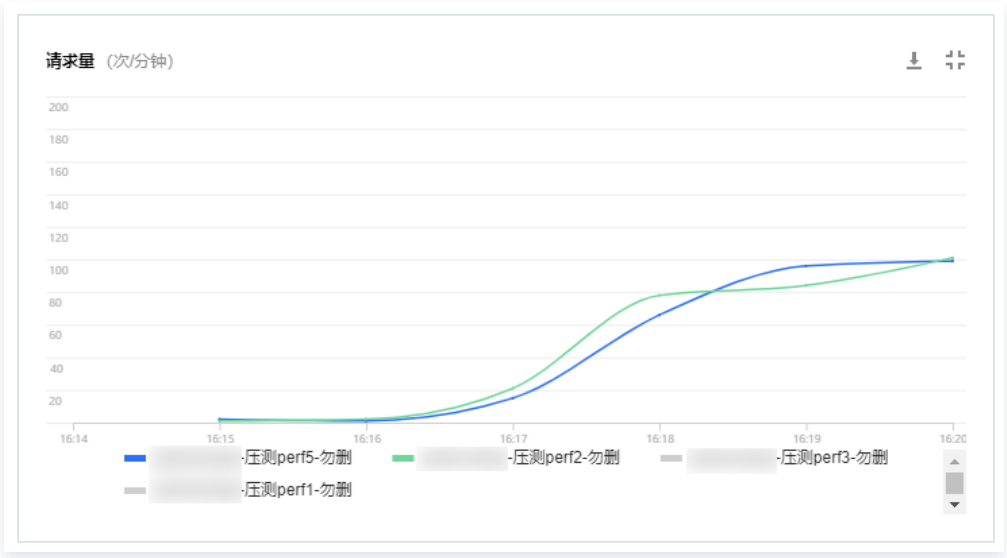



- 在稳定的业务流量下，可以看到新发布的应用所占流量比例呈预热流量系数缓慢增长。通过配置部署组的预热间隔以及预热流量系数来控制新发布的部署组流量比例，用户可以根据业务需求按需配置预热时长，并根据程序复杂度控制预热流量系数来调整应用刚启动时分配的流量权重。
- 应用部署时，配置不用的预热流量系数展示示例如下：

2.1.1 预热间隔为 180 秒，预热流量系数为 2。




2.1.2 预热间隔为 180 秒，预热流量系数为 5。



# 脚手架

最近更新时间：2025-03-10 18:08:33

## 操作场景

为提高用户开发微服务代码的效率， TSF 提供脚手架功能。脚手架可以实现快速生成基于 TSF 框架的微服务工程源码，帮助业务设计人员快速构建可运行的微服务程序。
TSF 引入了脚手架模板的概念用于保存工程信息，方便后续可以基于模板修改工程参数和下载工程源码。

## 操作步骤

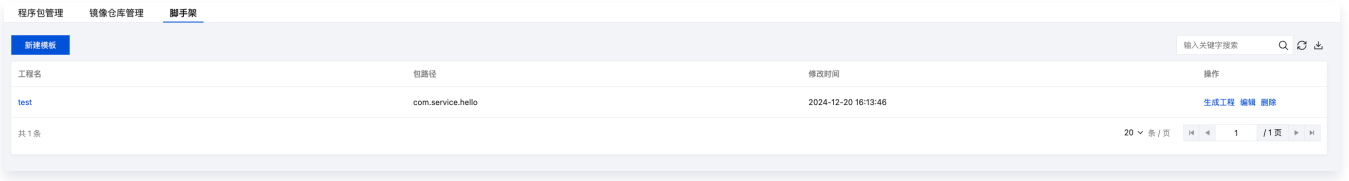
### 新建脚手架模板

1. 登录 腾讯微服务平台控制台 。
2. 单击左侧导航**制品管理 > 脚手架**。
3. 单击**新建模板**。
4. 填写工程配置和 POM 配置信息。
   - **工程名字**：Spring Boot 工程名，字母开头，支持大小写字母、数字组成，不超过24个字符长度。
   - **包路径**：package 路径，小写字母开头，支持小写字母、数字或小数点组成，不超过60个字符长度。
   - **GroupID**： pom.xml 文件中的 groupId。
   - **Artifact ID**： pom.xml 文件中的 artifactId。
   - **Name**：pom.xml 文件中的 name。
   - **Version**：pom.xml 文件中的 version。
   - **Description**：选填，pom.xml 文件中的 description。
5. 填写服务基本信息：单击**新增**可填写多个。
   - **服务名**：对应工程文件中的 `spring.application.name` 。
   - **端口**：服务监听端口，仅支持0 – 65535。
   - **Controller 类名前缀**：可填写多个，将会生成多个带有前缀的 Controller 类。
6. 单击**保存并生成工程**，生成工程并下载工程的 zip 文件。或者单击**保存**，仅保存模板，不生成工程和执行下载操作。

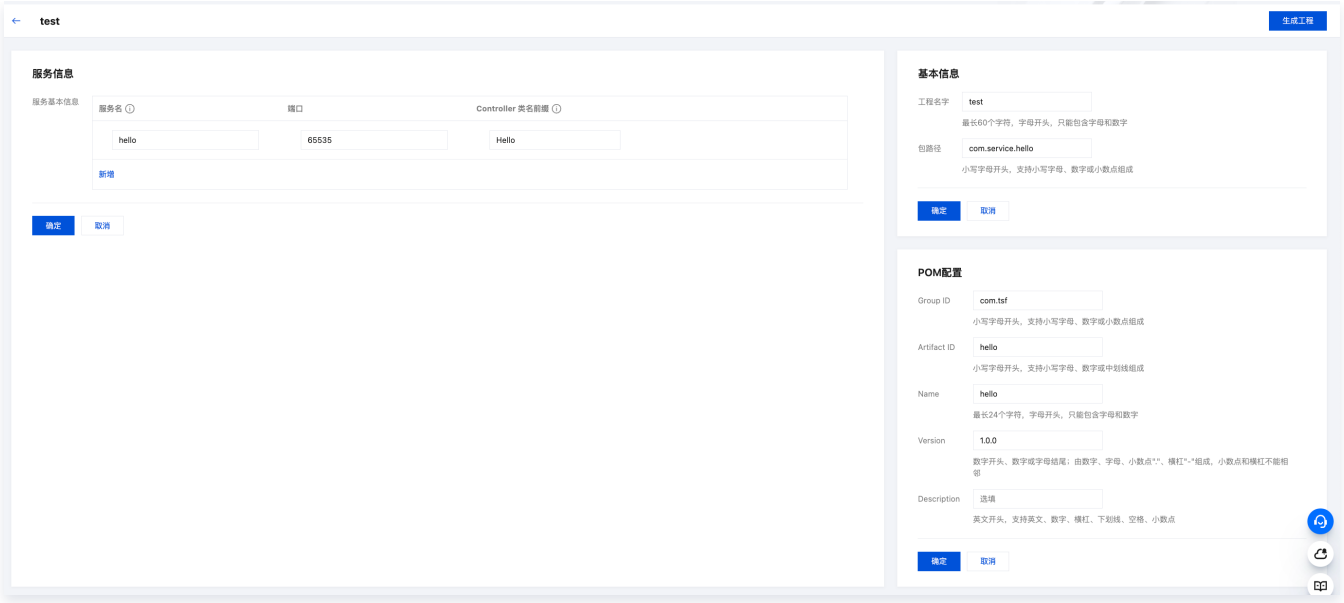
> **注意：**
> 界面将会根据 Controller 调用关系实时生成桑基图，以便用户更直观地看到服务相互间的依赖关系。

### 修改脚手架模板

1. 单击脚手架模板列表上目标模板的工程名。




2. 进入详情页可修改模板的参数信息，各参数字段含义参见 新建脚手架模板 。




## 删除脚手架模板

1. 单击脚手架模板列表上目标模板**操作**列的**删除**。
2. 在弹框中单击**确认**，删除脚手架模板。



# 业务应用管理（虚拟机）
# 虚拟机应用管理概述

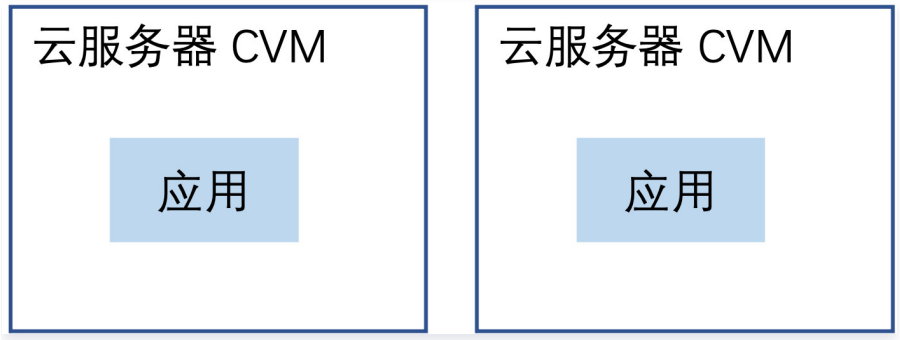
最近更新时间：2024-11-13 16:14:53

本文介绍虚拟机托管应用方式的应用场景，支持的应用类型和部署流程。

## 应用场景

- 应用托管方式：虚拟机部署方式通过程序包将应用部署在云服务器上，一台云服务器部署一个应用。
- 应用场景：适用于传统部署场景。
- 部署方式：支持 jar、war、tar.gz、zip 包部署方式，不同的部署方式需选择的应用运行环境也会有所不同。
- 应用举例：Spring Boot、Dubbo。

云服务器 CVM 和应用的关系如下。




## 支持的应用类型

TSF 支持以虚拟机的形式托管 Java 应用、Go应用以及其他多语言（如 PHP，C++ 等）应用，支持 jar 包、war 包、tar.gz/zip 包部署，不同的部署方式需选择的应用运行环境也会有所不同。

- Java：支持 SpringCloud、Dubbo 和其他框架接入。
- Go：支持 Go-GRPC 和其他框架接入。
- 多语言：支持 PHP，C++ 等应用类型。

## 部署流程

1. 创建集群
2. 将云主机导入集群
3. 创建应用
4. 上传程序包
5. 部署应用

## 常见问题

- 虚拟机集群内云主机的可用状态为何显示不可用?
- TSF 不同集群内的服务是否支持相互调用?
- TSF 如何设置开发环境和测试环境?
- CVM 集群应用的日志清理机制是什么?

版权所有：腾讯云计算（北京）有限责任公司

# 资源管理
# 集群管理

最近更新时间：2025-03-10 18:08:33

## 操作场景

集群是指云资源管理的集合，包含了运行应用的云主机等资源。TSF 中的集群分为虚拟机集群和容器集群。使用虚拟机或者容器计算资源时，用户需要提前先将云主机导入集群中，才能进行应用的部署。
本文介绍在 TSF 控制台创建虚拟机集群并对集群进行管理的操作方法。

## 创建集群

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中选择**资源管理 > 虚拟机集群**，选择好地域后，单击**新建虚拟机集群**。
3. 设置集群的基本信息。
   - 集群名：填写集群名称，最长60个字符，只能包含小写字母、数字及分隔符（ “-” ），且不能以分隔符开头或结尾。
   - 所在可用区：选择集群所在的可用区
   - 集群网络：选择与已有云服务器相同的 VPC 网络，用来保证后续导入集群的云服务器属于同一 VPC。
   - 标签：非必选。用于分类管理资源。管理方法详情请参见 标签 。
   - 备注：非必填。集群的描述，不超过200个字符。
   - 数据集：非必选。将新建的集群添加到已有数据集中。数据集使用，管理方法详情请参见 数据集管理 。
4. 单击**提交**，在弹窗中选择**导入云主机**，进入导入云主机页面。如果选择**取消**，您后续可以在集群列表中，单击操作列的**导入云主机**重新导入云主机。
5. 从集群所在 VPC 的云主机列表中，选择需要添加到集群的云主机。
6. 单击**下一步**，设置导入方式。
   - 重装系统：该方式会使 root 用户安装 agent。重装后，云主机系统盘内的所有数据将被清除，恢复到初始状态，请谨慎操作！
   - 安装 Agent（仅适用于虚拟机集群）： 该方式支持使用 root 用户和非 root 用户安装 agent。使用安装 Agent 部署时，不会重装机器。您可以单击以下页签，查看对应导入方式的操作步骤。

重装系统

> **说明：**
> CVM 竞价实例不支持重装系统。

1. 云主机配置
   - 导入方式：选择**重装系统**。
   - 操作系统：支持两种不同操作系统选择：Ubuntu 18.04版本与 CentOS 7.5版本。
   - 登录方式：提供三种对应登录方式。
     - 设置密码：请根据提示设置对应密码。
     - 立即关联密钥：密钥对是通过一种算法生成的一对参数，是一种比常规密码更安全的登录云主机的方式。详情参见 SSH 密钥 。

- 自动生成密码：自动生成的密码将通过站内信发送给您，也可登录 CVM 控制台重置密码。




2. 单击**提交**，导入的云主机将出现在云主机名列表中。等待几分钟，刷新列表，正常情况下云主机的状态将变为运行中，可用状态变为"可用"。

## 安装 Agent

1. 导入方式选择**安装 Agent**，单击**提交**，在导入结果中页面复制 agent 安装脚本。后续您也可以通过单击云主机列表操作列的**安装 Agent **获取 agent 安装脚本。

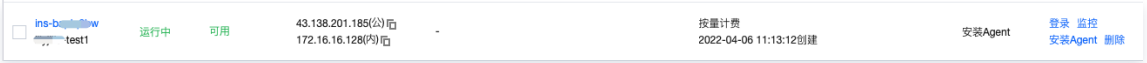



2. 单击**关闭**，返回云主机列表，此时导入的云主机可用状态显示“不可用”，选择云主机操作列的**登录**登录云主机。
   - root 用户安装 agent：直接粘贴脚本并执行。
   - 非 root 用户安装 agent：需要 root 用户新建好 `/data/tsf_apm/` 和 `/var/log/tsf/` 这两个目录，并赋予读写权限给tsf-agent的安装用户。tsf-agent 的安装用户就是部署应用的启动用户。粘贴脚本并执行。

3. 下图为脚本执行成功后终端最后几条打印日志的截图。

```
2022-04-14 15:19:17.060 [info] instance_id=ins-baplq8bw, local_ip=172.16.16.128
2022-04-14 15:19:17.062 [info] set conf for agent
2022-04-14 15:19:17.083 [info] start agent
2022-04-14 15:19:17.091 [info] add cron for agent
2022-04-14 15:19:17.098 [info] add cron for agent
2022-04-14 15:19:17.103 [info] start monit
2022-04-14 15:19:17.110 [info] add cron for monit
2022-04-14 15:19:17.116 [info] install tsf-agent success
[root@VM-16-128-centos ~]#
```

4. 脚本执行成功后，回到云主机列表，稍等片刻，云主机的可用状态会变为"可用"。




导入云主机后，您可以在虚拟机集群服务中创建并部署应用，参见 创建并部署应用 。

### 常见问题

导入云主机异常请参见 资源管理常见问题 。

## 移出云主机

> **注意：**
> 只有当节点上没有运行应用时，才可以将节点从集群中移出。停止应用需要到部署组的实例列表页面中操作，具体操作参见 应用管理 。

1. 进入 腾讯微服务平台控制台 ，选择**资源管理 > 虚拟机集群**，单击目标集群的**ID/集群名**，进入集群的云主机列表页面。
2. 在云主机节点列表页面，选择需要移出的云主机，单击**操作**列的**删除**。
3. 在弹出的提示框中，单击**确定**将云主机移出节点。

## 删除集群

> **注意：**
> - 集群内存在云主机时，无法删除集群，您需要先移除云主机。
> - 集群在删除期间，无法对外提供服务，请提前做好准备，以免造成影响。

1. 进入 腾讯微服务平台控制台 ，选择**资源管理 > 虚拟机集群**，单击目标集群操作列的**删除**。
2. 在弹出的提示框中，单击**确定**删除集群。

# 命名空间管理

最近更新时间：2025-03-10 18:08:33

命名空间（Namespace）是对 TSF 一组资源和对象的抽象集合，用于进行 TSF 资源的隔离。

## 命名空间分类

命名空间有三种类型：

| 类型 | 个数限制 | 说明 |
| --- | --- | --- |
| 系统命名空间 | 创建每个集群时会自动创建一个 | 命名规则是 `<cluster-name>_default`，不支持绑定到其他集群。 |
| 非全局命名空间 | 每个用户可以创建多个 | 不同非全局命名空间内的服务之间不能相互调用。 |
| 全局命名空间 | 每个用户在每个地域下只能创建一个 | • 非全局命名空间内的服务可以调用全局命名空间内服务。<br>• 全局命名空间内服务支持相互调用，不支持调用非全局命名空间内的服务。 |

**注意：**

- 全局命名空间特性仅适用于1.18.0版本之后的 SDK。
- 原生应用不支持全局命名空间特性。

## 与 K8s 命名空间、服务注册配置治理中心命名空间的关系

| 使用场景 | 说明 | 功能说明 |
| --- | --- | --- |
| K8s命名空间 | 容器资源空间 | 用于容器资源相互隔离 |
| 应用管理的命名空间 | TSF资源空间 | 用于TSF资源（集群、应用等）相互隔离，不同命名空间下的资源相互隔离。 |
| 注册配置治理中心的命名空间 | 服务空间 | 用于服务调用隔离，相同服务空间的服务可以互相调用和推送配置，不同服务空间的服务、配置相同隔离，不能进行服务调用和配置推送。 |

**说明：**

- 当注册配置治理中心为共享实例（TSF-Consul）时，服务空间默认创建，与 TSF 资源空间同名。
- 当注册配置治理中心为独占实例（北极星）时，服务空间与 TSF 资源空间需分别创建，独立使用，您可以在应用部署时选择该应用关联的服务空间。

## 应用场景

### 使用命名空间划分开发环境和测试环境

**场景：**用户希望部署一套开发环境和一套测试环境，两套环境支持部署同一个应用不同版本的程序包。两套环境可以共享一个集群作为底层部署资源。

**解决方案：**

1. 创建两个非全局命名空间分别为 `dev-env` 和 `test-env` 作为开发环境和测试环境。
2. 创建一个虚拟机集群，并绑定 `dev-env` 和 `test-env` 命名空间。
3. 创建应用（例如 `promotion`），并创建 2 个部署组（`dev-promotion` 和 `test-promotion`），分别选择 `dev-env` 和 `test-env` 命名空间。
4. 部署组 `dev-promotion` 部署程序包版本 v1.1，部署组 `test-promotion` 部署程序包版本 v1.0。

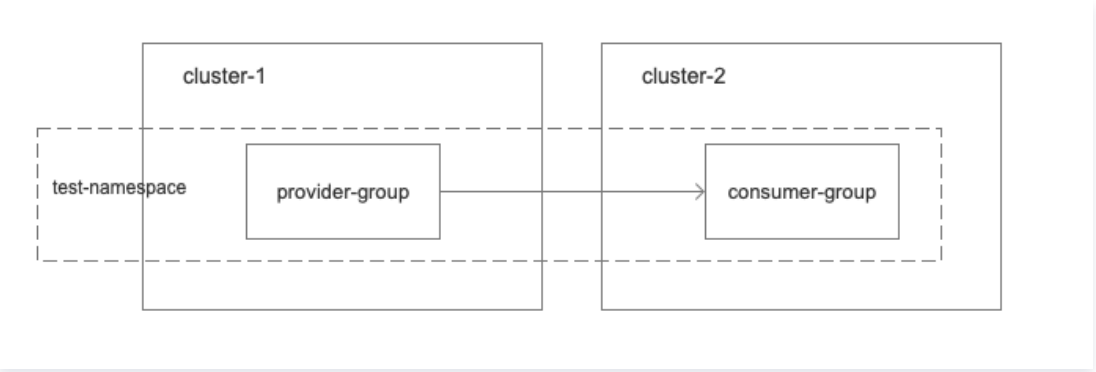
> ⚠ **注意：**
> 如果用户希望开发和测试环境的部署资源是隔离的，可以使用两个集群来划分开发和测试环境。

## 实现服务跨集群访问

在网络连通性的前提下，同一命名空间内的服务可以相互发现和相互调用。如果要实现**跨集群的服务访问**，有两个前提：

1. 集群内实例网络互通。
2. 集群关联相同的命名空间。**多个集群通过命名空间名称（而不是命名空间 ID）作为关联的 key**。用户可以在命名空间一级页面中看到关联的集群信息。

例如：有2个集群分别是 cluster-1 和 cluster-2，两个集群内实例网络互通（如在相同 VPC 内），并且都关联了命名空间 test-namespace。要实现跨集群访问，需要确保在创建部署组 provider-group 和 consumer-group 时，使用相同的命名空间 test-namespace。

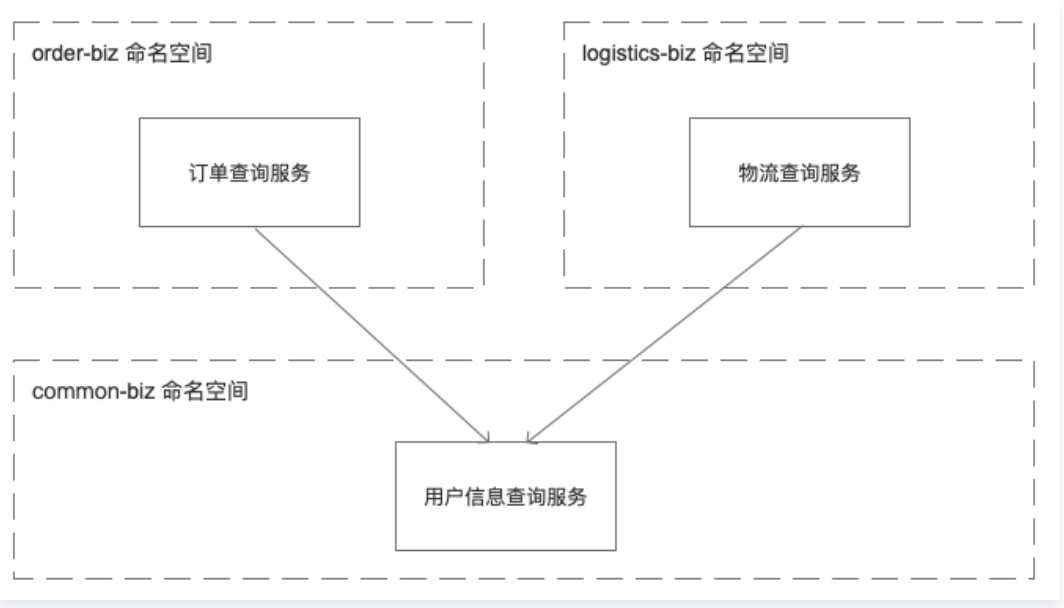



## 使用全局命名空间来部署公共服务

**场景：**在电商场景中，订单业务和物流业务都希望访问用户信息服务。此时可以使用不同命名空间来划分业务领域，将相对独立的业务（如订单和物流业务）部署在非全局命名空间，将公共服务（如用户信息服务）部署在全局命名空间中。

**解决方案：**

1. 创建两个非全局命名空间 *order-biz* 和 *logistics-biz* 作为订单和物流的业务领域。
2. 创建全局命名空间 *common-biz* 作为公共服务的业务领域。
3. 将订单业务的服务部署在 *order-biz* 命名空间内，将物流的服务部署在 *logistics-biz* 内，将用户信息服务部署在 *common-biz* 内。
4. *order-biz* 和 *logistics-biz* 命名空间内的服务可以访问 *common-biz* 内的用户信息服务。




## 创建命名空间并关联集群

## 创建命名空间

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中单击**资源管理**，选择好地域后，在**命名空间**标签页单击**新建命名空间**。
3. 填写命名空间信息。
   - 名称：不能为空。最长60个字符，只能包含字母、数字及分隔符（“-”），且不能以分隔符开头或结尾。
   - 勾选是否为全局命名空间：每个租户限制创建一个全局命名空间。
   - 标签：用于分类管理资源，可不选。详情参见 标签 。
   - 备注：集群的描述，不超过200个字符。
   - 数据集：将新建的集群添加到已有数据集中。非必填字段。数据集使用，请参见 数据集管理 。
4. 单击**确定**，完成创建。




## 集群关联命名空间

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中单击**资源管理**，选择目标集群，进入集群详情页，单击**命名空间**。
3. 在命名空间列表左上方，单击**关联命名空间**。
4. 选择要关联的命名空间，并单击确定。

> **注意：**
> 集群仅支持关联自定义命名空间。

# 删除命名空间

**前提条件**：删除命名空间前，需要先解除命名空间与集群的绑定关系（解除命名空间绑定时，需要先删除具有该命名空间属性的全部部署组）。不支持删除系统命名空间。

## 解除关联集群

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏中，单击**资源管理**。

2. 选择目标集群，进入集群详情页，单击**命名空间**标签页。
3. 在命名空间列表的**操作**列，单击**解除绑定**。

## 删除命名空间

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏中，单击**资源管理**，进入 命名空间 标签页。
2. 在命名空间列表的**操作**列，单击**删除**。
3. 单击**确认**，即可删除命名空间。

# 应用部署（虚拟机）
# 程序包管理

最近更新时间：2025-03-10 18:08:33

## 操作场景

虚拟机应用通过程序包部署在云服务器上。本文介绍通过 TSF 控制台上传程序包或删除程序包的操作步骤。

## 操作步骤

### 上传程序包

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择**应用管理**，单击目标应用的**ID/应用名**。
3. 进入应用详情页，选择**制品**标签页，单击**上传程序包**。




- 程序包类型：可选择 jar 包，war 包，tar.gz/zip 压缩包。
- 上传程序包：单击**选择文件**，选择提前准备好的压缩包。程序包格式说明参见 程序包格式说明 。
- 程序包版本：填写版本号，也可以单击**用时间戳作为版本号**。
- 备注：选填，填写备注。

4. 单击**提交**，程序包上传成功后出现在程序包列表中。

> **说明：**
> 上传程序包后，您需要创建部署组并部署应用，详情参见 虚拟机托管应用 。

### 删除程序包

1. 进入 腾讯微服务平台控制台 ，单击**应用管理**。
2. 在应用管理列表页，单击目标应用的 **ID/应用名**。

3. 在目标应用详情页，单击**制品**标签页，单击程序包列表右侧的**删除**。
4. 单击**确认**完成删除。

# 虚拟机集群部署应用

最近更新时间：2025-04-08 15:25:52

## 操作场景

在虚拟机服务集群中部署应用，您需要先在 TSF 控制台中创建好虚拟机集群和命名空间，再使用应用程序包来部署应用。本文介绍在容器服务集群中部署应用的操作方法。

## 前提条件

- 已创建好 集群 和 命名空间 。
- 已准备好应用程序包。

## 操作步骤

### 步骤1：创建应用

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏，选择**应用管理**。
3. 在应用列表，单击**新建**。设置应用基本信息。




- 应用名：填写应用名称，最长60个字符，只能包含小写字母、数字及分隔符（“-”、“_”），且不能以分隔符开头或结尾。
- 部署方式：选择**虚拟机部署**。
- **开发语言**：选择您的开发语言。

> **说明**

> 关于开发语言、开发框架和应用类型的选型原则和建议，请参见 技术栈选型 。

- **开发框架**：选择您的开发框架。
- **标签**：非必选。用于分类管理资源。详细管理方式可参见 标签 。
- **数据集**：非必选。您可以通过数据集管理配置不同的子账号和协作者使用不同资源的权限，详细管理方式可参见 数据集管理 。
- **备注**：选填，可留空。

4. 单击**下一步**，进入注册配置治理配置。




- **注册配置治理**：选择是否开启。关闭后，默认应用不注册到注册中心。
- **实例类别**：选择**独占实例（北极星）**或**共享实例（TSF-Consul）**。
- **接入方式**：

**共享实例（TSF-Consul）**

选择应用接入 TSF-Consul 的方式：

- SDK应用/框架接入：基于 SDK/Spring Cloud 等开发框架微服务模式，提供注册注册中心、配置中心等组件。
- Mesh接入：基于 Sidecar 的 Service Mesh 微服务模式
- 无侵入接入：使用了原生框架的应用0代码改造即可接入，支持服务发现、服务治理、应用性能监控能力。

**独享实例（北极星）**

- 注册中心：选择用于注册中心的北极星实例，关联后，应用管理将下发环境变量。
- 配置中心：选择用于配置中心的北极星实例，关联后，应用管理将下发环境变量。
- 治理中心：选择用于治理中心的北极星实例，关联后，应用管理将下发环境变量。

5. 单击**完成**，在弹窗中选择**导入程序包**，跳转至上传程序包页面。如果选择**关闭**，您后续可以在应用详情页面选择**程序包管理**页签，重新上传程序包。

## 步骤2：上传程序包

1. 进入 腾讯微服务平台控制台 ，在左侧导航栏选择**应用管理**，单击刚刚创建好的应用 ID/名称。
2. 在应用详情页，选择**制品**标签页，单击**上传程序包**。




- 程序包类型：可选择jar包，war包，tar.gz/zip压缩包。
- 上传程序包：单击**选择文件**，选择提前准备好的压缩包。程序包格式说明参见 程序包格式说明 。
- 程序包版本：填写版本号，您也可以单击**用时间戳作为版本号**。
- 备注：选填，填写备注。

3. 单击**提交**，程序包上传成功后出现在程序包列表中。

## 步骤3：部署应用

1. 进入 腾讯微服务平台控制台 ，在左侧导航栏选择**应用管理**，单击刚刚创建好的应用 ID。
2. 在**应用部署**组页面，单击**应用部署**，填写部署组信息。
   - 集群：选择应用要部署所在的集群。
   - 命名空间：选择应用要部署所在的命名空间。
   - 部署组名称：填写部署组名称，不超过60个字符，只能包含字母、数字及分隔符(“-”)，且必须以字母开头，数字或字母结尾。
   - 标签：选填，用于从不同维度对资源进行分类管理，详细管理方式可参见 标签 。
   - 备注：选填，可以留空。
3. 单击**下一步**，进入部署配置。
   - 选择实例：从关联集群的可用云主机列表勾选用于部署的云主机。
   - 软件仓库：选择软件仓库类型。
   - JDK版本：选择目标程序包版本。

| 程序包类型 | 说明 |
| --- | --- |
| jar | 启动命令固定为 java -jar；支持 JDK 版本 KNOA JDK8，KNOA JDK11，KNOA JDK11，OPEN JDK8 和 OPEN JDK17；支持用户设置启动参数 |
| war | 启动命令固定为 java -jar，支持用户设置启动参数 |
| zip/tar.gz | 程序包类型为 zip/tar.gz 时，支持 JDK 版本 KNOA JDK8，KNOA JDK11，KNOA JDK11，OPEN JDK8 和 OPEN JDK17；支持 **启停脚本配置**。<br>• 使用本地 start.sh 和 stop.sh：默认方式<br>• 控制台配置：需要填写启动脚本和停止脚本（推荐）<br>当部署组内的实例 agent 版本不支持 **控制台配置** 启停脚本时，不能选中该选项，请升级 agent 到最新版本。详细参见 应用管理相关 。 |

**说明：**

> 当用户切换运行环境，在部署组动态更新 JDK 运行版本，且重启部署组后生效。

4. （可选）设置启动参数JAVA_OPT。

> **说明**
> - 如果部署组关联的接入方式为TSF Mesh接入，则无须设置启动参数。

5. （可选）选择健康检查方式。关于健康检查的作用和配置详情参见 健康检查 。

> **注意**
> - 当部署组内的实例 agent 版本不支持存活检查和就绪检查时，不能开启检查功能，请升级 agent 到最新版本。参见 应用管理相关 。
> - 当部署组内的多个实例 agent 版本不一致时，不能开启检查功能。只有当所有实例的 agent 的版本支持存活检查和就绪检查才开启该特性。

| 健康检查方式 | 说明 |
| --- | --- |
| 存活检查 | 检查应用是否正常，不正常则重启实例。 |
| 就绪健康 | 检查应用是否就绪，不就绪会影响滚动更新。 |

**存活检查的检查方式：**

| 检查方式 | 说明 |
| --- | --- |
| HTTP 请求检查 | 任何大于200小于400的返回码都会认定是成功的返回码。其他返回码都会被认为是失败的返回码。HTTP检查需要设置端口和请求路径。 |
| TCP 端口检查 | 如果可以建立连接被认为是成功的。该检查方式需要设置检查端口。 |
| 执行命令检查 | 如果命令执行成功并且返回值为 0，认为是成功；其他返回值认为是失败。该检查方式需要填写执行命令。 |

**高级参数：**

| 参数 | 说明 | 参数范围 |
| --- | --- | --- |
| 启动延时 | 延时启动健康检查的时间。 | 最小值为0，默认值为10。 |
| 超时时间 | 每次健康检查响应的最大超时时间。 | 最小值为1，默认值为2。 |
| 检查周期 | 进行健康检查的时间间隔。 | 最小值为1， 默认值为10。 |
| 健康阈值 | 表示从失败到成功的连续健康检查成功次数。 | 存活检查不支持编辑，只能为1就绪检查支持编辑：最小值为1，默认值为1。 |
| 不健康阈值 | 表示从成功到失败的连续健康检查失败次数。 | 最小值为1，默认值为3。 |

6. 单击**下一步**，进入发布策略，选择发布方式。

| 更新方式 | 说明 |
| --- | --- |
| 立即部署 | 会先停止所有运行实例，然后使用新的程序包版本部署，会造成发布期间短暂停服。 |
| 分批部署 | 当部署组内有多个实例时，可以选择滚动更新方式进行分批发布。滚动更新支持设置发布策略：<br>• beta 批次：是否首次用一个实例来部署新版本，如果部署成功，才会部署后面的批次实例。 |

| | |
|---|---|
| | • 批次 N 实例占比：除了 beta 批次，可以将部署组内剩余实例按百分比划分为多个批次部署，只有当批次 N 部署成功后才会继续部署批次 N+1。<br>• 分批执行方式：批次之间可以选择自动或手动方式来开启下一批次的部署。<br>• 分批间健康检查：开启后，单批次中全部实例通过就绪检查后，再执行下一批次。<br>• 分批等待时间：如果分批执行方式选择自动时，等待分批等待时间后会自动部署下一批次。 |

7. 单击**下一步**，进入服务注册配置和可观测配置。
   - 注册配置治理：部署组默认使用应用对应的注册配置治理选项，确认注册配置治理信息是否正确。
   - 强制启动（可选）：开启强制启动则实例忽视 consul 服务注册报错信息正常启动。
   - 日志配置项：日志配置项将设定业务日志的采集路径和解析规则。
   - 日志投递：日志投递规则将设定日志的转储方式，将规则指定路径中的日志内容投递到指定的接收端。
   - 可观测 Agent：如您使用 Spring Cloud 2020，请勾选本选项，用于开启调用依赖等高级监控能力。相关开发文档，请参见 使用 Agent 进行无侵入应用开发 。
8. 单击**下一步**，进行信息确认。
9. 确认后，单击**发布**，进行应用发布。
10. 应用部署成功后，部署组状态变为运行中，**运行实例数/总实例数** 数值发生变化。

## 常见问题

- TSF 是否支持在同一台服务器上安装多个应用?
- 如何查看实例的 Agent 版本?
- 如何将实例 Agent 升级至最新版本?
- TSF 应用实例状态为什么显示 Agent 异常?
- 重启服务器后 TSF Agent 是否能自动重启?
- 如何排查应用是否部署失败?
- 程序包无法上传如何解决?
- 创建应用时，提示命名空间数达到配额如何解决?
- 程序包容量达到上限时如何解决?
- 将一台已在 CVM 创建的实例添加到 TSF 的时候，是否需要重装 Agent?

# 健康检查

最近更新时间：2025-03-10 18:08:33

## 概述

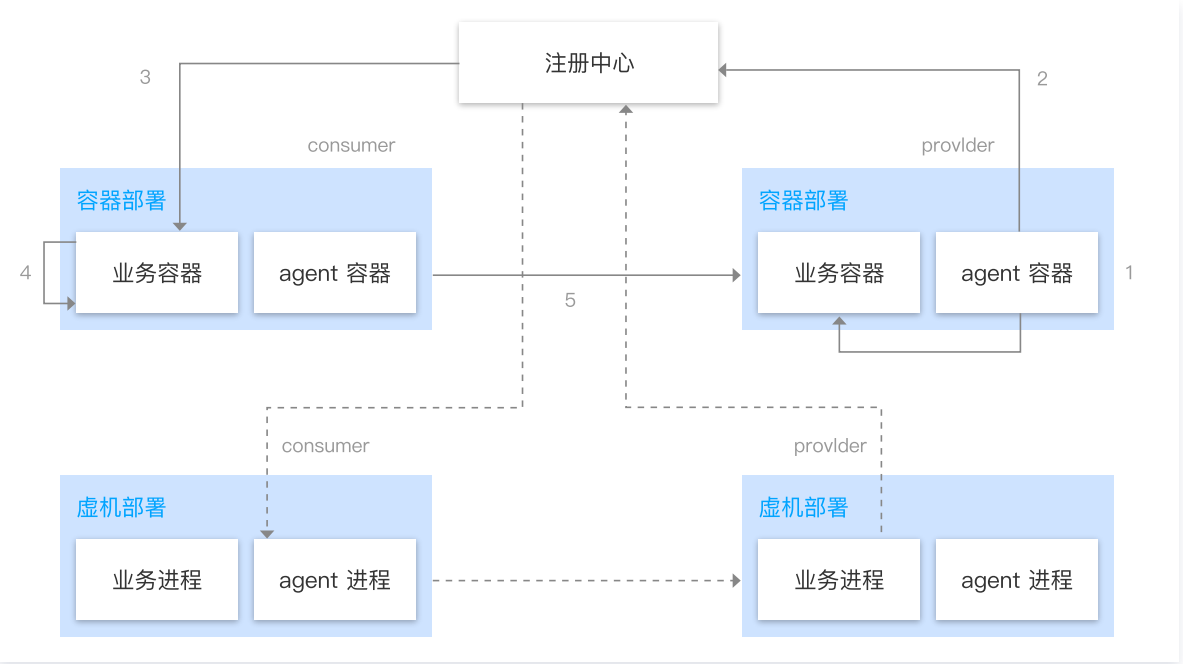
TSF 健康检查分为存活检查和就绪检查：

- **存活检查：**主要作用是确定进程存活状态，判断是否需要进行实例重启。例如存活检查可以捕捉到死锁（应用程序进程还存在，但是无法响应），重启进程或者容器可以让应用程序恢复可用。
- **就绪检查：**主要作用是确定服务实例能否支持对外服务，将健康检查结果与注册中心状态联动实现滚动更新及无损发布。当一个实例没有就绪，实例会在注册中心中被屏蔽（其他服务不会发送请求到微服务实例上）。

## 健康检查支持的产品能力

- 存活检查。
- 就绪检查。
- 无损发布。
- 检查顺序：HTTP 检查、TCP 端口、执行命令检查。
- 支持虚拟机部署组和容器部署组。

## 健康检查与注册中心联动流程




1. 就绪检查，检查实例状态是否 ready。
2. 如果就绪检查 ready 则更新实例注册状态为 passing，反之则检查状态为 Critical。
3. 监听注册中心服务提供方实例状态变更。
4. 存在状态变更更新缓存及本地文件。
5. 发起服务调用。

## 配置存活检查

1. 登录 腾讯微服务平台控制台，选择**应用管理**，单击对应的应用 ID，进入应用详情，在左侧菜单栏中，单击**应用部署**。
2. 在部署配置，在健康检查中开启存活检查。




3. 检查方式：
   - HTTP 请求检查：任何大于200小于400的返回码都会认定是成功的返回码。其他返回码都会被认为是失败的返回码。HTTP 检查需要设置端口和请求路径。
   - TCP 端口检查：如果可以建立连接被认为是成功的。该检查方式需要设置检查端口。
   - 执行命令检查：如果命令执行成功并且返回值为 0，认为是成功；其他返回值认为是失败。该检查方式需要填写执行命令。
4. 启动延时、超时时间、检测周期、健康阈值、不健康阈值使用默认值即可。

## 配置就绪检查

**场景：**服务 A 调用服务 B。服务 B 使用 `/health` 接口是否返回 `200` 状态码判断是否健康，当就绪检查失败时，服务 B 在注册中心中被屏蔽，服务 A 通过注册中心监听到服务 B 状态变化后更新本地路由表；当就绪检查成功时，服务 B 在注册中心恢复健康状态。

1. 登录 腾讯微服务平台控制台，选择**应用管理**，单击对应的应用 ID，进入应用详情，在左侧菜单栏中，单击**应用部署**。
2. 在部署配置，在健康检查中开启就绪检查。




   - **优雅滚动重启：**无须用户设置检查规则，会根据服务实例是否注册到注册中心来决定 虚拟机 变为 ready 状态。
   - **自定义：**自定义就绪检查通过的方式。
3. 检查方式：
   - HTTP 请求检查：任何大于200小于400的返回码都会认定是成功的返回码。其他返回码都会被认为是失败的返回码。HTTP 检查需要设置端口和请求路径。
   - TCP 端口检查：如果可以建立连接被认为是成功的。该检查方式需要设置检查端口。
   - 执行命令检查：如果命令执行成功并且返回值为 0，认为是成功；其他返回值认为是失败。该检查方式需要填写执行命令。
4. 启动延时、超时时间、检测周期、健康阈值、不健康阈值使用默认值即可。

## 就绪检查和滚动更新

部署组支持立即更新和滚动更新两种更新方式：立即更新会先停止**所有**应用实例，然后使用新的程序包或者镜像版本部署；滚动更新会根据更新策略，分批更新部署组内的实例。
如果部署组使用滚动更新并且开启就绪检查，当就绪检查失败时滚动更新会被阻塞。

# 应用升级与回滚
# 应用升级与回滚概述

最近更新时间：2025-03-10 18:08:33

## 概述

应用部署到 TSF 后，还会不断进行升级迭代，TSF 单应用发布支持单批发布（直接升级）、多部署组批次发布、蓝绿发布、全链路灰度发布等不同的发布方式，用户可以根据业务场景需求不同进行选择。当发现升级后的应用出现异常，可以将应用回滚到历史版本，待问题修复后再进行升级发布。
本文介绍 TSF 的发布类型，适用场景和发布回滚功能。

> **说明：**
> 发布计划功能仅专业版、铂金版支持。

## 升级应用的发布方式及适用场景

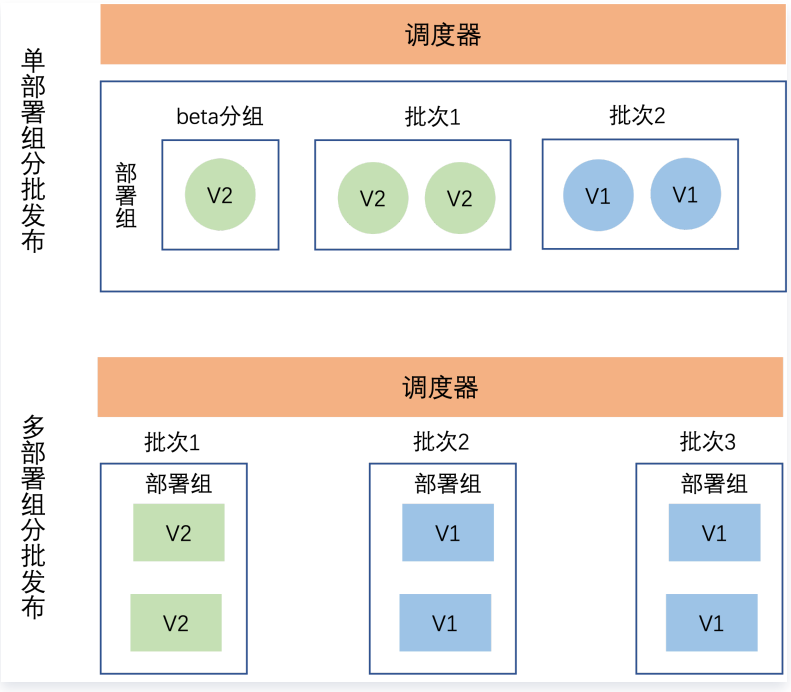
### 单批发布（直接升级）

- 适用场景：应用规模较小或服务架构比较简单，当完成开发测试后，可直接进行升级以便验证。
- 升级方式：应用部署发布策略。

### 多批次发布

- 适用场景：应用规模较大或服务架构比较复杂，可以按批次进行应用部署，分批发布过程中如果出现故障，可以及时终止并回滚，待问题修复后重新发布，降低因版本发布导致业务中断的风险。

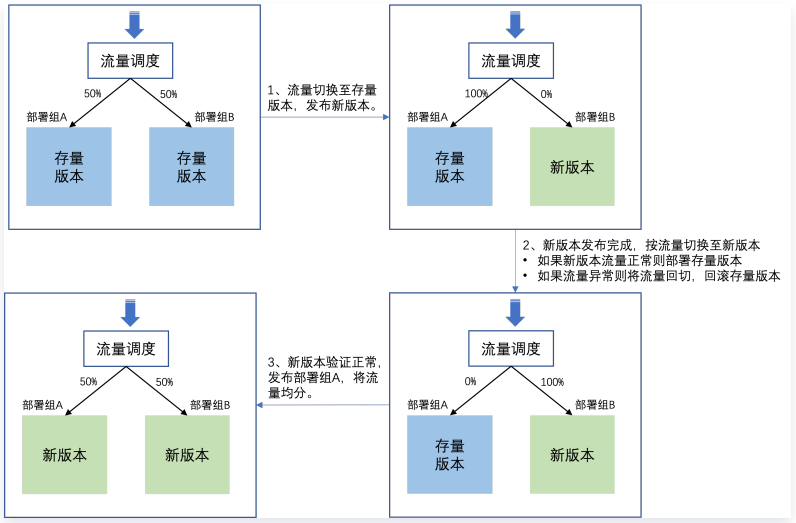
> **说明：**
> 下图所示，V1（蓝色）代表存量版本，V2（绿色）代表更新版本。




- 升级方式：发布计划。
- 相关文档：分批发布

### 蓝绿发布

- 适用场景：蓝绿发布的含义是分别蓝、绿两个分组部署服务不同版本，蓝组为存量版本，绿组为更新版本。发布步骤如下：
  - 步骤1：发布前先将流量切换至蓝组（存量版本）。
  - 步骤2：发布绿组，发布成功后将流量切换至绿组。
    - 如果流量异常则即时切换至蓝组，避免业务异常。
    - 如果流量正常则更新蓝组版本或下线蓝组版本。




- 升级方式：发布计划。
- 相关文档：蓝绿发布

### 全链路灰度发布

- 适用场景：当有新功能开发完成准备上线时，可以通过全链路灰度发布先选择一小部分用户流量导入到包含新功能的新版本应用及相关联的全链路服务中进行小规模验证，验证后，再将应用全量升级到新版本。
- 发布方式：结合注册配置治理中心。
- 相关文档：
  - 概述
  - 泳道配置
  - 灰度发布

## 发布回滚

若在发布过程中或者发布完成后发现新版本应用异常，可以使用回滚功能回滚应用，待问题解决后再次发布新版本。

- 发布中回滚：对于正在运行中的发布单，若发现异常，需要先终止发布再进行回滚，终止发布未执行的任务节点不再进行发布操作。
- 发布后回滚：对于已发布的发布单，若发现异常，直接回滚历史版本即可。

发布回滚的具体操作，请参见 发布回滚 。

# 分批发布

最近更新时间：2025-03-10 18:08:33

## 操作场景

应用规模较大或者服务架构比较复杂时，可以进行分批次发布，降低因版本发布导致业务中断的风险。
本文介绍如何在 TSF 控制台分批次发布应用。

## 操作步骤

### 步骤1：新建发布单 – 基本信息

1. 登录 腾讯微服务平台控制台，选择**应用管理**，单击应用 ID，进入应用详情。
2. 在左侧菜单栏选择**应用部署**，选择**发布计划**，单击**新建发布单**。
3. 填写发布单基本信息。

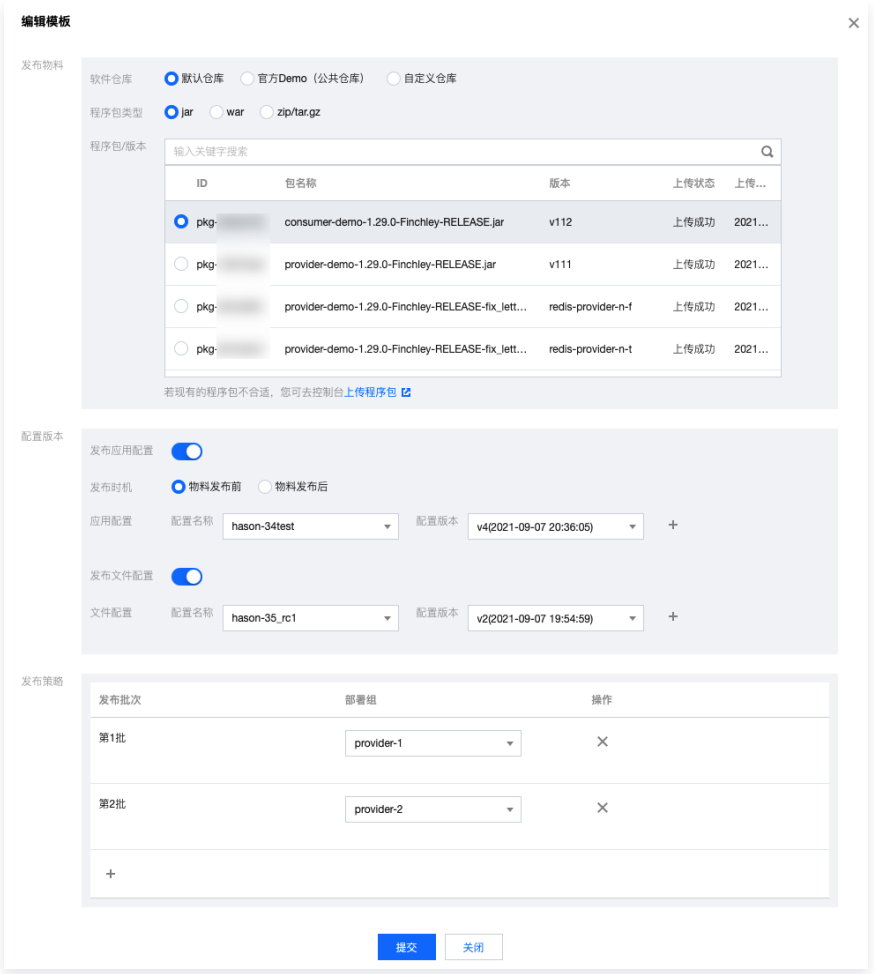



- 名称：填写发布单名称，最长60个字符，支持英文大小写数字、横杠（-）、下划线(_)、小数点（.）。
- 触发类型：
  - 手动触发：需要手动触发发布单。
  - 定时触发：按照定时时间自动触发发布计划。
- 发布类型：单应用发布。
- 数据集：选择**数据集**，关于数据集请参见 数据集管理 。
- 启用状态：启用后，发布单才能生效。

4. 单击**下一步**。

### 步骤2：新建发布单 – 发布类型

1. 在发布类型页面，选择应用后，发布类型选择**分批发布**。
2. 单击**创建模板**，在弹窗中填写发布模板信息。

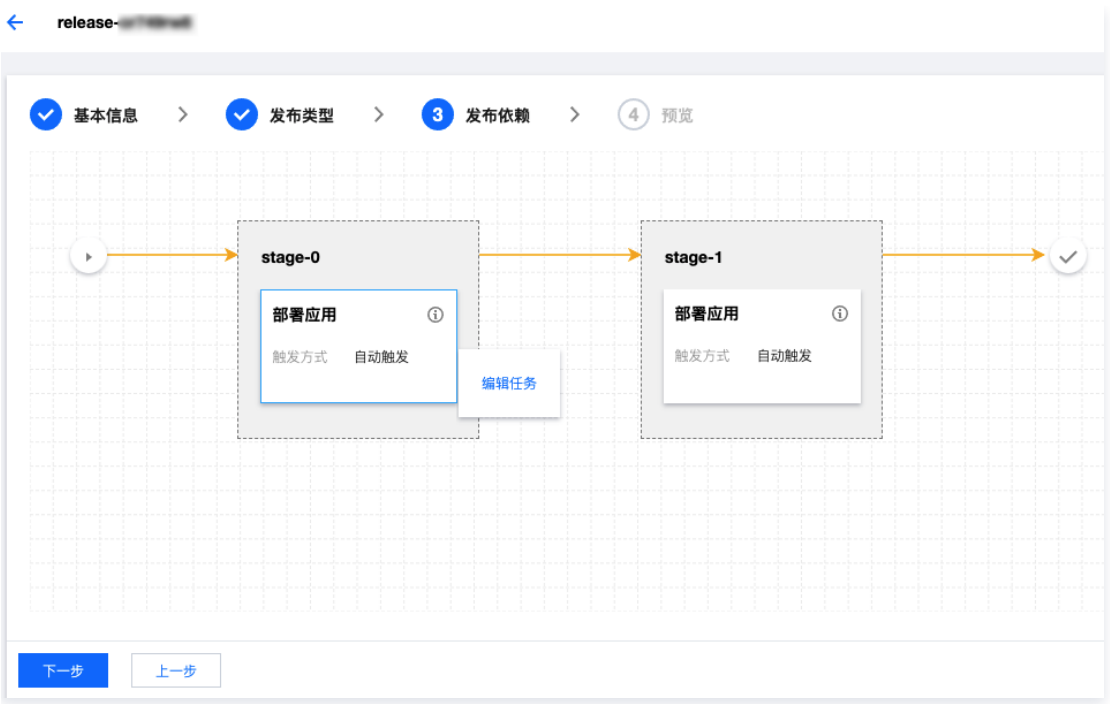



- 发布物料：选择要发布的程序包或者镜像（基于应用部署方式选择，例如应用部署时选择了 JAR 包部署，则只能选择 JAR 包）
- 配置版本：选择所要发布的应用配置和文件配置。关于配置详情参见 应用配置 和 文件配置 。
- 发布策略：在列表中选择分批发布的目标部署组。

3. 单击**提交**，完成模板配置，返回发布类型页面。
4. 单击**下一步**。

## 步骤3：新建发布单 – 发布依赖

在发布依赖页面，会按照模板生成发布依赖拓扑，支持针对部署应用阶段进一步确定发布参数配置，如果不选即为默认或原有部署组的发布参数配置。




1. 单击**部署应用**卡片，选择**编辑任务**，在弹窗中设置发布参数配置。




- 触发方式：支持自动触发和手动触发。
  - 自动触发：按照**任务时长**自动发布剩余批次的发布。
  - 手动触发：需要手动触发下一个批次的发布。

> **说明：**
> 虚机部署场景下批量更新不允许设置为手动触发模式。

- 任务时长：设置任务执行超时时间，设置手动触发但超过24小时未执行则判定为超时。

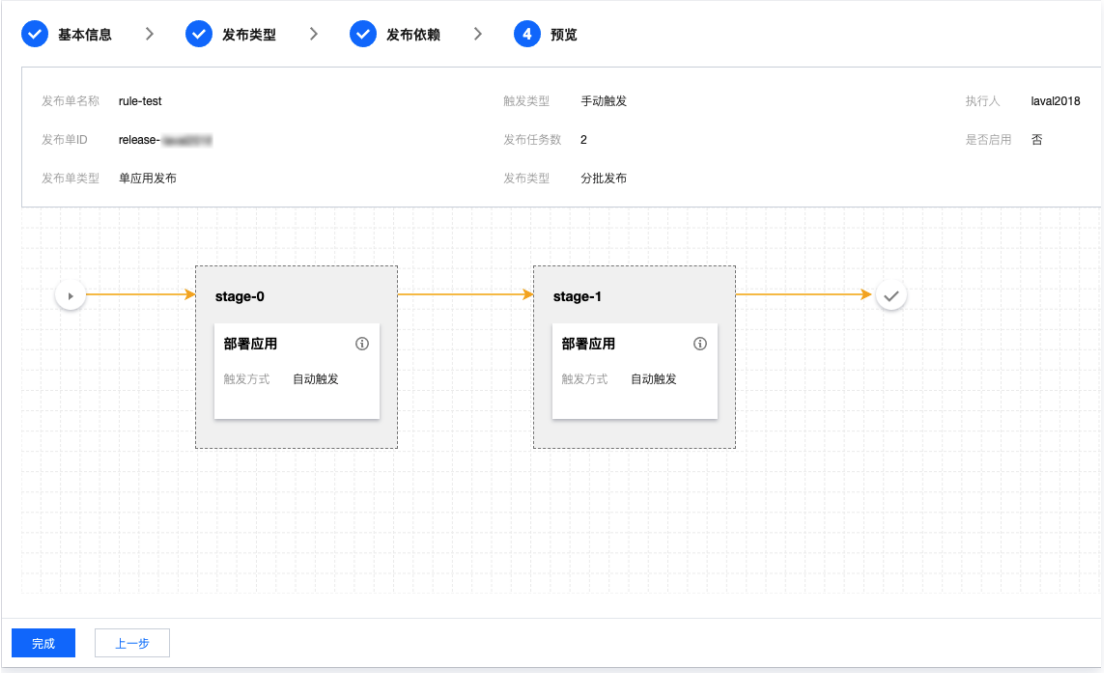
版权所有：腾讯云计算（北京）有限责任公司

- 发布物料：由模板确定不可编辑。
- 配置版本：由模板确定不可编辑。
- 发布参数：
  - 启动参数：填写应用启动参数。
  - 更新方式：
    - 立即更新：会先停止所有运行实例，然后使用新的程序包版本部署，会造成发布期间短暂停服。
    - 滚动更新：当部署组内有多个实例时，可以选择滚动更新方式进行分批发布。
  - 健康检查：详细参见 健康检查 。

2. 单击**提交**，完成模板配置，返回发布依赖页面。
3. 单击**下一步**。

## 步骤4：预览

在预览页面，会展示发布单的基本信息和发布任务的依赖拓扑图。




确认无误后单击**完成**即完成发布单创建，在发布计划列表中可看到刚刚创建的发布单。

## 步骤5：执行发布单

> **说明：**
> - 若发布单处于**未初始化**状态时，发布单无法执行，需要编辑完成发布单信息才能被初始化。
> - 发布单必须被启用后才能执行，若未启用，在发布单列表的启用状态栏单击**启用**按钮即可。

针对发布单的触发类型不同，发布单的执行方式也有所不同。

- 自动触发：对于自动触发的发布单，无需手动操作，满足条件后发布单会自动执行。
- 手动触发：对于手动触发的发布单，需要单击发布单操作列的**运行**手动运行发布单。




## 结果验证

### 查看变更记录

1. 单击发布计划的 ID，进入发布计划基本信息页面。
2. 在页面上方选择**发布记录**页签，通过发布记录查看此次批次发布的结果是否为**成功**。
   - 如果为**成功**，说明批次发布成功。
   - 如果为**失败**，说明批次发布失败。




### 查看应用详情

单击应用“ID”，进入部署组列表页面，查看目标部署组的状态是否为**正常**，已部署的程序包是否为需要发布的最新版本。

# 蓝绿发布

最近更新时间：2025-03-10 18:08:33

## 操作场景

在蓝绿发布过程中，有两套生产环境。蓝色是当前版本并拥有实时流量，绿色是包含更新代码的环境。无论任何时候，只有一套环境有实时流量。蓝绿发布的整体流程如下：

1. 发布前先将流量切换至蓝组（存量版本）。
2. 发布绿组，发布成功后将流量切换至绿组。
   - 如果流量异常则及时切换至蓝组，避免业务异常。
   - 如果流量正常则更新蓝组版本或下线蓝组版本。

本文介绍如何在通过 TSF 控制台上使用蓝绿发布方式发布应用。

## 操作步骤

### 步骤1：新建发布单 – 基本信息

1. 登录 腾讯微服务平台控制台，进入**应用管理**，单击应用 ID，进入应用详情。
2. 在左侧菜单栏选择**应用部署**，选择**发布计划**，单击**新建发布单**。
3. 填写发布单基本信息。




   - 名称：填写发布单名称，最长60个字符，支持英文大小写数字、横杠（-）、下划线（_）、小数点（.）。
   - 触发类型：
     - 手动触发：需要手动触发发布单。
     - 定时触发：按照定时时间自动触发发布计划。
   - 发布类型：单应用发布。
   - 数据集：选择**数据集**，关于数据集请参见 数据集管理 。
   - 启用状态：启用后，发布单才能生效。
4. 单击**下一步**。

### 步骤2：新建发布单 – 发布类型

1. 在发布类型页面，选择应用后，发布类型选择**蓝绿发布**。
2. 单击**创建模板**，在弹窗中填写发布模板信息。

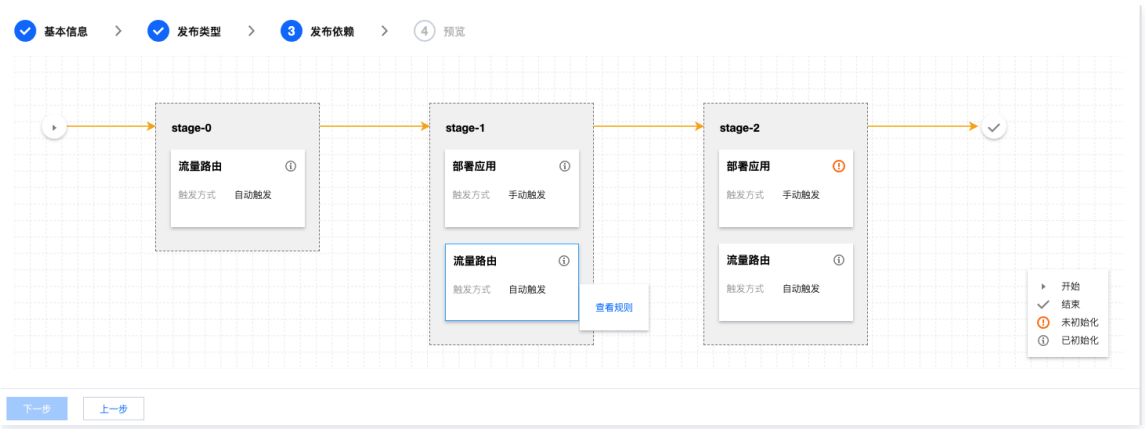



- 发布物料：选择要发布的程序包或者镜像（基于应用部署方式选择，例如应用部署时选择了 JAR 包部署，则只能选择 JAR 包）。
- 配置版本：选择所要发布的应用配置和文件配置。关于配置详情参见 应用配置 和 文件配置 。
- 发布策略：在列表中选择蓝组和绿组的目标部署组。

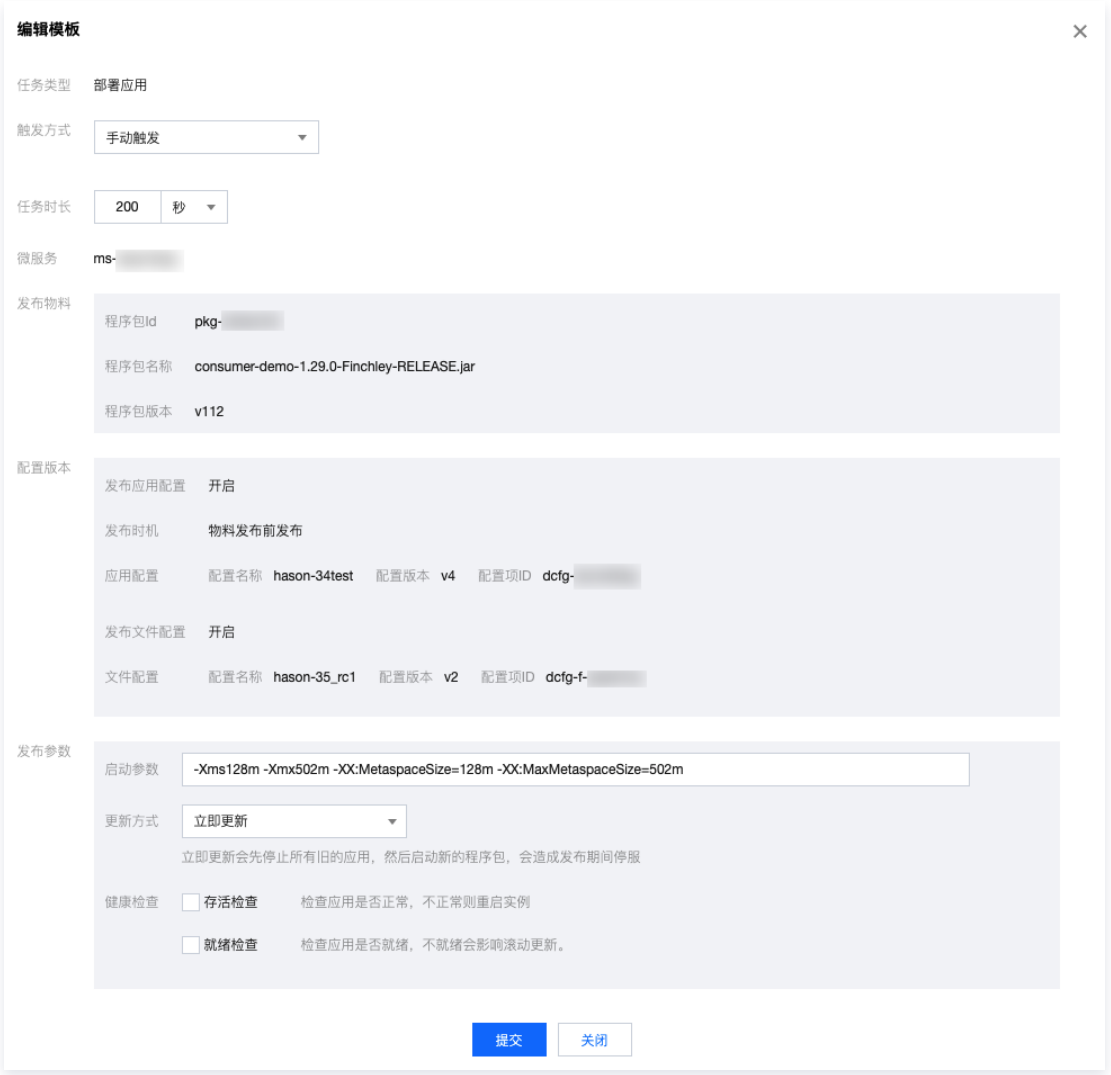
3. 单击**提交**，完成模板配置，返回发布类型页面。
4. 单击**下一步**。

## 步骤3：新建发布单 – 发布依赖

在发布依赖页面，会按照模板生成发布依赖拓扑，支持针对部署应用阶段进一步确定发布参数配置，如果不选即为默认或原有部署组的发布参数配置。




1. 单击**部署应用**卡片，选择**编辑任务**，在弹窗中设置发布参数配置。




- 触发方式：支持自动触发和手动触发。
  - 自动触发：按照任务流程自动发布剩余批次的发布。
  - 手动触发：需要手动触发下一个批次的发布。

> **说明：**
> 虚机部署场景下批量更新不允许设置为手动触发模式。

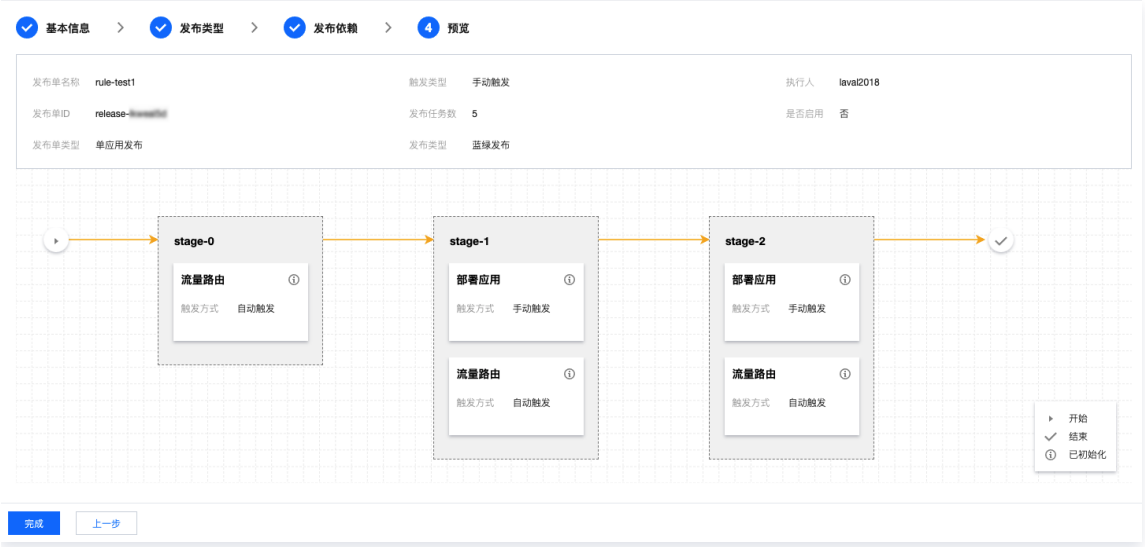
- 任务时长：设置任务执行超时时间，设置手动触发但超过24小时未执行则判定为超时。

- 发布物料：由模板确定不可编辑。
- 配置版本：由模板确定不可编辑。
- 发布参数：
  - 启动参数：填写应用启动参数。
  - 更新方式：
    - 立即更新：会先停止所有运行实例，然后使用新的程序包版本部署，会造成发布期间短暂停服。
    - 滚动更新：当部署组内有多个实例时，可以选择滚动更新方式进行分批发布。
  - 健康检查：参见 健康检查 。

2. 单击**提交**，完成模板配置，返回发布依赖页面。
3. 单击**下一步**。

## 步骤4：预览

在预览页面，会展示发布单的基本信息和发布任务的依赖拓扑图。




确认无误后单击**完成**即完成发布单创建，在发布计划列表中可看到刚刚创建的发布单。

## 步骤5：执行发布单

> **说明：**
> - 若发布单处于**未初始化**状态时，发布单无法执行，需要编辑完成发布单信息才能被初始化。
> - 发布单必须被启用后才能执行，若未启用，在发布单列表的启用状态栏单击**启用**即可。

针对发布单的触发类型不同，发布单的执行方式也有所不同。

- 自动触发：对于自动触发的发布单，无需手动操作，满足条件后发布单会自动执行。
- 手动触发：对于手动触发的发布单，需要单击发布单操作列的**运行**手动运行发布单。




# 结果验证

## 查看变更记录

1. 单击发布单的 ID，进入发布计划基本信息页面。
2. 在页面上方选择**发布记录**页签，通过发布记录查看此蓝绿发布的结果是否为**成功**。

- 如果为**成功**，说明蓝绿发布成功。
- 如果为**失败**，说明蓝绿发布失败。




### 查看应用详情

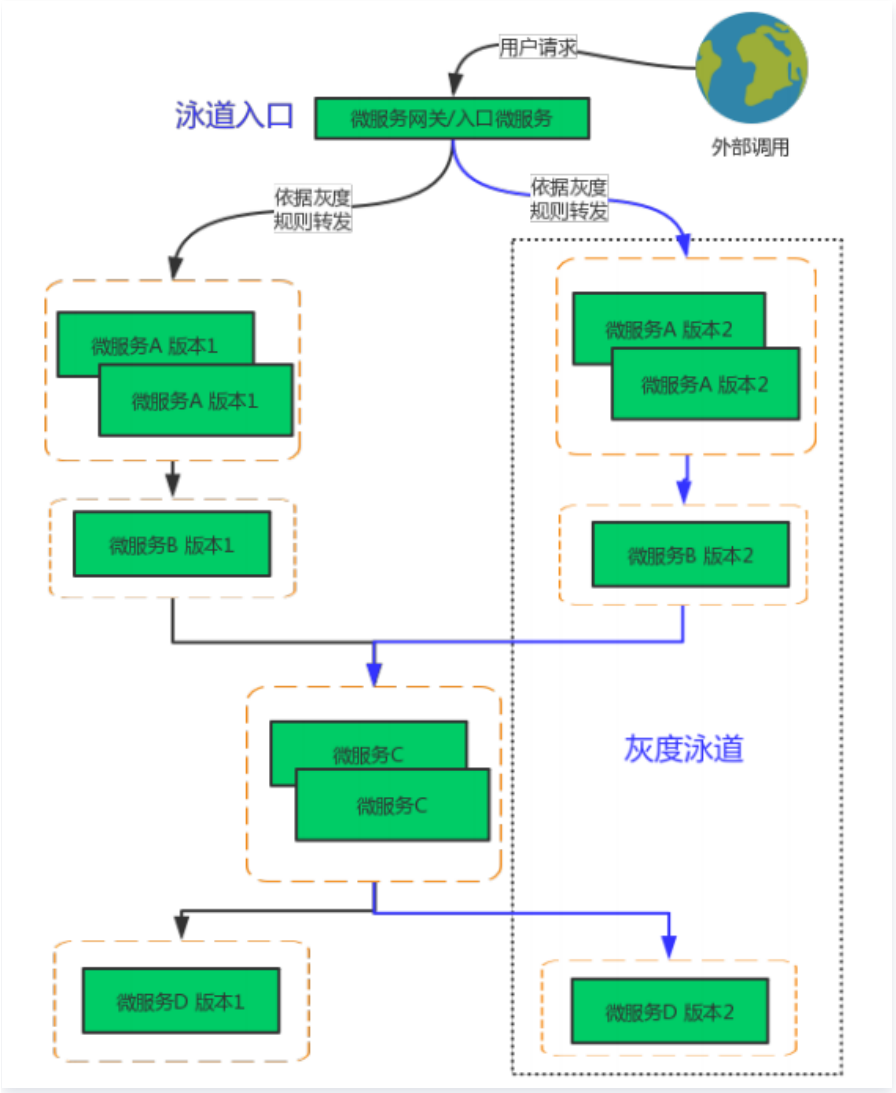
单击应用“ID”，进入部署组列表页面，查看目标部署组的状态是否为**正常**，已部署的程序包/镜像是否为需要发布的最新版本。

# 全链路灰度发布

## 概述

最近更新时间：2025-03-10 18:08:33

灰度发布是软件上线过程中常见的上线方式，是指在发布过程中，将具有一定特征或者比例的流量分配到需要被验证的版本中，用来观察新的验证版本的线上运行状态。
相比全量上线，灰度发布是更加谨慎的发布形式。当线上调用链路较为复杂时，全链路灰度发布可以将线上的各个服务隔离出一个单独的运行环境。
全链路灰度发布的常见场景如下：




如上图所示，我们划分出一个灰度环境，针对线上版本2的所有服务，我们希望都能配置一定规则。当满足一定规则后，所有请求都会流入灰度环境中，经过微服务 A 版本2的请求，一定会同样落入微服务 D 的版本2中。

### 名词解释

- **泳道**：泳道是一组部署组的集合，是灰度发布规则的目的地。泳道中的部署组属于不同的应用。可以认为用户通过划分泳道划分出了灰度环境。
- **灰度规则**：用户在灰度规则上配置请求需要满足的条件。当请求满足一定条件后，可以通过灰度规则将流量路由到某一个泳道中。

版权所有：腾讯云计算（北京）有限责任公司

# 泳道配置

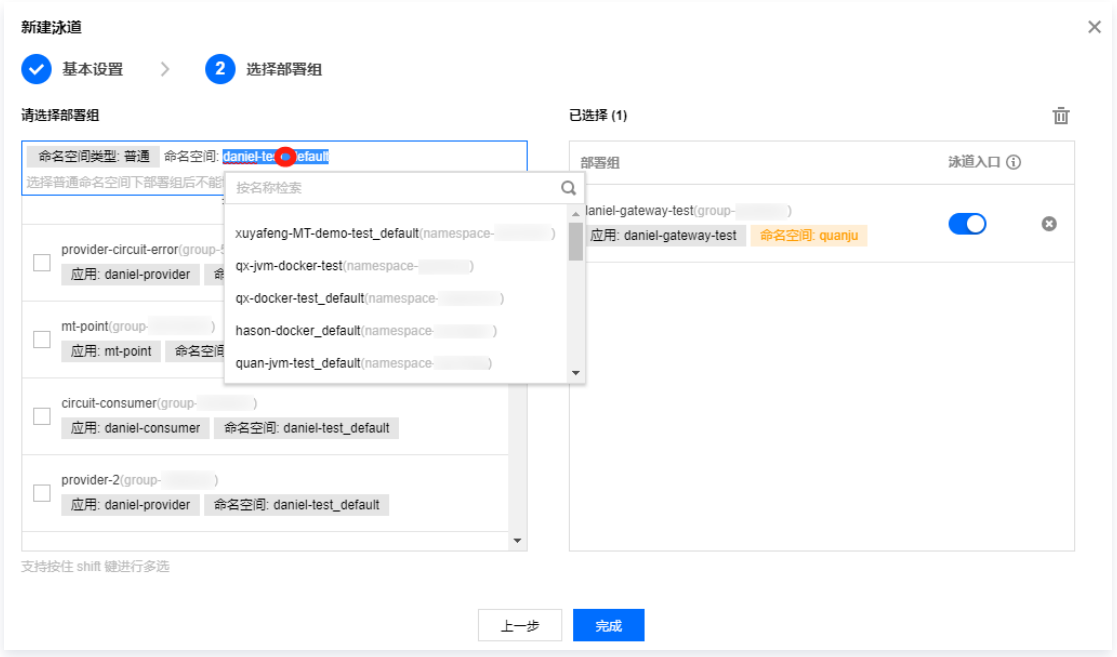
最近更新时间：2025-03-10 18:08:33

## 操作场景

使用全链路灰度发布之前，需要先配置泳道。一条泳道相当于一个灰度环境，环境中包含应用中需要进行灰度测试的部署组。在 全链路发布概述 中的示例图中，我们希望为不同微服务的版本2创建一条泳道，将微服务 A、B、D 部署版本2的部署组都放在一个泳道中。
假设当前访问链路为 gateway > consumer > provider。其中 consumer、provider 都有两个版本。希望通过请求参数 test = 1时配置流量通过网关访问到 consumer、provider 的版本1中。

## 新建泳道

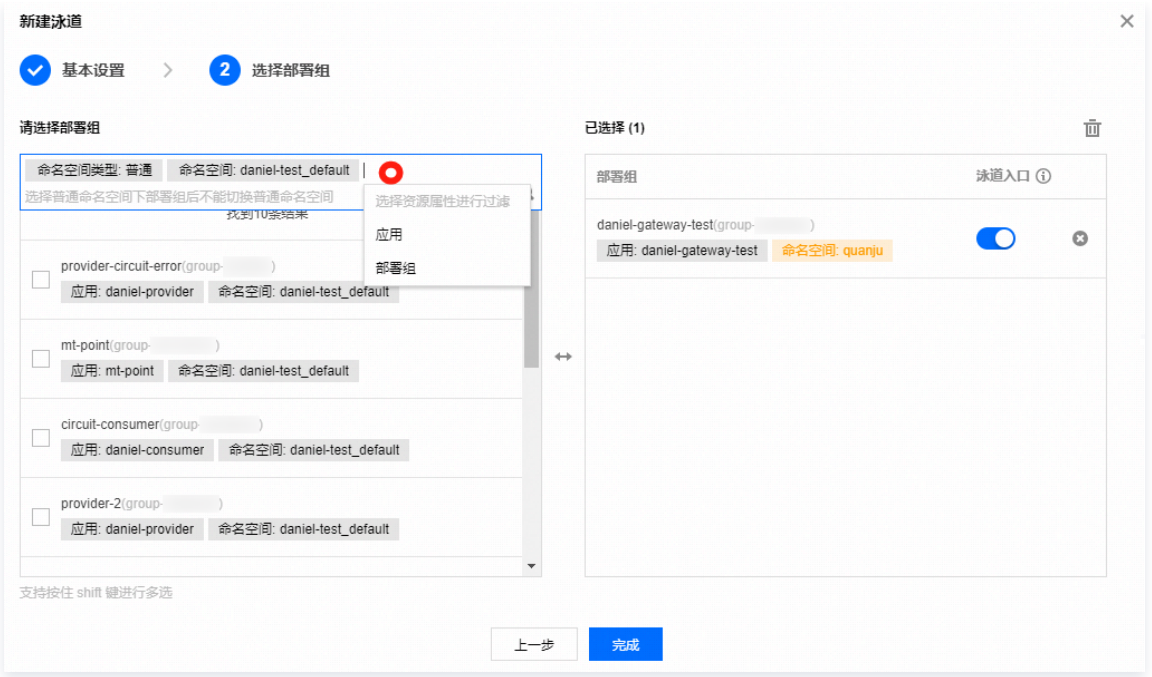
1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏单击 **TSF-Consul**，在页面上方选择 **全链路灰度发布** 页签，进入灰度发布页面。
3. 选择**泳道配置**页签，单击**新建泳道**，填写泳道基本信息。
   - 泳道名称：必填。最长60个字符，只能包括小写字母、数字及分隔符。
   - 标签：用于分类管理资源，可不选。详情参见 标签 。
   - 备注：选填。最长200字符。
   - 数据集：用于细粒度管理子账号权限。数据集使用，请参阅 数据集管理 。
4. 单击**下一步**，勾选需要导入泳道的部署组，单击**完成**。
5. 此时，我们需要创建一个泳道，将 gateway、consumer 的版本1部署组、provider 的版本1部署组添加到这个泳道中。设置 gateway 为泳道的入口。




### 过滤条件

- 用户可以通过选择命名空间类型、命名空间名称、应用名称和部署组名称进行筛选过滤部署组。默认提供了命名空间类型和命名空间名称作为过滤条件。
- 当用户希望切换命名空间时，单击当前已经选中的命名空间的名称，在弹出的下拉列表中切换。
- 当用户希望通过应用名称筛选部署组时，单击命名空间右侧的空白位置，便可弹出应用和部署组的下拉选项，如图所示。

版权所有：腾讯云计算（北京）有限责任公司




### 泳道入口

- 全链路灰度发布规则配置后，会在泳道的入口部署组上对请求规则进行校验，以此来判断请求是否应该进入某一个泳道中。通常情况下，泳道入口是一个网关。
- 同一个泳道中支持多个入口，在请求经过每一个入口部署组时，都会判断请求是否应该进入泳道中。
- 配置方法：在选择部署组页面勾选需要被导入到泳道中部署组后，在穿梭框的右侧通过泳道入口开关 配置。

### 泳道与命名空间

- 当前 TSF 支持两种 命名空间：全局命名空间和普通命名空间。
- 泳道中可以包含全局命名空间中的部署组和普通命名空间中的部署组。

> **说明：**
> 当前 Mesh 应用不支持部署在全局命名空间中，不支持泳道中同时出现全局命名空间中的部署组和 Mesh 应用的部署组。

## 删除泳道

> **注意：**
> 当泳道上已经绑定了全链路灰度发布的规则，则不能删除。

在泳道配置列表页面，选择目标泳道，单击操作列的**删除**。删除后，全链路灰度规则在该泳道上将不可用。

## 编辑泳道

1. 在泳道配置列表页面，单击目标泳道的“ID”，进入泳道详情页。
2. 在基本信息模块中，单击右上角的**编辑**，可以修改泳道名称和备注。
3. 在部署组管理模块中，可以向泳道添加或移除部署组，也可以设置某个部署组为泳道入口。

## 查看泳道监控信息

1. 在泳道配置列表页面，单击目标泳道的“ID”，进入泳道详情页。
2. 选择**泳道监控**页签，设置好时间范围后，即可查看该泳道的请求量、耗时和错误率信息。

# 灰度发布

最近更新时间：2025-03-10 18:08:34

## 操作场景

创建好泳道后，我们需要配置灰度发布规则。延续泳道配置中的场景：假设当前访问链路为 gateway > consumer > provider。其中 consumer、provider 都有两个版本。希望通过请求参数 test = 1 时配置流量通过网关访问到 consumer、provider 的版本1中。
当前，gateway 以及 consumer、provider 的版本1 已经放在了泳道中，我们需要配置一条规则，当请求参数 test = 1时，流量到版本1的泳道中。

## 前提条件

- 已 创建服务泳道 。
- 不支持创建子线程进行调用后的灰度发布。

## 新建灰度规则

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏单击 **TSF-Consul**，在页面上方选择 **全链路灰度发布** 页签，进入灰度发布页面。
3. 单击**新建灰度规则**，填写规则名称和备注。
   - 规则名称：必填。支持中英文字符，不超过60字符。
   - 标签：用于分类管理资源，可不选。详情参见 标签 。
   - 备注：选填。不超过200字符。
4. 单击**下一步**，填写请求参数规则和规则生效关系。




- **标签名**：是 TSF 中的自定义标签，自定义标签是用户自定义的业务参数转换的 TSF 中使用的标签。 参数名最多32字节，参数值最多128字符。此处，我们配置标签名为 test，设置标签值为1。
- 自定义标签的来源有两种：
  - 方式一：通过微服务网关将请求的参数转换为标签：如请求的 path、query、header 中包含的业务参数，通过微服务网关的 tag 插件转换为标签。详细使用说明可以参见 微服务网关 Tag 插件 。
  - 方式二：已在代码中配置标签，详细使用说明可以参见 参数传递 。
- **逻辑关系**：支持等于、不等于、包含、不包含、正则表达式五种逻辑关系。
  - 选择等于和不等于时，标签值只能填写为单个值。
  - 选择包含和不包含时，标签值可以填写多个，使用英文半角逗号分隔。

版权所有：腾讯云计算（北京）有限责任公司

- **规则生效关系**：上述条件的生效逻辑关系，或（满足任一规则）、与（满足全部规则）。每条全链路灰度规则最多包含10条子规则。

5. 单击**下一步**，配置流量目的地，选择已经创建好的泳道。




6. 单击**完成**。

## 切换灰度规则生效状态

在灰度发布页面，通过生效状态栏的按钮 调整规则是否生效。支持生效多条规则。

## 调整灰度规则优先级

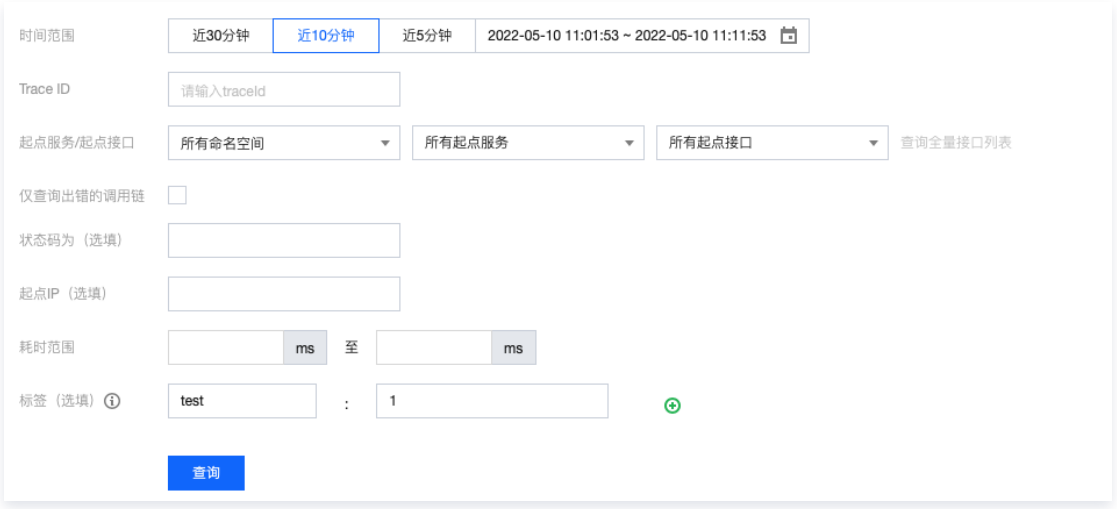
全链路灰度发布支持同时生效多条规则，并支持为规则配置优先级。当同一条请求同时满足多条规则时，会优先匹配高优先级的规则。

在灰度发布页面，单击**调整规则优先级**，拖动规则名左侧的拖拽图标，进行优先级配置，并单击**保存**。实际生效的优先级是生效状态的规则从页面上方向下的顺序进行排列的。

## 全链路灰度结果展示

在腾讯微服务平台控制台的**依赖分析 > 调用链查询**中，用户可以查询对应的请求是否按照预期走到了泳道中。

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏单击**观测中心 > 调用查询**，进入调用链查询页面。
3. 选择好时间、调用的命名空间和微服务，或通过其他条件筛选需要查询的调用链。如果需要对标签和标签值进行过滤，可以输入标签和标签值，进行过滤。




4. 单击**查询**，结果如下：




5. 单击非泳道入口的微服务的调用链详情或者蓝色的耗时条，可以查看调用链详情。




## 全链路灰度流量流向规则与说明

- 当请求流量没有命中任何灰度规则，流量将走到没有被添加到泳道的部署组中。
- 当某一个微服务下的部署组没有被加入任何泳道中，请求将在该微服务下所有部署组的所有实例中轮询。经过该微服务后，请求将继续按照规则配置流入对应泳道。
- 举例：A > B > C 微服务调用，A 和 C 微服务下有版本1和版本2，B只有一个版本。配置灰度规则test = 1 的流量进入版本2对应的泳道后，流量将从 A 的版本2进入，经过 B 微服务，流入到 C 微服务的版本2中。
- 一个部署组可以属于多个泳道。
- 在泳道中，服务路由规则不生效。
- 泳道是一个隔离环境。当泳道上所有规则关闭后，流量不会进入泳道中。如希望恢复建立泳道前的流量分配方式，请将泳道删除。

# 查看发布详情

最近更新时间：2025-03-10 18:08:34

## 操作场景

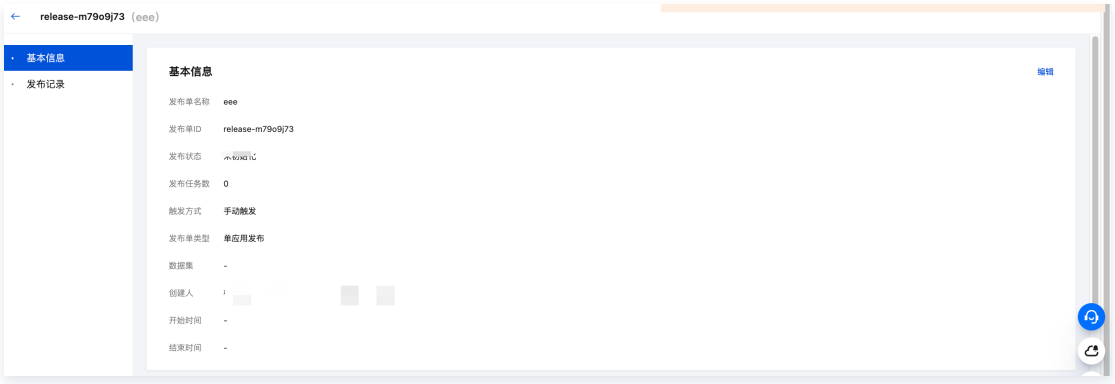
发布单执行后，可以在发布记录中查看发布详情，针对发布成功的发布记录可执行发布回滚操作，用户确认发布回滚操作后按照发布流程依次进行回滚操作。
关于回滚相关操作请参见 发布回滚 。本文介绍在发布单执行后如何查看发布详情。

## 前提条件

发布单已开始执行。

## 操作步骤

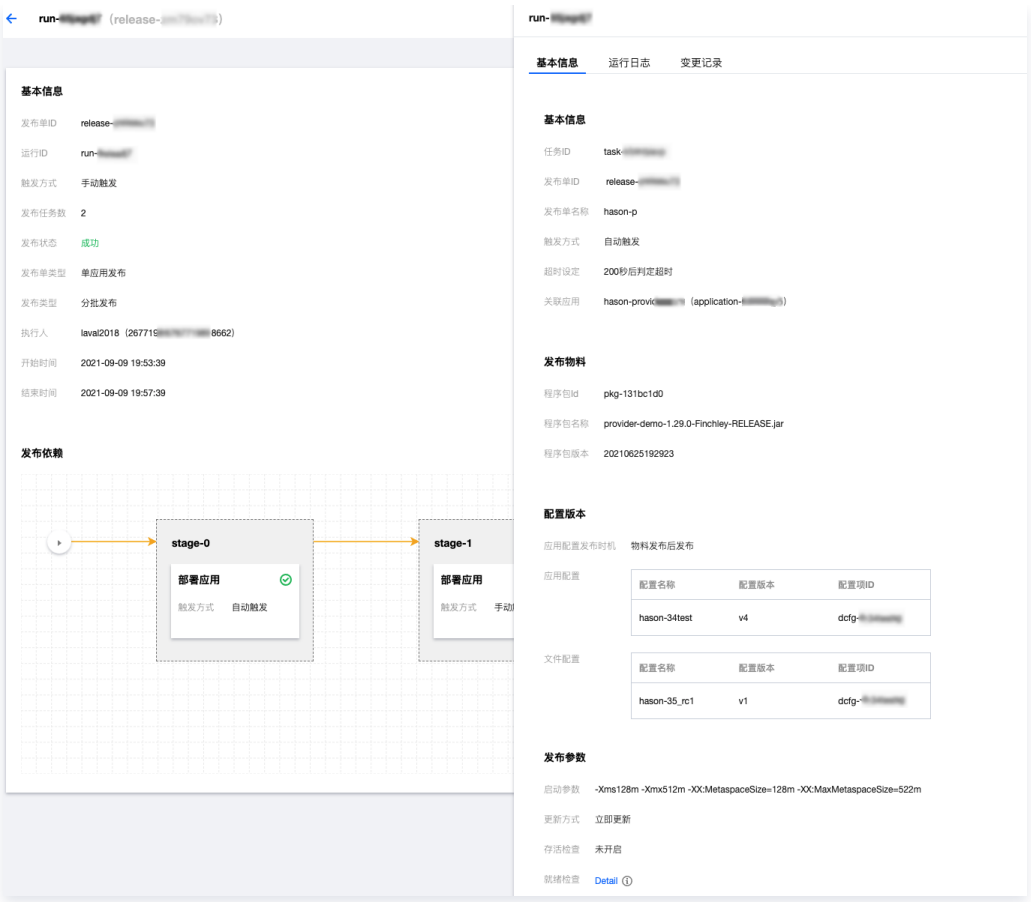
1. 登录 腾讯微服务平台控制台 ，选择**应用管理**，单击应用ID，进入应用详情。
2. 在左侧菜单栏选择**应用部署**，选择**发布计划**，但单击**发布单ID**。
3. 在发布单基本信息页面，选择左侧菜单栏的**发布记录**，可查看执行的发布记录列表。
4. 单击“运行 ID”，可查看发布记录详情，包含发布记录基本信息和发布依赖拓扑。



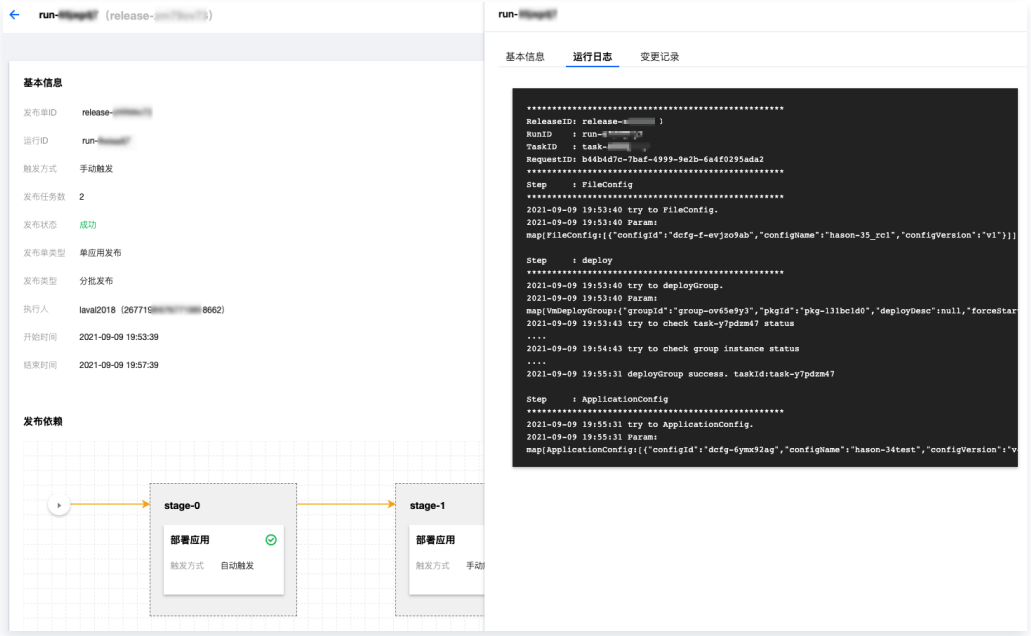

5. 单击**部署应用**或**流量路由**卡片，选择**查看详情**，可查看详细信息。

### 部署应用节点详情

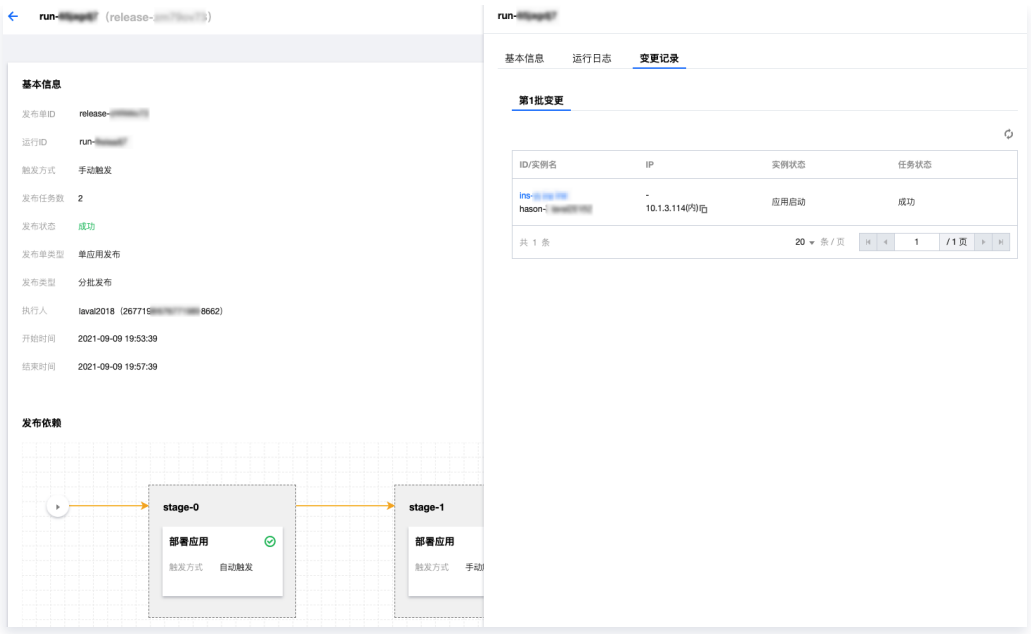
- 基本信息




- 任务节点执行日志



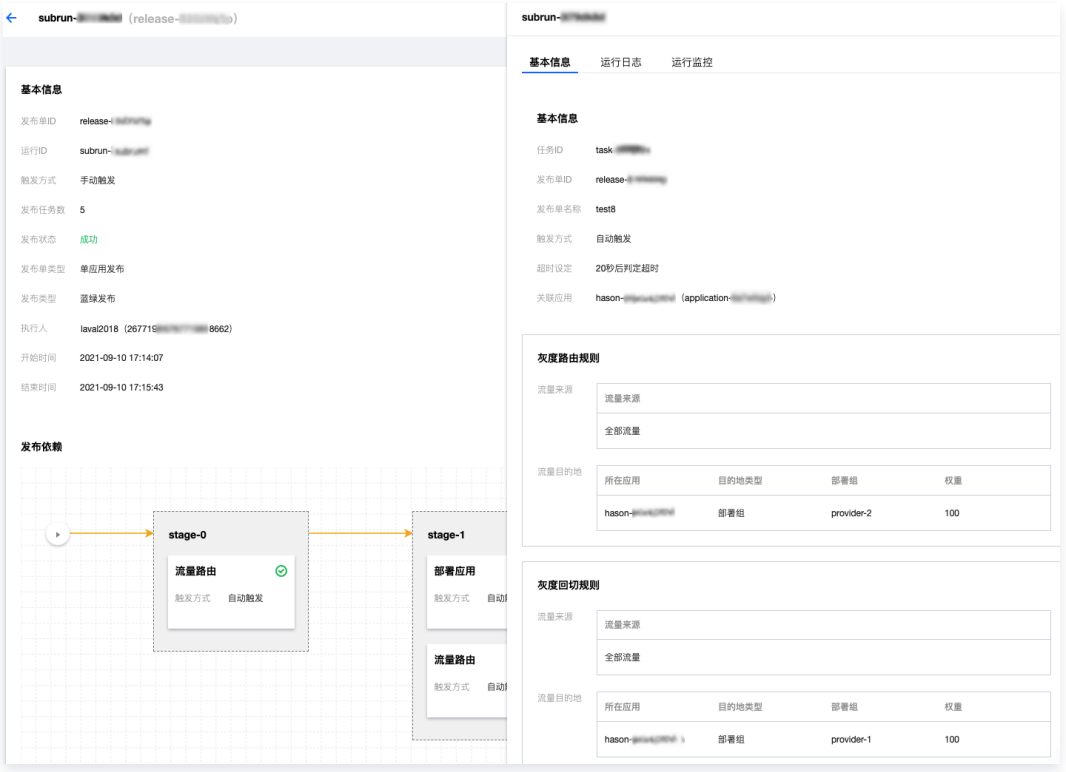

- 变更记录




## 流量路由节点详情

> **说明：**
> 只有蓝绿发布才会有流量路由节点。

- 基本信息




- 任务节点执行


subrun-
(release-
基本信息
发布单ID
release-
运行ID
subrun-
触发方式
手动触发
发布任务数
5
subrun-
基本信息
运行日志
运行监控
**********
ReleaseID: release-
RunID : subrun-
TaskID : task-
RequestID: 01647F78-
-a52dfb0ae3b9
**********

- 流量


subrun-
(release-
基本信息
发布单ID
release-
运行ID
subrun-
触发方式
手动触发
发布任务数
5
发布状态
成功
发布单类型
单应用发布
发布类型
蓝绿发布
执行人
laval2018（267715
8662）
开始时间
2021-09-10 17:14:07
subrun-
基本信息
运行日志
运行监控
流量详情
任务执行时
近10分钟
近30分钟
近24小时
2021-09-22 20:03:38 ~ 2021-09-23 20:03:38
暂无数据

# 发布回滚

最近更新时间：2025-03-10 18:08:34

## 操作场景

使用批次发布或者蓝绿发布新版本应用时，若在发布过程中或者发布完成后发现新版本应用异常，可以使用回滚功能回滚应用，待问题解决后再次发布新版本。

## 操作步骤

> **注意：**
> 运行开始超过7天的发布无法回滚。

### 发布中回滚

对于正在运行中的发布单，若发现异常，需要先终止发布再进行回滚，终止发布未执行的任务节点不再进行发布操作。

1. 单击发布计划的 ID，进入发布单基本信息页面。
2. 在页面上方选择**发布记录**页面，单击操作栏的**终止**，终止发布。




3. 发布终止后，单击操作栏的**回滚**。
4. 在回滚信息页面单击任务节点操作栏**详情**，可查看任务节点详情。




5. 在回滚信息页面选择**回滚**，弹窗中单击**确认**。




## 发布后回滚

对于已发布的发布单，若发现异常，直接回滚即可。

1. 单击发布计划的 ID，进入发布单基本信息页面。
2. 在页面上方选择**发布记录**页面，单击操作栏的**回滚**。
3. 在回滚信息弹窗中选择**回滚**，完成应用回滚。



# 应用运维
# 应用管理

最近更新时间：2025-03-10 18:08:33

## 操作场景

将虚拟机应用托管至 TSF 后，您可以在 TSF 控制台上对应用进行管理，包括创建、部署、扩缩容、停止和重启应用，以及部署新版本应用等操作。

## 操作步骤

### 创建应用

登录 腾讯微服务平台控制台，在**应用管理**页面，选择**业务应用**，单击**新建**创建应用，具体操作请参见 虚拟机托管应用 。

### 部署应用

在 腾讯微服务平台控制台 创建一个未部署的应用后，您可以单击应用 ID进入**应用详情**页，单击**应用部署**对应用进行部署。具体操作请参见 虚拟机托管应用 。
已部署过的应用，可以在部署组列表的操作栏选择**部署应用**来升级应用版本。

### 删除应用

> ⚠ **注意**
> - 删除应用会同时删除应用关联的程序包。
> - 当应用下有部署组时，无法执行删除操作，需要先删除所有部署组后才能删除应用。

1. 登录 腾讯微服务平台控制台，在**应用管理**页面的**操作**列，单击**删除**。
2. 在确认弹框中，单击**确认**完成删除。

### 删除部署组

> ⚠ **注意：**
> 部署组下有正在运行的实例时需要先下线实例才能删除部署组。部署组删除后将不再对外提供服务。

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的“ID”，进入应用详情页，左侧导航栏选择**应用部署**，进入部署组列表页。
2. 在部署组列表页面，单击目标部署组操作列的**更多** > **删除**。
3. 弹出提示页面，单击**确定**删除部署组。

### 下线实例

停止 CVM 云服务器上的应用，然后将实例从部署组中移除。
1. 在应用列表页面，单击目标应用的“ID”，进入应用详情页，点击应用部署菜单，进入部署组列表页。
2. 在部署组列表页面，单击目标部署组操作列的**更多** > **下线实例**。
3. 弹出提示页面，勾选需要下线的云主机实例，单击**提交**。

### 添加实例

1. 在应用列表页面，单击目标应用的“ID”，进入应用详情页，点击应用部署菜单，进入部署组列表页。
2. 在部署组列表页面，单击目标部署组操作列的**更多** > **应用扩容**。
3. 选择要添加进部署组的云服务器 CVM，单击**提交**。

版权所有：腾讯云计算（北京）有限责任公司

> **说明：**
> - 添加实例时，支持按照单实例名称模糊匹配，一次可添加多个云主机实例。
> - 部署组如果已选择在控制台配置启停脚本，扩容时如果实例 agent 版本不支持该特性，请升级agent到最新版本。请参见 应用管理相关。

4. 在部署组的实例列表页面中显示出刚才添加的 CVM。

## 停止应用

将部署组中所有的节点上运行的应用停止，停止后可以再启动。应用停止后，将不再对外提供服务。

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击目标部署组操作列的**更多** > **停止应用**。
3. 弹出提示页面，单击**确定**停止应用。

## 启动应用

应用停止后，您可以启动应用，应用将重新对外提供服务。

1. 登录 腾讯微服务平台控制台，在**应用管理**页面，单击目标应用的 “ID”，进入应用详情页。
2. 左侧导航栏选择**应用部署**，进入部署组列表页面，单击目标部署组操作列的**更多** > **启动应用**。
3. 弹出提示页面，单击**确定**启动应用。

# 弹性伸缩

最近更新时间：2025-03-10 18:08:34

在分布式应用管理中，弹性伸缩是十分重要的运维能力。它能够根据预先设定的弹性伸缩规则自动增加或减少部署组实例数量，即扩容或缩容，从而提高资源利用率、降低资源成本。

## 概述

TSF 弹性伸缩支持指标规则和定时规则两种类型的弹性伸缩策略。

- **指标规则**：适用于突发流量和典型周期性流量的应用场景。由规则名、扩容活动、缩容活动、冷却时间等参数构成的规则，用来描述弹性扩缩容的触发条件、实例数量变化和限制。

  **弹性伸缩触发指标：**

| 指标 | 说明 |
| --- | --- |
| CPU 利用率 | 在指定时间范围内，部署组内所有实例 CPU 利用率的平均值。 |
| 内存利用率 | 在指定时间范围内，部署组内所有实例内存利用率的平均值。 |
| 请求 QPS | 在指定时间范围内，部署组内所有实例请求 QPS 的平均值。 |
| 响应时间 | 在指定时间范围内，部署组内所有实例响应时间的平均值。 |

- **定时规则**：适用于资源使用率有周期性规律的应用场景。达到设定的周期时间点后，TSF 将依据设定的策略自动调整部署组实例数量。

## 使用限制

- 请勿同时开启容器平台的弹性伸缩，可能会导致 TSF 的弹性伸缩规则被容器平台的弹性伸缩规则覆盖，导致 TSF 弹性伸缩规则生效后实际结果不符合预期。
- 集群中存在超级节点时，当用户需要配置被动触发弹性时，可以使用 TSF 页面的弹性伸缩能力。当用户希望进行定时扩缩容的时候，建议通过 HPC/HPA 进行配置。

## 新建规则

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航单击**应用管理**，选择**弹性伸缩**，选择好地域后单击**新建规则**，填写弹性伸缩规则内容。

**指标规则**

| 参数 | | 说明 |
| --- | --- | --- |
| 扩容活动 | 触发条件 | 由指标（CPU 利用率、内存利用率、请求 QPS 和响应时间），阈值和持续时间构成。**多条触发条件为逻辑或（OR）的关系，满足任一条件，将触发扩容操作。** |
| | 增加实例数 | 每次部署组的指标达到了触发条件后，增加的实例数量。 |
| | 最大实例数 | 部署组的实例数量上限。 |
| 缩容活动 | 触发条件 | 由指标（CPU 利用率、内存利用率、请求 QPS 和响应时间），阈值和持续时间构成。**多条触发条件为逻辑与 AND 的关系，必须同时满足才能触发缩容操作。** |
| | 减少实例数 | 每次部署组的指标达到了触发条件后，减少的实例数量。 |
| | 最小实例数 | 部署组的实例数下限。 |
| 冷却时间 | | 冷却时间是指在同一个伸缩组内，一个伸缩活动（添加或移出实例）执行完成后的一段锁定时间，在这段时间内，该伸缩组不执行伸缩活动。建议设置冷却时间大于持续时间，如持续 |

| | |
|---|---|
| | 时间设置1分钟，冷却时间设置5分钟。 |

**定时规则**

| 参数 | 说明 |
|---|---|
| 周期 | 执行定时弹性伸缩策略的周期。<br>每天：每天指定时间执行定时策略。<br>每周：每周指定天数的指定时间执行定时策略。<br>每月：每月指定日期的指定时间执行定时策略。 |
| 触发时间与实例数 | 每日弹性扩缩容的生效时间与生效后的实例数，触发时间为部署地域时区（UTC+8） |

**说明：**
- 在一条弹性伸缩规则中定时规则最多设置10条。
- 如果定时规则中多个策略产生冲突，将按序执行，您可拖动调整顺序；若触发时间早于当前时间，则从下一个时间周期开始生效。

3. 单击**提交**，完成规则创建。

## 关联部署组

创建弹性伸缩规则后，需要将规则关联到部署组上并开启规则才能生效。

1. 在 弹性伸缩 列表页面，单击目标规则操作栏的**关联部署组**。
2. 在关联部署组页面，选择应用后勾选要关联的部署组。
3. 在关联部署组页面左下方，选择是否立刻开启规则。如果选择开启，则规则会在部署组上立刻生效，否则将不生效。用户可以后续在规则详情页的**关联部署组** tab 页中修改启用状态。

## 验证弹性伸缩策略

设置弹性伸缩规则后，TSF 会自动监控部署组实例状态，当满足触发条件时，TSF 会依据弹性策略进行部署组实例扩容或缩容。您可以通过以下步骤查看弹性伸缩策略是否生效。

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择 **观测中心 > 事件中心 > 事件概览**，进入事件概览页面。
3. 在顶部菜单栏选择地域并在页面上方选择要查看的应用，设置好时间范围（支持近6小时、近3天、近7天和自定义时间范围）。
4. 查看事件类型为弹性伸缩的事件及详情。若符合预期，说明弹性策略已生效。
5. （可选）您也可以在应用列表页面，查看运行实例数量。如果运行实例数与弹性策略设置的实例数一致，说明弹性策略已生效。

## 解除规则和部署组的关联

1. 在 弹性伸缩 列表页面，单击目标规则的“ID”进入详情页，单击**关联部署组**页签。
2. 在关联部署组列表右侧操作栏，单击**删除**，**确认**解除规则和部署组的关联。

## 删除规则

删除弹性伸缩规则前需要解除规则和部署组的关联，否则无法删除。

在 弹性伸缩 列表页面，单击目标规则操作栏的**删除**，**确认**后即可删除规则。

## 编辑规则

在 弹性伸缩 列表页面，单击目标规则操作栏的**编辑**，修改规则后单击**提交**即可完成规则的修改。

版权所有：腾讯云计算（北京）有限责任公司

# 服务预热

最近更新时间：2025-03-10 18:08:34

当一个线上应用正在运行时，用户需要进行重新部署、扩容或者更新的时候，新注册的节点需要进行一些懒加载、连接池初始化等动作，无法接受一些大流量的请求，此时新上线的服务可能会有短时间内无法处理突增流量导致进程异常的风险。
针对应用启动阶段，TSF 提供了服务预热能力进行保护，对刚刚上线的节点仅分配较小流量，以方便新增节点完成初始化。避免在大流量场景下，由于初始化不彻底出现大量请求报错情况。

## 应用场景

当用户开启了滚动部署后，可以配置服务预热策略，在如下情况下服务预热策略会生效：

1. 新增节点：如新部署组上线，新增节点等。
2. 更新节点：如部署组进行更新重启，应用扩容等。

## 使用限制

仅 Spring Cloud 普通应用支持服务预热能力，对应支持的版本如下：

| 限制项 | 支持版本 |
| --- | --- |
| Spring Cloud | Spring Cloud Finchley、Spring Cloud Greenwich、Spring Cloud Hoxton、Spring Cloud 2020，且SDK版本为1.40及之后的版本 |

## 配置服务预热

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏中，单击**应用管理 > 业务应用**，选择需要配置的应用ID，进入应用详情。
3. 选择**应用部署 > 部署组**，单击**应用部署**按钮，或找到目标部署组，单击**操作**列的**更新部署**。
4. 在发布策略，选择发布方式为**滚动部署**，并设置好更新策略。
5. 在**服务注册配置&可观测**步骤，打开**是否开启服务预热**开关，在展开的预热配置中根据您的业务需求配置服务预热策略。

> **说明：**
> 仅说明注册配置治理中心为TSF-Consul下的配置，注册配置治理中心为北极星时，请参考北极星配置使用说明。




| 参数 | 描述 |
| --- | --- |
| 预热间隔 | 应用实例启动的预热时间，默认预热时长为 60 秒。服务预热时长设置范围为 0~600 秒。建议该值小于分批等待时间，避免出现多批节点同时在预热、更新状态，从而降低更新中服务的流量承载能力。 |
| 预热流量系数 | 基于已配置的预热间隔，被预热的应用流量权重会根据配置的预热流量系数呈指数型增长。在指定预热间隔内，预热流量系数越大则应用刚启动时分配的权重越小，以满足需要较长时间进行预热的复杂应用的预热需求。预热流量系数默认值为 2，表示在预热间隔周期内服务提供者的流量接收曲线形状呈 2 次曲线形状。预热流量系数设置范围为 0~5。相同预热间隔，预热流量系数越大，表示预热开始时接收的流量越小，临近预热结束时接收的流量增幅越大。 |
| 预热终止保护 | 当超过50%节点处于预热阶段，则预热终止，避免流量聚集在少量节点导致过载。预热终止开关默认打开。 |

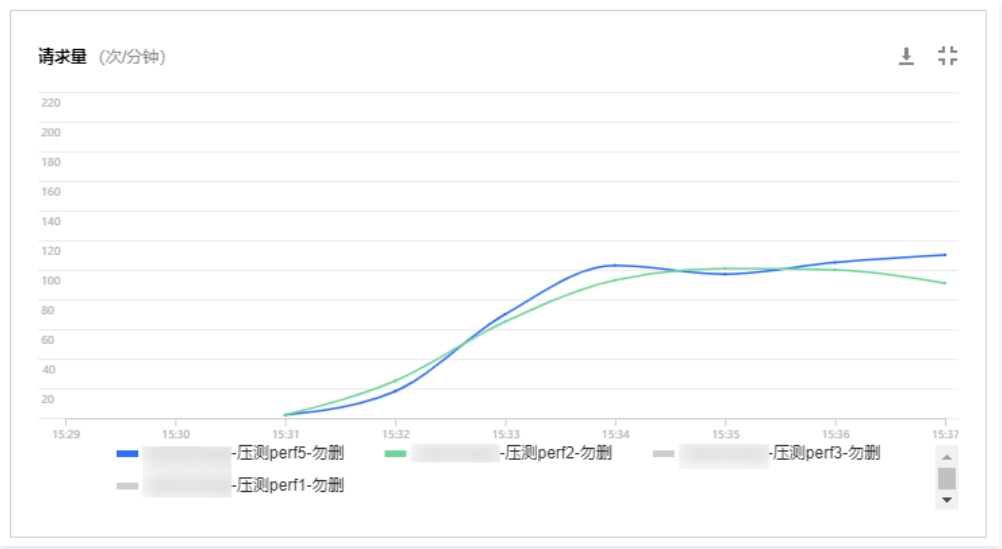
6. 完成设置后单击**提交**。

## 结果验证

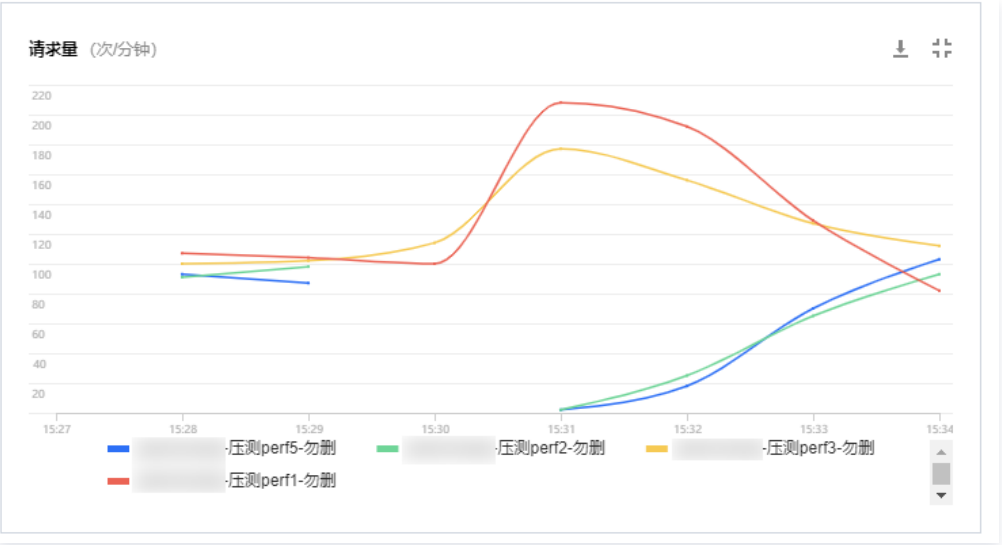
在应用部署后，可以查看预热效果。

1. 在左侧菜单栏选择TSF-Consul，在**服务监控**页签选择所属命名空间。
2. 选择对应的服务实例后，在监控页面可以查看预热效果。
   - 服务预热效果。

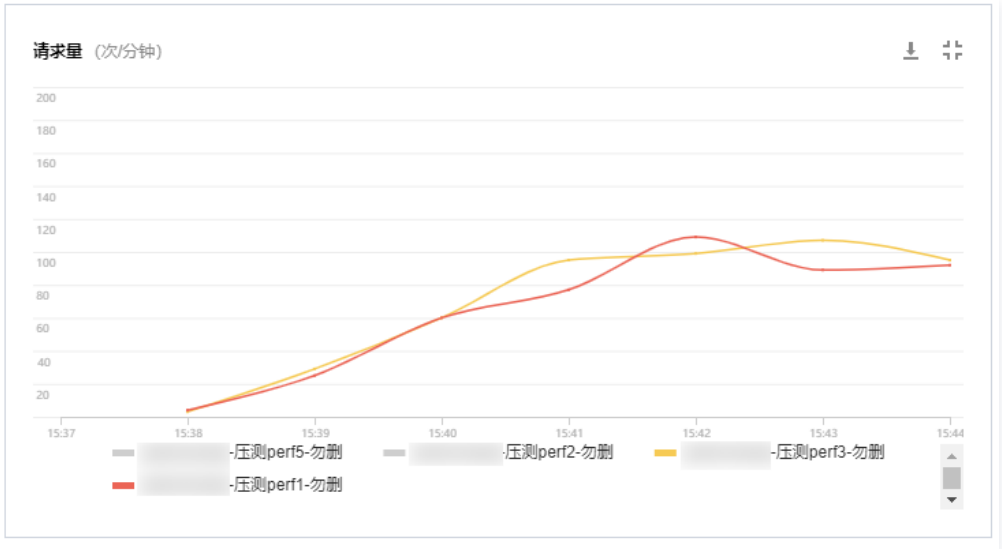
     2.1.1 分批重启，第一批重启两个节点，流量系数为 2，预热间隔为 180 秒。可以看到流量缓慢增长，三分钟后达到顶峰。



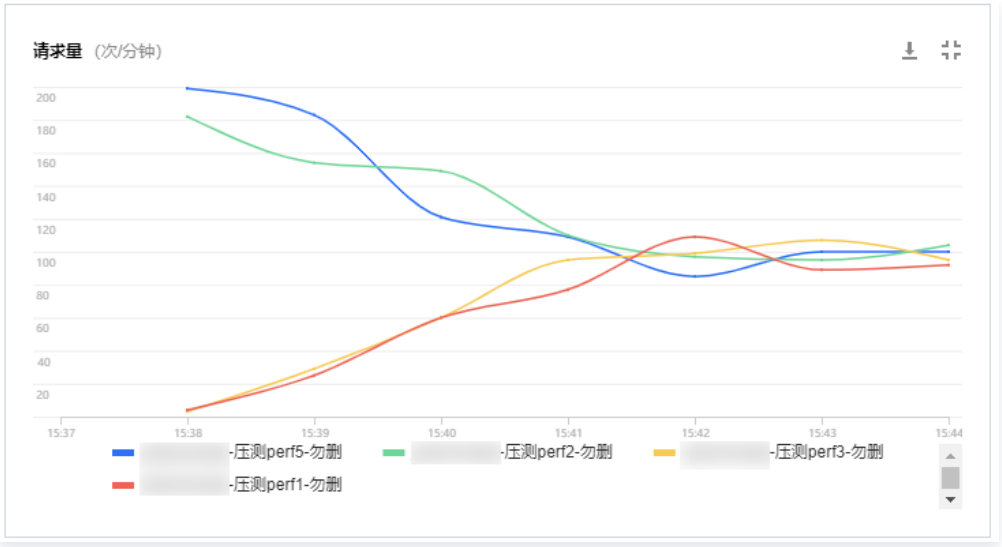

2.1.2 在第一批重启时，未重启的节点，承担了全部流量。



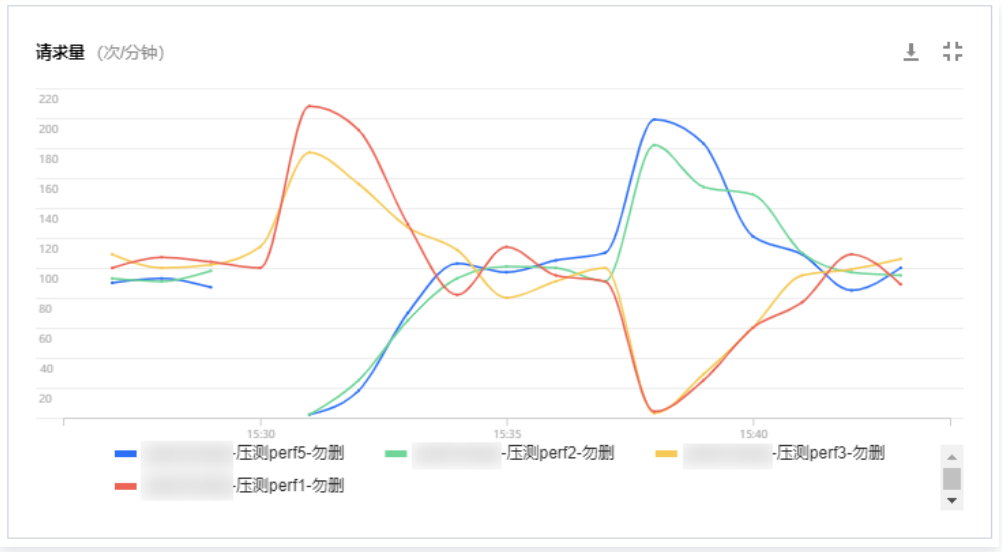

2.1.3 第二批重启时的流量接收表现。




2.1.4 第二批重启时，第一批重启的节点承担了大部分流量。

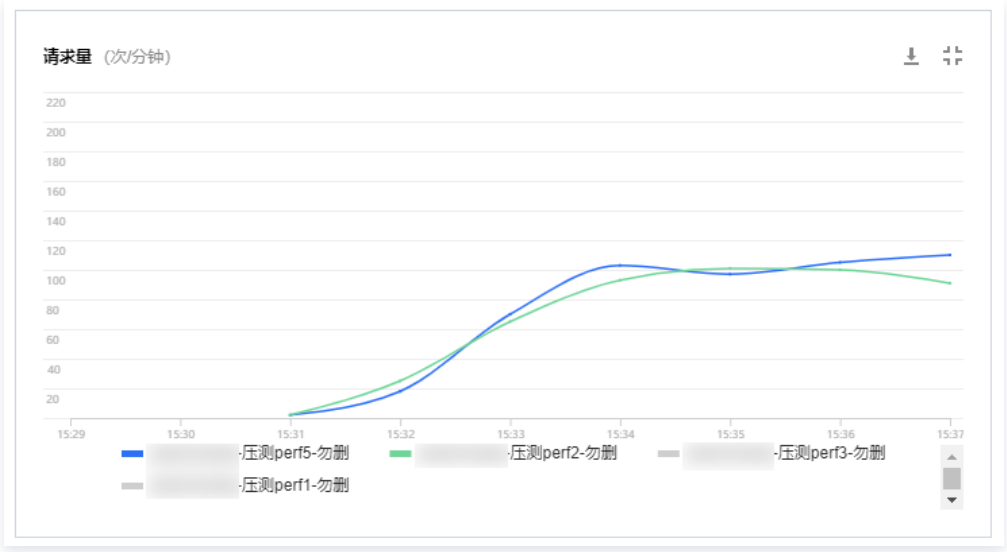



2.1.5 整体预热过程的表现

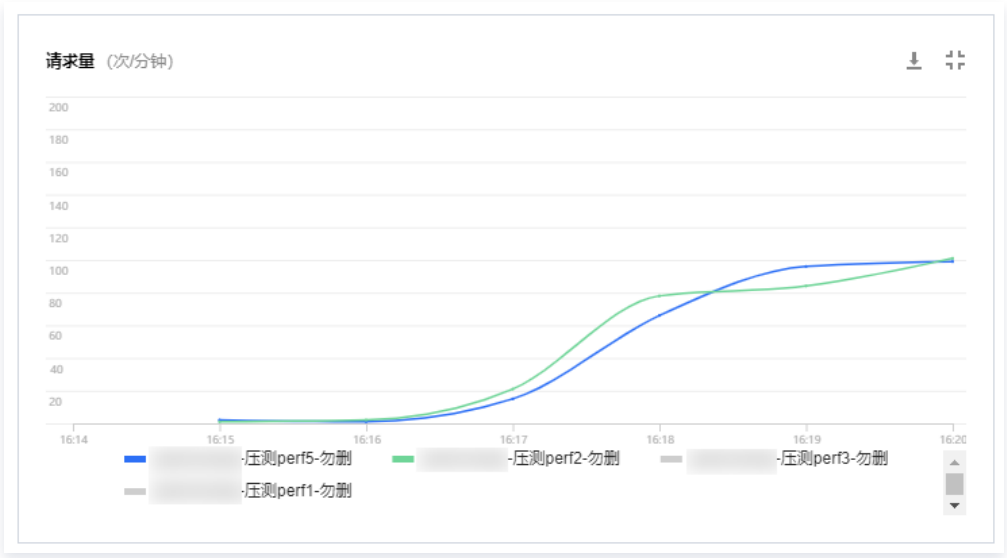



- 在稳定的业务流量下，可以看到新发布的应用所占流量比例呈预热流量系数缓慢增长。通过配置部署组的预热间隔以及预热流量系数来控制新发布的部署组流量比例，用户可以根据业务需求按需配置预热时长，并根据程序复杂度控制预热流量系数来调整应用刚启动时分配的流量权重。
- 应用部署时，配置不用的预热流量系数展示示例如下。

  2.1.1 预热间隔为 180 秒，预热流量系数为 2。




2.1.2 预热间隔为 180 秒，预热流量系数为 5。



# 脚手架

最近更新时间：2025-03-10 18:08:34

## 操作场景

为提高用户开发微服务代码的效率， TSF 提供脚手架功能。脚手架可以实现快速生成基于 TSF 框架的微服务工程源码，帮助业务设计人员快速构建可运行的微服务程序。
TSF 引入了脚手架模板的概念用于保存工程信息，方便后续可以基于模板修改工程参数和下载工程源码。

## 操作步骤

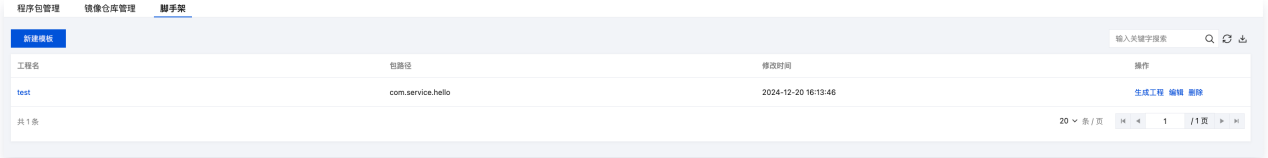
### 新建脚手架模板

1. 登录 腾讯微服务平台控制台 。
2. 单击左侧导航**制品管理 > 脚手架**。
3. 单击**新建模板**。
4. 填写工程配置和 POM 配置信息。
   - **工程名**：Spring Boot 工程名，字母开头，支持大小写字母、数字组成，不超过24个字符长度。
   - **包路径**：package 路径，小写字母开头，支持小写字母、数字或小数点组成，不超过60个字符长度。
   - **GroupID**： pom.xml 文件中的 groupId。
   - **Artifact ID**： pom.xml 文件中的 artifactId。
   - **Name**：pom.xml 文件中的 name。
   - **Version**：pom.xml 文件中的 version。
   - **Description**：选填，pom.xml 文件中的 description。
5. 填写服务基本信息：可填写多个
   - **服务名**：对应工程文件中的 `spring.application.name` 。
   - **端口**：服务监听端口，仅支持0 – 65535。
   - **Controller 类名前缀**：可填写多个，将会生成多个带有前缀的 Controller 类。
6. 调用方式：选择 Feign、RestTemplate 或 AsyncRestTemplate。
7. Controller 调用关系（仅服务个数多余1时可配置）：可填写多个，将在生成的工程中显示服务之间以 `/echo` 接口的调用逻辑代码。最多支持配置5个。
   - 主调服务名：选择主调服务的名称
   - 主调 Controller 类：选择主调服务的 Controller 类。
   - 被调服务名：选择被调服务的名称，不能与主调服务名称重复。
   - 被调 Controller 类：选择被调服务的 Controller 类。
8. 单击**保存并下载**，保存模板，并下载工程的 zip 文件。或者单击**保存**，仅保存模板，不执行下载操作。

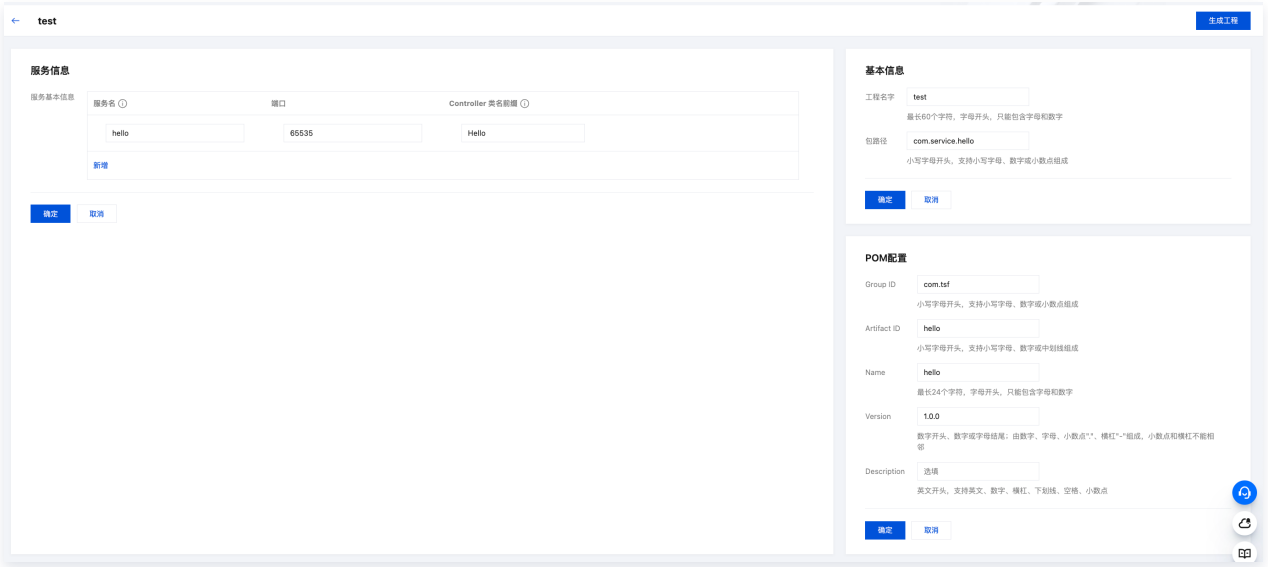
> **注意：**
> 界面将会根据 Controller 调用关系实时生成桑基图，以便用户更直观地看到服务相互间的依赖关系。

### 修改脚手架模板

1. 单击脚手架模板列表上目标模板的工程名。



2. 进入详情页可修改模板的参数信息，各参数字段含义参见 新建脚手架模板 。




## 删除脚手架模板

1. 单击脚手架模板列表上目标模板右侧的**删除**。
2. 在弹框中单击**确认**，删除脚手架模板。



# 中间件应用管理
# Java 网关概述

最近更新时间：2025-03-10 18:08:33

微服务网关作为后台架构的入口，提供路由转发、API 管理、访问过滤器等作用，是微服务架构中的重要组件。TSF 中的微服务网关基于 Spring Cloud 中的 Zuul/Spring Cloud Gateway 实现，提供了符合微服务体系的灵活可自定义的网关功能。

> **说明：**
> 仅专业版和铂金版支持使用 Java 网关，具体说明详见 版本对比 。

## 微服务网关的主要功能

### 请求转发

请求转发是微服务网关的基本功能。TSF 中的微服务网关可以通过页面配置灵活管理需要被转发请求的微服务 API。微服务网关会及时从注册中心感知后端服务节点健康状况，保证在后端服务节点变动情况下请求不中断。

### API 管理

微服务网关集中管理了所有需要对外暴露的 API，帮助用户进行 API 的生命周期管理。

### API 治理

支持 API 级别的限流、路由等能力，支持用户绑定系统插件或自定义插件。

## 典型使用场景

- 提供对外暴露 API 的统一入口。
- 穿透命名空间隔离，实现 set 化部署。
- 协议转换。
- 灰度发布入口。
- 自定义统一入口处理逻辑。

## 功能优势

### 支持完整的服务治理能力

TSF 中的微服务网关形态等同一个普通的应用，也需要用户创建应用（选择“微服务网关类型”应用），进行部署、发布。部署成功且注册到注册中心后也可以在服务治理页面看到微服务网关服务。
微服务网关可以通过服务注册发现感知后端微服务状态，当后端微服务某节点下线时，网关流量不会打到该节点上。
微服务网关支持 TSF 提供的完整的服务治理能力，包含服务路由、服务鉴权、服务限流、监控、调用链、全链路灰度发布、应用配置等。

### 支持灵活定制

微服务网关作为业务层网关，通常需要承担部分相对通用的业务逻辑。由于这部分业务逻辑有很强的业务属性（如请求去重、特殊的鉴权逻辑等），我们提供了微服务网关 SDK，方便用户进行定制化开发。依赖微服务网关 SDK 后，网关的页面功能依然可用。
当用户只需要使用我们页面上提供的微服务网关能力时（如请求转发、监控、密钥对鉴权、第三方鉴权、请求参数转 tag 等），可以直接下载并部署微服务网关 Demo 程序包，不需要额外改动。

### 贴近开源

提供基于 Zuul，Spring Cloud Gateway 两种选型的微服务网关。

### 支持灵活扩缩容

由于微服务网关本质也是一个微服务，用户可以通过手动或配置弹性伸缩规则的方式对网关进行横向扩展。

## 微服务网关开源对比优势

相比开源的微服务网关，TSF 中的微服务网关有以下使用优势：

| 对比项 | TSF 微服务网关 | 开源微服务网关 |
| --- | --- | --- |
| API 管控 | 支持页面化 API 管理能力。自动获取微服务 API 无需手动录入。 | API 需要手动录入到配置文件中，无法灵活调整。 |
| 可观测性 | 强，支持查看 API 各个维度监控指标，快速定位问题。 | 弱，需要配合其他监控组件。 |
| 服务与 API 治理 | 支持 API 限流、熔断、灰度等一系列服务与 API 治理能力。与 TSF 中提供的治理能力全面兼容，可实现全链路灰度发布等高级能力。 | 弱，需要用户通过插件自行开发。 |
| 插件 | 多。提供鉴权、小程序开发等一系列系统插件。同时支持自定义开发插件。 | 仅支持自定义开发。 |
| 使用场景 | 支持全链路灰度发布、单元化部署、set 化部署、高可用部署等多种复杂场景。 | 仅支持有限复杂场景。 |

版权所有：腾讯云计算（北京）有限责任公司

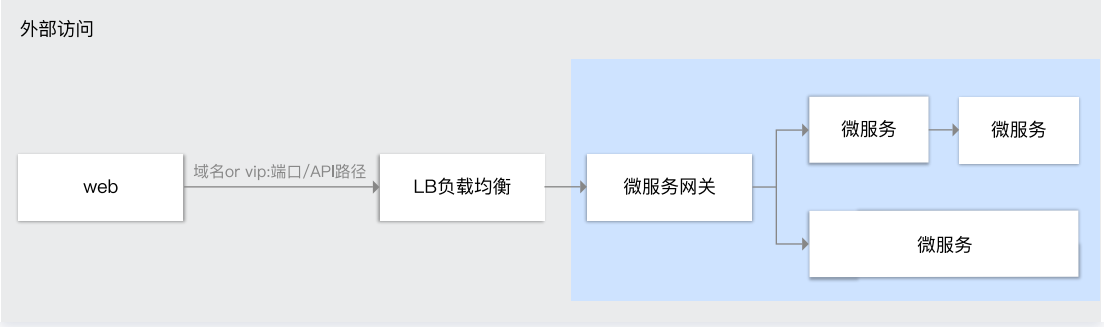
# Java 网关跨命名空间访问

最近更新时间：2025-03-10 18:08:33

## 使用场景

通常在一个微服务系统中，会有两个位置使用微服务网关：

- 微服务系统的边界，如前端与微服务后台之间的网关。




- 不同的微服务业务系统之间（如不同部门的业务后台都使用微服务架构）通过微服务网关进行调用；开发环境和预发布环境的业务都希望通过微服务网关访问某一个公共接口；在 set 化部署（分区部署）和单元化架构部署中，也常常通过微服务网关做不同业务系统之间访问的桥梁。在这类场景中，通常都希望不同系统之间的访问安全可控：不同系统之间进行物理或逻辑的隔离，通过网关实现“最小必须”的访问。




在腾讯微服务平台（TSF）中，我们通过命名空间来实现不同业务系统的访问逻辑隔离，本文主要介绍如何使用命名空间和微服务网关实现不同微服务业务系统之间的调用。

## 微服务网关与命名空间

### 注册配置治理中心使用 TSF-Consul

在 命名空间 中，介绍了 TSF 中的两种命名空间：全局命名空间与非全局命名空间。当注册配置治理中心使用 TSF-Consul 时，服务的命名空间与 TSF 命名空间保持一致，因此：

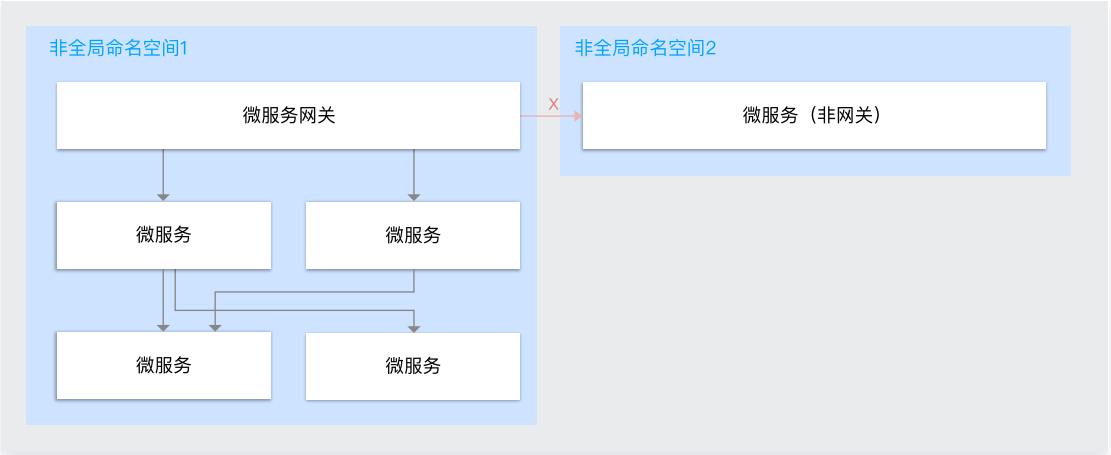
- 全局命名空间中的微服务可以被其他命名空间中的微服务调用（典型场景：公共服务），但是不能调用其他命名空间中的微服务。
- 不同的非全局命名空间（包含系统命名空间）中部署的微服务不能直接相互调用。

微服务网关也可以部署在全局命名空间和非全局命名空间中。在实际使用中，您创建微服务网关应用下的部署组时，可以选择是否将网关部署在全局命名空间中。不同命名空间与微服务网关的关系如下：

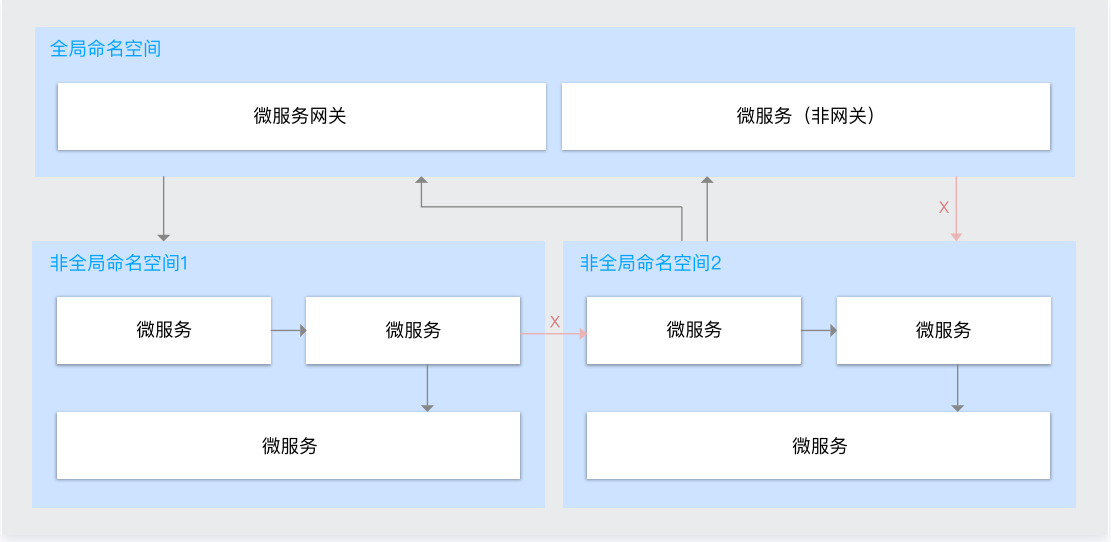
| 部署位置 | 微服务网关调用其他微服务 | 其他微服务调用微服务网关 |
| --- | --- | --- |
| 全局命名空间 | 微服务网关可以调用所有命名空间中的微服务 | 微服务网关可以被所有微服务调用 |
| 非全局命名空间 | 微服务网关仅可以调用所在命名空间下的微服务 | 可以被所在命名空间中的微服务调用（常见）或被全局命名空间中的网关调用（少见） |

示意图如下：

- 非全局命名空间中的微服务网关：




- 全局命名空间中的微服务网关：




您可以通过全局命名空间中的微服务网关达到跨命名空间访问的目的。非全局命名空间中的微服务调用全局命名空间中的微服务网关调用另一个非全局命名空间中的微服务。此时，全局命名空间中的网关就是不同非全局命名空间中微服务调用的桥梁。在实际的业务场景中，您可以将不同部门的业务部署在不同的命名空间中，通过全局命名空间的微服务网关来进行跨部门的调用。

## 调用方法

### 网关调用微服务

微服务网关调用微服务时，您可以通过网关托管 API 的路径进行调用。访问路径如下：
`IP 或域名/分组 context/命名空间名/微服务名/API 路径` 。访问路径规则与网关是否部署在全局命名空间中无关，但当微服务网关部署在非全局命名空间中，访问路径中却填写了错误命名空间名，则访问不通。

### 微服务调用网关

您可以通过网关的服务名调用网关。当非全局命名空间 namespace1 中的微服务 consumer 希望通过全局命名空间中的网关 zuul 访问 namespace2 中的微服务 provider 时，可以在 consumer 中直接访问微服务 zuul，访问路径为 `/context/namespace2/provider/api-path` 。

# 部署 Java 网关

最近更新时间：2025-03-10 18:08:33

## 操作场景

部署Java网关相当于部署一个微服务，本文以 TSF 提供的 官方 Demo 为例介绍在 TSF 平台上部署 Java网关的操作方法。

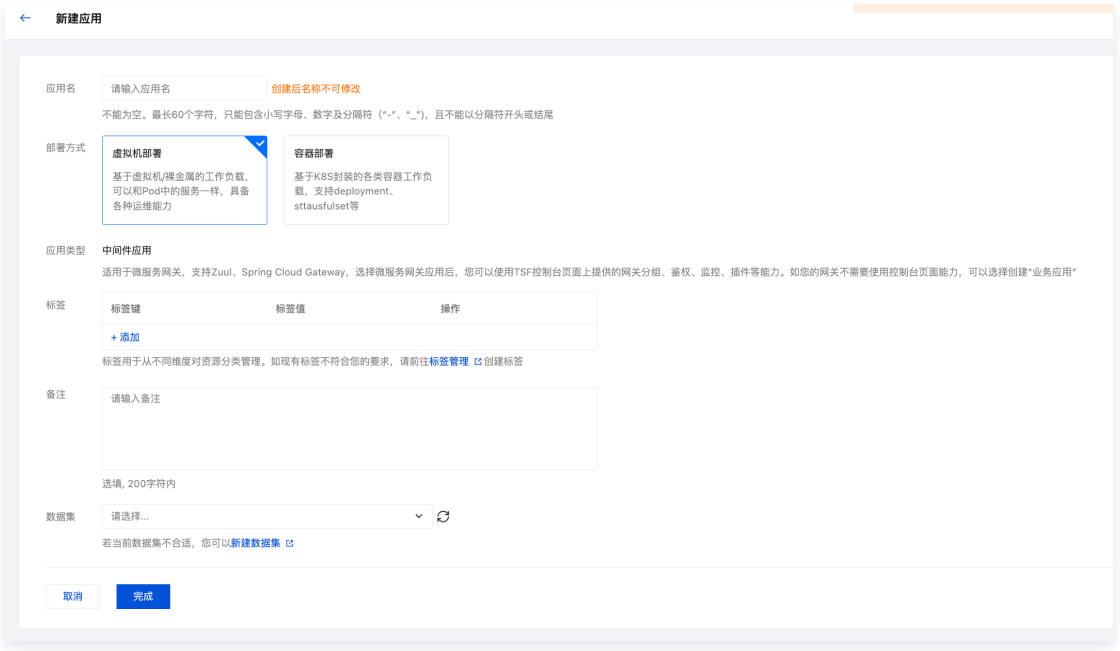
- 微服务网关兼具灵活性和便捷性。当用户仅需要使用请求转发、密钥对鉴权、服务治理等 TSF 已经提供的产品能力时，您仅需要使用 官方 Demo 中的 msgw - demo 程序包即可，可以选择其中的 msgw - zuul 以及 msgw - scg 来分别部署基于 Zuul 和基于 Spring Cloud Gateway 的微服务网关。
- 当您希望定制化微服务网关的转发逻辑，您可以选择自行开发微服务网关，并依赖 TSF 的网关 SDK 来进行开发。详情参见 微服务网关开发指南 。
- 直接使用开源的微服务网关（Demo 中的 opensource 版本）会导致 TSF 产品页面中的网关功能不可用。不建议您使用开源版本的微服务网关。

## 前提条件

1. 已创建好集群并导入云主机。
   - 虚拟机场景请参见 虚拟机集群管理 。
   - 容器场景请参见 容器集群管理 。
2. 下载 官方 Demo 并解压后，在pom.xml文件所在目录下执行 `mvn clean package` 将应用程序打包，在 `msgs-demo/target/` 目录下可以看到打包好的 jar 程序包。

## 操作步骤

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择**应用管理**，选择中间件应用，选择好地域后在**网关应用**列表单击**新建**，填写应用基本信息。




   - 应用名：填写应用名称
   - 部署方式：选择虚拟机部署或者容器部署方式，本文以容器部署为例。
   - 标签：用于分类管理资源，可不选。详情参见 标签 。
   - 备注：选填，可留空。
   - 数据集：用户可以通过数据集管理配置不同的子账号和协作者使用不同资源的权限，详情参阅 数据集管理 。
3. 单击**完成**，在弹窗中选择**导入程序包**，跳转至上传程序包页面。如果选择**关闭**，您后续可以在应用详情页面选择**制品**页签，重新上传程序包。

4. 在程序包管理页面，单击**上传程序包**，选择提前准备好的 jar 程序包。




5. 单击**提交**，等待程序包上传完成后，点击**应用部署**，进入部署页面。
6. 填写部署组信息。




- 集群：选择应用要部署所在的集群。
- 命名空间：选择应用要部署所在的命名空间。部署组名称：填写部署组名称，不超过60个字符，只能包含字母、数字及分隔符(“-”)，且必须以字母开头，数字或字母结尾。
- 部署组名称：填写部署组名称。
- 标签：用于分类管理资源，可不选。详情参见 标签 。
- 备注：选填，可留空。

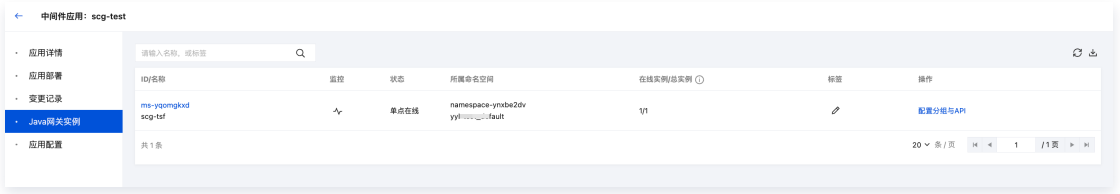
7. 单击**下一步**，进入部署配置，设置部署信息。

- 选择实例：选择用于部署的云主机。
- 软件仓库：选择**默认仓库**。
- 程序包类型：选择 **jar**。
- JDK 版本：选择**KONA JDK8**。
- 程序包类型：选择提前上传好的程序包。
- 启动参数JAVA_OPT：选填。
- 健康检查：可选。详情参见 健康检查 。
- 描述：可选。

8. 点击**下一步**，进入发布策略配置。
9. 点击**下一步**，进入服务注册配置和可观测配置。

> **注意：**
> - 当注册配置治理中心使用 TSF-Consul 时，如果微服务网关部署在全局命名空间中时，该微服务网关可以将请求路由转发到全部命名空间的微服务上。
> - 当微服务网关部署在非全局命名空间时，该微服务网关只能转发当前命名空间下的请求。即：**非全局命名空间中的微服务网关，无法穿透命名空间的逻辑隔离**。

10. 点击**下一步**，进行信息确认。
11. 单击**完成**，部署成功后，回到应用详情页。
12. 在左侧菜单栏选择 **Java网关实例**，可以看到刚部署好的网关。



# 配置分组与 API

最近更新时间：2025-03-10 18:08:33

## 操作场景

微服务网关通过分组管理微服务 API。分组是微服务网关管理 API 的维度，同一个分组下的 API 使用相同的鉴权方法，使用相同的一个或者多个密钥进行访问鉴权。每一个分组有一个固定的 context 作为访问路径中的 path 参数。

> **说明：**
> 仅专业版和铂金版支持使用 Java 网关，具体说明详细参见 版本对比 。

## 前提条件

已部署好微服务网关 。

## 新建分组

1. 登录 腾讯微服务平台控制台 。
2. 在左侧菜单栏选择**应用管理 > 中间件应用**，进入中间件应用列表页面。
3. 选择好地域后，单击目标应用 ID，进入应用详情页。
4. 单击 **Java网关实例**菜单栏，查看网关实例列表。

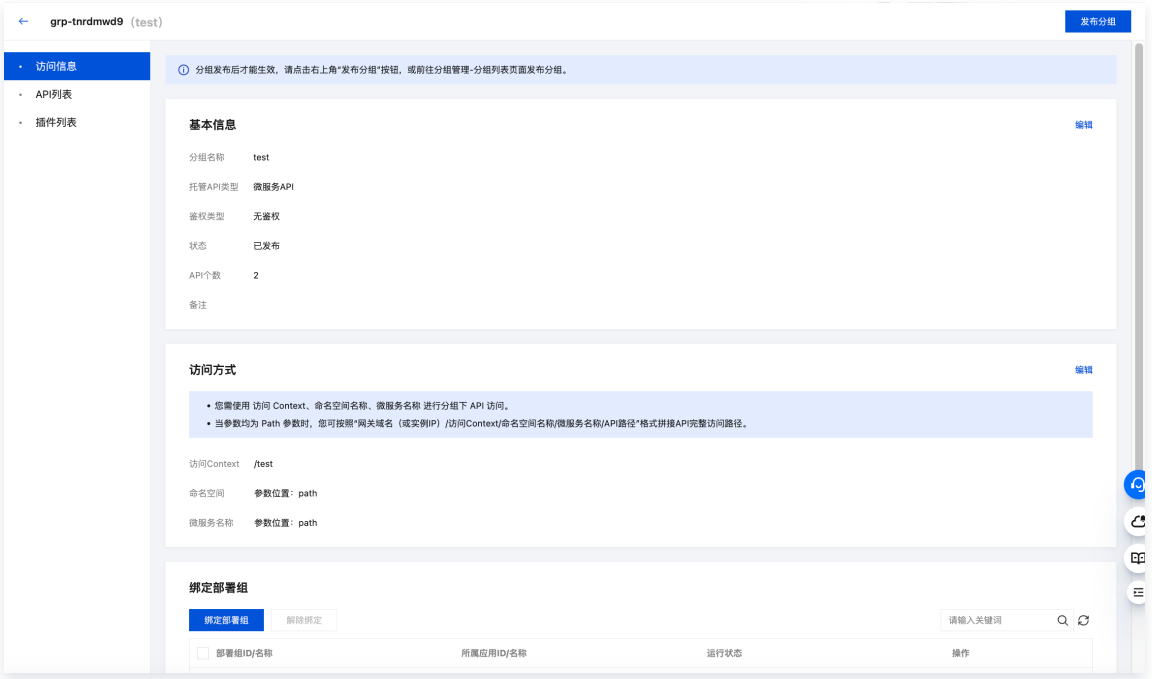



5. 单击目标实例操作栏的**配置分组与 API**，进入分组管理页面。
6. 单击**新建分组**，填写分组信息。
   - 分组名称： 最长为60个字符，只能包含小写字母、数字及分隔符（ `_` 、 `-` ），且不能以分隔符开头或结尾。
   - 访问 context：调用分组API所需参数，调用时需在访问路径中填写Context。
   - 托管 API 类型：当前微服务网关可以托管两种不同类型的 API。
     - 微服务 API：即部署在 TSF 上的微服务的 API。
     - 外部 API：即后端服务未部署在 TSF 上或未注册到 TSF 的注册中心的服务的 API。托管外部 API 时，需要用户手动填写后端服务的 Host 地址。
   - 鉴权类型：密钥鉴权或无鉴权。当选择密钥对鉴权时，请求参数中不带正确的 SecretId 和签名的访问会被拒绝。
   - 访问参数设置：
     - 命名空间：调用分组API所需参数，支持 Path，Header，Query 三种参数位置，调用时需在指定位置上填写命名空间名称。
     - 微服务名称：调用分组API所需参数，支持 Path，Header，Query 三种参数位置，调用时需在指定位置上填写微服务名称。
7. 单击**保存&下一步**，将分组与微服务网关应用的部署组进行绑定。

> **注意：**
> - 当微服务网关作为微服务内部调用网关时，建议绑定微服务下全部部署组。当微服务网关作为外部与微服务间调用的外部网关时，可以选择将某个分组与某个或全部部署组绑定，通过访问部署组域名或 IP 来访问网关托管的 API。
> - 只有将网关应用的部署组与分组绑定后，才能保证访问部署组 IP 后与对应 API path 访问畅通。
> - 此处分组绑定的部署组是微服务网关应用的部署组，也就是用作部署网关的节点所在的部署组。并非后端微服务的部署组。

- 同一个微服务网关部署组请使用不同的分组 context。

8. 新建分组成功后，单击分组的“ID”，进入访问信息页，可查看分组的基本信息，访问方式，并对绑定部署组和密钥进行操作管理。




## 新建 API

### 微服务 API

微服务 API 即部署在 TSF 上的微服务的 API。

1. 选择需要进行 API 配置的分组，进入分组详情页。
2. 在分组详情页，选择 **API 列表** 页签，单击**新建 API**。
   - 导入类型：支持3种导入方式。
     - 导入微服务 API：导入某个微服务当下的全部 API，如果后续该服务有新增的 API 需要重新进行导入。
     - 导入部分 API：导入某个微服务下部分 API。
     - 手动填写 API：用于对某服务下的所有接口进行放通或者按照某种通配规则进行放通，适用于API 频繁变动的场景，例如开发联调等。需选择请求方法并填写请求路径。请求路径不能为空，最长128字符，支持"/"开头，大小写字母、数字、和$-_.+!'(),/%等符合 URL 规则的符号，支持*（匹配0或者任意数量的字符）**（匹配0或者更多的目录）通配多个路径，Path 类型的请求参数必须以包裹，作为独立部分包含在路径中（示例：/{param}/）。




- 命名空间：选择微服务所在的命名空间。
- 微服务：选择需要导入 API 的微服务

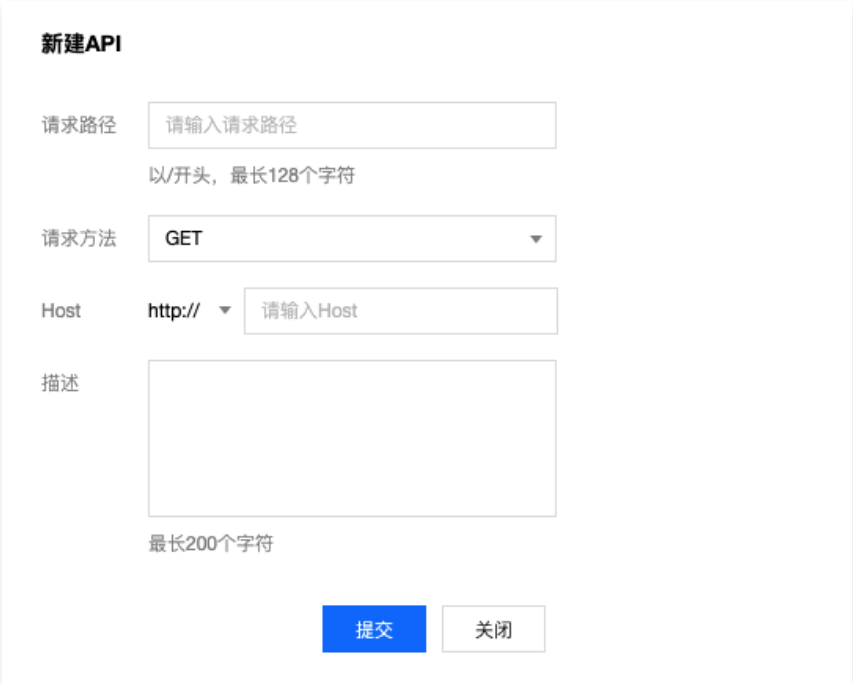
3. 单击**确认**，等待导入完成后，可以在 API 列表看到导入的 API。

> **说明：**
> - 当同时导入某个微服务下全部 API 时，后台导入完成前请勿重复导入。
> - 单部署组最多支持创建2000个API，超出后不保证发布成功。

## 外部 API

即后端服务未部署在 TSF 上或未注册到 TSF 的注册中心的服务的 API。托管外部 API 时，需要用户手动填写后端服务的 Host 地址。

1. 在分组详情页，选择 **API 管理** 页签，选择好分组后单击**新建 API**。
2. 填写请求路径、请求方法以及 host 地址。



3. 单击**提交**，完成导入。

## 发布/下线分组

**当且仅当分组被发布时，才能通过微服务网关访问微服务 API。**

- 导入 API 后，单击分组操作列的**发布分组**，该分组即可被发布。
- 如果某一个分组不想被访问，可以单击操作列的**下线分组**，将分组下线。

## 访问微服务 API

分组下某个微服务 API 的访问路径为：网关的域名或网关的 `IP + port/API` 访问路径 。API 完整访问路径拼接规则如下：

| 访问 context | 命名空间参数位置 | 微服务名称参数位置 | API 访问路径 |
| --- | --- | --- | --- |
| path | path | path | /context/命名空间名称/微服务名称/api 路径 |
| path | header/query | path | /context/微服务名称/apipath |
| path | header/query | header/query | /context/apipath |

**举例：**

当参数均为 Path 参数时，且分组 context 为 sell，命名空间为 test-env，微服务名称为 consumer，API 路径为 `/echo/slow/{param}` 时，则访问路径为 域名`/sell/test-env/consumer/echo/slow/{param}` 。
在 API 列表页面单击 API 路径后面的复制按钮，会自动复制从 context 开始的路径。




> **说明：**
> API 访问路径不支持中文，如访问路径包含中文，建议修改后重试。

## 访问外部 API

访问外部 API 的路径为：
微服务网关域名或 `VIP` 或服务名（仅当客户端为注册到 `TSF` 中的微服务时可以通过网关服务名访问）/ 分组 `context` / `API` 路径 。

> **说明：**
> API 访问路径不支持中文，如访问路径包含中文，建议修改后重试。

## 微服务网关域名

在用户公有云场景下，需要用户将部署网关的节点的 IP 配置在 DNS 或 LB 上，以获取统一的访问网关的统一域名或统一 VIP。

- 对于虚拟机部署组，用户可以将虚拟机部署组的机器 IP 绑定在 CLB 后端。
- 对于容器部署组，用户可以设置容器部署组的访问方式为公网访问。

# API 管理
# 配置鉴权

最近更新时间：2025-03-10 18:08:33

## 操作背景

该任务指导您通过TSF控制台，使用密钥对对访问进行鉴权。

> **说明：**
> 仅专业版和铂金版支持使用 Java 网关，具体说明详细参见 版本对比 。

## 操作步骤

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择**应用管理** > **中间件应用**，单击目标 Java 网关应用 ID，进入详情页，选择**Java网关实例**。
3. 选择目标 Java 网关实例，单击操作列的**配置分组与API**，进入**分组管理**页面。
4. 单击目标分组的**ID/名称**，进入分组详情页。
5. 在基本信息模块，确认鉴权类型为“密钥鉴权”。如不是，可单击模块右上角的**编辑**进行修改。
6. 在密钥管理模块，单击**新建密钥**，填写密钥名。
7. 密钥名最长50个字符，支持 `a-z` 、 `A-Z` 、 `0-9` 、 `_` 。
8. 单击**确认**，会自动生成密钥对，并展示在控制台上。
   - 您可以通过密钥末尾的复制图标进行复制。
   - 当密钥需要禁用和更换时，可以单击操作列的**禁用**或**更换**完成操作。
   - 有关如何计算密钥对签名的请求头，请参见 微服务网关密钥对鉴权 文档。

版权所有：腾讯云计算（北京）有限责任公司

# 设置 API 超时时间

最近更新时间：2025-03-10 18:08:33

## 操作场景

该任务指导您通过腾讯微服务平台控制台，为微服务网关托管的微服务 API 配置超时时间。

> **说明：**
> 仅专业版和铂金版支持使用 Java 网关，具体说明详细参见 版本对比 。

## 操作步骤

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择**应用管理** > **中间件应用**，单击目标Java网关应用，进入详情页，选择**Java网关实例**菜单栏。
3. 选择目标Java网关实例，单击操作列的**配置分组与API**，进入**分组管理**页面。
4. 选择一个需要调整 API 超时时间的分组，单击分组 ID，进入详情页的 **API 列表**页面。
5. 设置超时时间（0 < t ⩽ 600000ms，其中 t 为正整数），单击**提交**。
   - 方式一：选择一个 API，单击操作列的**更多** > **设置超时时间**，为 API 配置超时时间。
   - 方式二：通过页面上的**设置超时时间**按钮，批量为 API 配置超时时间。

   > **说明：**
   > - 若未对某个 API 配置超时时间，则使用微服务网关程序包中 application.yaml 文件中配置的超时时间。
   > - 当前提供的微服务网关 Demo 包的默认超时时间为10s。

版权所有：腾讯云计算（北京）有限责任公司

# 设置 API 限流规则

最近更新时间：2025-03-10 18:08:33

## 操作背景

该任务指导您通过腾讯微服务平台控制台，为网关下的每一个 API 配置限流规则设置最大请求次数，即限流阈值。如果请求数达到了限流阈值，任何到达的请求都会限流模块处理。如果该服务上的配额已经消耗完，会对请求返回 HTTP 429（Too Many Requests）；否则会正常放行。

> **说明：**
> 仅专业版和铂金版支持使用 Java 网关，具体说明详细参见 版本对比 。

## 操作步骤

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择**应用管理** > **中间件应用**，单击目标Java网关应用，进入详情页，选择**Java网关实例**菜单栏。
3. 选择目标Java网关实例，单击操作列的**配置分组与API**，进入**分组管理**页面。
4. 在分组详情页，单击 **API 列表**，进入API 列表页面，选择目标API操作栏的**更多** > **编辑限流规则**。
5. 在设置限流规则对话框中，选择限流策略为“设置最大请求数”，并填写最大请求次数（QPS）。

> **说明：**
> 配置了限流规则后，可以在配置 API 限流规则的微服务的服务限流列表中，找到该限流规则。

# 路径重写配置

最近更新时间：2025-03-10 18:08:33

## 操作场景

在很多场景下，为了不将内部接口信息过分暴露或为了保证上下游系统接口的一致性，需要对微服务对外暴露的接口配置新的请求路径，如内部接口写为 `/group1/prepub/provider/echo`，我们可以配置重定向接口为 `/echo`，这样所有通过对应网关访问 `/echo` 这个接口的调用都会被指向到后端 `/group1/prepub/provider/echo` 这个接口上。

> **说明：**
> 仅专业版和铂金版支持使用 Java 网关，具体说明详细参见 版本对比 。

## 操作步骤

### 新建路径重写配置

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择**应用管理 > 中间件应用**，单击目标Java网关应用，进入详情页，选择**Java网关实例**菜单栏。
3. 选择目标Java网关实例，在页面左侧选择**路径重写配置**页签，选择好网关部署组后，单击**新建路径重写配置**，填写以下内容：




- 请求路径：请求路径指的是外部请求访问到网关的路径。填写时要以 / 开头。
- 匹配顺序：当对一个微服务网关上配置多个路径重写规则的时候，可能会出现多个规则最终落在同一个实际的微服务 API 上的情况，此时可以通过匹配顺序指定规则之间的前后匹配顺序。匹配顺序值为1 – 2000之间正整数，1为最高优先级。匹配顺序允许相同，匹配顺序相同时按照规则创建时间顺序匹配。
- 原路径：指的是微服务实际对外暴露的路径，微服务开发者代码中定义的路径。
- 类别：在用户填写原路径配置时，可以通过类别筛选重定向规则匹配一个具体的 API 还是多个通配 API。
- 屏蔽原路径：屏蔽原有路径的访问，屏蔽后，微服务 API 只能通过配置后的请求路径访问。

### 相关说明

1. 配置路径重写规则前，请先确认原路径 API 所在的分组已经与页面上方选中的微服务网关部署组绑定。
2. 路径匹配规则支持正则表达式，详情如下：

| 正则表达式 | 说明 |
| --- | --- |
| . | 匹配除换行符以外的任意字符 |
| ? | 重复0次或1次 |
| + | 重复1次或更多次 |
| * | 重复0次或更多次 |
| \d | 匹配数字 |
| ^ | 匹配字符串的开始 |
| $ | 匹配字符串的介绍 |
| {n} | 重复 n 次 |
| {n,} | 重复 n 次或更多次 |
| [c] | 匹配单个字符 c |
| [a-z] | 匹配 a - z 小写字母的任意一个 |

**说明：**
小括号()之间匹配的内容，可以在后面通过$1来引用，$2表示的是前面第二个()里的内容。

### 使用场景举例

1. 当前 TSF 微服务网关提供的默认对外访问路径为 `域名/分组名/命名空间/微服务名称/API路径` ，可以通过路径重写配置支持将较长的路径映射为短路径：
   - 请求路径填写为： `/provider/(.*)`
   - 原路径填写为： `/groupone/defaultnamespace/provider/$1`
     此时，通过网关直接请求 `/provider/echo` 等请求会直接被转发到 `/groupone/defaultnamespace/provider/echo`
2. 可以通过路径重写配置为某一个 API 配置路径别名，如：
   - 请求路径填写为： `/tsf/config`
   - 原路径填写为： `/qcloudadmin/middleware/tsf/dispatch/config`

## 复制路径重写配置

复制路径重写配置支持将某一个网关下的配置快速复制到另一个微服务网关部署组下面。

1. 在路径重写配置列表，勾选已经配置好的规则，单击列表上方的**复制到**，在弹窗中选择目标网关的部署组。
2. 单击**确定**，完成复制。

## 导入导出路径重写规则

根据模板填写好配置规则后，可直接上传导入控制台，同时也可将控制台中的配置导出到本地。

- 导入：在路径重写配置规则列表页面，点击左上角的导入，选择提前准备好的配置文件（支持 xlsx、xls、csv 格式。最大允许上传 50 KB），确认后即可上传。
- 导出：在路径重写配置规则列表页面，在右上角点击下载按钮，可将控制台中的配置导出到本地，导出格式为xlsx。

# 网关监控

最近更新时间：2025-03-10 18:08:33

## 操作场景

微服务平台对运行在 TSF 上的网关微服务提供了多项监控指标，用以监测网关节点及服务的运行状况。您可以根据这些指标实时了解网关微服务的运行状况，针对可能存在的风险及时处理，保障系统的稳定运行。本文为您介绍通过 TSF 控制台查看网关监控的操作。

## 监控入口

1. 登录 腾讯微服务平台控制台 。
2. 在左侧菜单栏选择**观测中心 > 中间件应用监控**。
3. 选择好地域、网关微服务和部署组之后，可以查看网关监控信息。

> **说明：**
> 当前中间件应用监控仅支持对接到 TSF-Consul 上的 Java 网关应用监控。

## 网关监控

网关监控用于查看网关相关的指标。设置好时间范围（支持近1小时、近3天、近7天和自定义时间范围），可查看对应的监控曲线。
网关监控支持监控以下指标：

- 网关请求数 (次)
- 请求总耗时 (min)：请求进入网关到出网关总耗时
- 错误数 (次)
- 错误率（%）
- 4XX响应数 (次)
- 5XX响应数 (次)

## 节点监控

节点监控展示所选部署组下各实例节点的监控数据，支持展示以下数据：

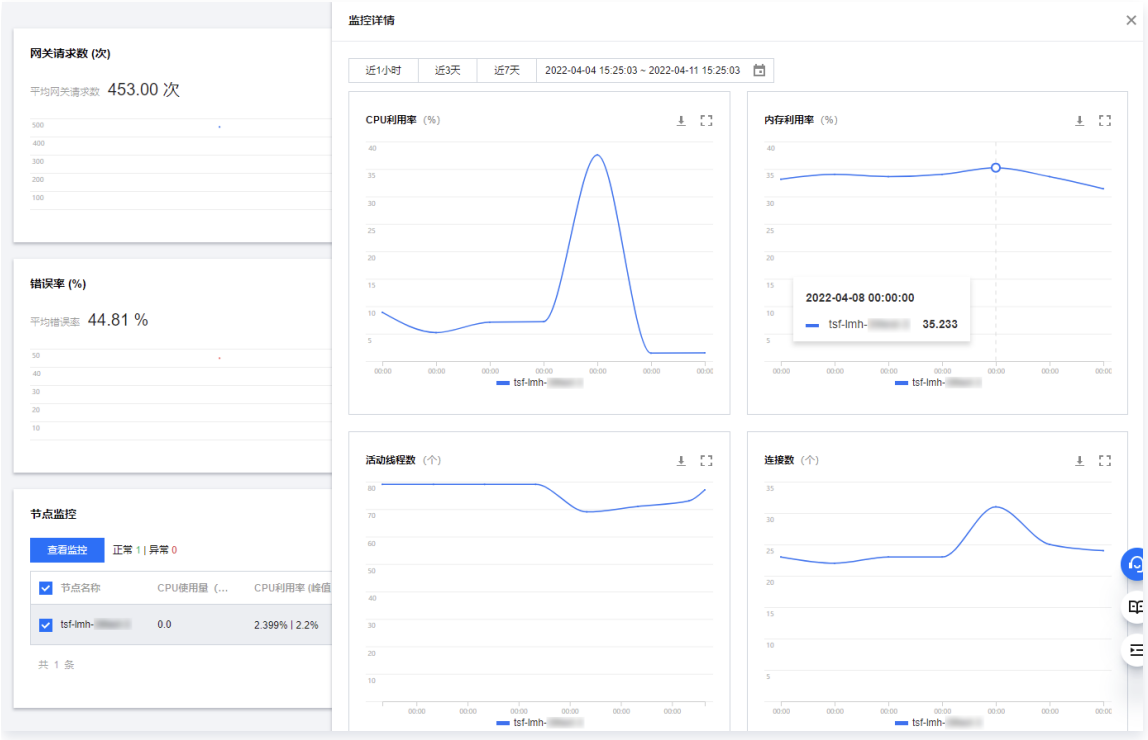
- CPU 使用量
- CPU 利用率（峰值/当前值）
- 内存利用率（峰值/当前值）
- 活动线程数
- 连接数
- 入流量
- NIO 线程数
- 守护线程数
- 总线程数




勾选要查看的节点，单击左上角**查看监控**，可查看对应节点详细的监控数据曲线，展示的指标如下：

- CPU 利用率

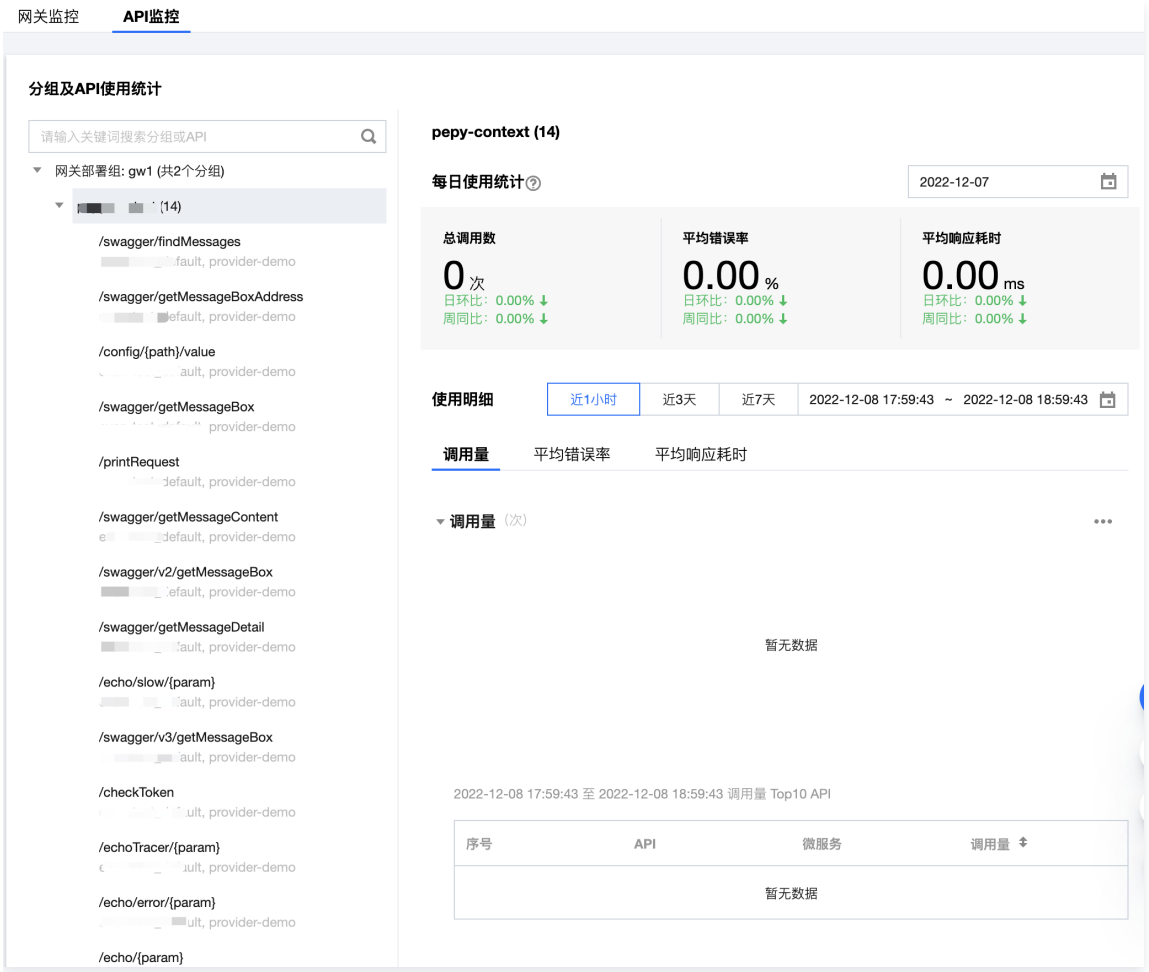
- 内存利用率
- CPU 核数
- 网络 IO
- 活动线程数
- 连接数
- 入流量
- nio 线程数
- boss 线程数
- work 线程数
- 业务线程数




## API 监控

选择API 监控页签，可查看所选网关下各分组与 API 节点的使用统计数据，包含调用数、平均错误率和平均响应耗时。

- 针对外部 API 调用的监控可以在 /external 分组中查询
- 针对单元化场景下跨命名空间的调用监控可以在 /crossunit 分组中查询


网关监控
API监控
分组及API使用统计
请输入关键词搜索分组或API
网关部署组: gw1 (共2个分组)
(14)
/swagger/findMessages
/swagger/getMessageBoxAddress
/config/{path}/value
/swagger/getMessageBox
/printRequest
/swagger/getMessageContent
/swagger/v2/getMessageBox
/swagger/getMessageDetail
/echo/slow/{param}
/swagger/v3/getMessageBox
/checkToken
/echoTracer/{param}
/echo/error/{param}
/echo/{param}
pepy-context (14)
每日使用统计
2022-12-07
总调用数
0 次
日环比: 0.00%
周同比: 0.00%
平均错误率
0.00 %
日环比: 0.00%
周同比: 0.00%
平均响应耗时
0.00 ms
日环比: 0.00%
周同比: 0.00%
使用明细
近1小时
近3天
近7天
2022-12-08 17:59:43 ~ 2022-12-08 18:59:43
调用量
平均错误率
平均响应耗时
调用量 (次)
暂无数据
2022-12-08 17:59:43 至 2022-12-08 18:59:43 调用量 Top10 API
序号
API
微服务
调用量
暂无数据

# 插件管理
# 配置 JWT 插件

最近更新时间：2025-03-10 18:08:33

## JWT 原理

JWT（JSON Web Token）本质是一个 Token，是一种紧凑的 URL 安全方法，用于在网络通信的双方之间传递声明。
JWT 的原理是，客户端通过 JWT 认证服务器认证以后，会返回给客户端一个 JWT 令牌（Token），示例如下（真实长度会更长）：

```
eyJraWQiOiJraWQiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJJc3N1ZXIiLCJhdW.iwMquJ1Vp-oo6P-sqY5Yg
HEADER                                PAYLOAD                SIGNATURE
```

JWT 分为三部分：Header（头部）、Payload（负载）、Signature（签名），中间用点（ `.` ）分隔成三个部分。
此后，客户端与服务端通信的时候，都要携带这个 JWT 令牌（Token）。微服务网关 JWT 插件凭此令牌（Token）来校验客户端权限，服务端就不再保存任何 session 数据，此时服务端变成无状态了，比较容易实现横向扩展。

## Header

Header 部分是一个 JSON 对象，描述 JWT 的元数据，示例如下：

```
{
   "alg": "HS256",
   "typ": "JWT"
}
```

- `alg` 属性表示签名的算法（algorithm），默认是 HMAC SHA256（写成 HS256）。
- `typ` 属性表示这个令牌（token）的类型（type），JWT 令牌统一写为 `JWT` 。

最后，将上面的 JSON 对象使用 Base64URL 算法转成字符串，即为 JWT 令牌中的 Header 部分。

## Payload

Payload 部分也是一个 JSON 对象，用来存放实际需要传递的数据，包括官方定义的字段和用户自定义字段。

| 参数 | 英文全称 | 是否必选 | 说明 | 取值要求 |
|---|---|---|---|---|
| iss | Issuer Identifier | 是 | 提供认证信息者的唯一标识。 | 一般是一个 HTTPS 的 URL（不包含 querystring 和 fragment 部分） |
| sub | Subject Identifier | 是 | iss 提供的 EU 的标识，在 iss 范围内唯一。它会被 RP 用来标识唯一 的用户。 | 最长为255个 ASCII 字符 |
| aud | Audience(s) | 是 | 标识 ID Token 的受众。 | 必须包含 OAuth2 的 client_id |
| exp | Expiration time | 是 | 过期时间，超过此时间的 ID Token 会作废不再被验证通过。 | – |
| iat | Issued At Time | 是 | JWT 构建的时间。 | – |
| auth_time | AuthenticationTime | 否 | EU 完成认证的时间 。如果 RP 发送 AuthN 请求的时候携带 max_age 的参数，则此 Claim 是必须的。 | – |

| | | | | |
|---|---|---|---|---|
| nonce | - | 否 | RP 发送请求的时候提供的随机字符串，用来减缓重放攻击，也可以来关联 ID Token 和 RP 本身的 Session 信息。 | - |
| acr | Authentication Context Class Reference | 否 | 表示一个认证上下文引用值，可以用来标识认证上下文类。 | - |
| amr | Authentication Methods References | 否 | 表示一组认证方法。 | - |
| azp | Authorized party | 否 | 结合 aud 使用。只有在被认证的一方和受众（sud）不一致时才使用此值，一般情况下很少使用。 | - |

下面是一个典型数据格式的示例，供参见：

```
{
    "iss": "https://cloud.tencent.com/tsf/msgw/jwt",
    "sub": "aaaaaaaa-bbbb-cccc-dddd-example",
    "aud": "tsf",
    "exp": 1500013000,
    "iat": 1500009400,
    "auth_time": 1500009400,
    "nonce": "x-03_si1h4t",
    "acr": "1",
    "azp": "tsf",
    "given_name": "Anaya",
    "email": "anaya@example.com"
}
```

最后，将上面的 JSON 对象使用 Base64URL 算法转成字符串，即为 JWT 令牌中的 Payload 部分。

### Signature

Signature 部分是对前两部分的签名，防止数据篡改。
生成 JWT 令牌（token）需要私钥，您可以在站点 https://mkjwk.org/ 生成与验证的公钥和私钥。使用 Header 里面指定的签名算法（默认是 HMAC SHA256），按照下面的公式产生签名。

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

算出签名以后，把 Header、Payload、Signature 三个部分拼成一个字符串，每个部分之间用"点"（`.`）分隔，即为JWT 令牌（token），最后返回客户端。

## 使用 Java 生成 JWT 令牌（Token）

1. 添加 JWT 的 pom 依赖。

   微服务网关使用 jose4j 来实现。在 pom 文件中添加以下依赖：

```
<!-- jwt -->
<dependency>
    <groupId>org.bitbucket.b_c</groupId>
    <artifactId>jose4j</artifactId>
    <version>0.6.5</version>
</dependency>
```

2. 编写生成令牌的 Java 代码（以下以 RSA 为例）。

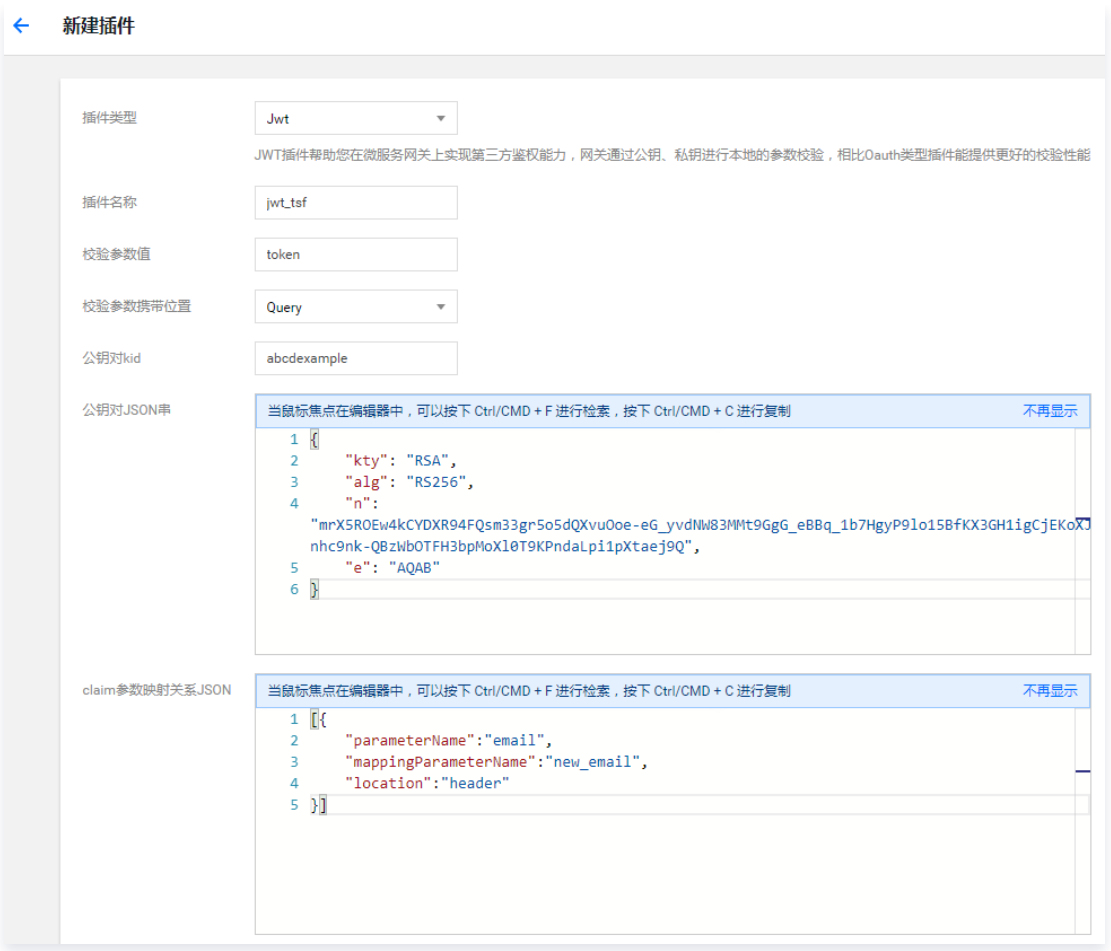
```
// 下面省略了无关代码
import org.jose4j.json.JsonUtil;
import org.jose4j.jwa.AlgorithmConstraints;
import org.jose4j.jwa.AlgorithmConstraints.ConstraintType;
import org.jose4j.jwk.PublicJsonWebKey;
import org.jose4j.jwk.RsaJsonWebKey;
import org.jose4j.jws.JsonWebSignature;
import org.jose4j.jwt.JwtClaims;
import org.jose4j.jwt.MalformedClaimException;
import org.jose4j.jwt.consumer.InvalidJwtException;
import org.jose4j.jwt.consumer.JwtConsumer;
import org.jose4j.jwt.consumer.JwtConsumerBuilder;
import org.jose4j.lang.JoseException;
public static void main(String[] args) throws JoseException, MalformedClaimException {
        // 通过私钥对生成jwk
        RsaJsonWebKey  jwk = new RsaJsonWebKey(JsonUtil.parseJson("{"
            + "    \"kty\": \"RSA\","
            + "    \"alg\": \"RS256\","
            + "    \"n\": \"mrX5ROEw4kCYDXR94FQsm33gr5o5dQXvuOoe-
eG_yvdNW83MMt9GgG_eBBq_1b7HgyP9lo15BfKX3GH1igCjEKoXJxcHC5xox4xC0tvbBNCDwG_987ZsqlgCb4f7X66DCc
Hh17AyAHYa8JhO2kXvtD1OIQxSajSgmk1C1sxI5kqJXfvJwRLcCEK87P6Bs9YNLnnJeouSkYce04AhspmyQKKax4GllbM
jcrUUMRoqpCBMklh5Pgl9sOGRLo-6uzzowtI_SyF03YsE2ejh9m-YWqTYsx7PIg6SdWrNRIprKtjnhc9nk-
QBzWbOTFH3bpMoXl0T9KPndaLpi1pXtaej9Q\","
            + "    \"e\": \"AQAB\","
            + "    \"d\":
\"lL9vqdUl7fMS_qTJPf1QYjPV6qBKrAQIJ28aV0DA6YF6pFCrCyJ3I5frC2E4nmbuZl0dPTpKaPiFIAQjUwsnvSb8Wb4
fLP-2El3-BcQSwX9FnalPrpnvwpwZw2gnvSgJn0EFRh6HBMCJSFf4QI7LWC01SDsTpj9xRsoQAHurf5YZ8YRpi_-
XEWBaL-4R_RxpoeAnSgSbJkcGoJNcxqwWbCun37KYBS_71sd155VWycMd-uHtTRnW6SenVG49pexXq-
tIQxOwmatpTj0XF7hhshKgdTF1xXPbMSX6XOvxiy929jPMercBL_-OUu0PPUZTTVp0gNziGdevzufHkEfHxQ\","
            + "    \"p\": \"5dOs8Q0SHIuCq-
gAOx2c2JaqXzPRsmmZ7nXx1P1jbddDIenVPA6q5zUVqXIRRQgMA7AdD9x7anJ2_kaKSFoE2D8peuObvmjrbmJeYE4F413
8pNESOHlBmhUH93Oo4i5TvmNZ5hxu3CmonGKMafeDoMpopN15yeDGkTVFKoOfuys\","
            + "    \"q\": \"rFRhmJIIj3o__CbjVOlUgiVzrk-
ZgK9jGXHhyt6LELQae1nUiZNZuwZeHwgzTonsTMJ2JHnmCuDpwuhjf_kha5KHeuczE7gmnlaGd3s6kaKDyB9bXbMTs122
SnNiB-lJcwm4wRNWI-irSh8PyHSQVnjvKkQCCEPi4i0Ky1KgDV8\","
            + "    \"dp\": \"qBUJNDn09v9pD8Ra9uEPZq-55mqFgFAPDgEgXj76yshWBqV3F7c6cmG2d_g-
fRgHgWL5vjHn6M_SCuEYHRYI2QZIleGEc9tT46T5lME7OS_xp7Bn_PlhawjajLT_3Hs5L9KFWu-
MfGPTNpw0SQOGNsARjBGWEnjbgDNPZGpjFYU\","
            + "    \"dq\": \"F_0rFNUHUgm_jHdRYAmXFQLnppU4FhzUG7-podb23t1jblZj6r7TV-
CcC4_VrJIwjcLoNU2uw0bp45L7_t2MVHAyYd57Urxoy9PZphpGXe2UkLAkxNdf37Ek5hpHxDgqXFQ3HtF1RUxnQ8stJEd
trEvrZyPOcJ4aoEeK4CDhXNs\","
            + "    \"qi\": \"QdwkKs9n6jswVsbYKprvpNr2Mbg-RBPp5xx1p-ypVJnFOd_lnCA6P3gRJ-
pe6tSCIB6AYViEhZpzKMSu47f27_38VHkH9qOOL38ZVeVFo8Yt8lkBwMEKQdDOghfF74L5Fczo9bH7QX679dC847cDEa1
oaV2Cdv6XcSKGywwvq3Y\""
            + "}"));
```

```
        // 创建Claims，放在JWT payload位置
        // Create the Claims, which will be the content of the JWT
        JwtClaims claims = new JwtClaims();
        claims.setIssuer("Issuer");  // who creates the token and signs it
        claims.setAudience("Audience"); // to whom the token is intended to be sent
        claims.setExpirationTimeMinutesInTheFuture(10); // time when the token will expire
(10 minutes from now)
        claims.setGeneratedJwtId(); // a unique identifier for the token
        claims.setIssuedAtToNow();  // when the token was issued/created (now)
        claims.setNotBeforeMinutesInThePast(2); // time before which the token is not yet
valid (2 minutes ago)
        claims.setSubject("subject"); // the subject/principal is whom the token is about
        claims.setClaim("email","mail@example.com"); // additional claims/attributes about
the subject can be added
        List<String> groups = Arrays.asList("group-one", "other-group", "group-three");
        claims.setStringListClaim("groups", groups); // multi-valued claims work too and will
end up as a JSON array

        JsonWebSignature jws = new JsonWebSignature();

        jws.setPayload(claims.toJson());
        jws.setKey(jwk.getPrivateKey());
        jws.setKeyIdHeaderValue("kid");
        jws.setAlgorithmHeaderValue(jwk.getAlgorithm());

        // 生成JWT令牌
        String jwt = jws.getCompactSerialization();

        long expirationTime = claims.getExpirationTime().getValueInMillis();
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss z",
Locale.SIMPLIFIED_CHINESE);
        String msg = String.format("JWT expired at (%d -> %s)", expirationTime,
dateFormat.format(expirationTime));

        // 打印JWT 令牌过期时间
        System.out.println(msg);
        // 打印JWT 令牌
        System.out.println(jwt);
    }
```

## JWT 插件配置步骤

### 步骤1：新建插件

1. 登录 腾讯微服务平台控制台，前往 **应用管理 > 中间件应用 > 网关插件**。
2. 创建 JWT 类型插件。




- 校验参数值：指用户存放 JWT 令牌（Token）参数名称，示例中参数名为 `token` 。
- 校验参数携带位置：指用户存放 JWT 令牌（Token）的位置，目前支持 Query 和 Header 两种携带方式，示例中为 Query 方式。
- 公钥对 kid：密钥 ID（“kid”），用来标识此密钥。
- 公钥对 JSON 串：公钥对，示例中值为：

```
{
    "kty": "RSA",
    "alg": "RS256",
    "n": "mrX5ROEw4kCYDXR94FQsm33gr5o5dQXvuOoe-
eG_yvdNW83MMt9GgG_eBBq_1b7HgyP9lo15BfKX3GH1igCjEKoXJxcHC5xox4xC0tvbBNCDwG_987ZsqlgCb4f7X66DCc
Hh17AyAHYa8JhO2kXvtD1OIQxSajSgmk1C1sxI5kqJXfvJwRLcCEK87P6Bs9YNLnnJeouSkYce04AhspmyQKKax4GllbM
jcrUUMRoqpCBMklh5Pgl9sOGRLo-6uzzowtI_SyF03YsE2ejh9m-YWqTYsx7PIg6SdWrNRIprKtjnhc9nk-
QBzWbOTFH3bpMoXl0T9KPndaLpi1pXtaej9Q",
    "e": "AQAB"
}
```

- claim 参数映射关系 JSON：指 JWT 令牌（Token）中 claim 携带的数据是否需要透传（可配置多组），示例中值为：

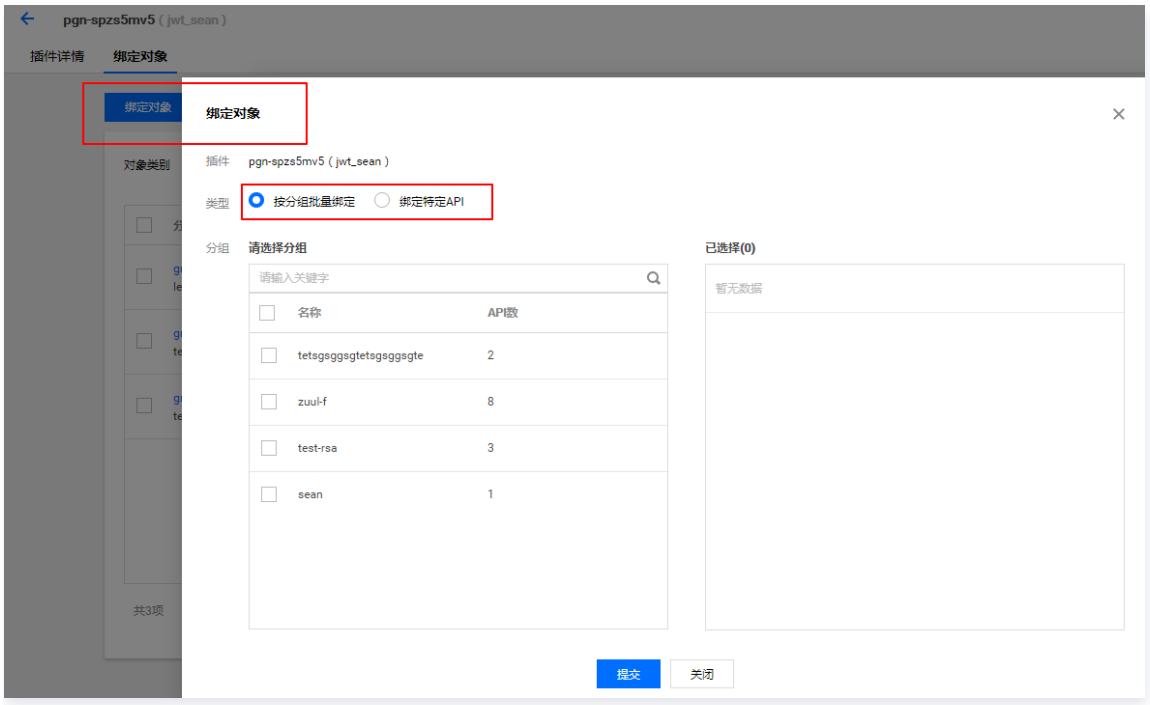
```
[{
    "parameterName":"email",
    "mappingParameterName":"new_email",
    "location":"header"
}]
```

该示例表示，将 claim 中参数为 `email` 的值，通过放入 HTTP header 透传给下游，并重新命名为 `new_email` 。

- `parameterName` 属性表示 claim 数据中待透传参数的名称。
- `mappingParameterName` 属性表示需要转换的参数名称。
- `location` 属性表示透传此值的位置。

## 步骤2：插件绑定对象

微服务网关插件通过绑定分组或 API 来生效。



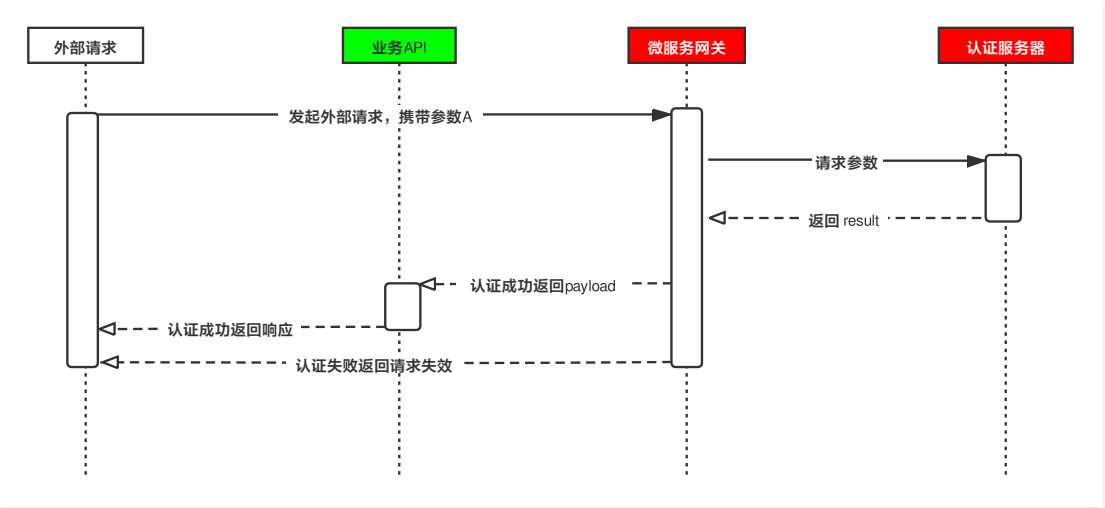

**说明：**
一个微服务网关最多支持绑定1500个插件。

# 配置 OAuth 插件

最近更新时间：2025-03-10 18:08:33

## 操作场景

该文档指导您通过腾讯微服务平台控制台，创建微服务网关标签（OAuth）插件。
微服务网关 OAuth 插件提供了简单的第三方鉴权的能力。外部请求请求到网关，网关向第三方服务器发送请求对参数进行校验，校验结果返回给第三方。
OAuth 插件的时序图如下：




外部请求请求到业务 API，通过微服务网关时网关访问认证服务器，认证服务器将结果返回给网关并返回相应给外部请求。

## 操作步骤

### 新建插件

1. 登录 腾讯微服务平台控制台，前往**应用管理 > 中间件应用 > 网关插件**。
2. 在插件管理页面，单击**新建插件**，填写基本信息。
   - 插件类型：选择 OAuth。
   - 插件名称：必填。
   - 插件描述：选填。
3. 设置请求参数。
   - 请求参数名称：请求参数名称，即上图中的 A 字段。
   - 请求参数携带位置：填写 A 参数的位置，网关将在这个位置将 A 参数传递给认证服务器。
   - 验证请求参数地址：填写认证服务器地址。以 http:// 或者 https:// 开头。
   - 请求方法：访问认证服务器的方法，GET 或 POST。
   - 超时时间：网关访问认证服务器的超时时间，0秒＜超时时间≤30秒。
4. 设置返回值。

   返回值指的是认证服务器认证成功后，认证服务器可以将某些业务字段返回给网关，网关可以将这些字段通过 Payload 的形式返回给业务 API，方便后续业务依据这些字段进行业务逻辑处理。
   - Payload 参数名称：选填。
   - Payload 参数携带位置：填写 Header 或者 Query。
5. 单击**完成**，跳转至插件管理列表。

> **说明：**

> 一个微服务网关最多支持绑定1500个插件。

## 使用说明

- 认证服务器返回认证结果将在 response body 中返回 result 字段，返回值为 true / false。返回示例如下：

```
{
    "result": true,
    "payload": "this is payload"
 }
```

- 认证失败后，网关将返回错误码401，错误信息为 “AuthCheckFailed”。
- 创建好插件后，需要将插件绑定在分组或者 API 上，才能生效。
- OAuth 插件与 JWT 插件的使用区别如下，用户可以酌情选择。

| 对比项 | 标鉴权认证实现方式 | 计算签名 | 性能 |
| --- | --- | --- | --- |
| OAuth 插件 | 在认证服务器上实现 | 需要请求方计算 | 请求量较大时，由于 OAuth 插件需要等待远程认证服务器的响应，可能会对性能有一定影响。 |
| JWT 插件 | 由网关完成 | 不需要 | - |

# 配置 Tag 插件

最近更新时间：2025-03-10 18:08:33

## 操作场景

该文档指导您通过腾讯微服务平台控制台，创建和绑定微服务网关 Tag 插件。
标签是 TSF 中传递客户业务参数的形式，用户可以通过标签来实现灵活的基于业务参数的服务治理能力并依据标签过滤调用链。
典型的标签的使用场景是：通过业务参数（例如 usertype 等字段）实现针对参数值的特殊路由、限流等。
TSF 中支持两种标签的配置方式：

- 通过微服务网关 Tag 插件将外部请求的请求参数转化为标签。
- 通过在代码或请求头中配置来实现标签。
  - Spring Cloud 应用配置标签的方法请参见 参数传递 。
  - Service Mesh 应用配置标签的方法请参见 Mesh 开发使用指引 。

## 操作步骤

### 创建 Tag 插件

1. 登录 腾讯微服务平台控制台 ，前往**应用管理 > 中间件应用 > 网关插件。**
2. 在插件管理页面，单击**新建插件**，填写基本信息。
   - 插件类型：选择 Tag。
   - 插件名称：必填。
   - 插件描述：选填。
3. 配置自定义标签。
   - 参数位置：可选 Path、Query、Header 和 Cookie。
   - 参数名：填写外部请求进来携带的参数名称。参数名请填写小写。
     如果参数在 Path中，可以按照这样的规则填写：/a/{parameter}/b，其中{}中的是参数名称。此时的路径不需要包含微服务名称、命名空间等信息。
   - 转换后标签名（选填）：如果不填写，会默认使用参数名作为转换后的标签名称。

     > **说明：**
     > - 通过微服务网关标签和代码中配置的标签总数是16个。
     > - 使用1.21.0版本的微服务网关 SDK，可以保证配置后的标签在全链路中进行传递。
     > - 标签名最长32字节。

   - 设置为 TraceID：是一个非必须功能。当请求中某个参数值具有唯一性时，且希望在部署在 TSF 的后台业务与其他前台、后台系统联动时，可以通过设置 TraceID 进行追踪。

     > **说明：**
     > - 同一类请求只允许一个参数设置为 TraceID。
     > - 当将某个参数设置为 TraceID 时，TSF 依然将同时生成 TraceID，参数名为 "X-B2-TraceID"，并将两者关联，且同时返回给调用方。
     > - 用户可以通过调用链查询中的标签检索能力搜索过滤所需要的调用链。

4. 单击**完成**，本步骤创建的插件将显示在插件管理列表中。

### 绑定插件

创建好 Tag 类型的插件需要绑定到分组或 API 上才会生效。

1. 登录 腾讯微服务平台控制台，在左侧导航栏单击组件中心下的**微服务网关 > 插件管理**。
2. 选择刚刚创建好的插件，单击操作列的**绑定对象**。
3. 在绑定对象中，将插件绑定到某个分组或者某个分组下的 API 上。
4. 单击**提交**。当用户访问对应的微服务网关分组时，插件就会生效。

**说明：**

- 有关使用微服务网关标签插件实现全链路服务治理能力并过滤调用链，请参见实践教程 > 基于业务参数的服务治理 。
- 一个微服务网关最多支持绑定1500个插件。

# 配置小程序登录插件

最近更新时间：2025-03-10 18:08:33

> **说明：**
> 当前 TSF 同时支持微信小程序和 QQ 小程序开发，QQ 小程序插件名为 QQMPLogin。关于小程序开发的详细说明请参见 QQ 小程序文档 。

## 操作场景

该文档指导您通过腾讯微服务平台控制台，创建和绑定微信小程序登录插件。
微信小程序开发者在开发小程序的过程中，需要考虑用户登录认证相关的业务逻辑，基于此插件，开发者可以方便地实现业务的登录认证服务。
典型的微信小程序登录插件使用场景是：

- 实现微信小程序用户登录：遵守微信小程序官方登录规范，在微服务网关侧完成登录业务，生成用户登录态凭证，并返回给小程序前端处理，同时，将微信登录成功返回的原生登录态数据传递至业务后台处理。
- 实现用户登录态凭证校验：微服务网关对用户登录态凭证进行校验，若校验通过，则网关将请求放行，并携带微信原生登录态数据传递至业务后台，否则，返回失败原因至小程序前台，小程序客户端可以根据异常状态码，进行对应的操作（例如因登录态过期失效进行重新登录操作）。

## 操作步骤

### 创建微信小程序登录插件

1. 登录 腾讯微服务平台控制台，前往**应用管理 > 中间件应用 > 网关插件**。
2. 在插件管理页面，单击**新建插件**，填写基本信息。
   - 插件类型：选择 **WeChatMpLogin**。
   - 插件名称：必填。
   - 插件描述：选填。
   - 标签：非必选。
   - 数据集：非必选。
3. 填写参数设置。
   - APPID：小程序唯一凭证密钥，获取方式同 APPID。
   - APPSECRET：小程序唯一凭证密钥，获取方式同 APPID。
4. 获取登录凭证 Code。
   请求 Code 携带位置：微信后台通过 Code 换取用户的登录态信息，包括用户的唯一标识（openid）及本次登录的会话密钥（session_key）。
5. 向小程序返回登录态。
   - 自定义登录态参数名：插件对微信登录态以及其他登录信息进行加密的参数名。
     返回自定义登录态参数位置：插件对登录态进行加密后的参数携带位置。
   - 登录态过期时间：登录态过期时间，以分为单位。
6. 选择登录态校验。
   - 前台业务请求自定义登录态参数位置：小程序请求业务服务时，登录态携带位置。
   - 向业务后台传输登录态参数位置：网关对登录态进行解密后，将提取的微信登录信息传递至用户后台服务的携带位置。
7. 单击**完成**，完成创建。

### 将请求参数转 Tag

TSF 提供了一系列将请求参数转 Tag 能力，方便进行细粒度的服务治理能力。
有关请求参数转 Tag 能力的描述可参见 微服务网关 Tag 插件 。

#### 默认 Tag 操作说明

- 平台提供了一系列微信默认参数的灰度发布能力，包含用户名、性别等信息。用户可以勾选这些参数并提供参数的填写位置，方便平台将这些请求参数转为 TSF 中的标签。
- 当某个参数的传入具有请求的唯一性（该参数值不重复），则可以将该参数设置为 TraceID，平台将对该参数与平台生成的 TraceID 进行关联，方便用户进行唯一性定位与检索。
- 仅支持一个参数设置为 `TraceID/` 。

### 自定义 Tag 操作说明

自定义 Tag 使用请参见 微服务网关 Tag 插件 。

### 绑定插件

创建好 WeChatMpLogin 类型的插件需要绑定到分组或 API 上才会生效。

1. 登录 腾讯微服务平台控制台，在左侧导航栏单击组件中心下的**微服务网关 > 插件管理**。
2. 选择刚刚创建好的插件，单击操作列的**绑定对象**。
3. 在绑定对象中，将插件绑定到某个分组或者某个分组下的 API 上。
4. 单击**提交**。当用户访问对应的微服务网关分组时，插件就会生效。

> **说明：**
> 一个微服务网关最多支持绑定1500个插件。

## 案例说明

1. 获取 APPID 与 APPSECRET。
   在开发者 微信小程序开发平台 中，可以查询到小程序的 AppID（小程序 ID）与 AppSecret（小程序密钥）。
2. 根据上述 操作步骤 ，完成微服务网关中微信小程序登录插件的创建与绑定。
3. 编写小程序登录业务。

```
// 登录
wx.login({
  success: res => {
        // console.log("code is : ", res.code);
  wx.request({
          url: ${LOGIN_BUSINESS_URL}, //登录业务访问路径
          header: {
            'content-type': 'application/json',
            'js_code': res.code, //携带js_code
          },
          success: res => {
              //本地缓存
             wx.setStorageSync('user_session',     res.header.user_session);
            //let user_session =     wx.getStorageSync('user_session') || ''
            //console.log('idsession is :', idsession);
            wx.showToast({
                title: '登录成功'
            })
          }
        }
        )
      }
    })
```

4. 通过小程序访问业务服务。

```
BusinessRequestFunc: function () {
   //获取登录时存储的user_session
   let user_session = wx.getStorageSync('user_session') || '';
   //console.log("user_session is : " , user_session);
   wx.request({
     url: ${BUSINESS_URL}, //业务访问路径,
     header: {
       'content-type': 'application/json',
       'user_session': user_session
     },
     success: res => {
       //console.log("response : ", res.data)
       // your business here
     }
   }
   )
    //具体接口使用文档参照微信小程序官方使用接口文档
 }
```

# 持续集成
# 使用 Python 脚本部署应用

最近更新时间：2022-07-27 16:47:07

TSF 应用可以使用 Python 脚本来部署。

## 前提条件

在开始持续集成之前，需要完成下述的准备工作：

1. 保证机器上安装的 Python 版本不低于 `3.0` 版本，并已安装 PIP 等 Python 包管理工具。
2. 获取腾讯云的 访问密钥（SecretId 和 SecretKey）。
3. 在 TSF 平台创建了 部署组 。
4. 了解 Python 脚本使用。

## 虚拟机应用部署准备

1. 机器上保证安装的 Python 版本不低于 `3.0` 版本，并已安装 PIP 等 Python 包管理工具。
2. 从 GitHub仓库 下载虚拟机部署 Python 脚本。
3. 修改脚本中的secret_id 、secret_key 为腾讯云访问密钥，region 为 TSF 服务所在地域。

```
secret_id = "改为您的 SecretId"
secret_key = "改为您的 SecretKey。"
region = "改为您的服务所在地域，如 ap-guangzhou"
```

4. 安装脚本依赖包。

```
pip install requests cos-python-sdk-v5
```

5. 准备脚本参数，要严格保证参数顺序。
   - `path`：本地文件路径，可以是针对脚本的相对路径，支持 `.tar.gz` 、 `.jar` 、 `.war` 、 `.zip` 结尾的文件。
   - `applicationId`：应用 ID。在 TSF 控制台 > **应用管理**中，选择目标应用第一列的应用 ID（如 `application-qab76pxv` ）。
   - `pkg_version`：程序包版本，最长32个字符，支持 a-z、A-Z、0-9、横杠（-）、下划线（_）。
   - `appId`：用户 APPID。在腾讯云控制台 账号中心 > **账号信息**中获取 APPID（如 `1300555551` ）。
   - `group_id`：部署组 ID。在 TSF 控制台 > **应用管理** 中，单击目标应用 ID，进入详情页，在**部署组**标签页中获取部署组的 ID（如 `group-gvk5pbdv` ）。
   - `startup_params`：启动参数。用户视情况可以自定义。

     ```
     -Xms128m -Xmx512m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=512m
     ```

     > ⚠ **注意**
     > 执行的脚本参数中如果版本号不变时，脚本会选择 TSF 平台已有的 jar 包进行再次部署。

6. 将以上参数按照顺序整理待用，格式如下：

```
 python2.7.14 upload_virtual_machine_deploy.py ./consumer-demo/target/consumer-demo-1.10.0-
RELEASE.jar application-qab76pxv v001 1300555551 group-gvk5pbdv '-Xms128m -Xmx512m -
XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=512m'
```

## 容器应用部署准备

1. 保证机器上能够构建、上传镜像（镜像及 Dockerfile 编写参见 制作容器镜像，仓库使用请参见 镜像管理 ）。保证 Python 版本不低于2.7.14版本，并已安装 PIP 等 Python 包管理工具。
2. 从 GitHub仓库 下载容器部署 Python 脚本。
3. 修改脚本中的 secret_id 、secret_key 为腾讯云访问密钥，region 为 TSF 服务所在地域。修改脚本中的 docker_build_command、docker_push_command 为实际的 docker build 和 push 命令。

```
secret_id = "改为Access Key ID"
secret_key = "改为Access Key Secret。"
docker_build_command = "改为 docker build 命令"
docker_push_command = "改为 docker push 命令"
region = "改为您的服务所在地域，如 ap-guangzhou"
```

4. 安装脚本依赖包。

```
pip install tencentcloud-sdk-python
```

5. 准备脚本参数，要严格保证参数顺序。
   - `group_id`：部署组 ID。在 TSF 控制台 > **应用管理**中，单击目标应用 ID，进入详情页，在**部署组**标签页中获取部署组的 ID（如 `group-zvw397wa` ）。
   - `tag_name`：镜像版本名称，如v1。

   > **注意**
   > 目前容器应用部署脚本只将 group_id 和 tag_name 作为参数，但实际上用户可以修改脚本，将其他字段（如 JVM 启动参数 JvmOpts 等）作为脚本参数，参见 容器应用部署 API 。

6. 将以上参数按照顺序整理，并且加入第一步中用户自己的 docker build 、push 命令待用，格式如下：

```
python2.7.14 upload_container_deploy.py group-zvw397wa v1
```

> **注意**
> docker 相关命令必须按照 制作容器镜像 和 镜像管理 调整为用户自己的账号和应用名。

# 使用 Jenkins 创建持续集成

最近更新时间：2024-11-04 10:09:12

TSF 应用可以使用 Jenkins 构建持续集成方案。

## 准备工作

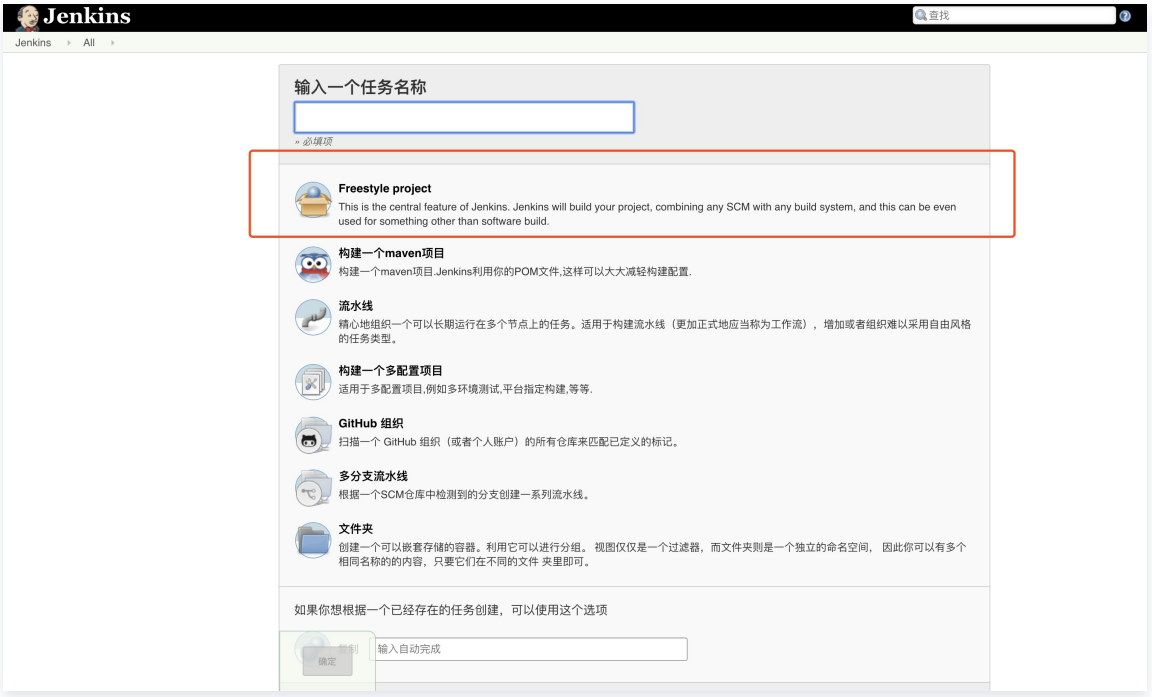
在开始持续集成之前，需要完成下述的准备工作。

1. 参见 使用 Python 脚本部署应用 获取部署脚本及使用方法的说明。
2. 安装 Jenkins ，确保 Jenkins 安装机器上的 Python 不低于 `2.7.14` 版本，并已安装 PIP 等 Python 包管理工具。
3. 在 TSF 平台创建了 部署组 。

## 安装和配置 Jenkins

1. 进入 Jenkins 官网 下载安装 Jenkins。
2. 安装 Maven 并配置 TSF 私服地址，请参见 下载 Maven 。
3. 在 Jenkins 控制台的菜单栏中选择**系统管理** > **插件管理**，安装 Maven Integration plugin 插件，如已安装请忽略。
4. 创建 Jenkins 项目

   4.1 在 Jenkins 首页左侧导航栏中单击新建，创建 Jenkins 任务，并选择构建一个自由风格的软件项目。

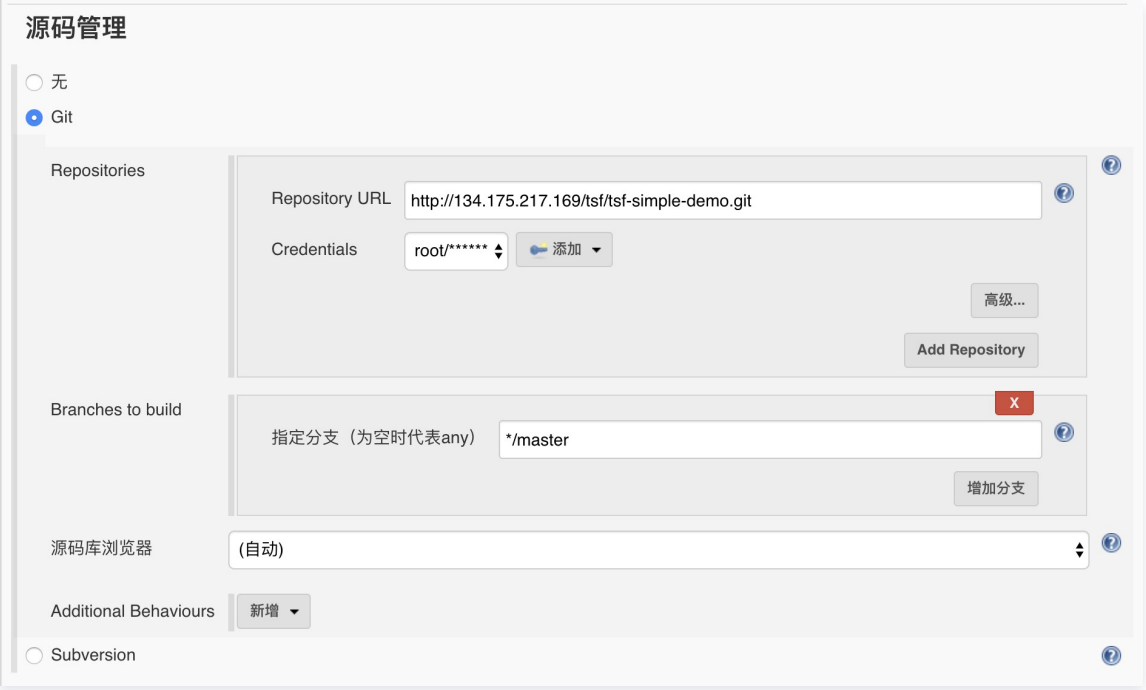


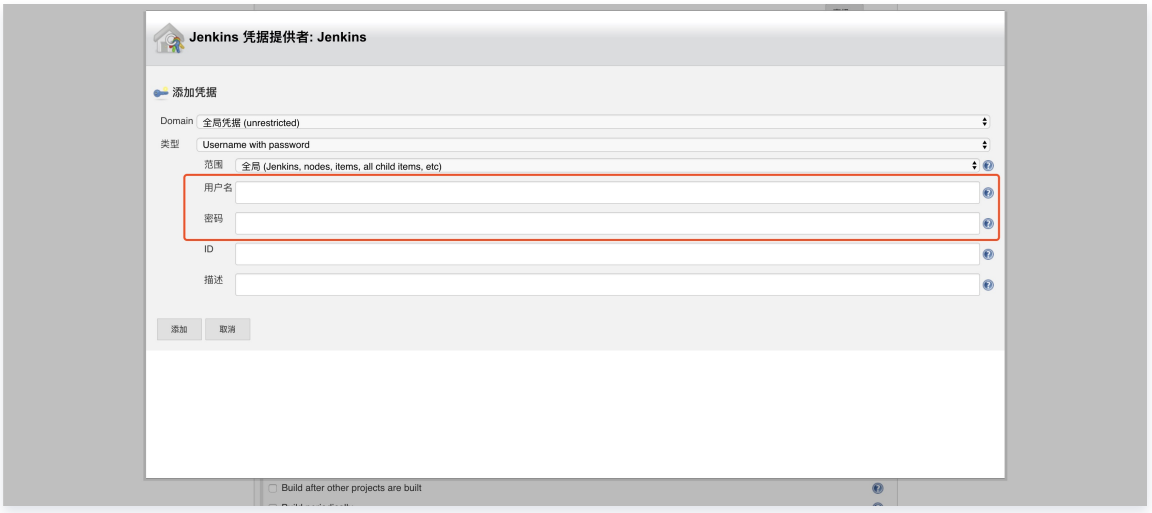

   4.2 在 **源码管理** 页面中选择 Git，并设置相关参数。

   Repository URL：您的项目的 Git 协议地址。

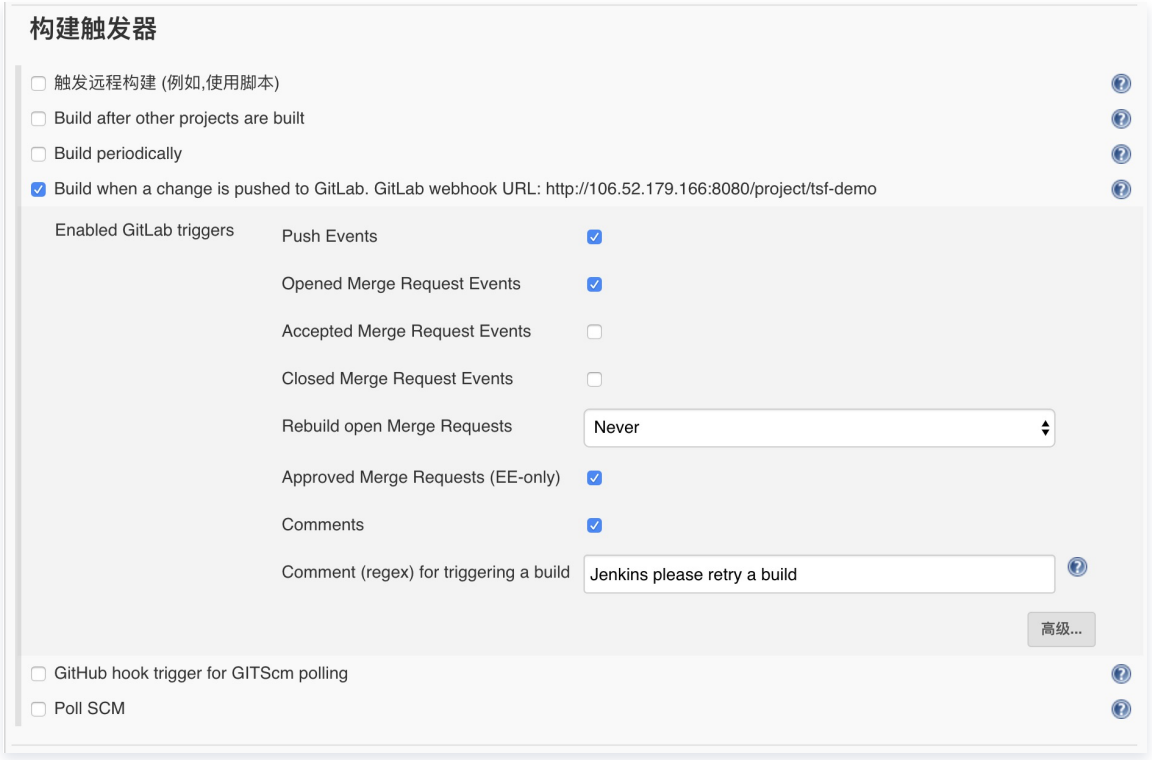
   Credentials：安全凭证，选择无（前提是运行 Jenkins 软件的用户的 SSH RSA 公匙已添加到该 Git 项目所在的 GitLab 或 GitHub 中，否则这里会报错）或者添加用户名密码。

版权所有：腾讯云计算（北京）有限责任公司

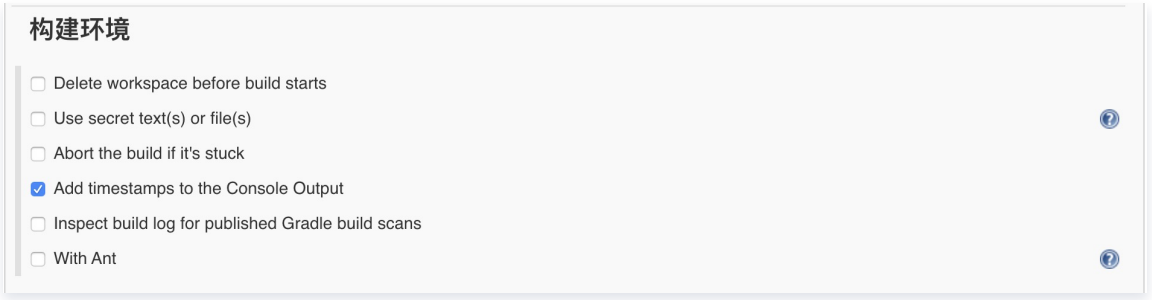

源码管理
无
Git
Repositories
Repository URL
http://134.175.217.169/tsf/tsf-simple-demo.git
Credentials
root/******
添加
高级...
Add Repository
Branches to build
指定分支（为空时代表any）
*/master
增加分支
源码库浏览器
(自动)
Additional Behaviours
新增
Subversion


Jenkins 凭据提供者: Jenkins
添加凭据
Domain
全局凭据 (unrestricted)
类型
Username with password
范围
全局 (Jenkins, nodes, items, all child items, etc)
用户名
密码
ID
描述
添加
取消
Build after other projects are built

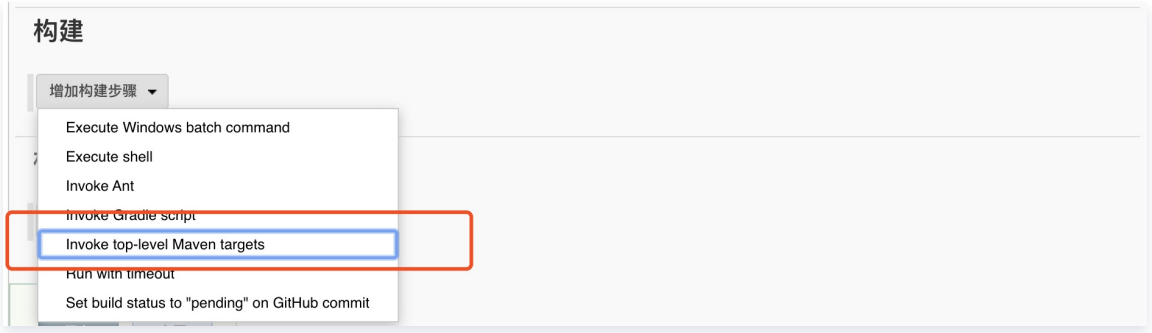
单击构建触发器页签，勾选 Build when a change is pushed to GitLab。GitLab webhook URL 中的 IP 要确保 GitLab 能访问。



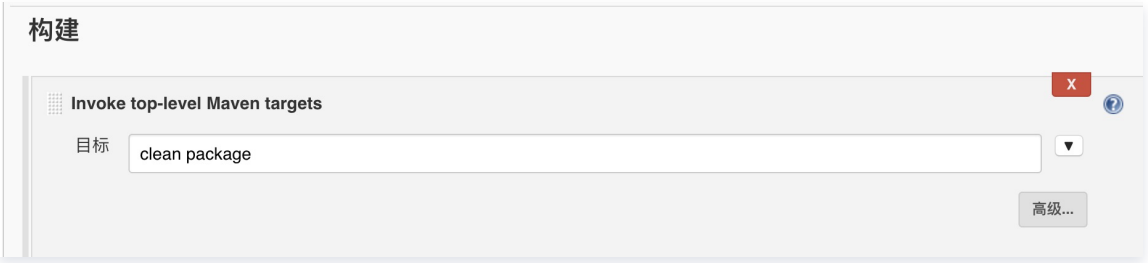

4.4 单击构建环境页签，勾选 Add timestamps to the Console Output（为控制台输出的信息添加时间戳）。



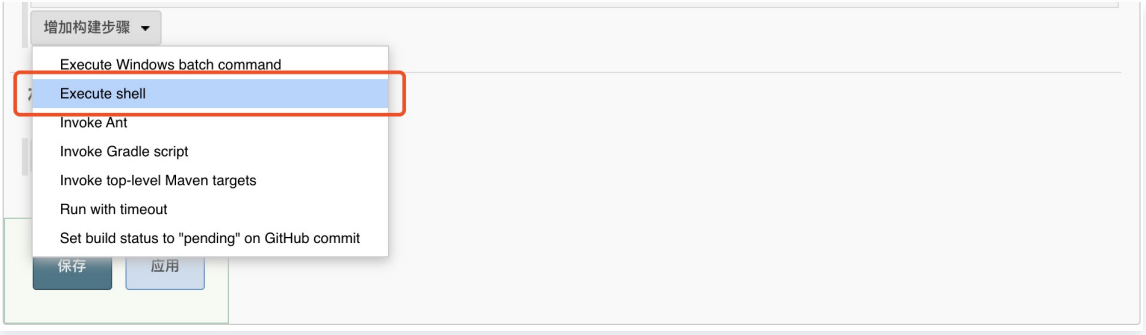

4.5 单击构建页签，然后单击增加构建步骤，选择 Invoke top-level Maven targets。



目标：填入 clean package（如有其它参数，请根据实际情况填入）。



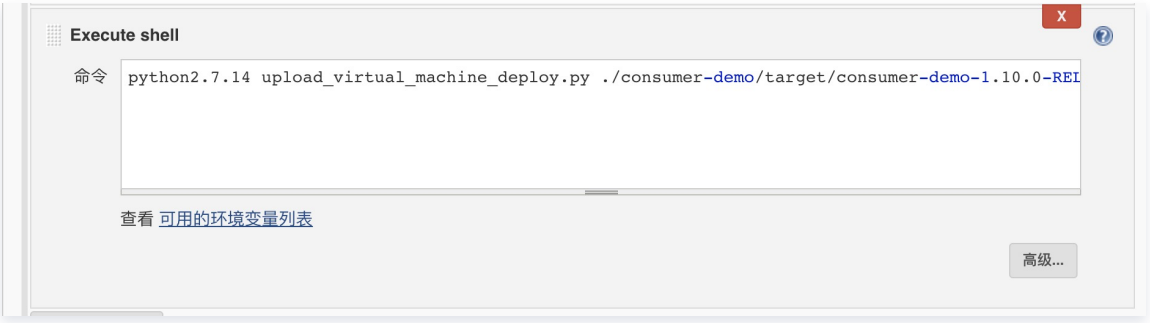

4.6 单击构建页签，然后单击增加构建步骤，选择 Execute shell。




命令中填入应用部署准备 步骤中准备好的 Shell 命令。

4.6.1 虚拟机如下：

```
python2.7.14 upload_virtual_machine_deploy.py ./consumer-demo/target/consumer-demo-1.10.0-RELEASE.jar application-qab76pxv v001 1300555551 group-gvk5pbdv '-Xms128m -Xmx512m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=512m'
```



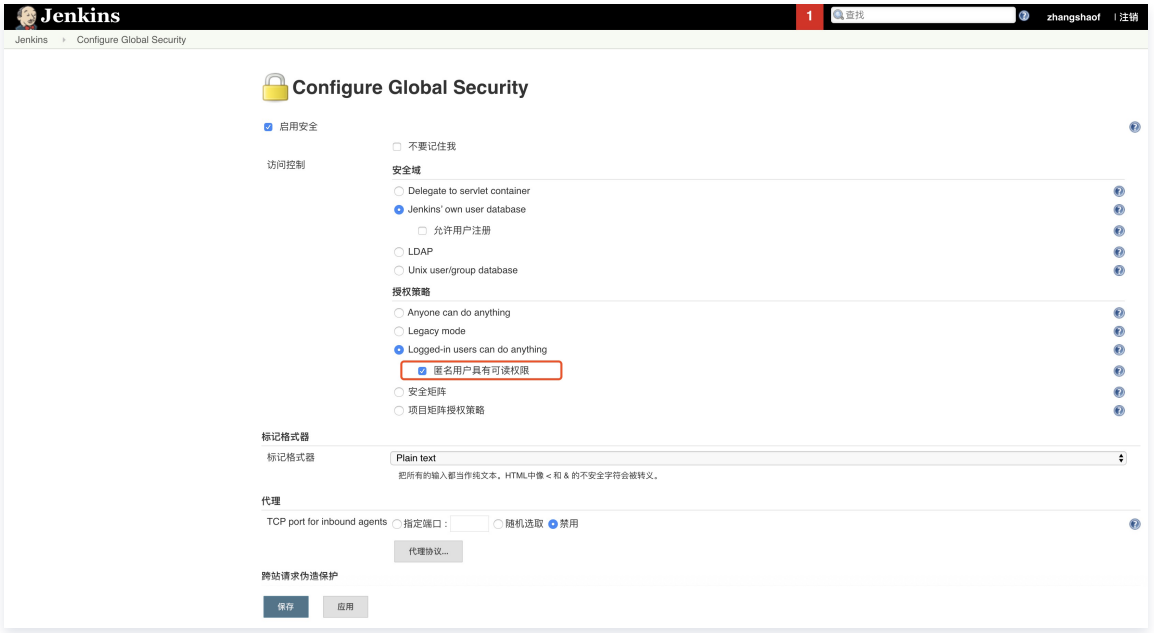

4.6.2 容器如下：

```
python2.7.14 upload_container_deploy.py group-zvw397wa v1
```

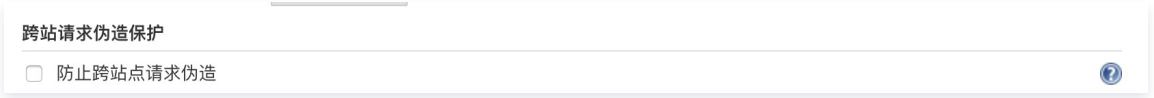


### 4.7 配置 GitLab 所需权限：

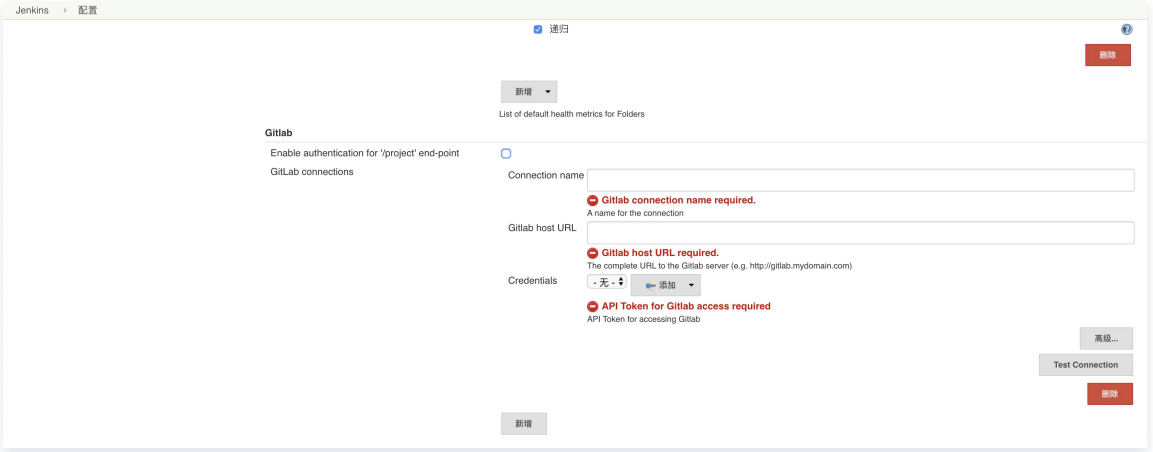
#### 4.7.1 管理 Jenkins > Configure Global Security 勾选 匿名用户具有可读权限。




取消防止跨站点请求伪造。

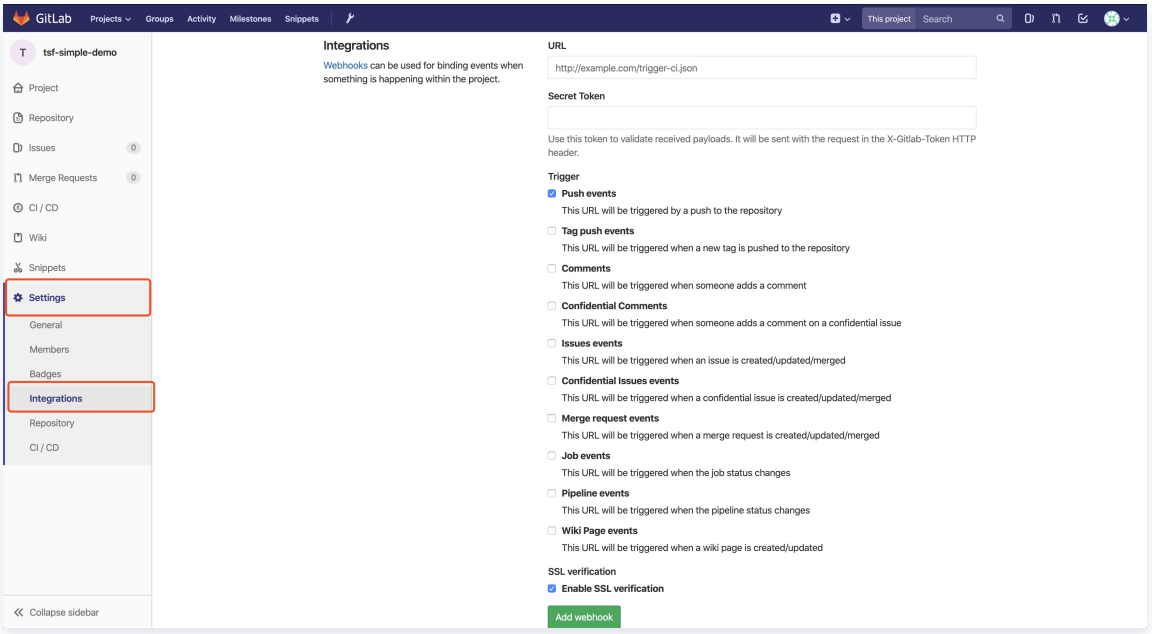



#### 4.7.2 管理 Jenkins > 系统配置 取消 Enable authentication for '/project' end-point。

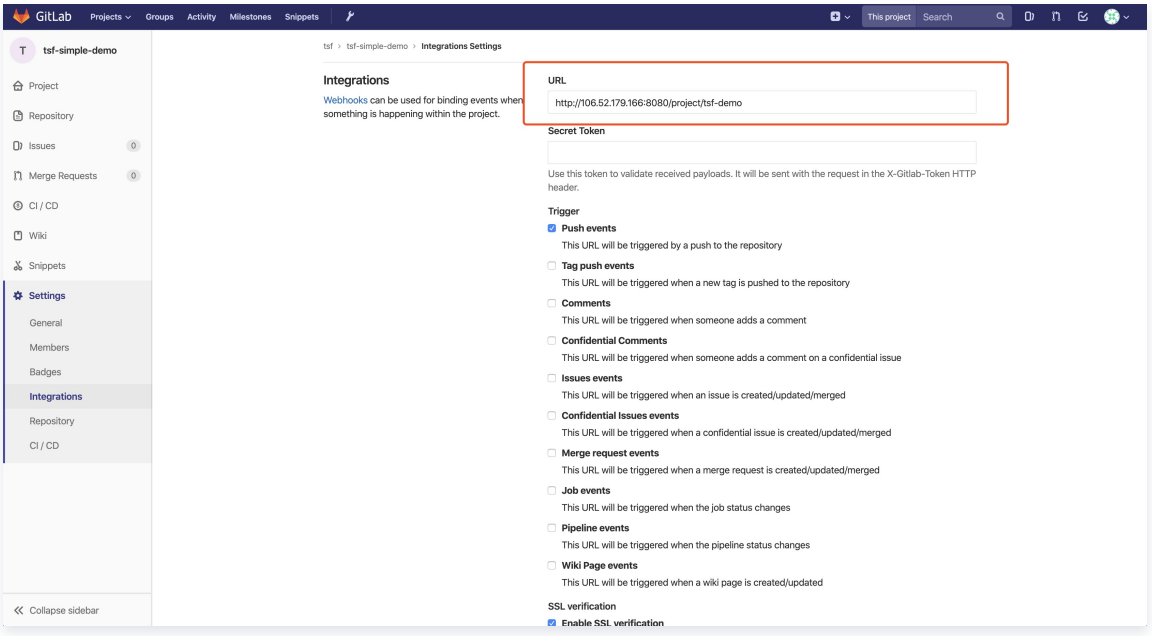



### 5. 配置 GitLab 的 Web Hook，实现自动构建：

5.1 项目 > Settings > Integrations 进入添加 webhook 界面。

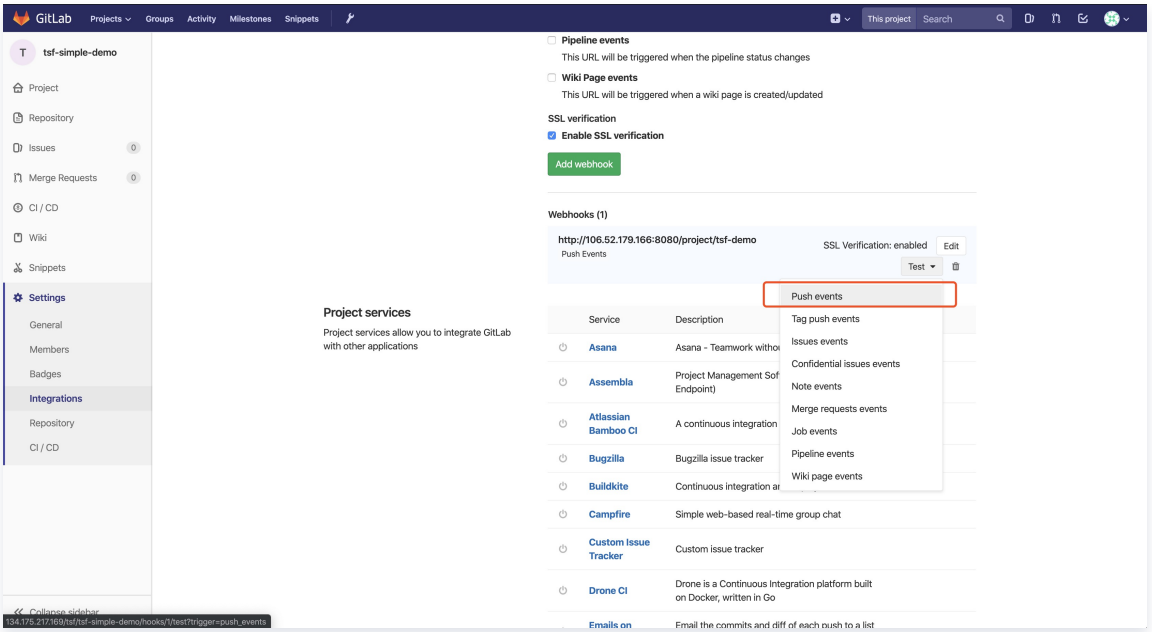



5.2 将 4.3 中 Jenkins 产生的 GitLab webhook URL 填入 URL，其他选项使用默认设置，单击"Add webhook"。

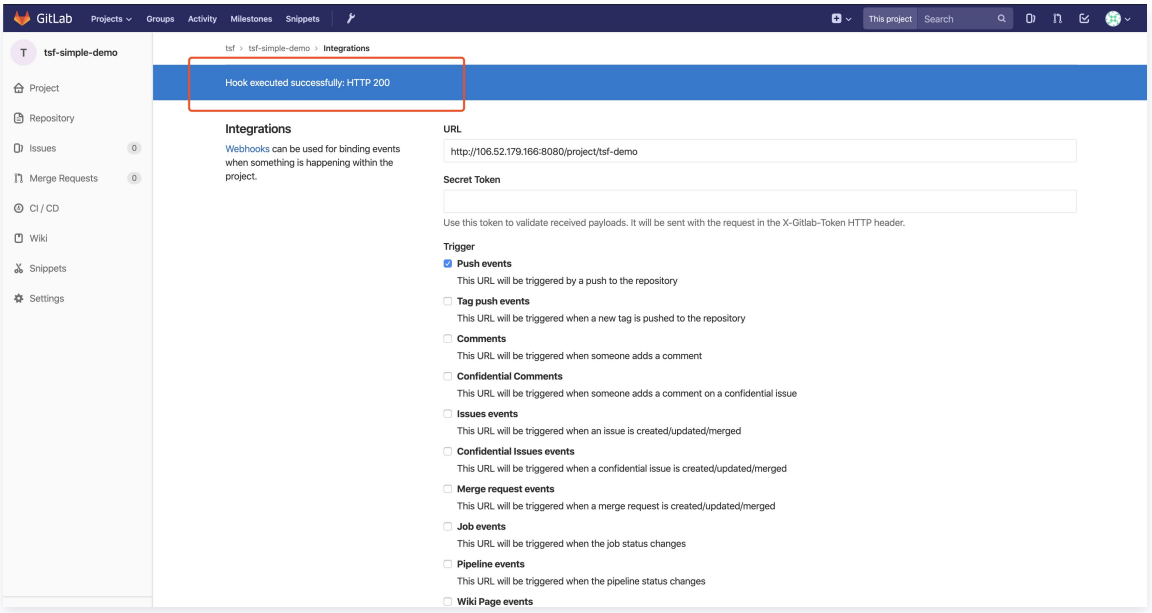



版权所有：腾讯云计算（北京）有限责任公司

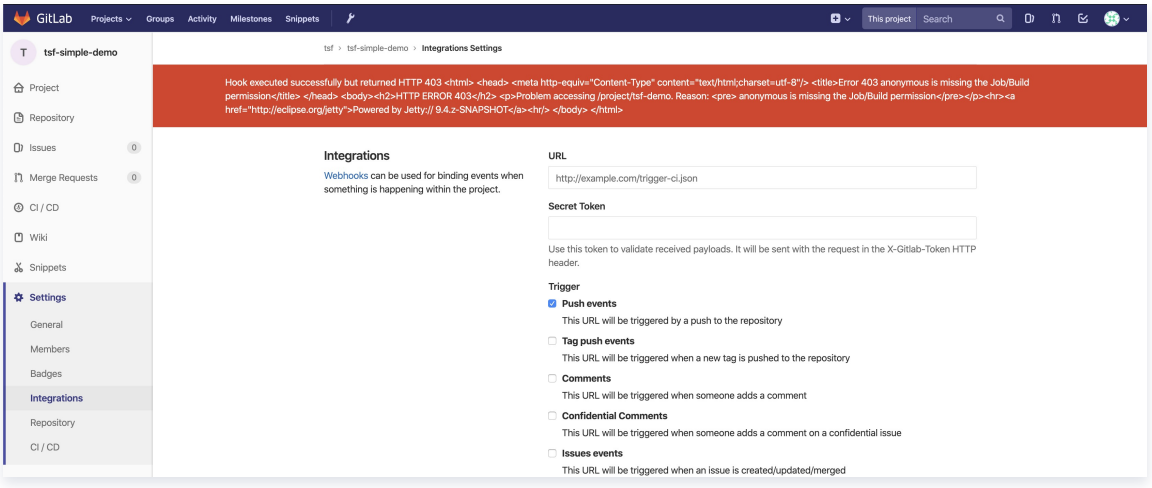
### 5.3 测试 WebHook。




**成功如图：**




**如出现 403 异常，请检查 4.7 配置。**



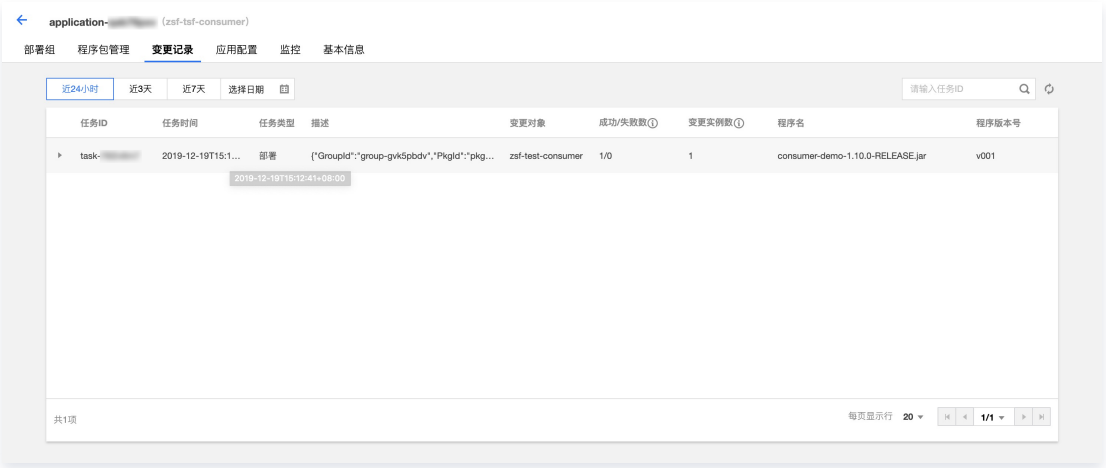
## 配置正确后，提交变更到 GitLab

如果上述步骤配置正确，这次提交会触发一次 GitLab Hook。Jenkins 在接收到这个 Hook 后会构建您的 Maven 项目，并在构建过程中触发 Python 脚本自动部署 TSF 应用。

1. 虚拟机提交部署成功输出的日志信息（Build Number > 控制台输出）。

```
    15:12:40 + python2.7.14 upload_virtual_machine_deploy.py ./consumer-demo/target/consumer-
demo-1.10.0-RELEASE.jar application-qab76pxv v001 1300555551 group-gvk5pbdv '-Xms128m -
Xmx512m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=512m'
    15:12:41 POST
    15:12:41 /
    15:12:41
    15:12:41 content-type:application/json
    15:12:41 host:tsf.tencentcloudapi.com
    15:12:41
    15:12:41 content-type;host
    15:12:41 0c2f4a50b1dc4df621a84d1731ec07cbd756ac04b1f4853fa792c5dbcf104f28
    15:12:41 TC3-HMAC-SHA256
    15:12:41 1576739561
    15:12:41 2019-12-19/tsf/tc3_request
    15:12:41 46cdc18f96456edef21e4b01e787030d62ce54bb7b300994fed4d5d473709701
    15:12:41 1d501d695675696fef1c2b2576bb21f42f6f31f55f07c7b27c5ca051d20f242e
    15:12:41 TC3-HMAC-SHA256
Credential=**********************************************************/2019-12-
19/tsf/tc3_request, SignedHeaders=content-type;host,
Signature=1d501d695675696fef1c2b2576bb21f42f6f31f55f07c7b27c5ca051d20f242e
    15:12:41 {"Response":{"RequestId":"0b6aa4ee-38fe-4782-97e4-be1061e7cdb1", "Result":
{"Content":[{"PkgId":"pkg-f51038e0","PkgName":"consumer-demo-1.10.0-
RELEASE.jar","PkgType":"fatjar","PkgVersion":"v001","PkgDesc":"","UploadTime":"2019-12-18
11:03:26","Md5":"4ad6f32f20c1d2dcd60575fa50b16e1d","PkgPubStatus":1}],"TotalCount":1}}}
    15:12:41 [INFO] application-qab76pxv has uploaded version v001, no need upload
    15:12:41 POST
    15:12:41 /
    15:12:41
    15:12:41 content-type:application/json
    15:12:41 host:tsf.tencentcloudapi.com
    15:12:41
    15:12:41 content-type;host
    15:12:41 99f3e6ddeaf00f7b64315835fdd6f29d7a0a5520743184c21a43d8a609d6891a
    15:12:41 TC3-HMAC-SHA256
    15:12:41 1576739561
    15:12:41 2019-12-19/tsf/tc3_request
    15:12:41 e5ebc373c1c432910ca0110e48f1fe4cf29ffcdedc21dc79dd59a99bf802ea23
    15:12:41 73e21ed01f4ad54668e94a2839d6de17c07e6188e1ff9e176452b80fb776be15
    15:12:41 TC3-HMAC-SHA256
Credential=**********************************************************/2019-12-
19/tsf/tc3_request, SignedHeaders=content-type;host,
Signature=73e21ed01f4ad54668e94a2839d6de17c07e6188e1ff9e176452b80fb776be15
    15:12:41 200
    15:12:41 {"Response":{"RequestId":"86c93d7e-ee44-4c8d-8d34-537a0dbe6871","Result":
{"TaskId":"task-76l2v6m7"}}}
    15:12:41 Finished: SUCCESS
```

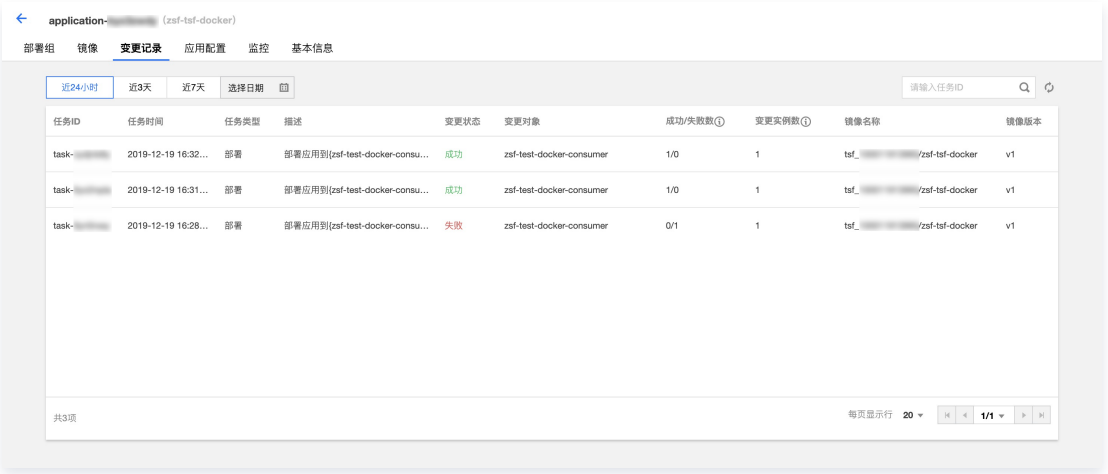
TSF 平台可以查看虚拟机部署变更记录。




2. 容器部署成功日志信息。

```
    16:32:28 + python2.7.14 upload_container_deploy.py group-zvw397wa v1
    16:32:31 {"Result": {"InstanceNum": 1, "NamespaceName": "zsf-test-jenkins-
docker_default", "ApplicationType": "C", "CurrentNum": 1, "MicroserviceType": "N", "Status":
"Running", "LbIp": "", "MemRequest": "128", "ClusterId": "cls-52tydz2q", "UpdateIvl": 0,
"Envs": [{"Name": "tsf_consul_ip", "Value": "169.254.0.77"}, {"Name": "tsf_consul_port",
"Value": "8000"}, {"Name": "PATH", "Value":
"/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"}, {"Name": "workdir", "Value":
"/app/"}, {"Name": "jar", "Value": "consumer-demo-1.10.0-RELEASE.jar"}, {"Name":
"tsf_app_id", "Value": "1300555551"}, {"Name": "tsf_token", "Value":
"VjCCe9gixVmdeBfTpM74PikHFfU8ZmZVL7Z4FMGvXjI="}, {"Name": "tsf_application_id", "Value":
"application-byx3zwdy"}, {"Name": "tsf_cluster_id", "Value": "cls-52tydz2q"}, {"Name":
"tsf_namespace_id", "Value": "namespace-py5e9zjy"}, {"Name": "tsf_group_id", "Value": "group-
zvw397wa"}, {"Name": "tsf_prog_version", "Value": "v1"}, {"Name": "tsf_region", "Value": "ap-
guangzhou"}, {"Name": "tsf_ratelimit_master_ip", "Value": "169.254.0.77"}, {"Name":
"tsf_ratelimit_master_port", "Value": "7000"}], "TagName": "v1", "ClusterIp":
"172.18.255.122", "MemLimit": "256.00", "CpuLimit": "0.50", "ApplicationId": "application-
byx3zwdy", "ApplicationName": "zsf-tsf-docker", "ProtocolPorts": [{"Protocol": "TCP",
"TargetPort": 80, "Port": 80}], "UpdateType": 0, "Server": "ccr.ccs.tencentyun.com",
"GroupName": "zsf-test-docker-consumer", "NodePort": 30215, "AccessType": 2, "ClusterName":
"zsf-test-jenkins-docker", "NamespaceId": "namespace-py5e9zjy", "Reponame":
"tsf_100011913960/zsf-tsf-docker", "CpuRequest": "0.25", "Message": "", "CreateTime": "2019-
12-17 15:09:08", "GroupId": "group-zvw397wa"}, "RequestId": "aff9a2b6-24f5-44b1-8173-
112a6a24b581"}
    16:32:31 {"Result": true, "RequestId": "263c8a64-1bb1-427e-80e2-870aa0bc7d68"}
    16:32:31 Finished: SUCCESS
```

TSF 平台可以查看容器部署变更记录。




3. 容器部署成功日志信息如果部署失败，需要分析 Jenkins console 日志。

# 使用 Coding 创建持续集成

最近更新时间：2024-09-04 14:29:31

## 操作场景

TSF 应用可以使用 CODING 构建持续集成方案。腾讯云 CODING 持续集成全面兼容 Jenkins 的持续集成服务，更多关于 CODING 持续集成功能和使用的说明，请参见 CODING 持续集成 。

### CODING DevOps 工作流

下图展示了使用 CODING 持续集成，TSF 作为应用部署平台的工作流。




## 准备工作

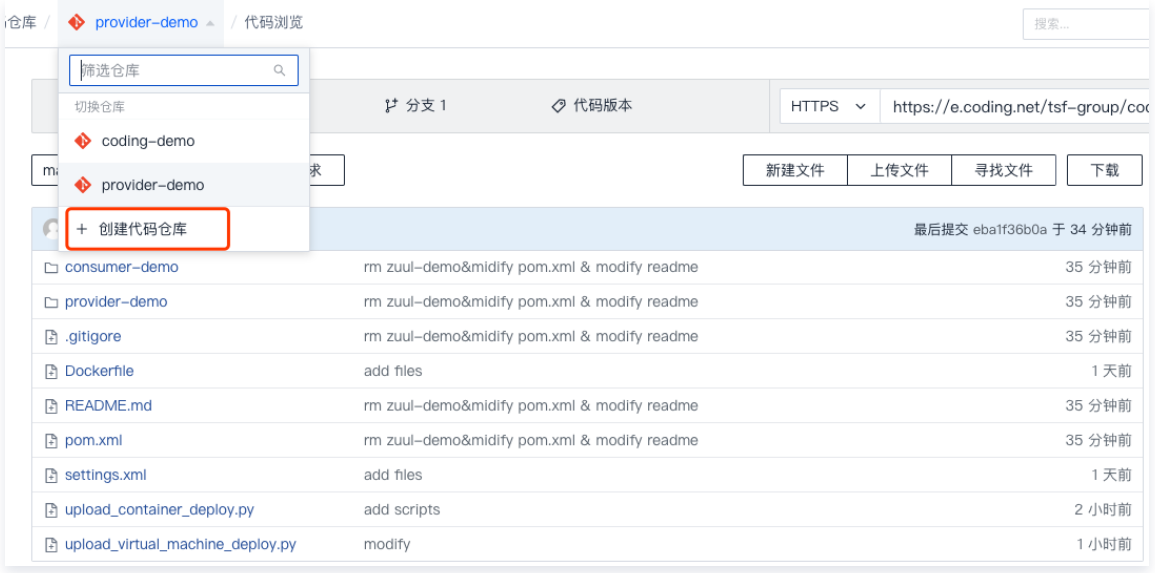
在开始持续集成之前，需要完成下述的准备工作：

1. 参见 使用 Python 脚本部署应用 获取部署脚本及使用说明。
2. 在 TSF 平台创建了 部署组 。
3. 获取 TSF 私服地址，详情请参见 SDK 下载 。
4. 腾讯云主账号已开通 CODING DevOps ，并创建 CODING 项目、代码仓库等基础环境。

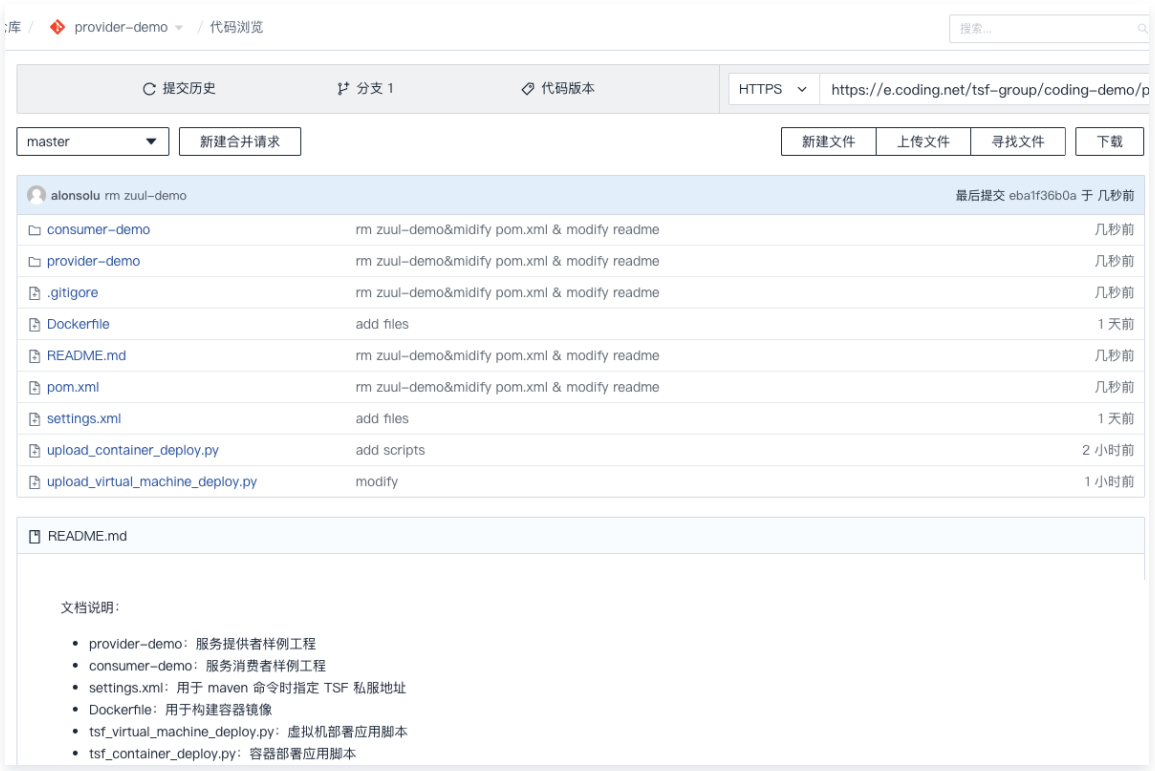
## CODING 平台操作

### 步骤1：创建代码仓库并提交代码

1. 登录 CODING DevOps 控制台 ，进入示例项目。
2. 单击左侧导航栏**代码仓库** > **代码浏览**，单击顶部导航栏，创建代码仓库（如 `provider-demo` ）。



3. 下载 演示工程源码，上传到 Coding 的 `provider-demo` 代码仓库中。




更多关于如何创建代码仓库，以及如何提交代码的说明请参见 CODING 代码托管 。

> **注意**
> 用户需要修改示例代码中的 Python 部署脚本，替换其中访问密钥、地域等信息，参见 使用 Python 脚本部署应用 。

## 步骤2：创建并配置构建任务

### 创建构建任务

1. 单击 Coding 控制台左侧导航栏**构建与部署 > 构建**。
2. 单击**新建**。
3. 填写新建信息，选择代码仓库 `provider-demo` ，使用静态配置的 Jenkinsfile ，选择简易模板。

### 设置环境变量

单击**变量与缓存**选项卡，输入环境变量：

- PKG_VERSION：程序包/镜像版本号
- TSF_GROUP_ID: TSF 平台的部署组 ID
- TSF_APPLICATION_ID：TSF 平台的应用 ID（仅适用于虚拟机应用部署）
- TSF_APPID：用户在腾讯云上的 APPID（仅适用于虚拟机应用部署）
- TSF_STARTUP_PARAM：Java 应用启动参数，注意**不要**带上引号（仅适用于虚拟机应用部署）




### 流程配置-构建

1. 在**流程配置**选项卡**图形化编辑器**中，选中**构建**阶段方块。
2. 单击**执行 Shell 脚本**。




3. 编辑**Shell 脚本**，填写 maven package 命令 ，并使用工程中包含 TSF 私服地址的settings.xml。单击**保存**。

```
mvn clean package -U --settings settings.xml -Dmaven.test.skip=true
```




### 流程配置-部署

1. 在**流程配置**选项卡**图形化编辑器**中，选中**部署**阶段方块。
2. 单击**增加步骤**。
3. 单击**执行 Shell 脚本**，输入 Shell 脚本，填写 Python 部署脚本执行命令。




- 虚拟机应用部署脚本命令：

```
python upload_virtual_machine_deploy.py provider-demo/target/provider-demo-0.0.1-SNAPSHOT.jar ${TSF_APPLICATION_ID} ${PKG_VERSION} ${TSF_APPID} ${TSF_GROUP_ID} ${TSF_STARTUP_PARAM}
```

- 容器应用部署脚本命令：

```
python upload_container_deploy.py ${TSF_GROUP_ID} ${PKG_VERSION}
```

## 步骤3：执行构建任务

构建任务可以通过代码提交等方式自动触发，也可以手动执行。

1. 选中目标构建任务，单击**立即构建**。




2. 填写环境变量，将值替换为用户自己的参数，单击**立即构建**。

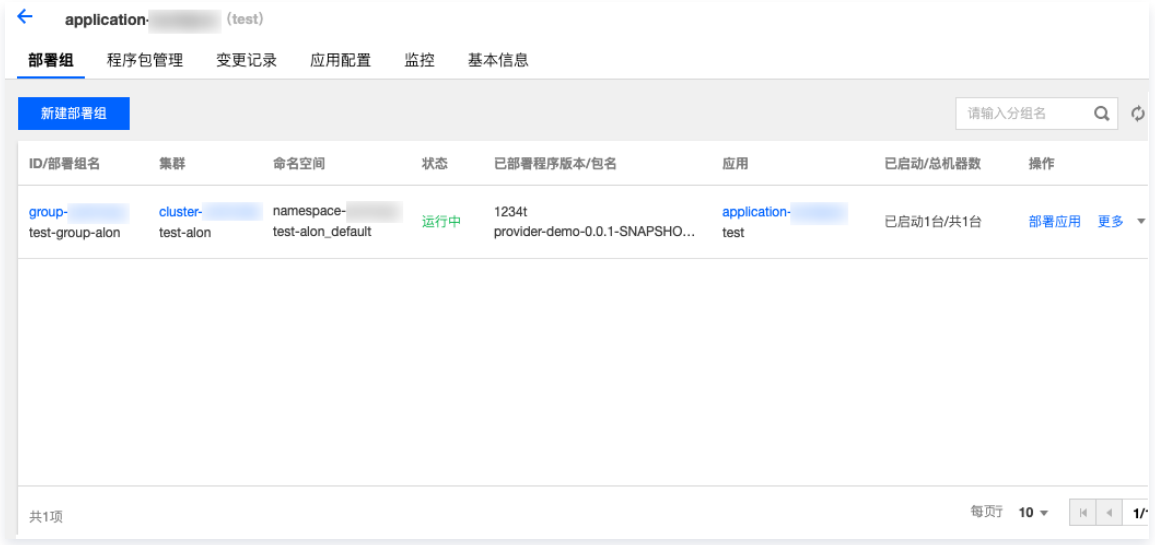



## 步骤4：验证构建结果

1. 构建任务通常会执行几分钟，查看构建任务的结果。




2. 在 TSF 控制台，查看对应部署组的状态。



# 使用插件部署应用
# 安装配置 IntelliJ IDEA 插件

最近更新时间：2022-06-10 10:43:17

## 操作场景

通过本篇文档，您将会了解到如何在 IntelliJ IDEA 中安装和配置 Tencent Cloud Toolkit。安装配置完成后，您可以将本地应用快速部署到腾讯微服务平台 TSF（包括虚拟机部署、容器部署），还可以使用 Tencent Cloud Toolkit 编译、上传程序包和镜像。

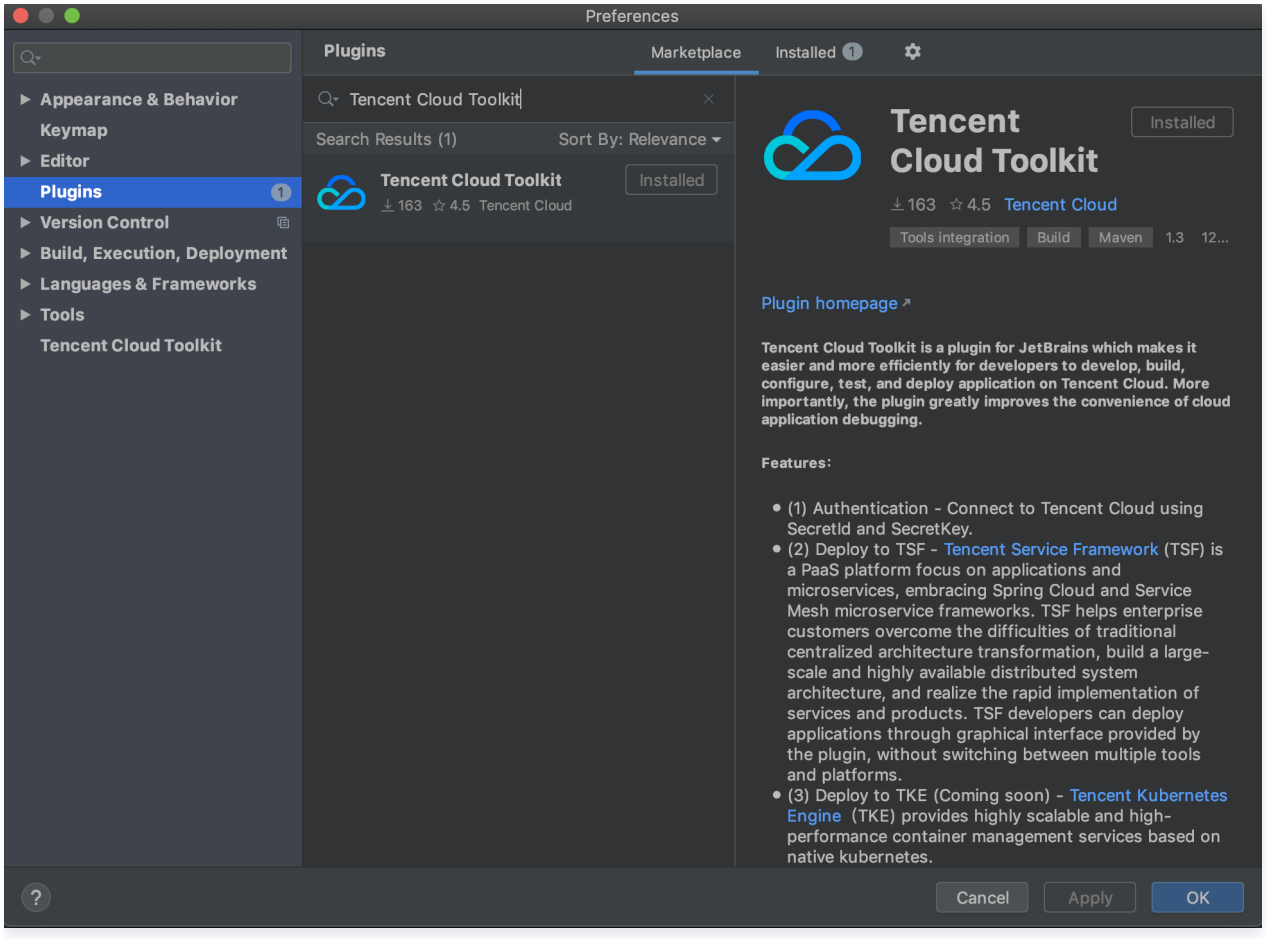
## 准备工作

- 下载并安装 JDK1.8或者更高的版本 。
- 下载并安装 IntelliJ IDEA（2019.2或更高版本）。

## 安装 Tencent Cloud Toolkit

您可以在 JetBrains 插件市场搜索 Tencent Cloud Toolkit 并下载安装。

1. 在插件市场中下载安装。
2. 在 IntelliJ IDEA 顶部菜单栏中选择 **IntelliJ IDEA** > **Preferences**。
3. 在 Preferences 对话框的左侧导航栏中单击 **Plugins**。
4. 在 Plugins 区域单击 **Marketplace**。
5. 在搜索栏中输入“Tencent Cloud Toolkit”，并单击 **install**。



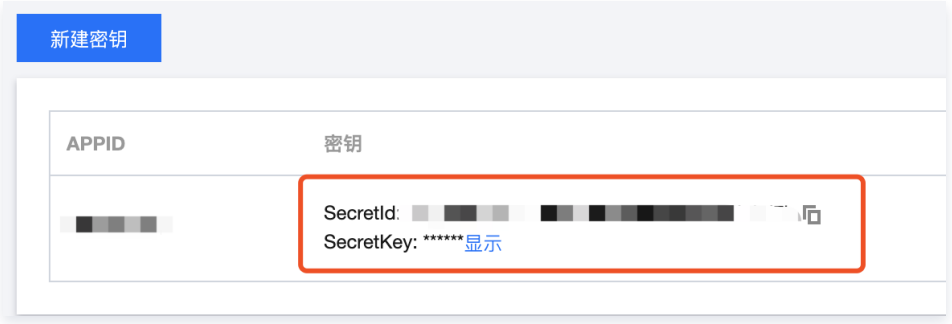

## 验证结果

IntelliJ IDEA 重启后，在菜单栏中选择 **IntelliJ IDEA** > **Preferences**，在 Preferences 对话框左侧的导航栏中可以看到 Tencent Cloud Toolkit，则说明安装成功。

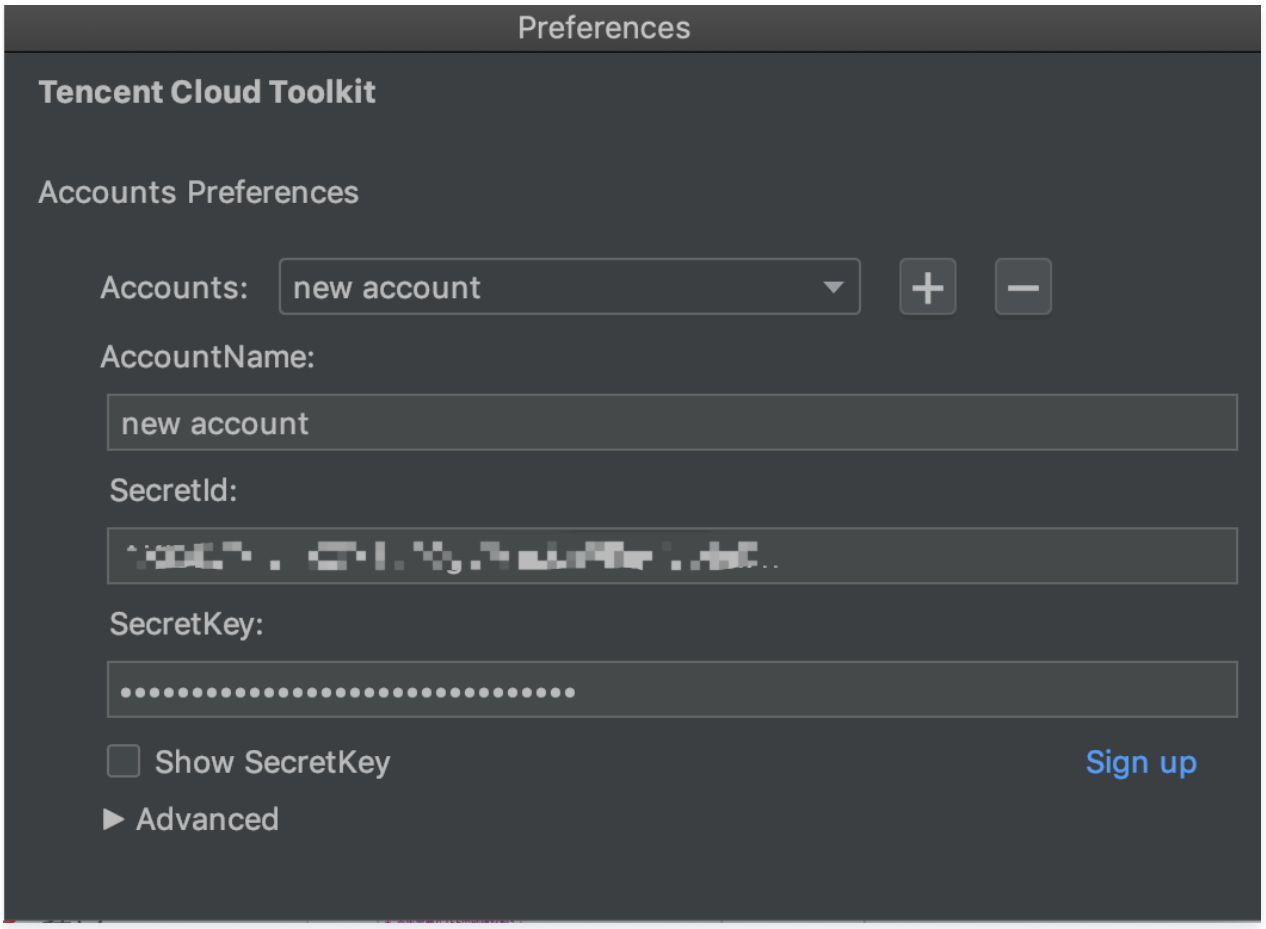
## 配置账户信息

使用 Tencent Toolkit 部署应用到云端时，需要调用 TSF 的 API，调用 API 时需要使用访问密钥（SecretId、SecretKey）进行云端身份验证。因此在部署应用之前，需要先在 Tencent Toolkit 中配置账户信息，操作步骤如下：

1. 获取 SecretId 和 SecretKey （如果未在腾讯云注册账号请先完成 注册 ）。
   1.1 登录 访问管理控制台 。
   1.2 在左侧导航栏中，单击**访问密钥** > **API密钥管理**。
   1.3 在密钥列表中，可以查看并复制 SecretId 和 SecretKey。您也可以新建密钥信息，详情请参见 访问密钥 文档。




2. 在 IntelliJ IDEA 菜单栏中选择 Preferences，在 Preferences 对话框左侧的导航栏中选择 Tencent Cloud Toolkit。
3. 配置 Account 、SecretId、 SecretKey 等信息。
   3.1 输入AccountName 、SecretId、 SecretKey（设置后为默认配置）。
   3.2 单击 **Advanced** 可以为不同环境配置不同的域名(默认为tencentcloudapi.com)。
   3.3 单击 **apply** 应用配置。




## 后续操作

完成安装和配置 Tencent Cloud Toolkit 后，即可 使用 Tencent Toolkit 部署应用 。

# 通过 Tencent Cloud Toolkit 部署应用

最近更新时间：2024-01-19 11:28:31

## 操作场景

使用 Tencent Cloud Toolkit 可以将应用快速部署到 TSF，适用于快速迭代更新、开发阶段快速验证等场景。在 IntelliJ IDEA 安装和配置 Tencent Cloud Toolkit 后，只需在配置界面设置部署参数即可实现自动化部署，支持虚拟机部署场景和容器部署场景。

## 前提条件

您需要先在 TSF 中创建好相关的集群（Cluster）、应用（Application）、部署组（Group），具体操作部署如下（如已有相关资源请跳过这一步骤）：

- 创建集群
- 创建应用
- 创建虚拟机部署组
- 创建容器部署组

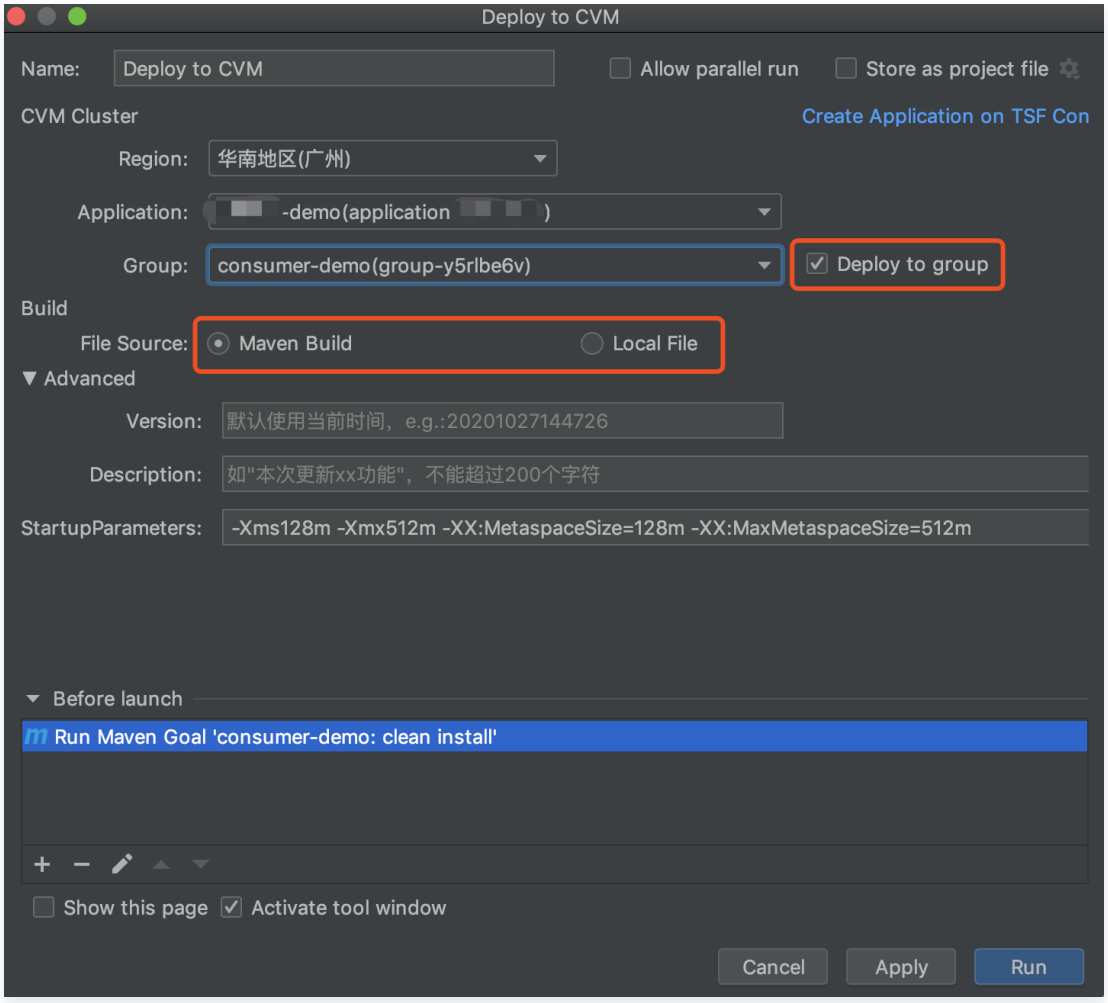
## 部署到 TSF 部署组

这里以部署 consumer-demo 为例，具体步骤如下：

### 虚拟机部署

1. 在 IntelliJ IDEA 中打开 consumer-demo 工程。
2. 在 IntelliJ IDEA 顶部菜单栏中选择 **Tools** > **Tencent Cloud Toolkit** > **Deploy to TSF for CVM Application...**。
3. 在 Deploy to CVM 对话框中部署配置。根据部署的程序包来源以及是否部署，插件提供三种选项：
   - Maven build：使用 maven 编译项目并打包、上传程序包至 TSF、部署。
   - Local file：选择本地文件上传至 TSF、部署。

- Deploy to group：是否执行部署，默认执行。如果取消勾选，则仅上传程序包至 TSF 程序包仓库。




**说明**

如果您尚未在 TSF 上创建应用，可在对话框右上角单击 Create application on TSF Console，跳转到 TSF 控制台创建应用。

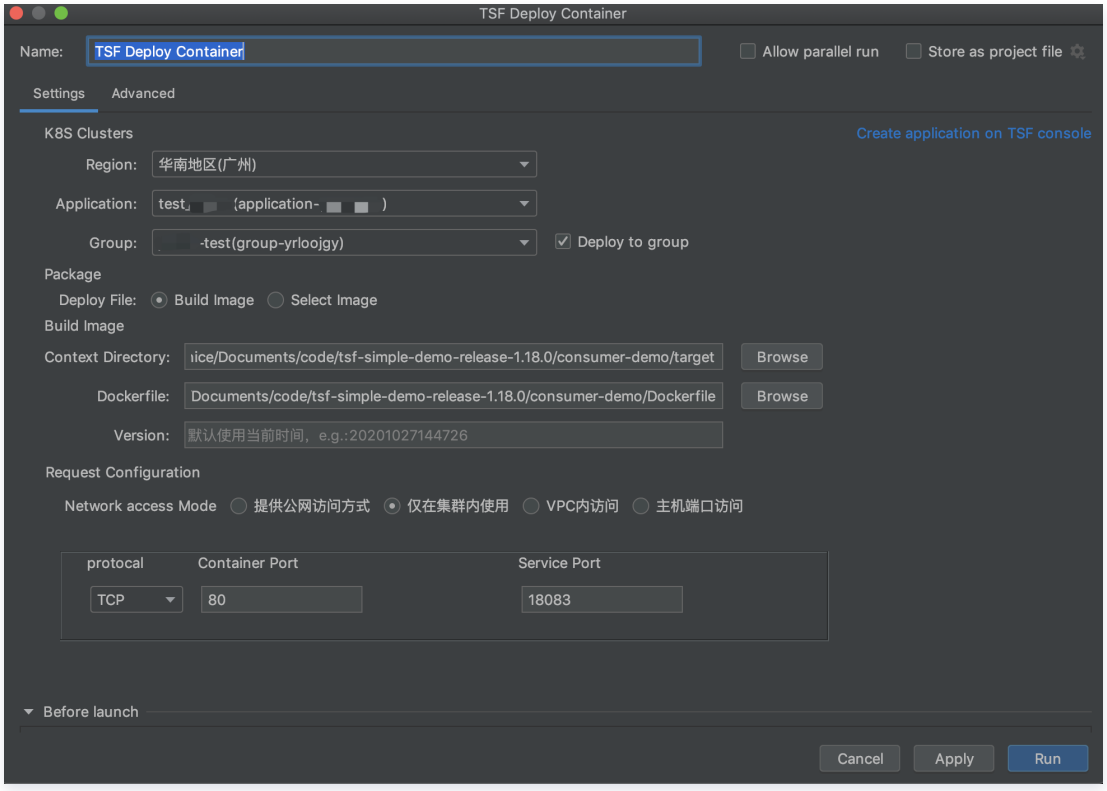
部署参数说明如下：

| 参数名 | 参数 | 描述 |
| --- | --- | --- |
| 地域 | Region | 服务所在地区。 |
| 应用信息 | Application | 服务所在应用。 |
| | Group | 服务所在部署组。 |
| 部署方式 | Maven Build | 若当前工程采用 Maven 构建，可以直接构建并部署。 |
| | Local File | 若当前工程不是采用 Maven 构建，或本地已存在部署文件的压缩包，则上传本地的部署文件即可。 |
| 其他 | Version | 部署版本，缺省使用时间戳作为版本号。 |
| | Description | 部署信息描述。 |
| | StartupParameters | 启动参数。 |

4. 先单击 Apply，然后单击 Run。
5. 在 TSF 平台即可查看部署结果。

## 容器部署

1. 在 IDEA 中打开 consumer-demo 工程。
2. 在 IntelliJ IDEA 顶部菜单栏中选择 **Tools** > **Tencent Cloud Toolkit** > **Deploy to TSF for Kubernetes Application...**。
3. 在 TSF Deploy Container 中部署配置，跟虚拟机部署一样，根据镜像来源以及是否部署，插件提供三种选项：
   - Build image：制作镜像、上传镜像至TSF镜像仓库、部署。
   - Select image：选择镜像仓库中的镜像部署。
   - Deploy to group：是否执行部署，默认执行。如果取消勾选，则仅上传镜像至 TSF 镜像仓库。




   - ServicePort：服务端口（该端口为服务启动时占用端口）。
   - ContainerPort：服务映射端口。
   - InstancePort：K8S 中使用 NodePort 方式部署时指定 NodePort 端口（端口范围：30000-32767）。这部分逻辑同 TSF。

Settings 页签部署参数配置说明如下：

| 参数名 | 参数 | 描述 |
| --- | --- | --- |
| 地域 | Region | 服务所在地区。 |
| 应用信息 | Application | 服务所在应用。 |
| | Group | 服务所在部署组。 |
| 部署方式 | Build Image | 制作镜像、上传镜像至 TSF 镜像仓库、部署。 |
| | Select Image | 选择镜像仓库中的镜像进行部署。 |
| 请求配置 | Network AccessMo | 部署方式。 |

| | | |
|---|---|---|
| | de | |
| | Port | ContainerPort：服务映射端口；ServicePort（应用服务启动端口）；IntancePort：K8S 中使用 NodePort 方式部署时指定 NodePort 端口（端口范围: 30000-32767）。 |
| 其他 | Version | 部署版本，缺省使用时间戳作为版本号。 |
| | Context Directory | Docker 命令执行目录，Dockerfile 文件中依赖的 context 目录。 |
| | Dockerfile | Dockerfile 所在文件目录。 |

4. Advanced 提供容器部署的相关高级参数设置，包括 pod 更新、调度规则、健康检查相关的配置。

| 参数名 | 参数 | 描述 |
|---|---|---|
| 资源配置 | ApplicationContainer | 配置应用所在 POD 占用的内存和 CPU 大小。 |
| 应用管理 | Update Method | POD 更新策略配置。 |
| | Schedule Rules | POD 调度策略。 |
| | HealthCheck | 就绪检查（Readiness），存活检查（Liveness）。 |
| 其他 | StartupParameters | 容器应用启动参数。 |
| | Envs | 容器应用环境变量。 |

> **说明：**
> 相关参数详情可参见 TSF 容器部署 。

5. 先单击 **Apply**，然后单击 **Run**。
6. 在 TSF 平台即可查看部署结果。

# 平台管理
# 权限管理
# 概述

最近更新时间：2022-06-10 11:22:24

为满足企业客户不同成员使用 TSF 平台的接口操作权限、资源操作权限的需求，当前腾讯微服务平台支持主账号为子账号灵活分配不同角色、不同数据集的权限。

## 操作场景

当前 TSF 可以支持以下几种场景的使用：

- 为子账号或协作者配置全部资源的全读写策略。
- 为子账号或协作者配置全部资源的部分操作权限，如可以为部分用户配置应用、微服务、配置的全读写策略，以及集群、命名空间的只读策略。
- 为子账号和协作者配置某些资源（一个或多个）的读或写权限，如可以为不同的子账号配置不同命名空间、不同应用的不可见、只读、全读写权限。

## 操作步骤

在子账号和协作者使用 TSF 平台之前，需要进行以下几个步骤：

1. 按照 访问管理 中，“子账号和协作者使用 TSF”部分文档进行操作，这一步保证了用户使用 TSF 时，TSF 访问 VPC、 TKE 等云资源的服务角色权限。
2. 按照 角色管理 、数据集管理 、授权管理 三个文档进行操作，配置子账号和协作者使用 TSF 平台时操作和数据集权限。

## 预设策略

TSF 提供了 QcloudTSFFullAccess 和 QcloudTSFReadOnlyAccess 两条预设策略。

| 策略名称 | 作用 |
| --- | --- |
| QcloudTSFReadOnlyAccess | 只读策略，包含 TSF 全部集群、命名空间、应用、服务等资源的全部只读操作 |
| QcloudTSFFullAccess | 全读写策略，包含 TSF 全部集群、命名空间、应用、服务等所有资源的全部操作 |

预设策略只限制用户操作，没有对数据进行过滤。当用户对数据过滤没有需求时，仅需绑定预设策略，无需进行上述使用步骤中的步骤2（步骤1还需要进行）。

> **说明**
>
> - 截止至2019年10月10日，TSF 现网所有子账号和协作者都默认绑定了 QcloudTSFFullAccess 权限，当用户需要精细化的配置操作和数据集权限时，需要解绑全读写策略 QcloudTSFFullAccess 并按照上述步骤2进行操作。
> - 当为用户绑定了多条策略时，多条策略以白名单的形式取并集生效。
> - QcloudTSFReadOnlyAccess 相当于配置一条只读角色操作权限和全数据集权限。当主账号想要设置某子账号对某些资源不可见时，不要绑定 QcloudTSFReadOnlyAccess 策略。

版权所有：腾讯云计算（北京）有限责任公司

# 角色管理

最近更新时间：2024-11-13 16:14:53

## 操作场景

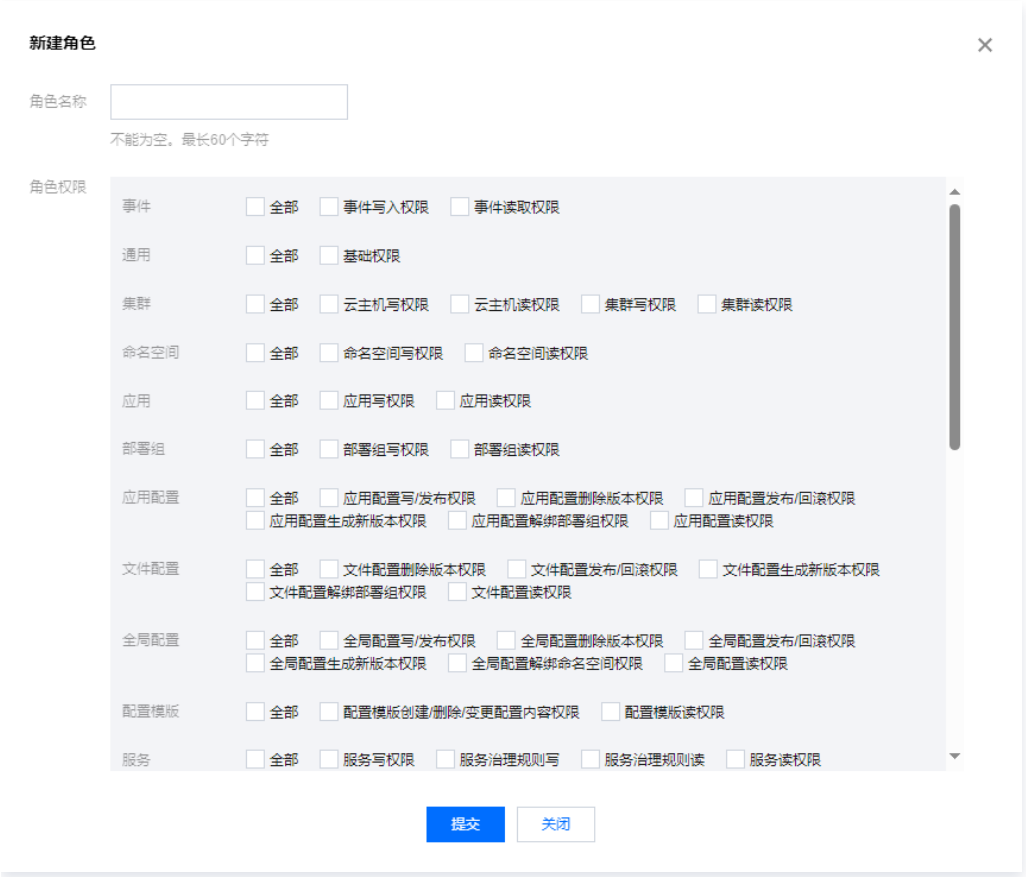
角色管理是配置子账号与协作者使用 TSF 的操作权限。当前 TSF 平台采取白名单机制，如子账号和协作者没有被授权任何角色，则无法使用 TSF 平台。

## 操作步骤

### 新建角色

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击**权限管理**，进入**角色管理**标签页。
3. 在角色管理页面，单击页面左上角的**新建角色**。
4. 在新建角色页面，填写角色名称并选择角色权限。
5. 单击**提交**，完成角色创建。

> **说明：**
> - 角色权限中的通用权限包含的 TSF 访问概览页面、监控等页面的基本操作，建议对所有用户授权。
> - 日志告警、服务告警、部署组告警等告警策略的配置在腾讯云可观测平台页面配置，不受 TSF 权限管控。
> - 镜像仓库列表不受 TSF 权限管控。




### 编辑角色

1. 登录 TSF 控制台 。

2. 在左侧导航栏，单击**管理中心**下的**角色管理**。
3. 单击选择目标角色。
4. 单击右上角的**编辑**即可对角色进行编辑。

## 删除角色

> ⚠ **注意：**
> 当已经给某用户绑定角色和策略组后，删除角色会导致策略不可用，请谨慎删除。

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击**管理中心**下的**角色管理**。
3. 将鼠标悬停在目标角色上，单击角色名称右侧的**删除**即可删除该角色。

# 数据集管理

最近更新时间：2024-11-13 16:14:53

## 操作场景

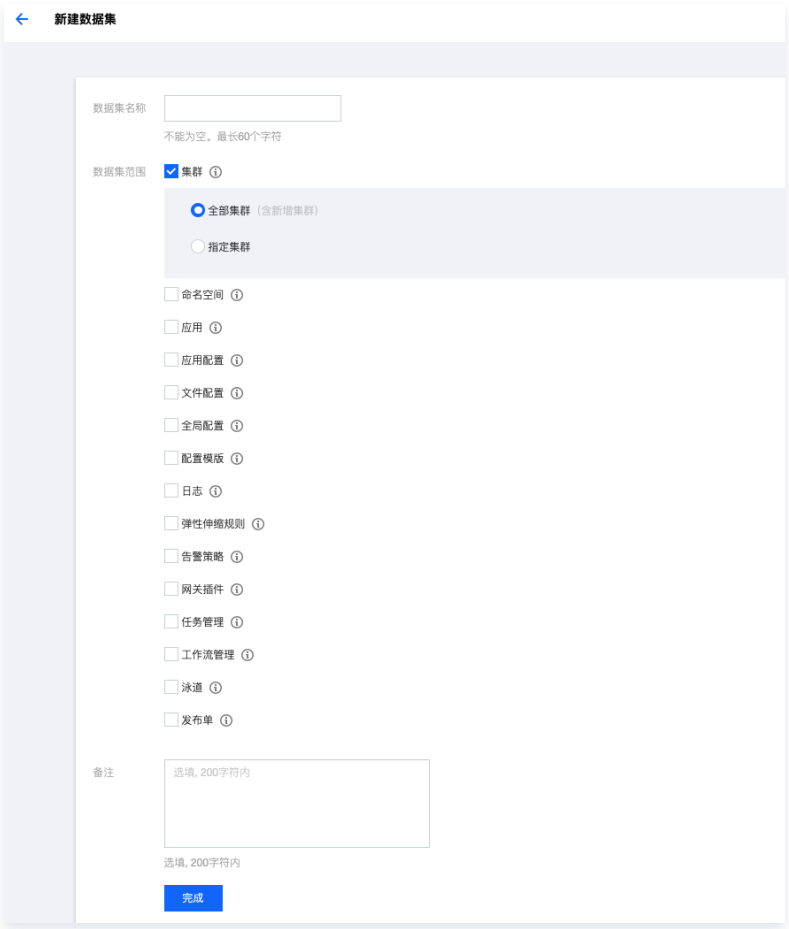
用户可以通过数据集管理配置不同的子账号和协作者使用不同资源。数据集是白名单机制，未配置数据集会导致无法看到 TSF 上任何资源。

## 操作步骤

### 新建数据集

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击**权限管理**下的**数据集管理**标签页。
3. 在数据集管理页面，单击左上角的**新建数据集**。
4. 填写数据集名称和备注，勾选资源范围。
5. 单击**完成**，完成数据集创建。

> **说明：**
> - 全部资源代表的是当前全部数据以及未来可能新增的资源。如果仅将指定集群权限授予某个用户，则新增集群对用户不可见。
> - 用户对某一类资源不勾选时，该类别资源对被授权的用户不可用。




### 编辑数据集

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击**管理中心**下的**数据集管理**。

3. 单击目标数据集进入编辑页面。
4. 单击基本信息卡片右侧的**编辑**修改数据集名称，单击资源范围卡片右侧的**编辑**修改数据集权限。

## 删除数据集

> ⚠ **注意：**
> 当某个数据集已经对用户授权后，删除数据集将导致授权策略对用户不生效。

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击**管理中心**下的**数据集管理**。
3. 选择目标数据集，单击操作列的**删除**删除该数据集。

# 授权管理

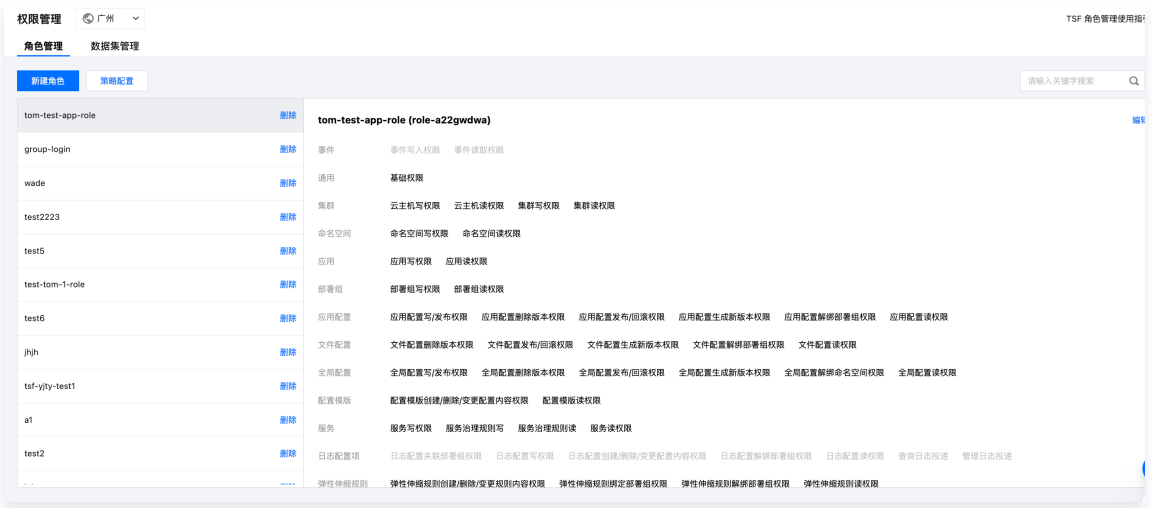
最近更新时间：2025-03-28 14:41:42

## 操作场景

该任务指导您通过 TSF 控制台配置并生成策略，并通过访问管理（CAM）将该策略管理用户/用户组。

## 操作步骤

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击**权限管理**下的**角色管理**。
3. 单击**策略配置**，进入策略生成器页面。




4. 配置策略名称和权限。
   - 策略名称：支持英文字母、数字、以及符号“+ = , . @ _ -”，限制50个字符以内。
   - 权限配置：选择对应角色与资源组。示例：当用户希望对集群1配置全读写权限，对集群2配置只读策略时，可以将包含全读写集群的角色与包含集群1的数据集关联作为权限配置的第一条数据，将包含集群只读的角色与包含集群2的数据集关联作为第二条数据。
5. 单击**前往 CAM 进行授权**，并在访问管理产品页面单击**创建策略**。

> **注意**
> 为避免策略不生效，建议不要在此处修改策略脚本。

6. 将生成的策略 关联用户/用户组 。
7. 单击**确定**，完成授权管理。

# 标签管理

最近更新时间：2023-11-03 10:42:04

## 使用场景

标签是腾讯云提供的用于标识云上资源的标记，是一个键-值对（Key-Value）。
您可以根据各种维度（例如业务、用途、负责人等）使用标签对TSF中的集群、命名空间和应用资源进行分类管理，通过标签非常方便地筛选过滤出对应的资源。标签键值对对腾讯云没有任何语义意义，会严格按字符串进行解析匹配。

## 使用限制

标签数量限制如下：

- 资源维度：一个资源最多 50 个不同的 key。
- 标签维度：单个用户最多 1000 个不同的 key。
- 一个 key 最多有 1000 个 value。

其他标签命名相关限制请参见 使用限制 。

## 使用案例

以下案例以集群为例进行说明，命名空间和应用的标签使用方法类似，可以选择在创建资源时设置标签或者在编辑标签中设置标签。

### 案例描述

某公司在腾讯云上拥有5个集群，这5个集群的使用部门、业务范围以及负责人的信息如下：

| 集群/ID | 使用部门 | 业务范围 | 负责人 |
| --- | --- | --- | --- |
| Cluster-abcdefg1 | 电商 | 营销活动 | 张三 |
| Cluster-abcdefg2 | 电商 | 营销活动 | 王五 |
| Cluster-abcdefg3 | 游戏 | 游戏 A | 李四 |
| Cluster-abcdefg4 | 游戏 | 游戏 B | 张三 |
| Cluster-abcdefg5 | 娱乐 | 前端制作 | 王五 |

以 Cluster-abcdefg1 为例，我们可以给该集群添加以下三组标签：

| 标签键 | 标签值 |
| --- | --- |
| dept | ecommerce |
| business | mkt |
| owner | zhangsan |

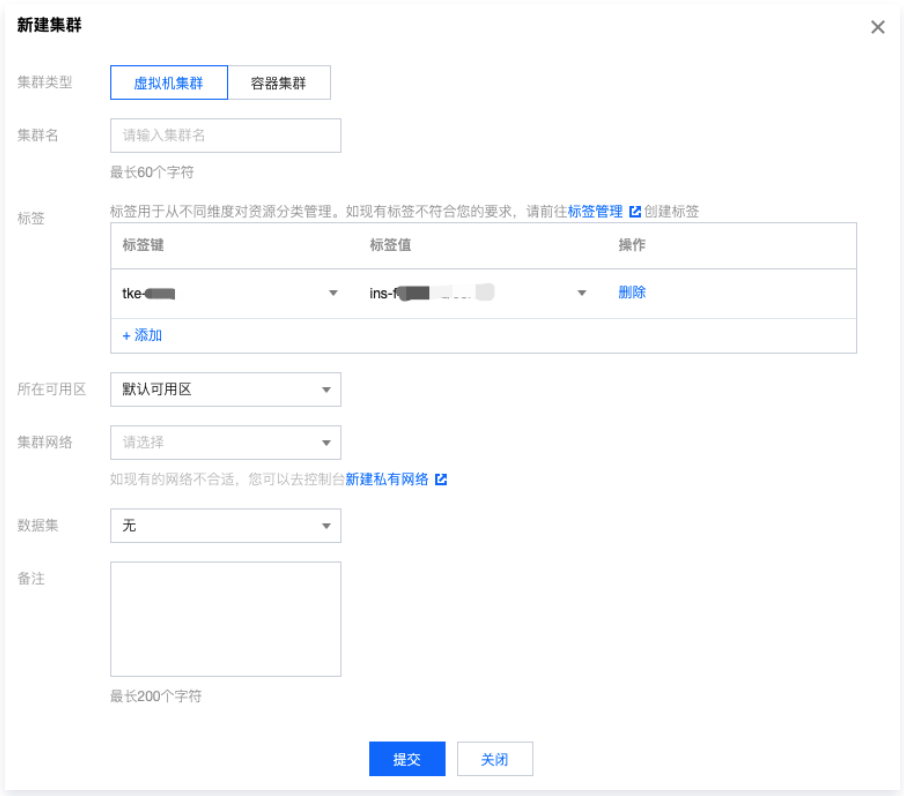
类似的，其他实例也可以根据其使用部门、业务范围和负责人的不同设置其对应的标签。

### 在 TSF 控制台上设置标签

以上文场景为例，当您完成标签键和标签值的设计后，可以登录TSF控制台进行标签的设置。

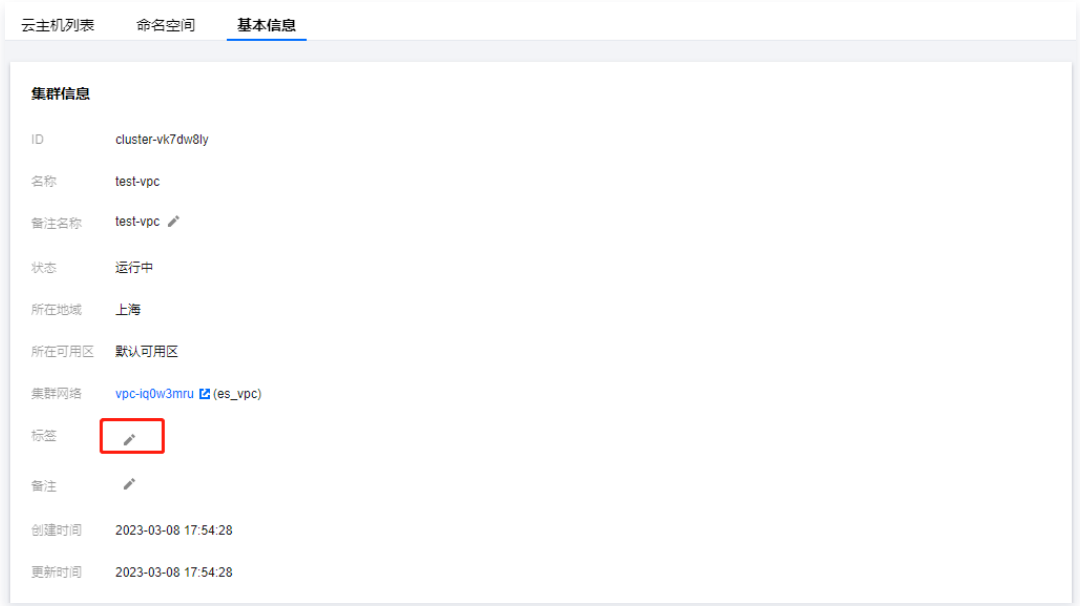
1. 登录 TSF 控制台 。

2. 在左侧导航栏选择**集群**，单击**新建集群**，在新建集群页面添加标签。




> **说明**
> 若现有标签不符合要求，可以单击**标签管理**创建标签。

3. 在集群列表，单击目标实例 ID，进入集群详情，在**基本详情 > 标签**，编辑标签。



4. 在弹出的窗口中，可以添加、删除或者修改标签。




## 通过标签键筛选实例

1. 在搜索框中，选择**标签**。
2. 在**标签：**后输入标签或者选择标签，单击 进行搜索。您可以同时根据多个标签键进行筛选。例如，输入 `标签键：key1|key2` 可筛选出绑定了标签键 `key1` 或 `key2` 的实例。



# 注册配置治理

最近更新时间：2025-04-24 10:32:43

注册配置治理完全兼容开源版本的使用方式，在功能、可用性和可运维性等多个方面进行增强，助力您快速构建微服务架构。支持无改造、无缝平滑迁移、多开发语言接入等特性，助力您的业务轻松快速上云。详细的操作方法可参见 注册配置治理 相关操作文档（以 Polaris（北极星）为例）。

# 云原生 API 网关
# 云原生 API 网关概述

最近更新时间：2025-06-09 16:07:01

**云原生 API 网关**是腾讯云基于开源微服务网关推出的一款**高性能高可用**的云原生 API 网关产品，减少用户自建网关的开发及运维成本。云原生 API 网关作为云上微服务架构的流量入口，集成请求分发、API 管理、流量监控、访问限制等功能，是微服务架构中的重要组件。
微服务平台 TSF 提供基于开源 Kong 的云原生 API 网关服务，结合灰度分流、流量限制、安全控制等场景，提供基于微服务体系和云原生体系的可插拔的网关功能。

> **说明：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。如果您有新购需求，请 提交工单 处理。

## 产品特色

- **兼容开源网关：**开源网关内核，兼容多个版本，开箱即用，降低自建复杂度。
- **高性能高可靠：**提供多种规格的实例，支持无缝扩缩容，轻松应对流量洪峰。
- **高性价比：**采用腾讯云底层资源，由腾讯云托管网关，无自建网关的人力成本及运维成本。
- **腾讯云特色功能：**提供与腾讯云环境、腾讯云产品的标准化对接方式，支持腾讯云开发的特色网关插件。

## 业务场景




## 功能特性

| 功能点 | 说明 |
| --- | --- |
| 数据可视化 | 在腾讯云控制台可查看云原生 API 网关的集群信息，可对集群配置、系统参数进行修改。云原生 API 网关提供控制台与 Konga 控制台两种路由管理方式，可在 Konga 控制台操作 Kong 网关的 Service、Upstream、Plugins 等资源，使用体验与开源一致 |
| 批量修改系统参数 | 支持在腾讯云控制台批量调整云原生 API 网关的系统参数，如请求体大小、TLS 版本等不停服、安全、可靠 |
| 日志监控 | 支持 CPU 利用率、内存使用率、入带宽流量、出带宽流量等底层资源的监控指标支持通过插件将监控数据输出到 Prometheus 监控等开源平台或腾讯云监控平台与腾讯云日志系统 CLS 高度集成，可直接在云上分析日志 |
| 不停服扩缩容 | 集成腾讯云 CLB 的探活、负载均衡能力，进行自动节点故障流量切换支持灵活升降配、支持按需扩展，过程客户透明、支持资源规格扩缩容、支持副本数扩缩容 |

版权所有：腾讯云计算（北京）有限责任公司

| | |
|---|---|
| **IP 黑白名单** | 云原生 API 网关中支持创建 IP 黑白名单，名单中可配置 IP 或网段，对访问进行限制 |
| **容灾备份** | 提供高安全可靠性的网关托管，可备份网关数据和多可用区 |
| **免运维** | 由腾讯云托管网关，无自建网关的人力成本及运维成本 |
| **轻松迁移** | 使用迁移插件和官方工具，无门槛平滑迁移，开箱即用 |

版权所有：腾讯云计算（北京）有限责任公司

# 快速入门
# 使用云原生 API 网关访问 HTTP 服务

最近更新时间：2025-06-09 16:07:01

本文将快速引导您如何在云原生 API 网关中，访问后端服务为 HTTP 类型的 API。

> **注意：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。

## 概述

您需要依次完成以下步骤：

1. 创建云原生 API 网关
2. 定义后端服务信息
3. 为服务配置路由规则
4. 调用 API

## 操作步骤

### 步骤1：创建云原生 API 网关实例

参见 新建网关，创建一个云原生 API 网关。

### 步骤2：定义后端服务信息

1. 单击创建好的网关实例的“ID”，进入实例详情页。
2. 在顶部菜单栏选择**服务路由 > 服务**，在服务列表单击**新建**。
3. 在新建服务窗口中配置相关参数后，单击**提交**。

> **说明：**
> 本步骤是指定在云原生 API 网关收到 Client 端的请求后，转发到哪个后端服务。本例中的后端服务可以为腾讯云上或在互联网上的任何可以访问到的地址。




### 步骤3：为后端服务配置路由

1. 后端服务添加完成后，点击后端服务名称进入服务路由页面。
2. 单击**新建**，设置好路由信息，单击下一步。

> **说明：**
> 请求方法、请求路径、Host 需至少配置一种。

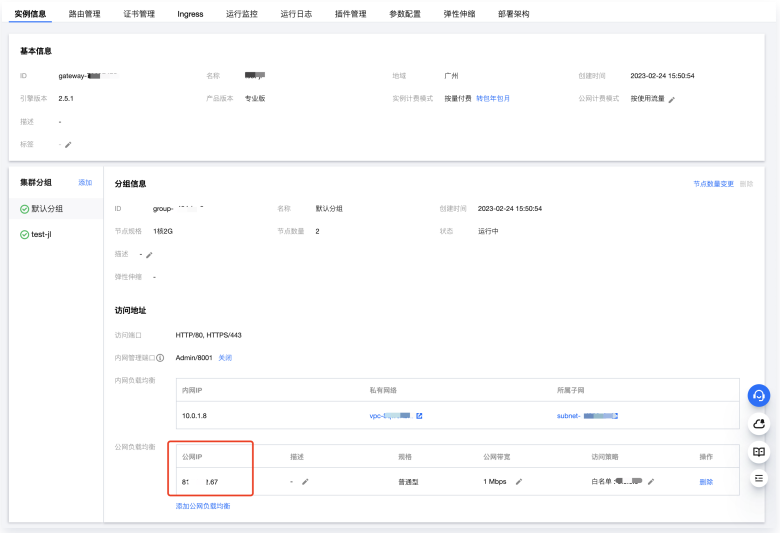



3. 配置路由后端信息，确认后端服务信息无误后单击**提交**，完成路由创建。

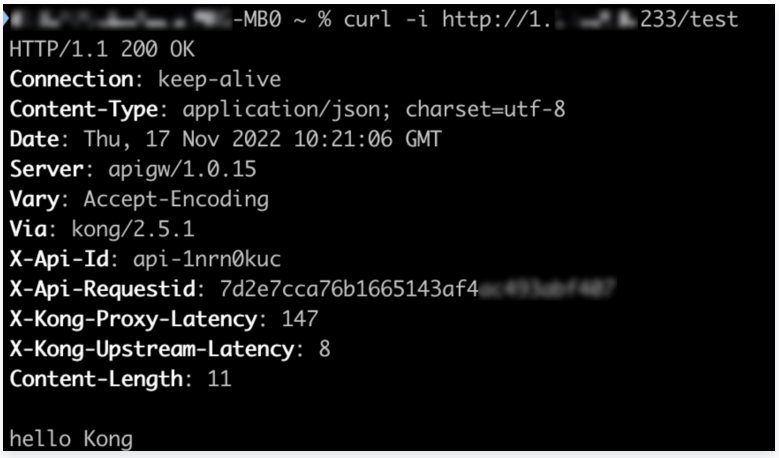



## 步骤4：调用 API

1. 进入云原生 API 网关的基本信息页面，获取负载均衡地址。




2. 使用负载均衡地址和配置的路由路径访问 API，执行 curl 命令测试，已通过云原生 API 网关访问到后端服务。



# 使用云原生 API 网关访问 TKE 服务

最近更新时间：2025-06-09 16:07:01

## 概述

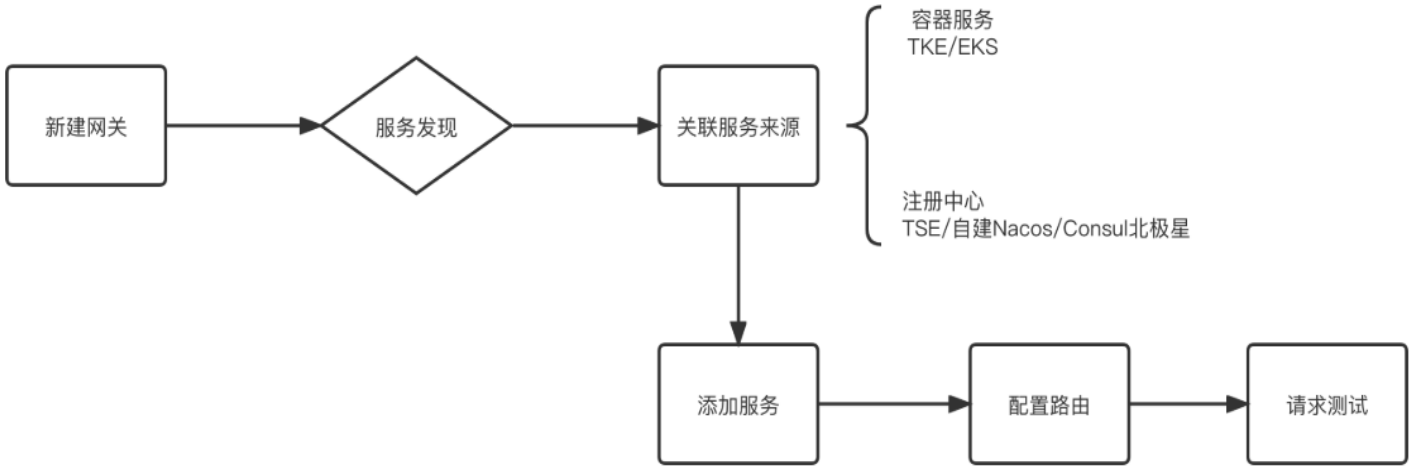
您可以为容器服务创建一个云原生 API 网关，从注册中心或容器服务 TKE 关联服务，然后在网关中为服务创建路由策略，以便该服务通过网关对外提供服务。本指导以容器服务为例，您需要完成以下步骤：

1. 创建云原生 API 网关
2. 关联服务来源
3. 添加服务
4. 为服务配置路由规则
5. 调用 API

> **注意：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。

## 体验流程

微服务如果部署到容器服务 TKE，云原生 API 网关可直接从 TKE 关联服务。




## 操作步骤

### 步骤1：创建云原生 API 网关

参见 新建网关，创建一个云原生 API 网关。

### 步骤2：关联服务来源

1. 单击创建好的云原生 API 网关的 ID，进入基本信息页面。
2. 在左侧菜单栏选择**服务路由 > 服务来源**，在服务来源单击**新建**，服务来源类型选择**容器服务**，服务实例选择与云原生 API 网关 在同一 VPC 下的可用实例，单击**提交**。




## 步骤3：添加服务

1. 选择**服务路由 > 服务**，在服务列表单击**新建**，服务类型选择 K8S 服务，服务来源选择上一步关联的服务来源。填写容器服务的命名空间名称和服务名称。




2. 单击**提交**，创建完成后可以看到服务列表中服务的信息。

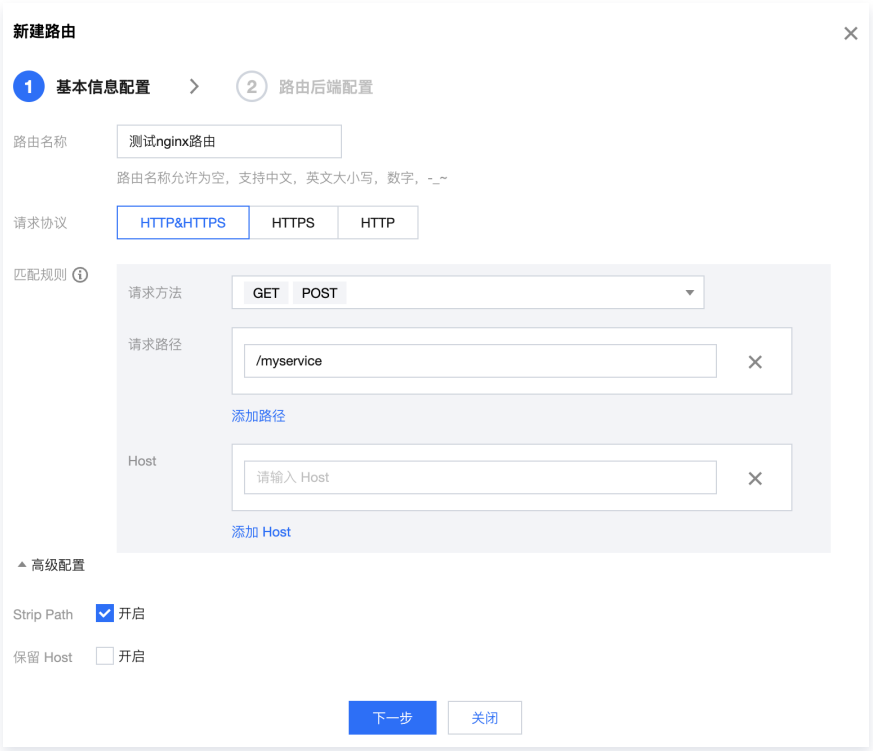



## 步骤4：为服务配置路由规则

1. 后端服务添加完成后，单击服务名称进入服务路由页面。
2. 单击**新建**，设置好路由信息，单击下一步。

> **说明：**

> 请求方法、请求路径、Host 需至少配置一种。

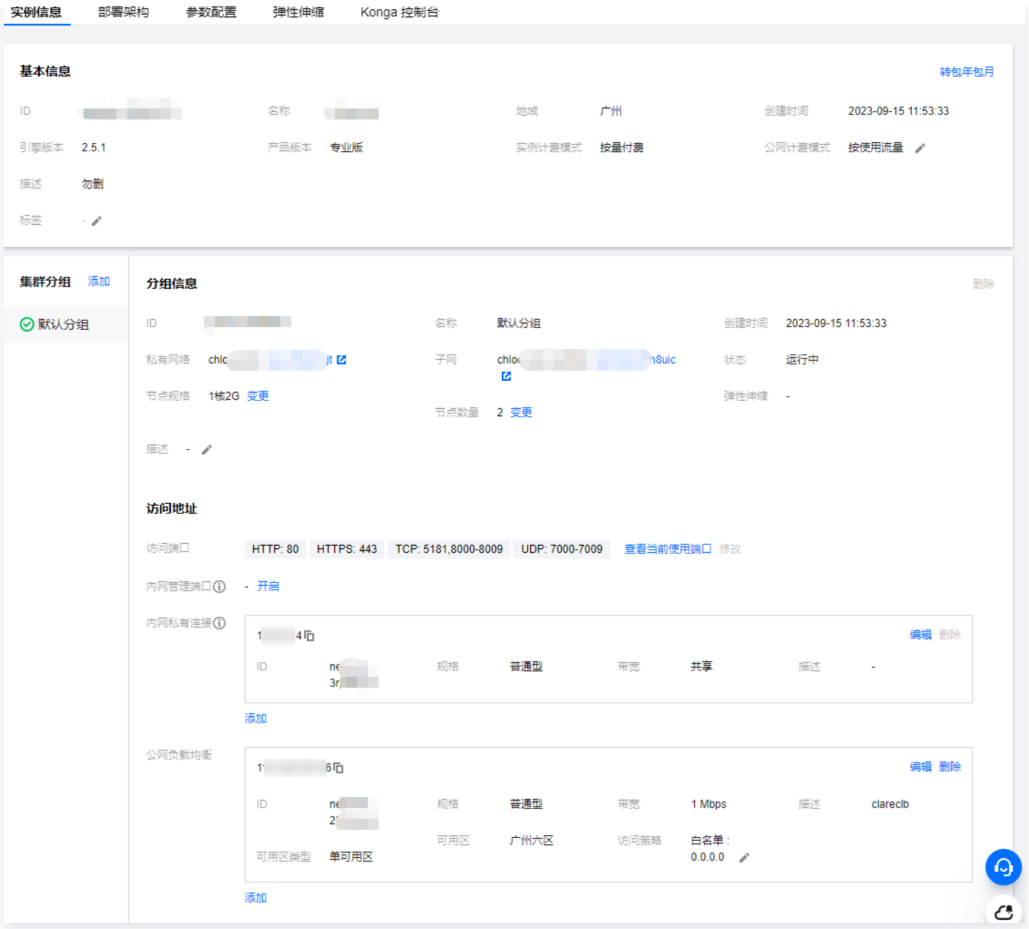



3. 配置路由后端信息，确认后端服务信息无误后点击**确定**，完成路由创建。

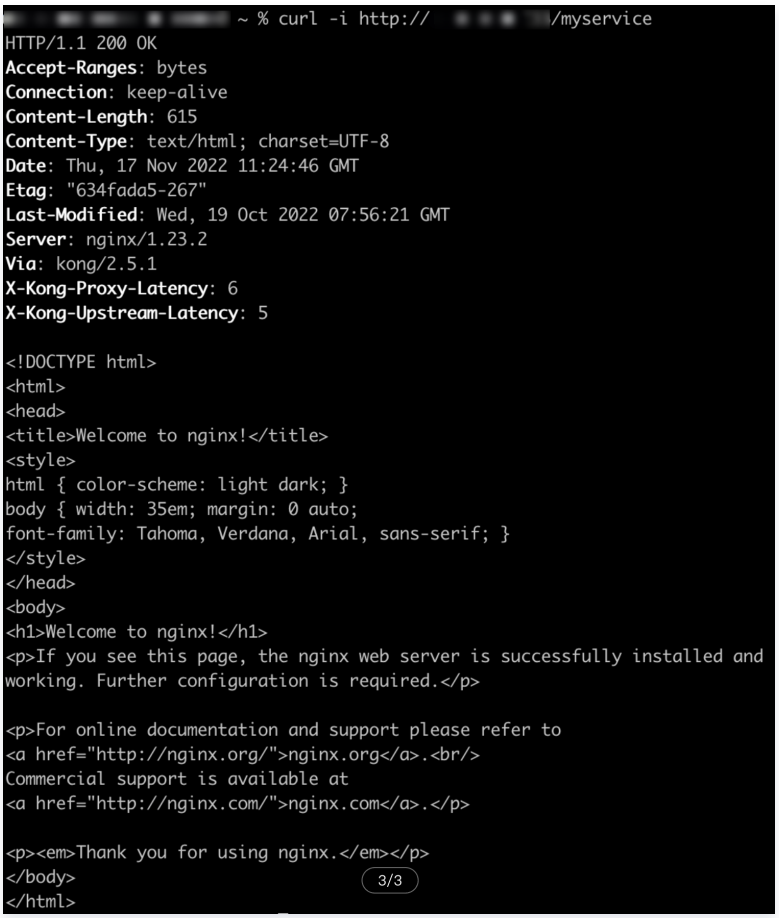



## 步骤5：调用 API

1. 进入云原生 API 网关的基本信息页面，获取负载均衡地址。




2. 使用负载均衡地址和配置的路由路径访问 API，执行 curl 命令测试，已通过 云原生 API 网关访问到容器服务。


~ % curl -i http://  /myservice
HTTP/1.1 200 OK
Accept-Ranges: bytes
Connection: keep-alive
Content-Length: 615
Content-Type: text/html; charset=UTF-8
Date: Thu, 17 Nov 2022 11:24:46 GMT
Etag: "634fada5-267"
Last-Modified: Wed, 19 Oct 2022 07:56:21 GMT
Server: nginx/1.23.2
Via: kong/2.5.1
X-Kong-Proxy-Latency: 6
X-Kong-Upstream-Latency: 5
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.
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>.
Thank you for using nginx.
</body>
3/3
</html>

# 使用云原生 API 网关访问 Nacos 服务

最近更新时间：2025-06-09 16:07:01

## 概述

您可以为注册中心服务创建一个云原生 API 网关，从注册中心关联服务，然后在网关中为服务创建路由策略，以便该服务通过网关对外提供服务。本指导以 Nacos 服务为例，您需要完成以下步骤：

1. 创建云原生 API 网关
2. 关联服务来源
3. 添加服务
4. 为服务配置路由规则
5. 调用 API

> **注意：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。

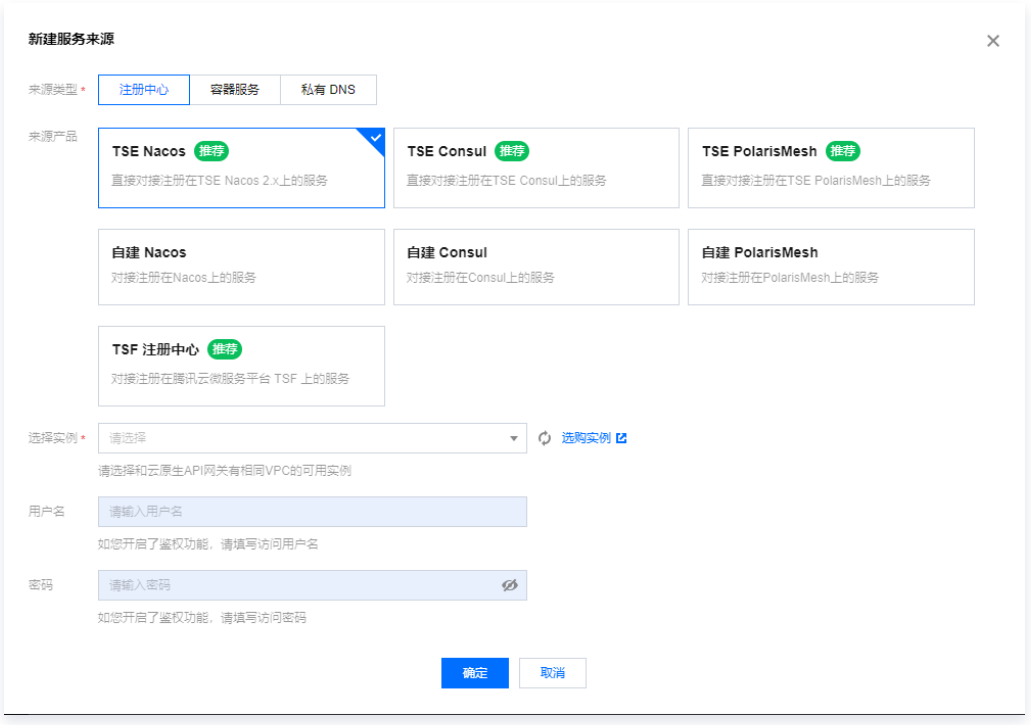
## 操作步骤

### 步骤1：创建云原生 API 网关

参见 新建网关，创建一个云原生 API 网关。

### 步骤2：关联服务来源

1. 单击创建好的云原生 API 网关的 ID，进入基本信息页面。
2. 在左侧菜单栏选择**服务路由 > 服务来源**，在服务来源单击**新建**，服务来源类型选择**注册中心**，来源产品选择 **TSE Nacos**，服务实例选择与云原生 API 网关在同一 VPC 下的可用实例，单击**提交**。




### 步骤3：添加服务

1. 选择**服务路由 > 服务**，在服务列表单击**新建**，服务类型选择**注册中心**，服务来源选择上一步关联的服务来源。填写 Nacos 服务的命名空间 ID 和服务名称。




> **注意：**
> 1. Nacos 默认命名空间请填写 empty_ns，其余命名空间请填写命名空间 ID。
> 2. 服务列表支持填写多个服务，多个服务请使用换行分隔。
> 3. 服务填写规范为：{Nacos 分组}__{Nacos 服务名称}，Nacos 分组与服务名称之间使用两个下划线(_)分隔。

2. 单击**提交**，创建完成后可以看到服务列表中服务的信息。

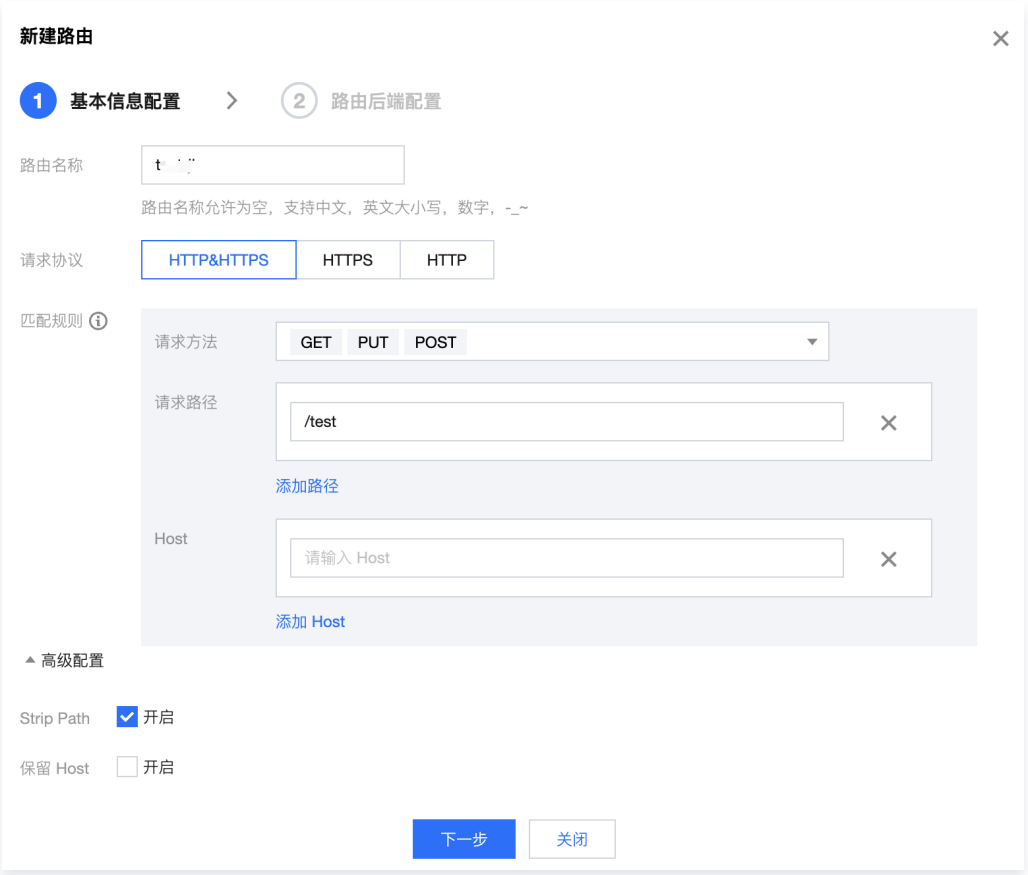



## 步骤4：为服务配置路由规则

1. 后端服务添加完成后，单击服务名称进入服务路由页面。
2. 单击**新建**，设置好路由信息，单击下一步。

> **说明：**

> 请求方法、请求路径、Host 需至少配置一种。

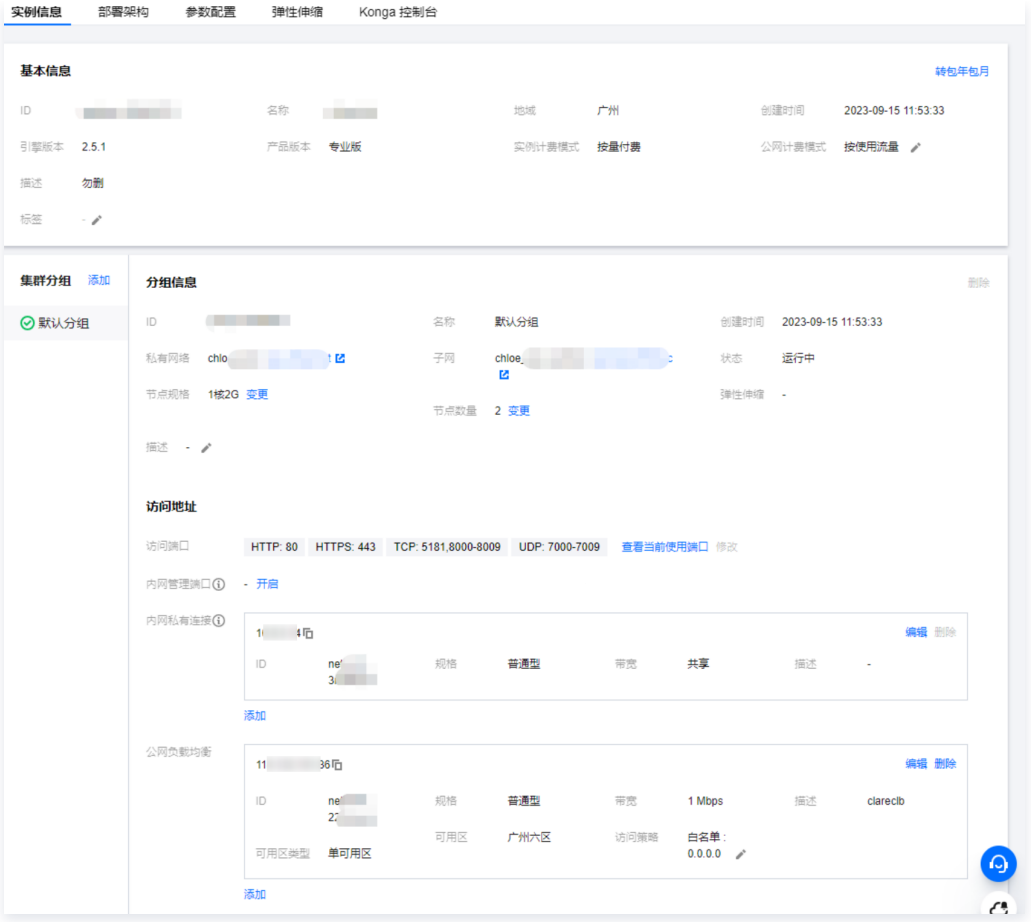



3. 配置路由后端信息，确认后端服务信息无误后单击**确定**，完成路由创建。




## 步骤5：调用 API

1. 进入云原生 API 网关的基本信息页面，获取负载均衡地址。




2. 使用负载均衡地址和配置的路由路径访问 API，执行 curl 命令测试，已通过**云原生API网关**访问到后端服务。

# 使用云原生 API 网关访问 TSF 服务

最近更新时间：2025-06-09 16:07:01

## 概述

您可以为注册中心服务创建一个云原生 API 网关，从注册中心关联服务，然后在网关中为服务创建路由策略，以便该服务通过网关对外提供服务。本指导以 TSF 注册中心为例，您需要完成以下步骤：

1. 创建云原生 API 网关
2. 关联服务来源
3. 添加服务
4. 为服务配置路由规则
5. 调用 API

> **注意：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。

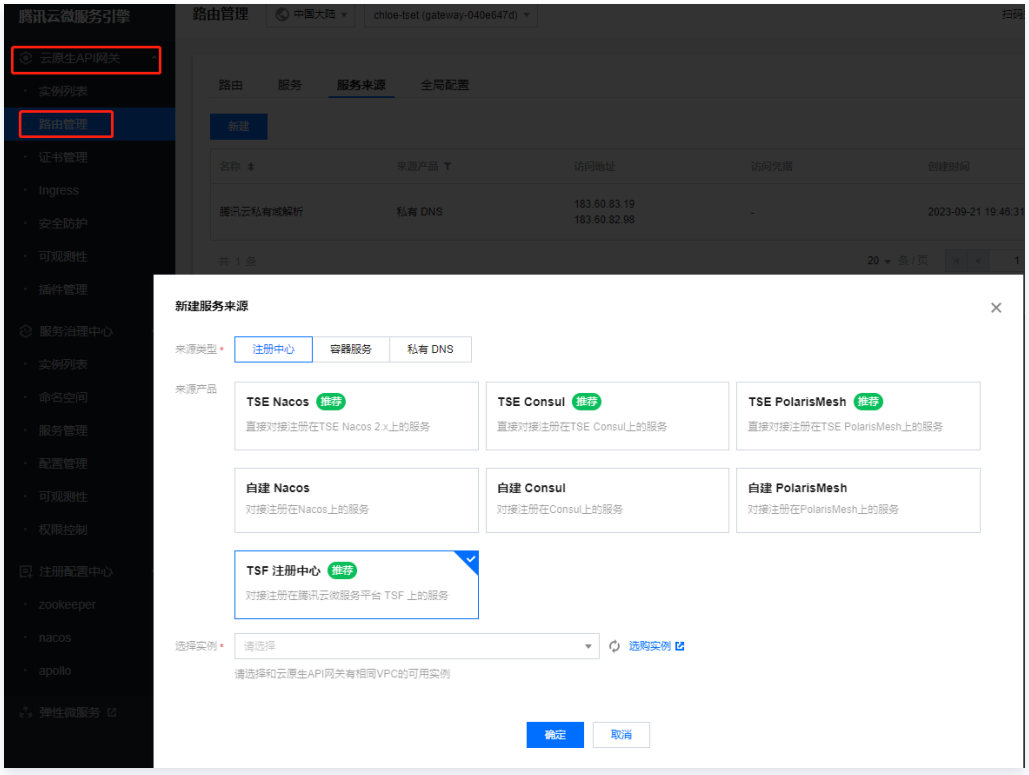
## 操作步骤

### 步骤1：创建云原生 API 网关

参见 新建网关，创建一个云原生 API 网关。

### 步骤2：关联服务来源

1. 单击创建好的云原生 API 网关的 ID，进入基本信息页面。
2. 在左侧菜单栏选择**服务路由** > **服务来源**，在服务来源单击**新建**，服务来源类型选择**注册中心**，来源产品选择 **TSF 注册中心**，服务实例选择与云原生 API 网关在同一 VPC 下的可用实例，单击**确定**。




### 步骤3：添加服务

1. 选择**服务路由 > 服务**，在服务列表单击**新建**，服务类型选择**注册中心**，服务来源选择上一步关联的 TSF 注册中心，下拉选择命名空间和服务。




2. 单击**提交**，创建完成后可以看到服务列表中服务的信息。




## 步骤4：为服务配置路由规则

1. 后端服务添加完成后，单击服务名称进入服务路由页面。
2. 单击**新建**，设置好路由信息，单击下一步。

> **说明：**
> 请求方法、请求路径、Host 需至少配置一种。

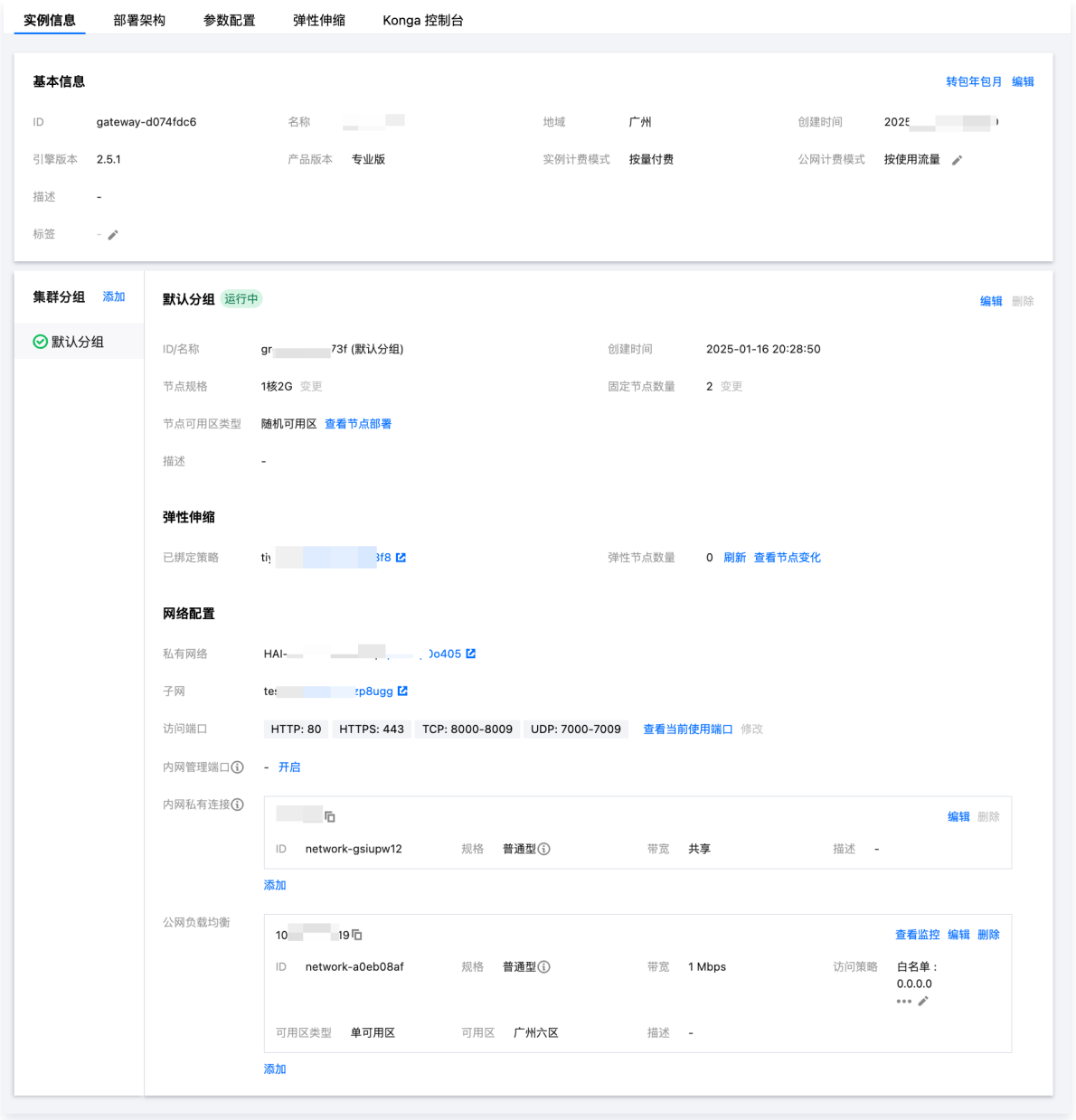



3. 配置路由后端信息，确认后端服务信息无误后单击**确定**，完成路由创建。




## 步骤5：调用 API

1. 进入云原生 API 网关的基本信息页面，获取负载均衡地址。




2. 使用负载均衡地址和配置的路由路径访问 API，执行 curl 命令测试，已通过云原生API 网关访问到后端服务。

# 使用云原生 API 网关访问 PolarisMesh

最近更新时间：2025-06-09 16:07:01

## 概述

您可以为北极星实例创建一个云原生 API 网关，从注册中心关联服务，然后在网关中为服务创建路由策略，以便该服务通过网关对外提供服务。本指导以 TSE PolarisMesh 为例，您需要完成以下步骤：

1. 创建云原生 API 网关
2. 关联服务来源
3. 添加服务
4. 为服务配置路由规则
5. 调用 API

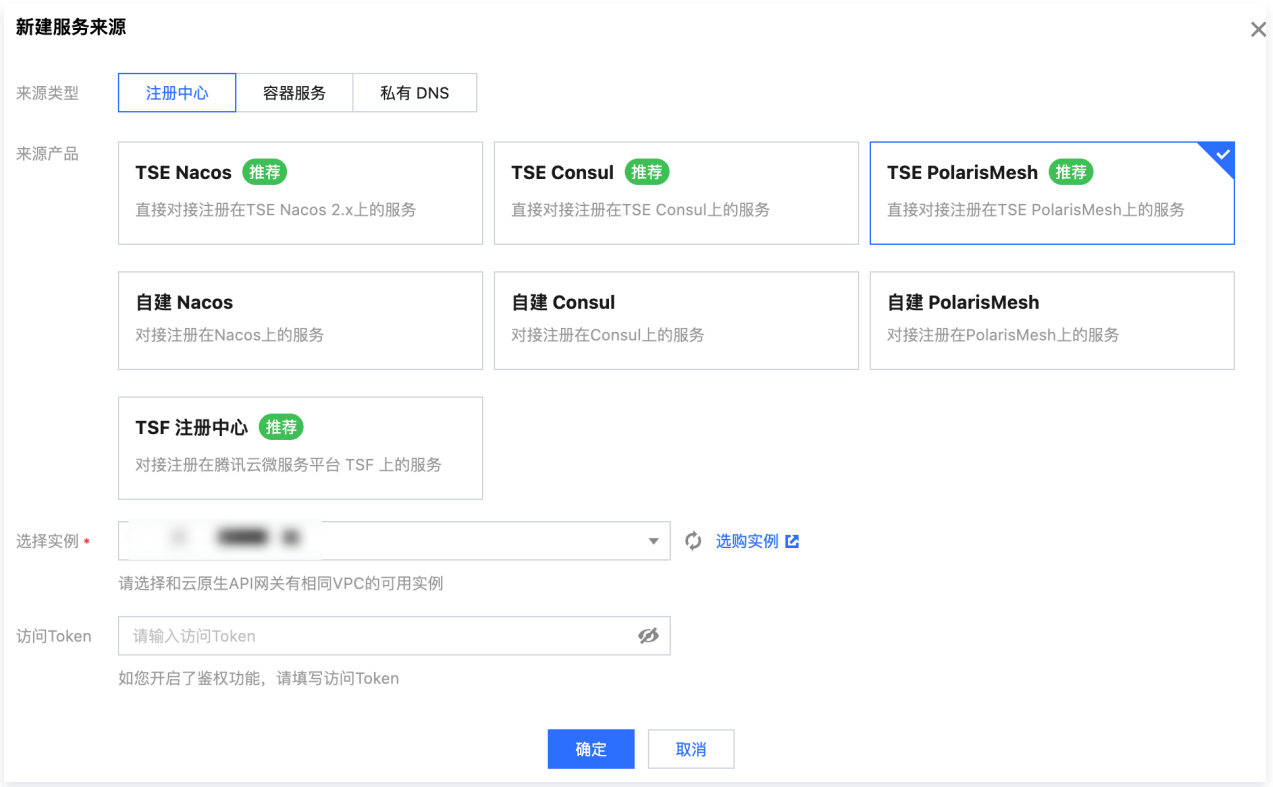
## 操作步骤

### 步骤1：创建云原生 API 网关

参见 新建网关，创建一个云原生 API 网关。

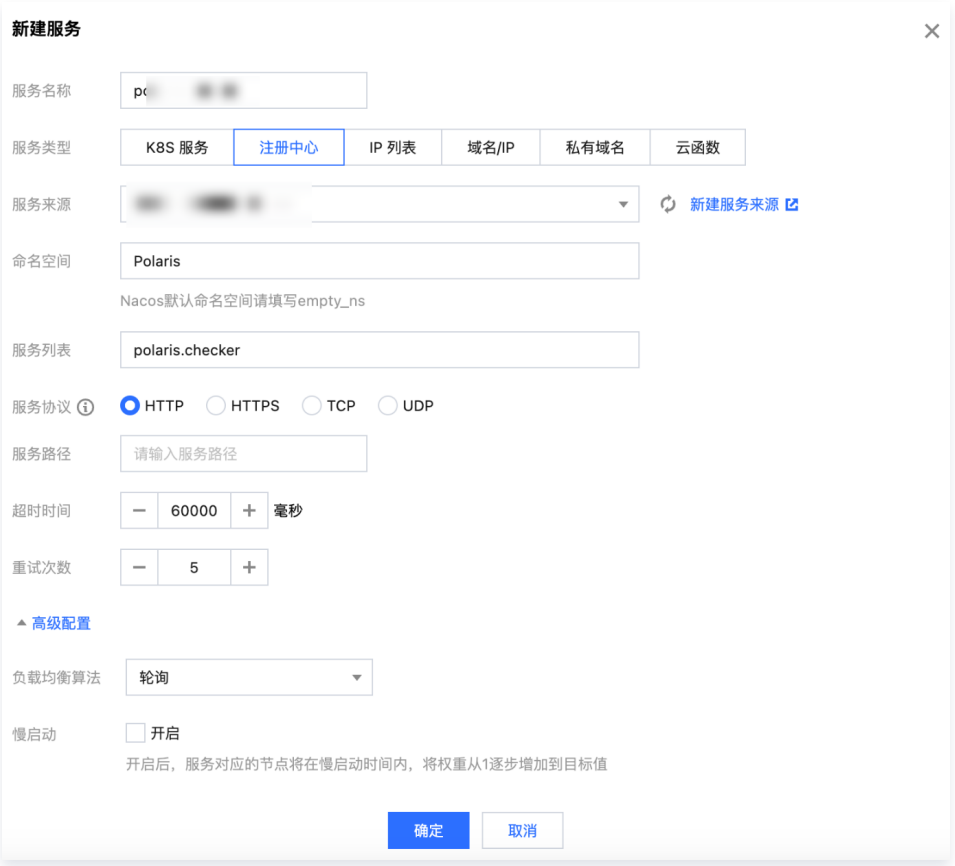
### 步骤2：关联服务来源

1. 单击创建好的云原生 API 网关的 ID，进入基本信息页面。
2. 在左侧菜单栏选择**服务路由 > 服务来源**，在服务来源单击**新建**，服务来源类型选择**注册中心**，来源产品选择 **TSE PolarisMesh**，服务实例选择与云原生 API 网关在同一 VPC 下的可用实例，单击**提交**。




### 步骤3：添加服务

1. 选择**服务路由 > 服务**，在服务列表单击**新建**，服务类型选择**注册中心**，服务来源选择上一步关联的服务来源。填写 PolarisMesh 服务的命名空间 ID 和服务名称。




2. 单击**提交**，创建完成后可以看到服务列表中服务的信息。




## 步骤4：为服务配置路由规则

1. 后端服务添加完成后，单击服务名称进入服务路由页面。
2. 单击**新建**，设置好路由信息，单击**下一步**。

> **说明：**
> 请求方法、请求路径、Host 需至少配置一种。




3. 配置路由后端信息，确认后端服务信息无误后单击**确定**，完成路由创建。




## 步骤5：调用 API

1. 进入云原生 API 网关的基本信息页面，获取负载均衡地址。




2. 使用负载均衡地址和配置的路由路径访问 API，执行 curl 命令测试，已通过**云原生 API 网关**访问到后端服务。

# 使用 Kong Ingress 实现容器服务访问

最近更新时间：2025-06-09 16:07:01

## 操作场景

本文介绍使用 Kong Ingress 实现容器服务访问的操作步骤。

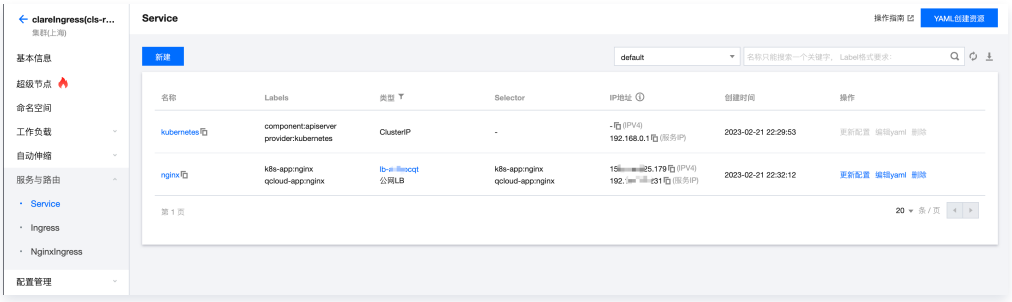
> **注意：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。

## 操作步骤

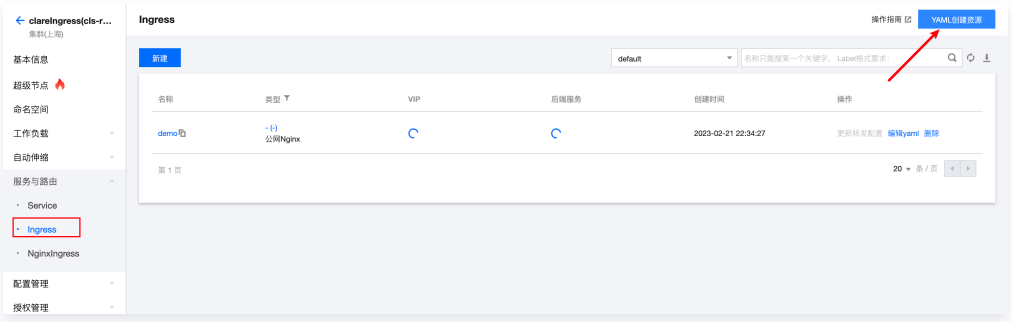
### 步骤1：创建 Ingress 规则

以下以 TKE为例，介绍 Ingress 规则配置。

1. 新建容器集群，参考文档快速 创建一个标准集群 。
2. 在创建好的容器集群上，创建 Nginx 服务，参见文档 创建简单的 Nginx 服务 。




3. 在左侧页签单击 Ingress，选择使用 YAML 创建资源。

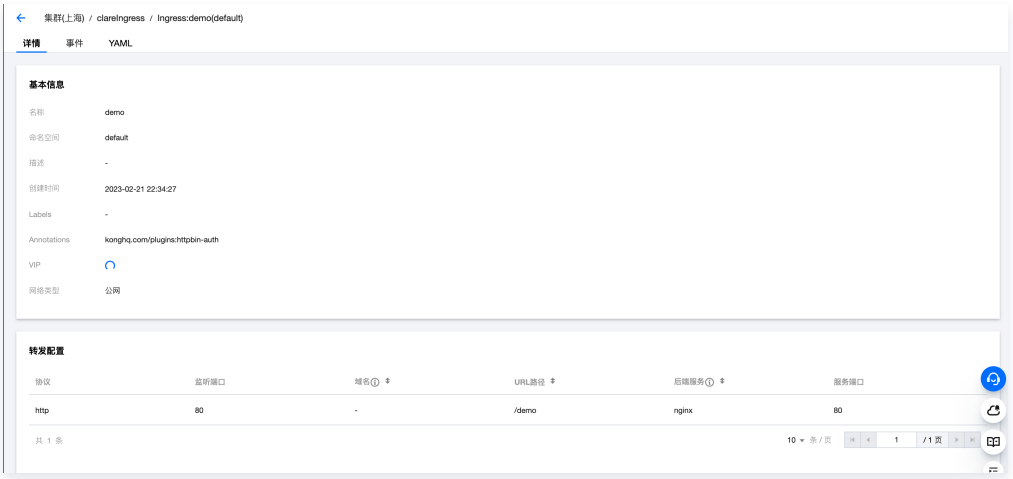



4. 创建 Ingress 资源，基础路由配置可参见如下 Ingress 示例：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: demo
  namespace: default
spec:
  ingressClassName: kong
  rules:
  - http:
      paths:
```

```
      - backend:
          service:
            name: nginx
            port:
              number: 80
        path: /demo
        pathType: Prefix
```

5. 单击 Ingress 详情确认转发配置正确。




## 步骤2：启用 Kong Ingress Controller

1. 登录 TSE 控制台 。
2. 在左侧导航栏，单击云原生 API 网关下的**实例列表**。
3. 单击目标实例的“ID/名称”，进入实例详情页面。
4. 在基本信息页面。展示 Kong Ingress Controller 当前状态，默认为关闭。
5. 单击当前状态后的编辑， 选择对接集群类型和集群信息，单击确定，即启用 Kong Ingress Controller。
6. 确认当前状态变更为已开启，同时展示选择的容器集群信息。

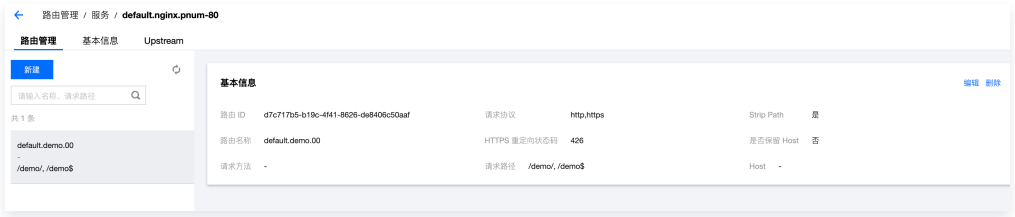



## 步骤3：访问容器服务

1. 单击**服务路由 > 服务**，查看服务列表，确认 Kong Ingress Controller 将容器服务同步过来。




2. 单击**服务名称**，查看路由配置，确认 Ingress 规则对应的路由。




3. 访问容器服务。使用网关地址 + 请求路径，验证容器服务访问成功。

# 使用云原生 API 网关访问私有域名服务

最近更新时间：2025-06-09 16:07:01

## 概述

您可以为私有域名服务创建一个云原生 API 网关，从私有域名关联服务，然后在网关中为服务创建路由策略，以便该服务通过网关对外提供服务。

> **注意：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。

## 操作步骤

### 步骤1：创建云原生 API 网关

参见 新建网关，创建一个云原生 API 网关。

### 步骤2：关联服务来源

在左侧导航栏选择**服务路由 > 服务来源**，在服务来源单击**新建**，服务来源类型选择**私有 DNS**，单击**提交**。




### 步骤3：添加服务

1. 选择**服务路由 > 服务**，在服务列表单击新建，服务类型选择**私有域名**，服务来源选择上一步关联的**私有 DNS**，输入私有域名和服务端口。




2. 单击**提交**，创建完成后可以看到服务列表中服务的信息。




## 步骤4：为服务配置路由规则

1. 后端服务添加完成后，单击服务名称进入服务路由页面。
2. 单击**新建**，设置好路由信息，单击下一步。

> **说明：**
> 请求方法、请求路径、Host 需至少配置一种。




3. 配置路由后端信息，确认后端服务信息无误后单击提交，完成路由创建。

## 步骤5：调用 API

1. 进入云原生 API 网关的基本信息页面，获取负载均衡地址。




2. 使用负载均衡地址和配置的路由路径访问 API，执行 curl 命令测试，已通过云原生API 网关访问到后端服务。

# 使用云原生 API 网关实现 WebSocket 服务转发

最近更新时间：2025-06-09 16:07:02

WebSocket 的服务可以按照正常的 HTTP（HTTPS）服务配置，只要访问网关的时候使用 WebSocket 的协议，网关会自动升级协议为 WebSocket，用户不需要明确指定协议为 WebSocket。本文将快速引导您如何在云原生 API 网关中，访问后端服务为 WebSocket 类型的 API。

> **注意：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。

## 操作步骤

### 步骤1：创建云原生 API 网关实例

参见 新建网关，创建一个云原生 API 网关。

### 步骤2：定义后端服务信息

1. 单击创建好的网关实例的“ID”，进入实例详情页。
2. 在顶部菜单栏选择**服务路由 > 服务**，在服务列表单击**新建**。
3. 在新建服务窗口中配置相关参数后，单击**提交**。

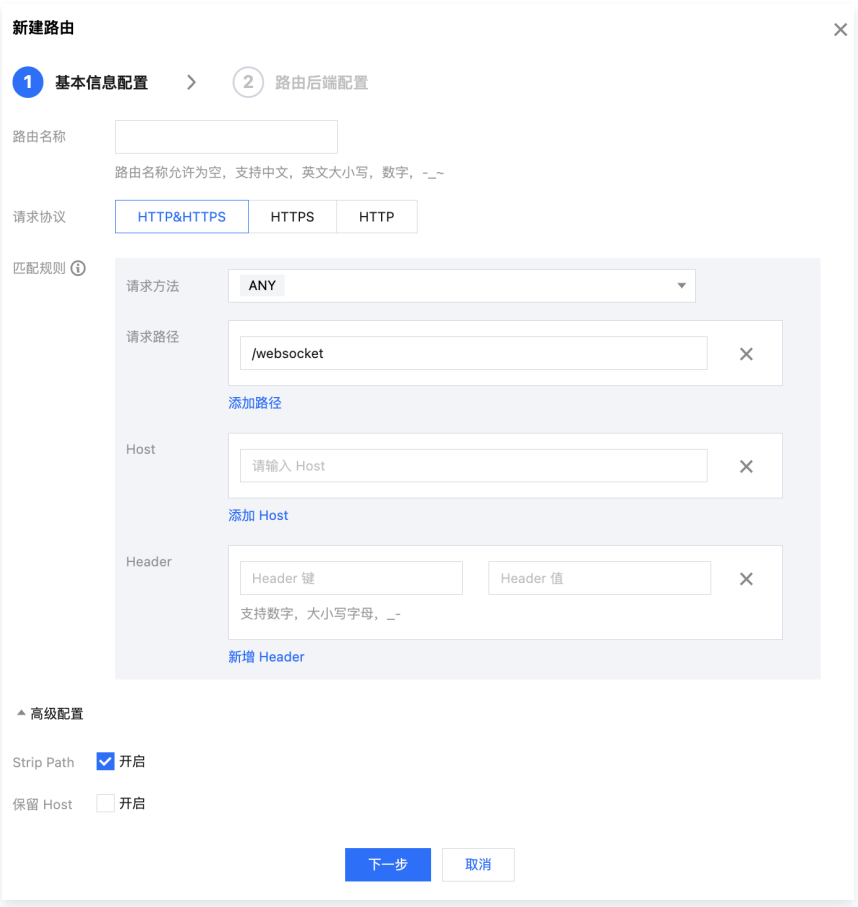
> **说明：**
> 新建网关服务，如果后端服务 WebSocket 提供的是 WS 协议，选择服务协议为 HTTP，如果后端服务提供的是 WSS 协议，选择服务协议为 HTTPS。服务的类型也可以为 K8S 服务/ IP列表等。



## 步骤3：为后端服务配置路由

1. 后端服务添加完成后，点击后端服务名称进入服务路由页面。
2. 单击**新建**，设置好路由信息，单击**下一步**。

> **说明：**
> 请求方法、请求路径、Host 需至少配置一种。

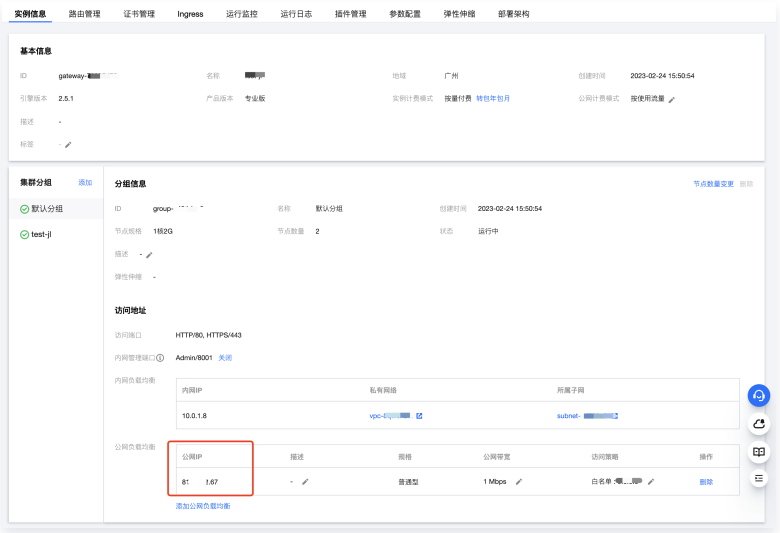



3. 配置路由后端信息，确认后端服务信息无误后单击**提交**，完成路由创建。




## 步骤4：访问 WebSocket 服务

1. 进入云原生 API 网关的基本信息页面，获取负载均衡地址。




2. 使用负载均衡地址和配置的路由路径访问 API。
   - 使用 WS 协议访问网关：websocat ws://<网关负载均衡IP或域名>/websocket。
   - 使用 WSS 协议访问网关：websocat wss://<网关负载均衡IP或域名>/websocket -k。



# 操作指南
# 网关管理
# 新建网关

最近更新时间：2025-06-09 16:07:02

> **说明：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。如果您有新购需求，请 提交工单 处理。

## 操作场景

本文将介绍如何通过微服务平台控制台创建一个云原生 API 网关实例。

## 操作步骤

1. 登录 微服务平台控制台 ，在左侧导航栏单击**云原生 API 网关** > **实例列表**。
2. 单击**新建**，进入云原生 API 网关 Kong 创建页。

| 参数 | 是否必选 | 说明 |
| --- | --- | --- |
| 计费模式 | 是 | 支持**包年包月**和**按量计费**两种模式，如果您的网关使用时间在一个月以上，建议采用**预付费（包年包月）**模式。具体价格请参考 价格说明 文档。 |
| 网关配置 | 是 | 普通场景选择**默认配置**即可，如果对网关版本有要求，可以选择**自定义配置**（目前只有一种网关类型kong，版本支持2.5.1和2.4.1）。 |
| 地域 | 是 | 选择与您部署业务最靠近的地域。 |
| 引擎版本 | 是 | 所选择的网关类型的开源版本号，当前支持2.5.1和2.4.1。 |
| 产品版本 | 是 | 支持开发版、标准版、专业版，专业版提供弹性伸缩等高级特性，可用于测试、生产环境。具体区别请参考 规格对比 。 |
| 节点规格 | 是 | 云原生 API 网关集群由多个节点组成，此处选择每个节点的规格。 |
| 节点数量 | 是 | 云原生 API 网关集群由多个节点组成，此处选择节点的数量。<br>• 节点数量为2时，仅支持随机分配可用区。<br>• 节点数量大于等于3时，支持选择随机可用区和指定可用区（最少选择 2 个可用区，最多选择 3 个可用区）。 |
| 部署架构 | 是 | • 开发版：同城单可用区。<br>• 标准版、专业版：同城三可用区，提供高可用版注册网关，默认支持同城多活。 |
| 节点网络 | 是 | 选择当前云原生 API 网关所部署在的腾讯云 VPC 网络。 |
| 公网负载均衡 | 否 | 开启公网后，将产生费用，具体价格参考 公网流量费用 。<br>• 计费模式：支持**按使用流量**和**按带宽计费**两种。<br>• 带宽上限：支持1-2048Mbps范围。<br>• 可用区类型：支持**单可用区**和**多可用区**。选择多可用区时，您需要选择一个主可用区和一个备可用区，当主可用区故障时，负载均衡可在较短时间内自动切换到备可用区并恢复服务，以保障网关的可用性。 |

| | | |
|---|---|---|
| 日志采集 | 否 | • 开启实时日志服务：默认为您提供网关实时日志服务和简单搜索能力，免费使用。如需日志能力持久化存储，用于排障、审计等场景，建议开启 CLS 日志服务。<br>• 开启 CLS 日志服务：为您开启网关日志投递，提供日志分析。在勾选本选项前，请先确认您已开通 CLS 日志服务。日志服务由 CLS日志 服务提供，会产生费用，具体计费项查看 费用详情 。 |
| 名称 | 是 | 输入当前云原生 API 网关的名称，最长60个字符，支持中英文大小写，-，_。 |
| 描述 | 否 | 输入当前云原生 API 网关的描述信息。 |
| 资源标签 | 否 | 用于分类管理资源，具体使用方法可参见 标签管理 。 |

3. 单击**保存**，耐心等待实例创建完成。

## 结果验证

返回网关列表页面，查看创建的网关信息和状态。网关信息和创建时一致，且状态为运行中，则说明网关新建成功。

## 注意事项

- 创建云原生 API 网关 Kong 要求账号必须拥有 ApiGateWay_QCSRole 角色，并且角色下必须有 `QcloudAccessForApiGateWayRoleInCloudNativeAPIGateway` 策略。如无相关角色和策略，请在创建过程中弹出的授权弹窗中完成授权。
- 云原生 API 网关 Kong 日志功能依赖于 腾讯云日志服务（简称 CLS），请您确保已经开通 CLS 服务，否则在创建云原生 API 网关 Kong 前请前往 CLS 控制台 开通服务。

# 查看网关详情

最近更新时间：2025-06-09 16:07:02

本文介绍在创建一个网关实例后，如何查看网关的基本信息以及访问地址等。

## 操作步骤

1. 登录 微服务平台控制台 ，在左侧导航栏单击**云原生 API 网关 > 实例列表**。
2. 在实例列表页面，单击刚刚创建好的实例的“ID”，进入网关实例基本信息页面，可查看实例的基本信息和集群分组信息。
   - **实例信息**




   您可以在基本信息区域进行如下操作：
   - 单击基本信息模块右上角的**编辑**，可修改网关实例的名称和说明。
   - 单击公网计费模式的**编辑**，可以修改公网的计费方式。
   - 单击标签的**编辑**，可以修改网关绑定的标签信息。
   - **集群分组**：集群分组支持多个集群使用相同的路由配置，方便实现对多个云原生 API 网关实例进行统一管理。每个网关实例创建后均会生成一个默认分组。




     - 分组信息：关于集群分组的详细介绍和操作请参考 集群分组 。
     - 网络配置：
       - 访问端口：云原生 API 网关的访问端口。
       - 内网管理端口：云原生 API 网关 的管理端口。管理端口没有权限控制，仅支持通过内网私有连接访问。
       - 内网私有连接：云原生 API 网关 内网地址。客户端可以通过内网节点地址访问云原生 API 网关的网关节点。
       - 公网负载均衡：展示目前网关实例配置的公网负载均衡，支持修改公网带宽和配置访问策略，公网访问建议设置访问安全策略。

**说明：**

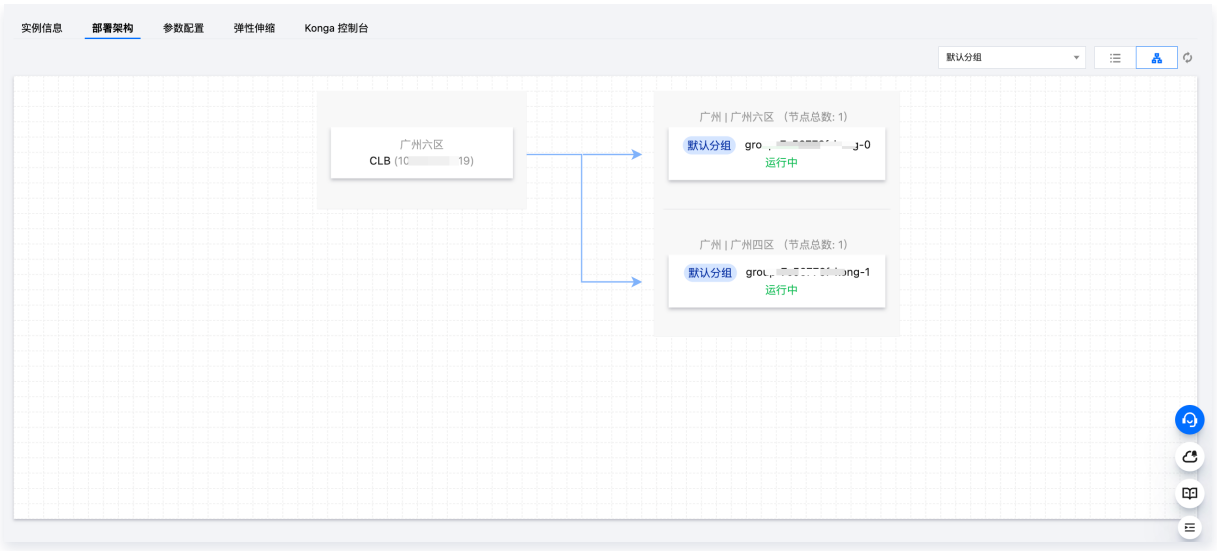
内网和公网负载均衡支持选择负载均衡规格，单击**添加负载均衡**后，设置好负载均衡规格，单击**确定**即可。




3. 在实例详情页面，单击**部署架构**页签，展示当前所选分组下网关节点的状态及可用区信息。




点击切换视图，切换到部署架构图，展示当前所选分组下接入层（CLB）和网关节点的部署情况。




**说明：**

云原生 API 网关标准版、专业版节点默认为多可用区部署。

# 升级网关版本

最近更新时间：2025-06-09 16:07:02

## 操作场景

如果当前的网关版本功能不能满足您的业务需求，您可以通过控制台将您的网关升级到更高的版本，从而体验更丰富的功能。

## 操作步骤

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击**云原生 API 网关 > 实例列表**，找到目标网关，单击网关的“ID”，进入基本信息页面。
3. 在基本信息的产品版本处，单击**升级**，选择好目标版本、节点规格和节点数量。




4. 单击**确定**，等待大约3-5分钟完成网关版本升级，您可以在网关列表的状态栏可以查看任务状态。



# 升级网关规格

最近更新时间：2025-06-09 16:07:02

## 操作场景

如果当前的网关规格不能满足您的业务需求，可以通过控制台提升您的网关节点数量和节点规格。

> **说明**
> 如需降低网关节点数量和节点规格，请 提交工单 处理。

## 前提条件

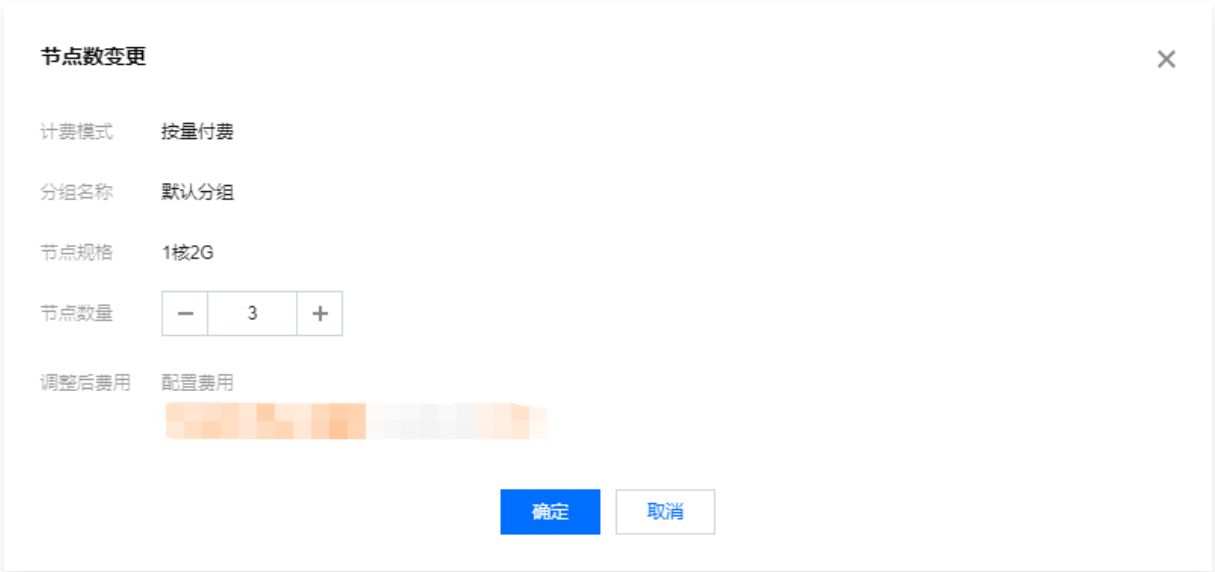
只有**运行中/升级失败**状态的实例支持升级规格。

## 节点数量变更

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击**云原生 API 网关 > 实例列表**，找到目标网关，点击网关的“ID”，进入基本信息页面。
3. 在分组信息模块，单击节点数量处的**变更**，进入节点变更页。




4. 在规格变更页，选择目标升级节点数，单击**确定**，在确认弹窗中选择**确定**，等待大约3-5分钟完成实例升级，您可以在网关列表的状态栏可以查看任务状态。



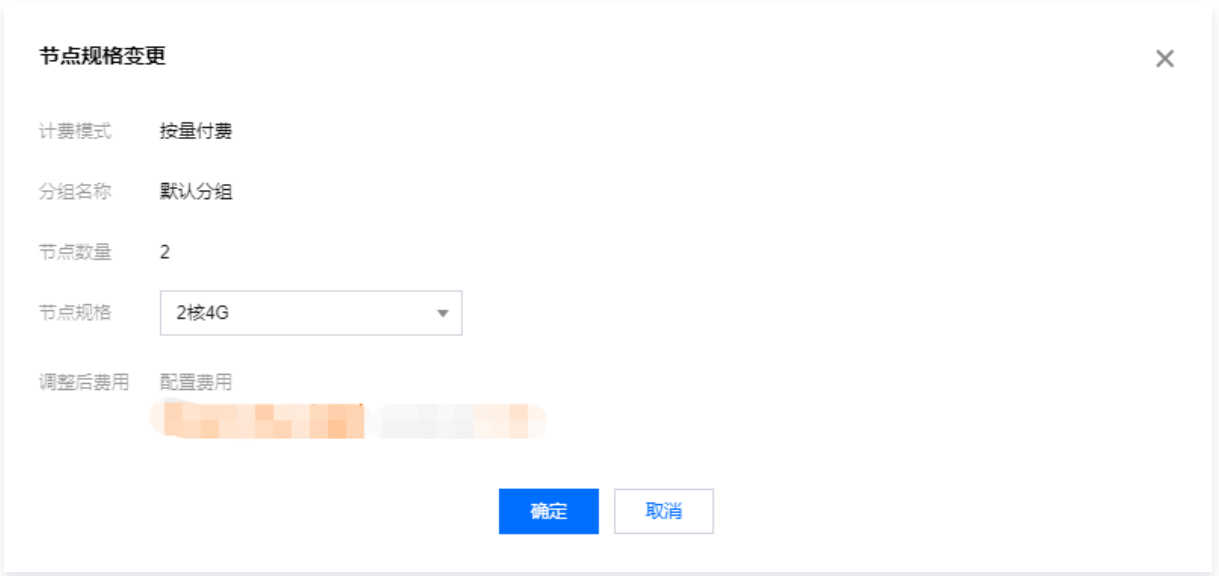

## 节点规格变更

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击**云原生 API 网关 > 实例列表**，找到目标网关，点击网关的“ID”，进入基本信息页面。

3. 在分组信息模块，点击节点规格处的**变更**，进入规格变更页。




4. 在规格变更页，选择目标升级节点规格，单击**确定**，在确认弹窗中选择**确定**，等待大约3-5分钟完成实例升级，您可以在引擎列表的状态栏可以查看任务状态。



# 集群分组管理

最近更新时间：2025-06-09 16:07:02

## 操作场景

云原生 API 网关集群分组支持多个分组共用相同的路由配置，方便实现对多个分组进行统一管理。
集群分组的适用场景如下：

1. 为相同业务分配不同规格的资源。不同分组支持配置不同的资源，包括节点数量和节点规格，您可以为开发用途的分组配置较低的节点规格和数量，为生产用途的分组配置较高的节点规格和数量，实现资源按需购买。
2. 为网关提供多个访问入口。不同分组支持使用不同的负载均衡 CLB，提供多个入口，您可以为不同的 CLB 配置不同的访问策略，并将其分配给不同的角色，实现权限控制。

> **说明：**
> 每个网关实例创建后均会生成一个默认分组，您可以按需添加自定义分组。

## 前提条件

已创建云原生 API 网关实例 。

## 操作步骤

### 步骤1：添加分组

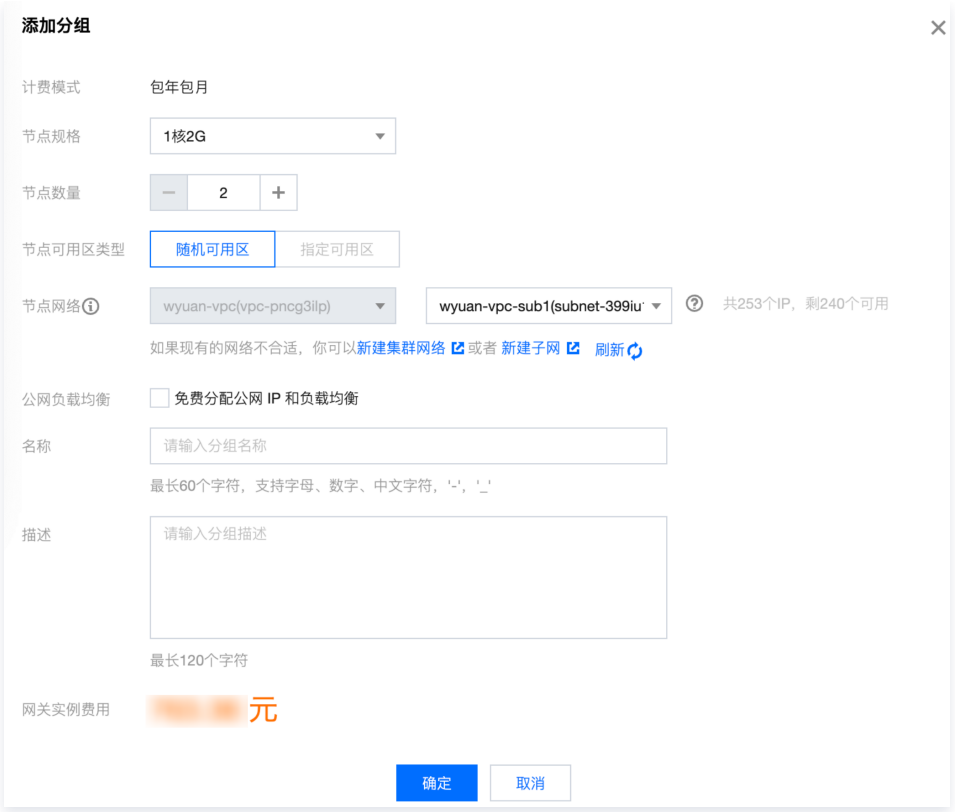
1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击 **实例列表**，进入 kong 网关实例列表页面。
3. 单击目标实例的“ID/名称”，进入实例详情页面。




4. 在基本信息页面下方的 集群分组 卡片，展示 默认分组。每个网关实例创建后均会生成一个默认分组。




5. 单击**添加分组**， 填写新添加的分组信息。

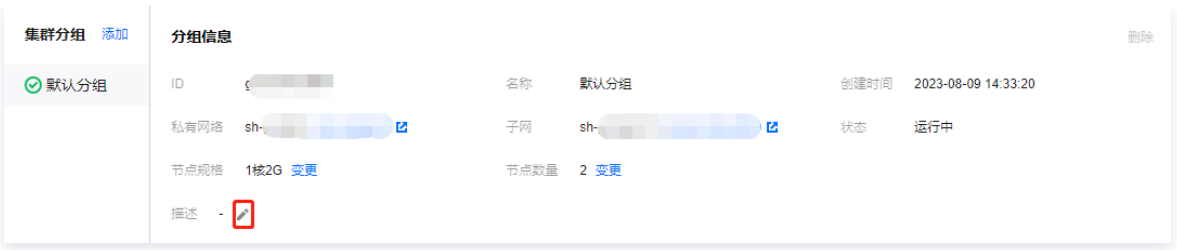



| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 计费模式 | - | 默认与 kong 实例计费模式保持一致，不支持修改。 |
| 节点规格 | 是 | 集群分组每个节点的规格。 |
| 节点数量 | 是 | 集群分组节点的数量。 |
| 节点可用区类型 | 是 | • 节点数量为2时，仅支持随机分配可用区。<br>节点数量大于等于3时，支持选择随机可用区和指定可用区（最少选择 2 个可用区，最多选择 3 个可用区）。 |
| 节点网络 | 是 | 选择集群分组所部署在的腾讯云 VPC 网络和腾讯云子网。VPC 网络默认与 kong 实例相同，支持选择腾讯云子网。 |
| 所属子网 | 是 | 选择当前云原生 API 网关所部署在的腾讯云子网，子网属于 VPC 网络。 |
| 公网带宽 | 是 | 选择集群分组的公网带宽。 |
| 分组名称 | 是 | 输入当前分组的名称。 |
| 描述 | 否 | 输入当前分组的描述信息。 |

6. 单击**确定**，耐心等待分组创建完成。

## 步骤2：修改分组基本信息

1. 在分组信息卡片，单击描述旁边的**编辑**。




2. 在修改分组名称/描述弹窗修改分组名称/描述。



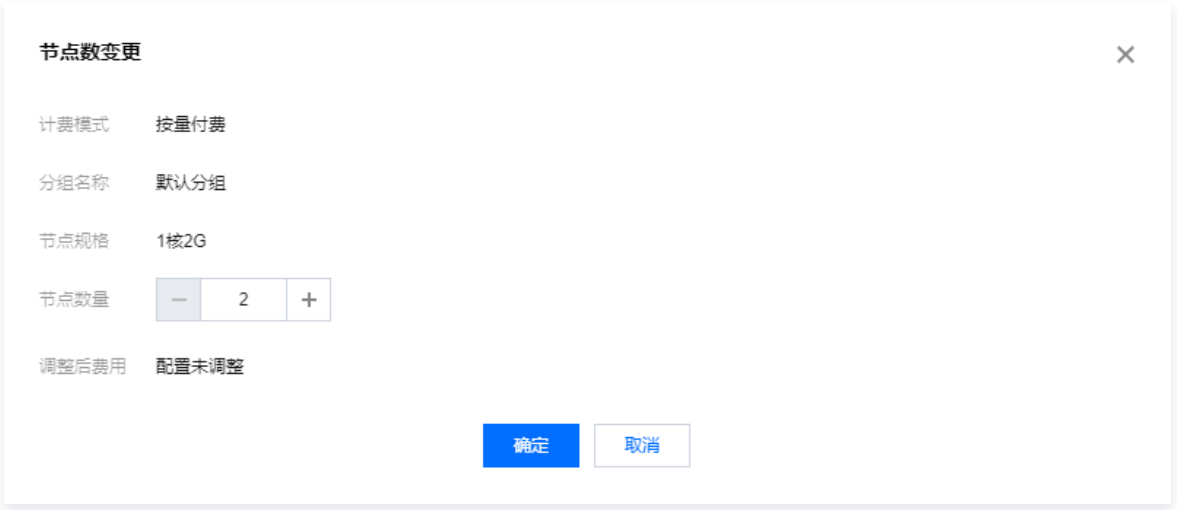

3. 单击**确定**，完成修改。

## 步骤3：分组规格变更

1. 在分组信息卡片，单击右侧的**节点数变更**。




2. 在节点数变更弹窗修改节点数量。

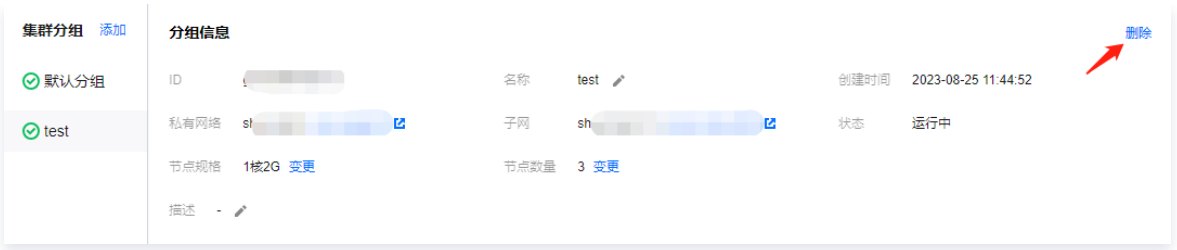



3. 单击**确定**，完成修改。

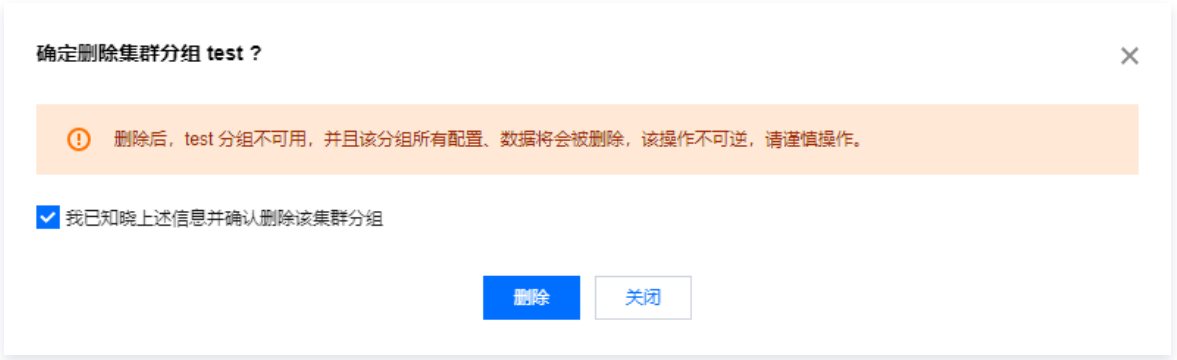
## 步骤4：删除分组

当您不再需要集群分组时，可以在控制台手动删除该分组。

1. 在分组信息卡片，单击右侧的**删除**。




2. 在删除分组弹窗中二次确认，即可删除分组。




**说明**

- 只有运行中/失败状态的分组可以被删除。
- 分组删除后，该分组节点及其数据均会被删除，请谨慎操作。

## 步骤5：访问路由

当您访问路由时，需使用「分组的公网负载均衡/内网私有连接 IP + 路由路径」拼接访问 URL。



# 网关节点管理

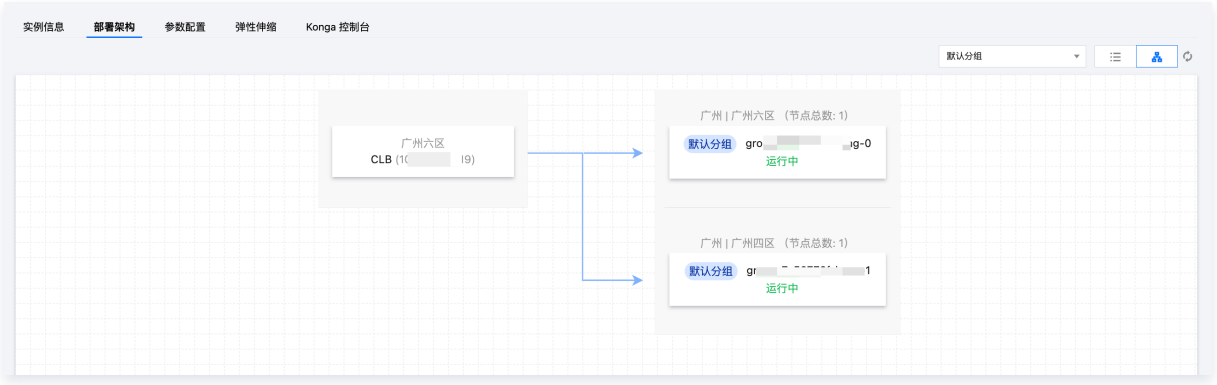
最近更新时间：2025-06-09 16:07:02

本文介绍在创建网关实例后，如何查看网关的节点信息以及进行网关节点管理（**仅专业版支持**）。

## 操作步骤

### 步骤1：查看部署架构

1. 登录 微服务平台控制台 ，在左侧导航栏单击**云原生 API 网关 > 实例列表**。
2. 在实例列表页面，单击网关实例的“ID”，进入网关实例基本信息页面，单击**部署架构**页签，可以查看网关实例的部署架构。
3. 部署架构图展示当前所选分组下接入层（CLB）和网关节点的部署情况。

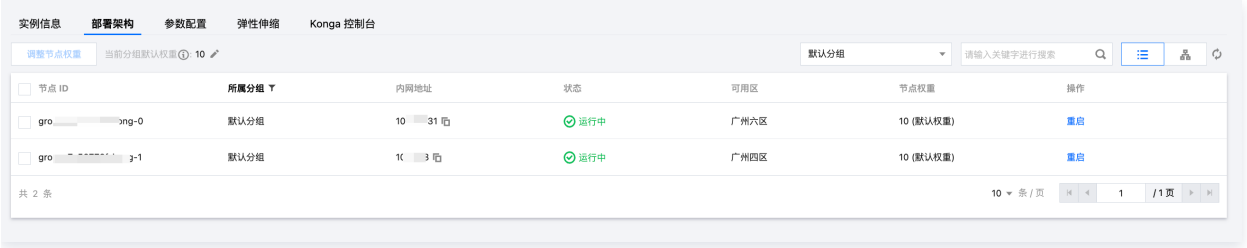



> **说明：**
> 云原生 API 网关标准版、专业版节点默认为多可用区部署。

### 步骤2：查看节点信息

节点信息展示当前所选分组下网关节点的信息。

- 节点名称：云原生 API 网关节点名称。
- 所属分组：云原生 API 网关节点所属集群分组。
- 内网地址：云原生 API 网关节点内网地址，
- 状态：云原生 API 网关节点当前运行状态，包括运行中和修改中两种。
- 可用区：云原生 API 网关节点当前可用区。
- 节点权重：云原生 API 网关节点当前权重，权重越大承载的流量越多，初始状态为分组默认权重值，取值范围 0-100。
- 分组默认权重：云原生 API 网关分组内节点的默认权重，在该分组中增加节点后以该值作为初始权重值。




### 步骤3：修改节点权重

您可以通过控制台调整节点权重进行节点流量管理。

1. 在节点信息处，勾选需要调整权重的节点，单击**调整节点权重**，在弹窗中填写目标权重值。
2. 单击**确定**，完成节点权重修改。




**说明：**

1. 仅专业版支持调整节点权重。
2. 节点权重调整实时生效，流量不会转发到权重为 0 的节点，请谨慎操作。
3. 为保证网关可用性，不支持所有节点权重同时调整为 0。

## 步骤4：修改分组默认权重

分组默认权重决定新增节点和未进行自定义权重配置的节点权重值，您可以通过控制台调整分组默认权重进行节点的默认权重管理。

1. 在节点默认权重信息处，点击 **修改**，在弹窗中填写目标节点默认权重，默认为 10，取值范围 0-100。
2. 单击 **确定**，完成分组默认权重修改。

**说明：**

- 仅专业版支持调整节点权重。
- 分组默认权重调整实时生效。
- 已进行自定义权重配置的节点不受影响。

# 共享带宽包管理

最近更新时间：2025-06-09 16:07:02

云原生 API 网关支持使用共享带宽包，您可以将多个网关实例公网负载均衡加入共享带宽包中合并计费。

> **说明：**
> 共享带宽包功能已转内测，不支持新增使用。

## 限制说明

云原生 API 网关当前仅支持常规 BGP 带宽包：

| 带宽类型 | 线路说明 | 计费模式 | 计费模式说明 |
| --- | --- | --- | --- |
| 常规 BGP 带宽包 | 普通 BGP IP 线路类型的带宽包 | 后付费模式-月 TOP5计费 | 按月结算费用，按实际用量削去每日的 TOP5 峰值计费 |

## 新建共享带宽包

1. 登录 TSF 控制台 。
2. 在左侧导航栏，单击 **实例列表**，选择好地域后在页面上方单击**共享带宽包**页签。
3. 单击**新建**，在弹窗中输入共享带宽包名称，并选择好地域。



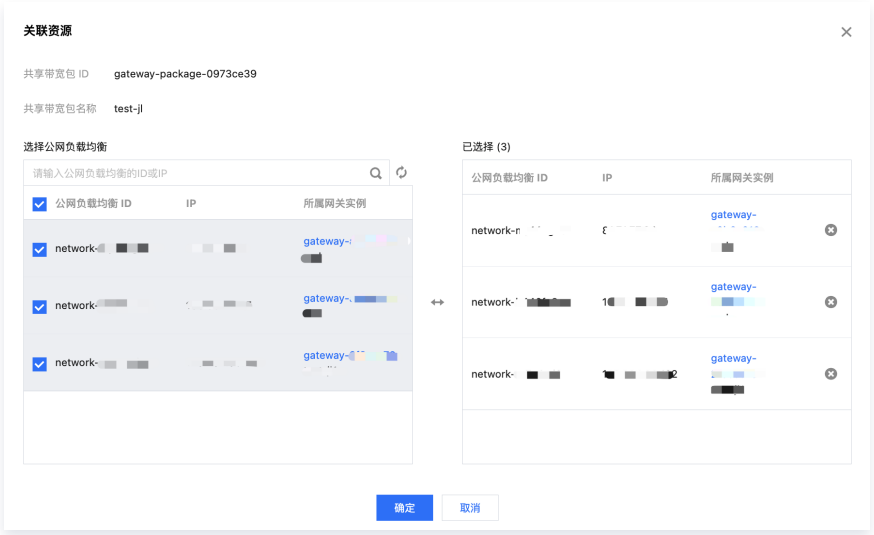

4. 单击**确定**即可在共享带宽包列表页面看到创建好的共享带宽包，单击共享带宽包的“ID”，即可进入详情页面。




## 添加资源

1. 在共享带宽包详情页面，单击**添加资源**，勾选要添加的公网负载均衡。

2. 单击**确定**，即加入共享带宽包。




## 移除资源

在共享带宽包详情页面，您可以单击公网负载均衡操作列的**移除资源**；也可以单击上方的**批量移除资源**，勾选要移除的公网负载均衡，批量移除。




## 删除共享带宽包

> **说明：**
> 只有共享带宽包没有关联的资源才可以删除。

在共享带宽包列表页，单击要删除的共享带宽包操作栏的**删除**，二次确认后即可删除。

# 删除网关

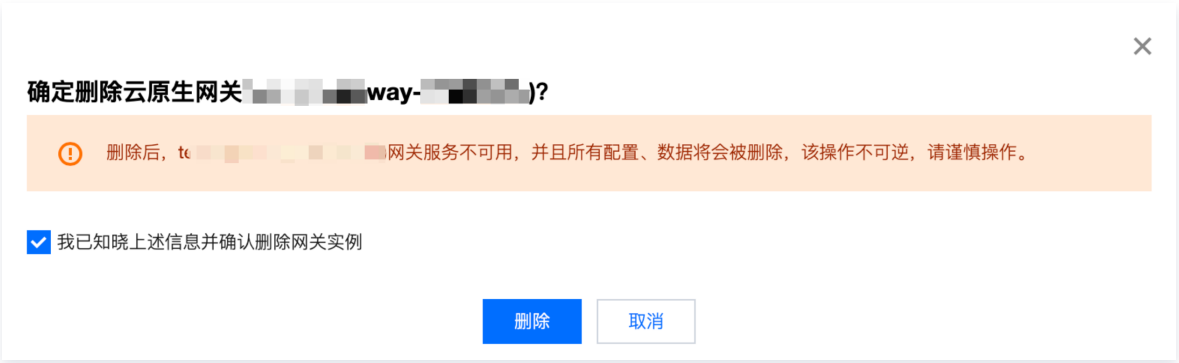
最近更新时间：2025-06-09 16:07:02

当您不再需要云原生 API 网关实例时，您可以删除该实例。

## 操作步骤

### 手动删除

1. 登录 TSF 控制台 。
2. 在云原生 API 网关实例列表页面，单击目标实例操作列的**删除**。
3. 在删除的确认弹框中，单击**删除**，即可删除集群。




**说明**

- 只有**运行中/失败状态**的实例可以被销毁。
- 实例销毁后，该实例及其数据均会被删除，该操作不可逆，请谨慎操作。

### 到期/欠费自动删除

您的网关实例在到期未续费/账户欠费后，最多在 TSE 控制台中保留7个自然日。到期后7天内完成续费可继续使用，详情请查看 欠费说明 。
若您的实例在到期7天（包括第7天）后未进行续费，系统将在到期后第8天的0点开始对资源释放，到期实例中的数据将被清除且不可恢复。

**说明：**

- 处于已隔离状态的实例不能生产和消费，已保存在实例内的数据和配置不会被删除。
- 对于7天内已隔离状态的集群，可以在控制台实例列表页，单击操作列的**续费**。续费成功后，实例即可恢复正在运行状态，集群可正常使用。

# 服务路由
# 服务

最近更新时间：2025-06-09 16:07:02

## 操作场景

您需要将服务添加到云原生 API 网关中，以便网关能查找对应的服务地址从而实现动态代理。云原生 API 网关支持添加 K8S 服务、注册中心、IP 列表、域名/IP、私有域名、云函数几种类型的服务。本文介绍如何为云原生网关添加服务。

## 操作步骤

### 添加服务

1. 登录 微服务平台控制台 。
2. 在左侧导航栏单击**服务路由**，在页面上方选择好实例后，单击**服务**页签，在服务列表单击**新建**。
3. 在新建服务窗口中配置相关参数。

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 服务名称 | 是 | 输入服务名称。 |
| 服务类型 | 是 | 服务类型，支持 注册中心，其余服务类型不支持新接入。 |

**注意：**
目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。

#### K8S 服务

当服务来源为 K8S 服务时，填写以下参数：

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 服务来源 | 是 | 选择已创建的容器服务来源。 |
| 命名空间 | 是 | 输入命名空间名称。 |
| 服务列表 | 是 | 填写需要访问的容器服务列表，多个服务使用换行分隔。 |
| 服务协议 | 是 | 网关请求后端服务使用的协议，支持 HTTP/WS、HTTPS/WSS、TCP、UDP。 |
| 服务路径 | 是 | 后端服务的请求路径。 |
| 服务端口 | 否 | 后端服务的端口，支持1~65535的任意端口。 |
| 超时时间 | 是 | 连接建立、写入和读取后端返回的最大超时时间 |
| 重试次数 | 是 | 与后端服务建立连接的最大重试次数。 |

版权所有：腾讯云计算（北京）有限责任公司

### 注册中心

**当服务来源为注册中心时，填写以下参数：**

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 服务来源 | 是 | 选择已创建的服务来源。 |
| 命名空间 | 是 | 当来源产品为 Nacos/PolarisMesh 时，输入命名空间 ID。注意：Nacos 默认命名空间请填写 empty_ns。当来源产品为 TSF 注册中心时，下拉选择命名空间。 |
| 服务列表 | 是 | 当来源产品为 Nacos/PolarisMesh/Consul 时填写需要访问的服务列表，多个服务使用换行分隔。针对不同注册中心，服务列表书写方式为：<br>• Nacos：{Nacos分组}__{服务名称}注意：Nacos 分组与服务名称之间使用两个下划线(_)分割<br>• Consul：{服务名称}<br>• Polaris Mesh：{服务名称}<br>当来源产品为 TSF 注册中心时，下拉选择单个微服务名称。 |
| 服务协议 | 是 | 网关请求后端服务使用的协议，支持 HTTP/WS、HTTPS/WSS、TCP、UDP。 |
| 服务路径 | 是 | 后端服务的请求路径。 |
| 超时时间 | 是 | 连接建立、写入和读取后端返回的最大超时时间。 |
| 重试次数 | 是 | 与后端服务建立连接的最大重试次数。 |

### IP 列表

**当服务类型为 IP 列表时，需要填写以下参数：**

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 地址列表 | 是 | 填写后端服务节点的 IP、端口、权重信息。 |
| CVM 弹性伸缩 | 否 | 开启后，Kong 将自动关联所选伸缩组对应的 CVM 实例。<br>• 伸缩组：勾选同 VPC 网络下的弹性伸缩组。关联后，Kong 将在弹性伸缩组创建生命周期挂钩，以及在云服务器-我的命令创建弹性伸缩命令，对上述资源的修改和删除将导致伸缩失效。<br>• 伸缩组端口：填写伸缩组端口。 |
| 服务协议 | 是 | 网关请求后端服务使用的协议，支持 HTTP/WS、HTTPS/WSS、TCP、UDP。 |
| 服务路径 | 是 | 后端服务的请求路径。 |
| 超时时间 | 是 | 连接建立、写入和读取后端返回的最大超时时间。 |
| 重试次数 | 是 | 与后端服务建立连接的最大重试次数。 |

### 域名/IP

当服务类型为域名/IP 时，需要填写以下参数：

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 服务地址 | 是 | 支持填写 IP 或域名，例如：10.0.0.1或 www.tencent.com。 |
| 服务端口 | 是 | 后端服务的端口，支持1~65535的任意端口。 |
| 服务协议 | 是 | 网关请求后端服务使用的协议，支持 HTTP/WS、HTTPS/WSS、TCP、UDP。 |
| 服务路径 | 是 | 后端服务的请求路径。 |
| 超时时间 | 是 | 连接建立、写入和读取后端返回的最大超时时间。 |
| 重试次数 | 是 | 与后端服务建立连接的最大重试次数。 |

### 云函数

当服务类型为云函数时，需要填写以下参数：

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 函数类型 | 是 | 事件函数或者 Web 函数。 |
| 命名空间 | 是 | 选择函数所在的命名空间。 |
| 函数 | 是 | 选择函数。 |
| 版本/别名 | 是 | 选择函数所在的服务版本或者别名。 |
| CAM 鉴权 | 是 | 默认开启，网关转发请求到云函数进行 CAM 鉴权。 |
| 超时时间 | 是 | 连接建立、写入和读取后端返回的最大超时时间。 |
| 重试次数 | 是 | 与后端服务建立连接的最大重试次数。 |

### 私有域名

当服务来源为 私有域名时，填写以下参数：

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 服务来源 | 是 | 选择已创建的腾讯云私有解析来源。 |
| 私有域名 | 是 | 填写服务的私有域名。 |
| 服务端口 | 是 | 填写需要访问的容器服务列表，多个服务使用换行分隔。 |
| 服务协议 | 是 | 网关请求后端服务使用的协议，支持 HTTP/WS、HTTPS/WSS、TCP、UDP。 |
| 服务路径 | 是 | 后端服务的请求路径。 |
| 超时时间 | 是 | 连接建立、写入和读取后端返回的最大超时时间。 |

版权所有：腾讯云计算（北京）有限责任公司

| | | |
|---|---|---|
| 重试次数 | 是 | 与后端服务建立连接的最大重试次数。 |

**说明：**

1. 网关仅对 SRV记录、A记录、CNAME 记录类型进行解析。
2. 如果 SRV 记录已填写端口，控制台输入端口会覆盖 SRV 记录的端口内容。

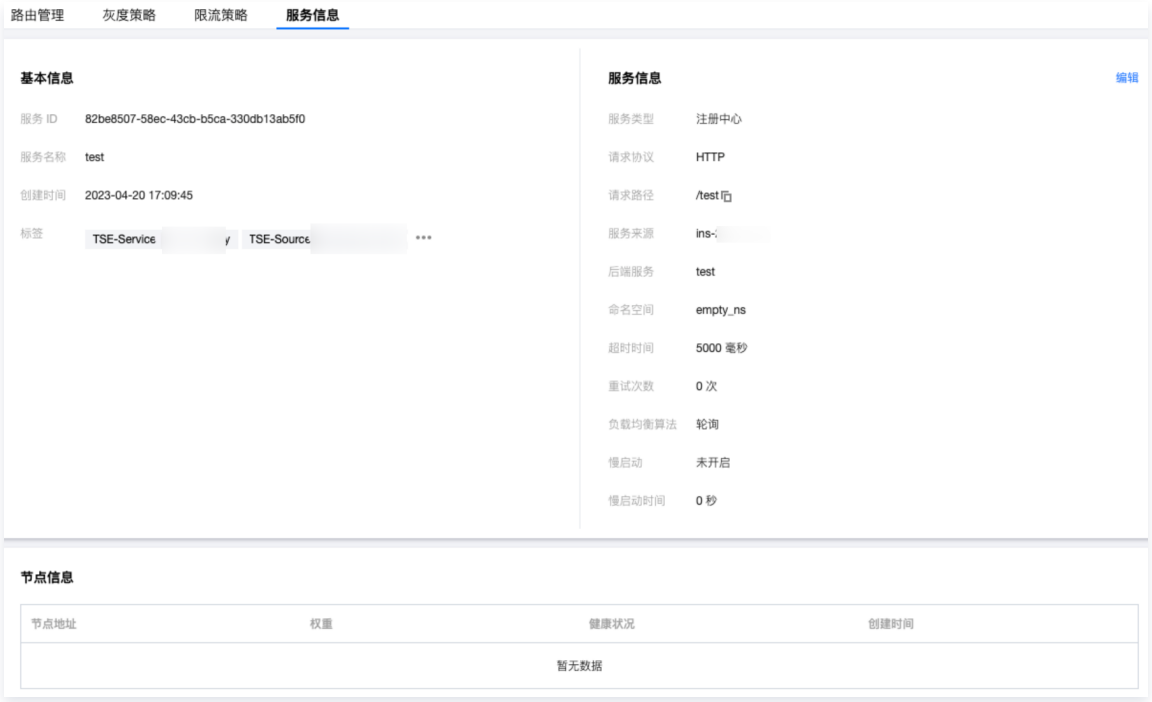
4. （可选）配置高级参数。
   - 负载均衡算法：支持选择轮询和最小连接数。
   - 慢启动：开启后，服务对应的节点将在慢启动时间内，将权重从1逐步增加到目标值。若引擎版本较低，可能不支持使用此功能，建议前往实例基本信息页面升级引擎版本。

**说明：**

支持在容器服务集群中修改 YAML 配置慢启动，增加 annotation：

```
annotations:
    tse.konghq.com/TSE-Upstream-SlowStart: "30"
```

5. 添加后，查看服务列表，可以看到添加的服务，单击服务名称，可以查看详细的服务信息。




## 编辑服务

单击**操作** > **编辑**，修改服务信息，确认后即可。

## 删除服务

单击**操作** > **删除**，确认是否删除服务。确认后将删除服务。

# 路由

最近更新时间：2025-06-09 16:07:02

## 操作场景

您需要为需要转发的服务配置路由，以便网关能根据路由匹配条件进行路由匹配，并将请求转发到对应的服务地址从而实现动态代理。本文介绍云原生 API 网关路由匹配方式，以及如何为云原生网关添加路由并验证转发效果。

## 路由匹配说明

### 路由属性

根据不同的协议，云原生 API 网关支持配置不同属性用于进行路由匹配：

| 请求协议 | 用于路由匹配的属性 |
| --- | --- |
| HTTP/HTTPS | 请求 HTTP 方法（Request HTTP method） |
| | 主机名（Host） |
| | 请求头（Request Header，自定义请求头） |
| | 请求路径（Request path） |
| | 请求 SNI（Request SNI），仅用于 HTTPS 场景 |
| TCP/TLS 协议 | 请求目标（Request Destination） |
| | 请求 SNI（Request SNI），仅用于 TLS 场景 |
| gRPC/gRPCS 协议 | 主机名（Host） |
| | 请求头（Request Header，自定义请求头） |
| | 请求路径（Request path） |
| | 请求 SNI（Request SNI），仅用于 GRPCS 场景 |

配置路由时，每个属性都是可选的，但至少需要指定上述属性中的一个。例如，仅指定 Host 为 “`*.example.com`”，则所有来自该域名及其子域名的请求都可能匹配此路由。

## 路由属性说明

### 主机名（Host）

云原生 API 网关支持以下两种主机名：

- 精确主机名：基于 Host 头进行路由是最常见的方式，可通过路由的 Host 字段配置。例如，配置 Host 为 `["example.com", "service.com"]`，则客户端请求的 Host 头必须为这两个值之一才能匹配该路由。
- 通配符主机名：为了提供灵活性，云原生 API 网关支持配置带有通配符的主机名，通配符主机名在域的最左侧 或最右侧标签中只能包含一个星号。例如 `*.example.com` 将允许主机值（例如 `a.example.com`）与 `x.y.example.com` 匹配。`example.*` 将允许主机值（例如 `example.com`）、与 `example.org` 匹配。

### 请求头（Request Header）

除 Host 头外，也可以使用其他自定义请求头进行路由。例如，配置 headers.region=north，则请求头中包含 Region: North 的请求将匹配该路由。

说明：多个请求头之间是逻辑 “与” 关系，而其值是逻辑 “或” 关系，即请求必须包含所有配置的 headers 键，且对于每个键，其值只需匹配其中一个配置值即可。

### 请求路径（Request path）

云原生 API 网关支持配置以下路径匹配方式：

- 前缀匹配：请求 URL 以配置的路由路径前缀开头即能匹配，例如配置 `/api/v1/`，则 `/api/v1/users`、`/api/v1/orders` 等都可匹配。
- 正则匹配：通过正则表达式定义路由匹配规则，请求 URL 符合正则模式才能匹配，例如 `/api/v1/[a-z]+/[0-9]+` 可匹配 `/api/v1/users/123`。

### 请求 HTTP 方法（Request HTTP method）

请求方法指根据 HTTP 方法（例如 GET、POST、PUT 等）匹配请求。请求方法支持配置多个值，默认值为空（即不根据方法进行路由）。例如，配置请求方法为 `["GET", "HEAD"]` 的路由，将匹配 GET 和 HEAD 请求，但不匹配 POST 或 DELETE 请求。您可以为不同的 HTTP 方法配置不同的插件，例如对 GET 请求不进行认证，对 POST 请求应用认证和限流插件。

### 请求目标（Request Destination）

仅适用于 TCP 和 TLS 路由，通过目标端口进行匹配。

### 请求 SNI（Request SNI）

在使用安全协议（https、grpcs、tls）时，Server Name Indication 可作为路由属性。例如，配置 SNI 为 `["foo.test", "example.com"]` 的路由，将匹配在 TLS 连接的 SNI 扩展中设置了相应主机名的传入请求。SNI 路由适用于 TLS 及基于 TLS 的其他协议（如 HTTPS），且在路由匹配时，只要路由中指定的任何一个 SNI 与请求的 SNI 匹配即可。SNI 在 TLS 握手时确定，连接建立后无法修改，因此使用同一 keepalive 连接的多个请求在进行路由匹配时将具有相同的 SNI 主机名，与 Host 头无关。虽然可以创建 SNI 和 Host 头不匹配的路由，但通常不建议这样做。

### 请求匹配条件

对于一个请求要匹配路由，必须满足以下条件：

1. 请求中必须包含路由配置中设置的所有字段。例如，如果路由配置了 Host 和 paths，则请求中必须有对应的主机名和路径信息。
2. 请求中字段的值必须至少匹配配置值中的一个。例如，路由的 Host 配置为 `["example.com", "test.com"]`，则请求的 Host 头为“example.com”或“test.com”时满足条件；若 paths 配置为 `["/api/v1/products", "/api/v1/orders"]`，请求路径为“/api/v1/products/list”或“/api/v1/orders/123”等以配置路径开头的都可匹配。

### 请求匹配优先级

当一个请求可能匹配多个路由时，按照以下优先级进行匹配：

1. 根据匹配到的路由属性数量进行判断：匹配到的路由属性数量越多，优先级越高。
2. 根据主机名 Host 进行判断：使用精确 Host 的请求>使用通配 Host 的请求。
3. 根据请求头进行判断：匹配到的请求头数量越多，优先级越高。
4. 根据路径进行判断：正则匹配 > 前缀匹配。

对于正则匹配，存在以下优先级：

1. 根据正则优先级（Regex priority）进行匹配：默认为0，值越大优先级越高。
2. 正则优先级相同时，依据创建时间进行判断：创建时间越早，优先级越高。

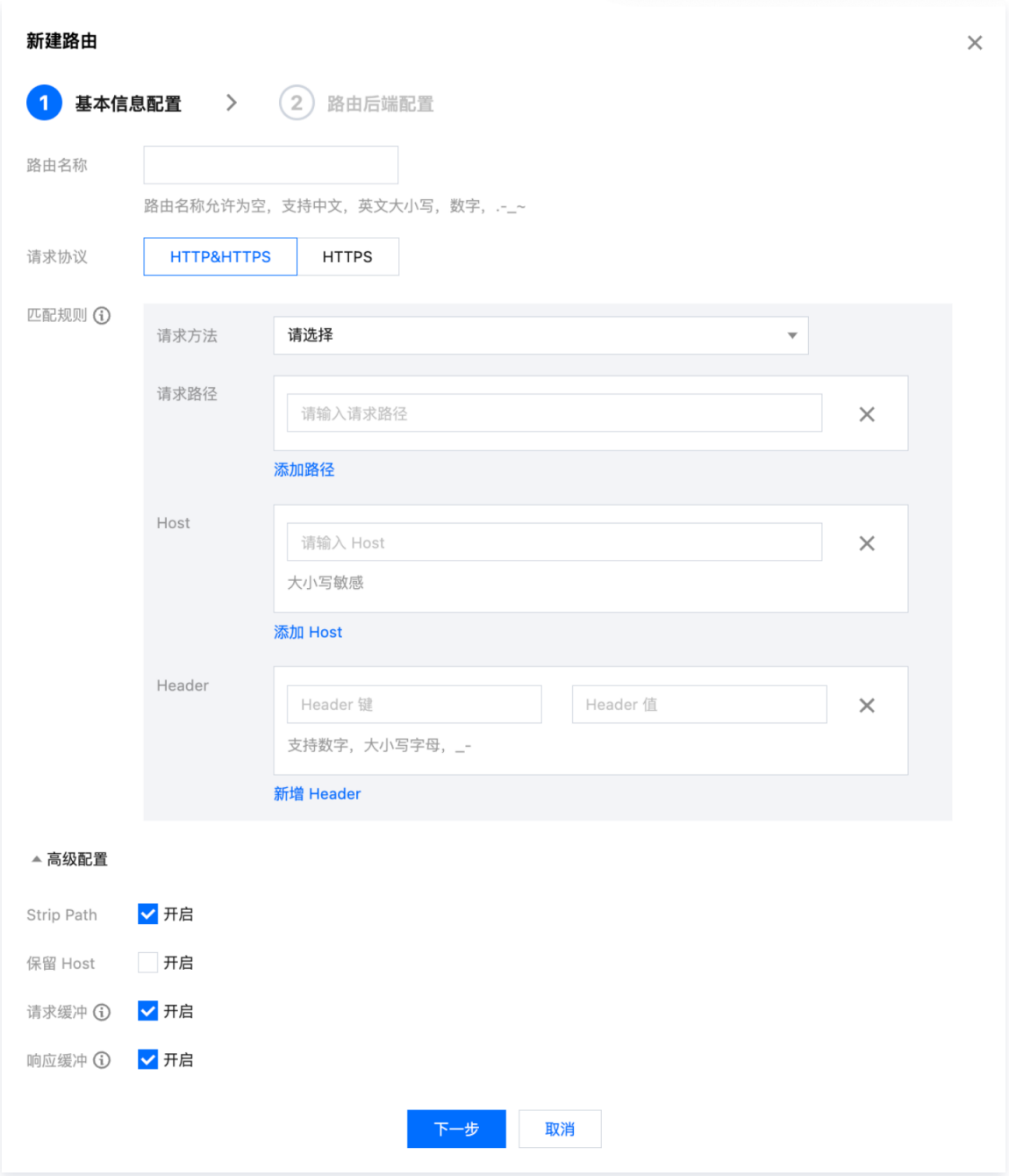
## 操作步骤

### 添加路由

1. 登录 微服务平台控制台。
2. 在左侧导航栏单击**路由管理**，在页面上方选择好实例后，单击**服务**页签。
3. 点击要配置的服务名称，进入服务路由管理页面。
4. 在左上角单击**新建**，填写路由基本信息。
    - 当后端请求协议为 HTTP、HTTPS 时，需要配置的参数如下：

| 参数 | 是否必选 | 说明 |
| --- | --- | --- |
| 路由名称 | 否 | 路由名称允许为空，支持中文，英文大小写，数字，-_~。 |
| 请求协议 | 是 | 支持 HTTP&HTTPS、HTTPS。 |
| 匹配规则 | 是 | 请求方法、请求路径、Host 需至少配置一种；Header 可选配置。 |
| Strip Path | 否 | 当通过请求路径匹配到路由时，从后端服务请求 URL 中去除匹配前缀。默认开启。 |
| 保留 Host | 否 | 请求后端时是否使用请求 Host，默认关闭。 |
| 请求缓冲 | 否 | 默认开启。开启时，网关会缓存客户端请求，再分块转发到后端服务；关闭后，网关在接受到客户端请求后将直接转发给后端服务。如果配置的插件需读取请求 body，不支持关闭缓冲。 |
| 响应缓冲 | 否 | 默认开启。开启时，网关会先缓存后端服务响应，再统一返回客户端；关闭后，网关将直接将响应数据返回客户端。如果关联的插件需读取响应 body，不支持关闭缓冲。 |

版权所有：腾讯云计算（北京）有限责任公司




**说明：**

1. 如需配置正则路由，只需将请求路径配置为正则表达式即可。
2. Host 大小写敏感。

- 当后端请求协议为 TCP/UDP 时，填写路由名称和请求端口即可。




5. 单击**下一步**，进入路由后端配置，确认后端服务信息无误后提交。

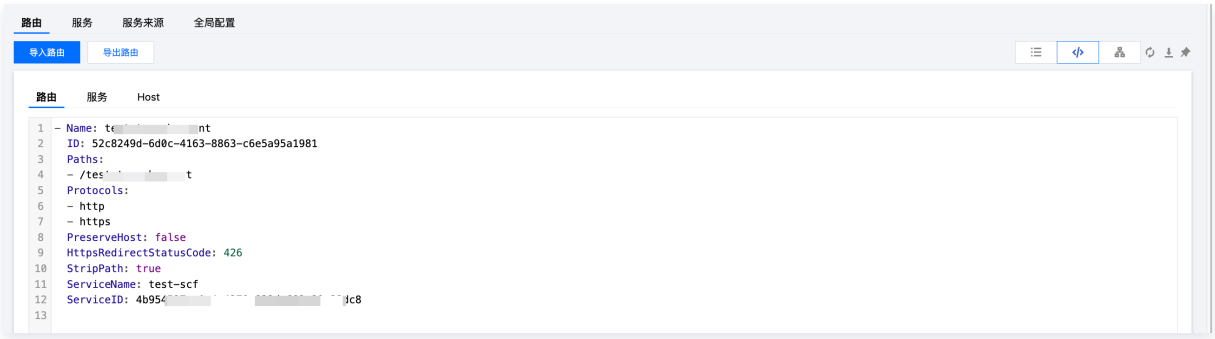



- 单服务：表示当前路由转发请求到后端单个服务。
- 多服务：暂不支持多服务。

6. 单击提交，完成路由创建，在**路由管理 > 路由**列表页面，可以看到创建好的路由，代表创建成功。




7. 在路由列表右上角，您可以切换至 YAML 视图查看信息，




## 验证请求转发

发起 API 请求，验证转发到注册中心/容器集群中的服务。

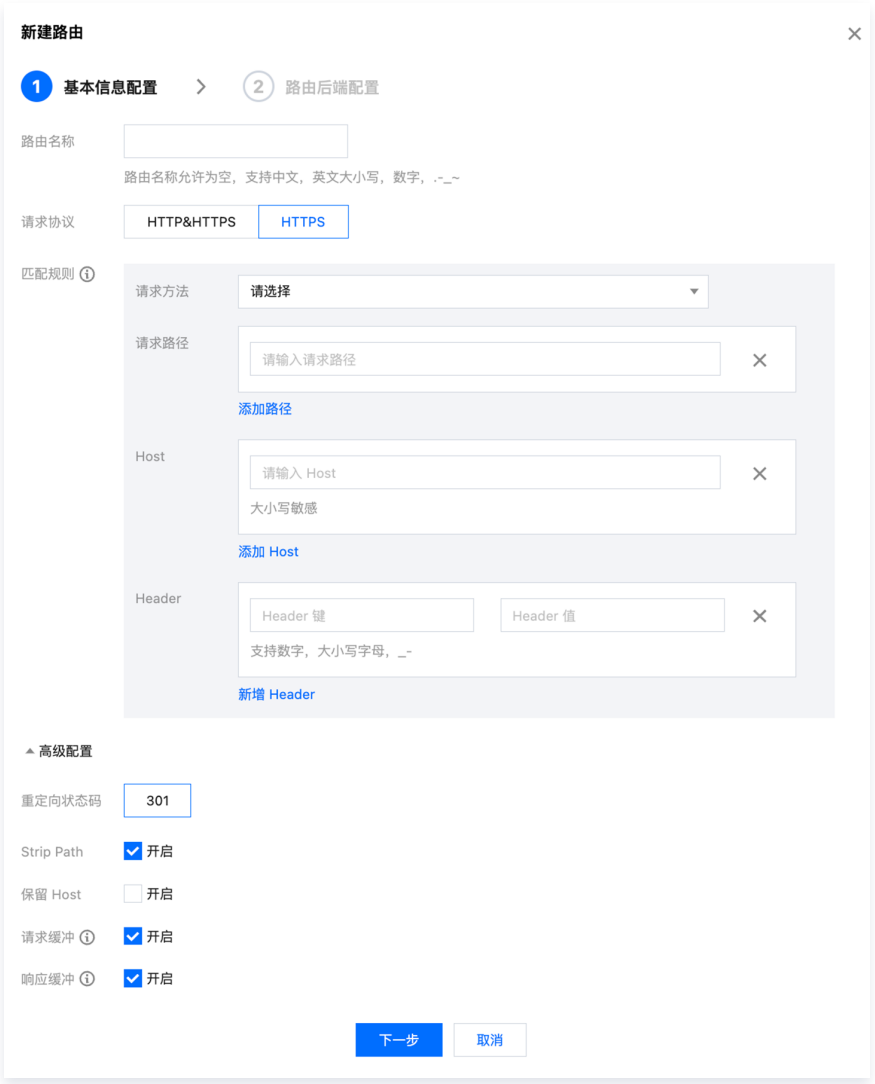
# 应用场景1：强制 HTTPS

## 场景说明

HTTPS 通过数字证书、加密算法以及非对称密钥等技术对信息进行加密处理，有效的保证了数据在传输过程中的安全性和完整性。在一些信息安全要求极高的场景下，需要实现只有 HTTPS 请求才能访问服务，如果是 HTTP 请求则需要完成 HTTP 到 HTTPS 的自动跳转，整个过程对用户来讲是透明的、无感知的。

## 操作方式

1. 创建/编辑路由指定请求协议为 HTTPS，设置重定向状态码为 301（即 Moved Permanently，永久重定向）。




2. 使用 HTTP 请求路由，可以看到请求返回 301 状态码（表示跳转），Location 重定向为 HTTPS 请求。

```
cl[redacted]o-3 ~ % curl -i http://1.1[redacted].80/httpbin
HTTP/1.1 301 Moved Permanently
Connection: keep-alive
Content-Length: 134
Content-Type: text/html
Date: Wed, 09 Aug 2023 12:21:36 GMT
Location: https://1.1[redacted].80/httpbin
Server: kong/2.5.1
X-Kong-Response-Latency: 0

<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
</body>
</html>
```

# 服务来源

最近更新时间：2025-06-09 16:07:02

## 操作场景

目前云原生 API 网关支持注册中心（包括 Nacos、Consul、PolarisMesh 和 TSF 注册中心）、容器服务（包括 TKE 和 EKS）以及私有 DNS 三类服务来源，本文介绍如何将部署了后端服务的容器集群或注册有后端服务的注册中心添加到网关服务来源中，以便网关能获取服务列表，查找对应的服务地址从而实现动态代理。如您的服务使用私有 DNS，可以使用腾讯云私有解析（Private DNS）作为服务来源，从而实现服务私有域名解析。

## 前置条件

已购买云原生 API 网关实例，相关操作详情请参见 网关实例管理 。

## 操作步骤

### 添加服务来源

1. 登录 微服务平台控制台 。
2. 在左侧导航栏单击**云原生 API 网关 > 路由管理**，在页面上方选择好网关实例，单击**服务来源**页签，在服务来源页面单击**新建**。
3. 在新建服务来源窗口中配置相关参数。

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 来源类型 | 是 | 服务来源类型，支持注册中心、容器服务和私有 DNS 三类来源。 |

> **注意：**
> 目前云原生 API 网关仅对微服务场景，即作为 Polaris（北极星）服务接入层支持开通白名单新购，其余场景不支持新接入。非上述场景访问控制台时将重定向至 Polaris（北极星）控制台。

- 当服务来源为注册中心时，填写以下参数：

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 来源产品 | 是 | 来源产品，支持 TSF 注册中心、自建注册中心和 TSF 注册中心，注册中心类型包括 Nacos，Consul，PolarisMesh 和 TSF 注册中心。<br>**说明：**推荐使用TSF PolarisMesh 。 |
| 选择实例 | 是 | 当来源产品为 TSF 注册中心时，选择后端服务所在的注册中心实例。 |
| 来源名称 | 是 | 当来源产品为自建注册中心时，填写来源名称。 |
| 访问地址 | 是 | 当来源产品为自建注册中心时，填写访问地址，多个访问地址使用换行分隔。针对不同注册中心，访问地址书写方式为：<br>• 自建 Nacos：填写域名或 IP：Port<br>• 自建 Consul：填写域名或 IP：Port<br>• 自建 PolarisMesh：填写 http 和 grpc 访问地址作为一组访问地址，每组格式为：http://IP:{HTTP_PORT}、grpc://IP:{GRPC_PORT} |
| 用户名 | 否 | 对于类型为 Nacos 的注册中心，如开启了注册中心鉴权功能，填写访问用户名。 |
| 密码 | 否 | 对于类型为 Nacos 的注册中心，如开启了注册中心鉴权功能，填写访问密码。 |
| 访问 Token | 否 | 对于类型为 Consul 和 PolarisMesh 的注册中心，如开启了注册中心鉴权功能，填写访问 Token。 |

- 当服务来源为容器服务时，填写以下参数：

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| 来源产品 | 是 | 来源产品，支持 TKE 容器服务和 TKE Serverless集群。 |
| 选择实例 | 是 | 选择后端服务所在的容器集群。 |

- 当服务来源为私有 DNS 时，选择腾讯云私有解析即可使用 Private DNS 进行域名解析。

4. 添加后， 查看服务来源列表，包含新建的服务来源信息，则说明服务来源创建成功。




## 删除服务来源

单击**操作 > 删除**，确认是否删除服务来源。确认后将删除服务来源。

> **注意**
> 如果当前服务来源已有服务关联，则不允许删除，需删除服务后重试。

# 灰度发布

最近更新时间：2025-06-09 16:07:02

## 操作场景

灰度发布是软件上线过程中常见的上线方式，是指在发布过程中，将具有一定特征或者比例的流量分配到需要被验证的版本中，用来观察新的验证版本的线上运行状态。相比全量上线，灰度发布是更加谨慎的发布形式。当线上调用链路较为复杂时，全链路灰度发布可以将线上的各个服务隔离出一个单独的运行环境。
使用云原生 API 网关的灰度策略能力，可以在不修改任何业务代码的情况下，可视化配置灰度规则，实现云上轻松易用的全链路灰度发布。本文介绍如何在 Kong 云原生 API 网关控制台上配置灰度发布策略，包括以下两种场景：

- 配置参数路由，以实现根据请求特征（header、query、cookie、path、method 等）进行请求转发。
- 配置百分比路由，以实现根据比例转发到不同的服务。

## 前置条件

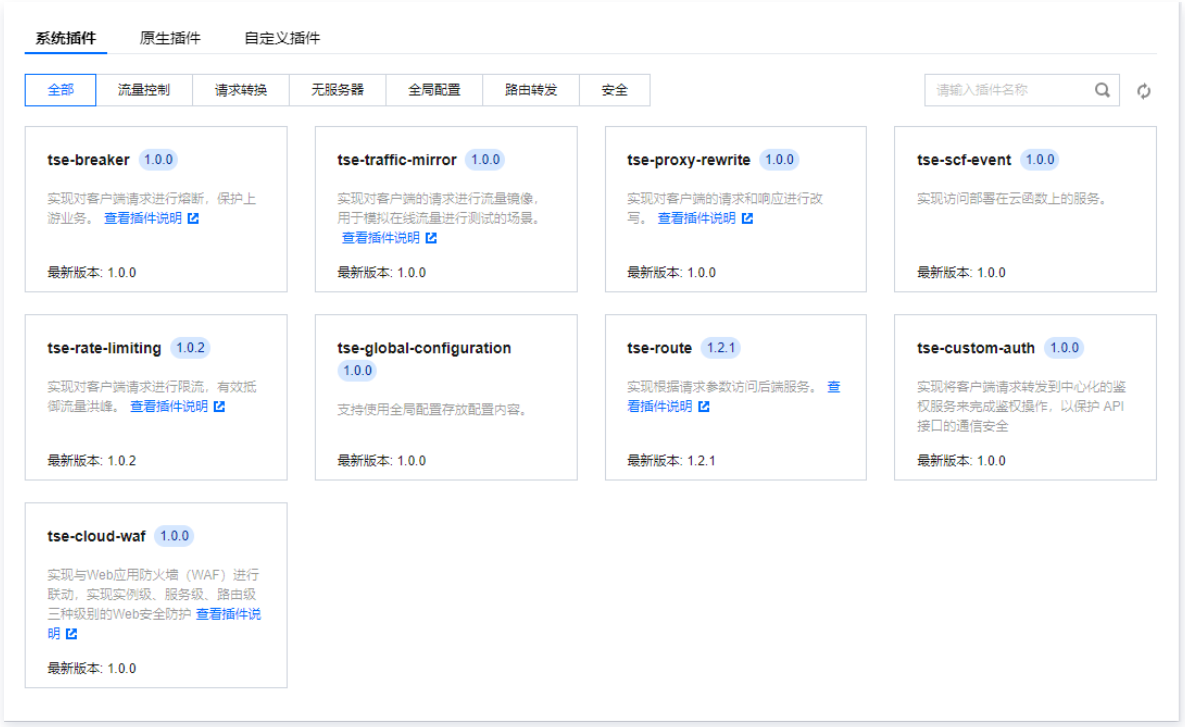
- 已购买云原生 API 网关 Kong 网关实例，详情请参见 操作文档 。
- 配置了后端服务（Service）以及路由（Route），详情参见 服务与路由 。

## 操作步骤

### 步骤1. 更新路由插件

在配置灰度发布策略前，确保您的 tse-route（路由转发）插件已升级到最新版本。

1. 登录 TSF 控制台 。
2. 在左侧导航栏单击**云原生 API 网关 > 插件管理**，在页面上方选择好网关实例。
3. 单击**系统插件**页签，查看插件是否已更新到最新版本，若未更新，单击操作列的**安装最新版本**即可。




### 步骤2.（可选）创建全局配置

> **说明：**
> 当灰度策略中的逻辑关系选择了包含或者不包含，参数值可选择全局配置，此时需要提前创建好全局配置。

1. 在左侧导航栏选择**服务路由 > 全局配置**页签，单击**新建**。




2. 在新建配置页面，输入配置名称和配置内容，配置内容可以选择直接**导入文件**。



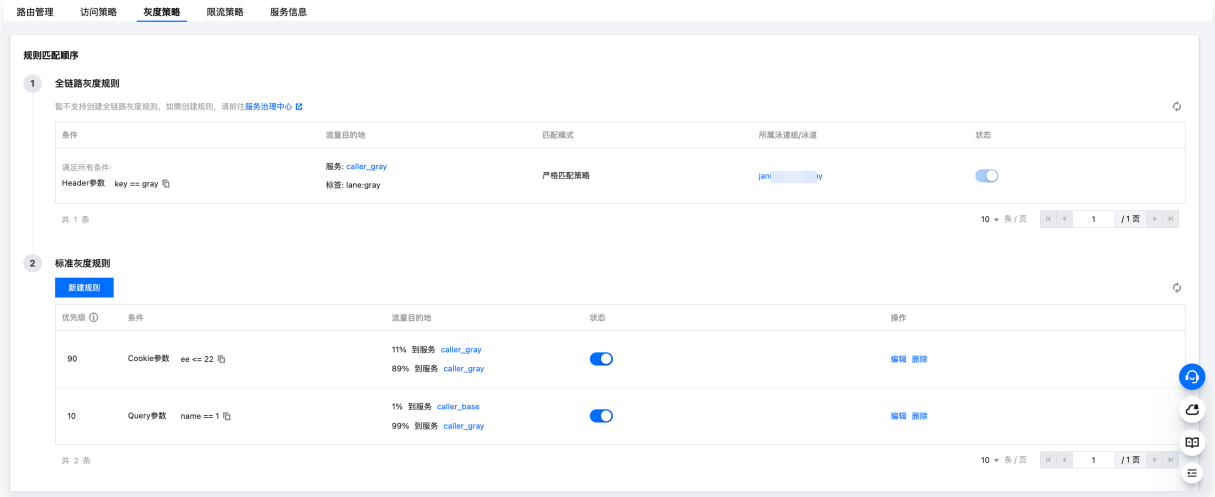

3. 单击**提交**，完成创建。

## 步骤3. 配置灰度策略

1. 在左侧导航栏选择**服务路由 > 服务**页签，单击服务名称，进入服务详情页面。




2. 在页面上方选择灰度策略页签，单击**新建规则**，配置规则信息。

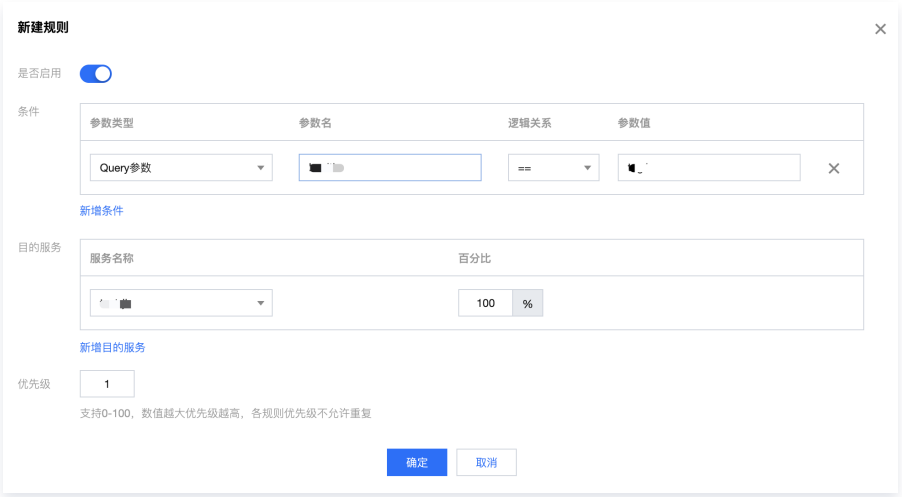



- **是否启用**：启动后，创建好的灰度策略将生效。
- **条件**：路由规则内的多个条件为逻辑与（AND）关系，即一个路由规则内条件都满足了，才会转发到对应的后端服务。

| 参数 | 说明 |
| --- | --- |
| 参数类型 | 选择参数类型，支持的参数类型说明如下：<br>• 支持特定位置的请求参数，目前只支持 Header、Path、Body、Query、Cookie 参数。<br>注意：<br>○ Body参数为请求json中的参数，仅支持content-type为application/json类型。<br>○ Body参数仅对JSON第一层进行解析，当存在同名键，会选择后一个参数值。<br>• 支持请求方法（method）匹配，满足数组中的一个方法即可。<br>• 支持以下常量类型：<br>○ STRING: 字符串类型，支持单引号或双引号，如： "Hello"、'hello'。<br>○ NUMBER: 数字类型，如： 0.1, 100.0, 1。<br>• 支持以下系统参数：<br>○ domain：请求的域名。<br>○ clientIp：客户端 IP。<br>○ httpScheme：请求的协议 HTTP,HTTPS,WS,WSS。<br>○ clientUa：客户端上传的 UserAgent 字段。 |
| 参数名 | 输入参数名称。 |
| 逻辑关系 | 选择逻辑关系：<br>• 支持存在与正则表达式<br>• 支持<，<=, >, >=, ==, != 运算符。<br>• 支持包含和不包含。<br>• 不支持双目运算。<br>• 不支持数学运算。 |
| 参数值 | 输入参数值。当逻辑关系选择了**包含**或者**不包含**，参数值可选择**手动输入**或者**全局配置**，若无全局配置可参考全局配置文档创建配置。 |

- **目标服务**：路由规则内条件都满足后，路由转发到的目标后端服务。支持一个或多个后端服务，所有后端服务权重总和必须为100%。
- **优先级**：支持0-100，数值越大优先级越高，各规则优先级不允许重复。支持配置多条规则，网关将按优先级进行规则匹配，匹配到了即转发到对应的目的后端。

3. 单击**确定**，完成规则创建。




## 步骤4. 验证策略是否生效

请求服务，网关将根据请求参数动态路由到对应的目的后端。

# 配置限流策略

最近更新时间：2025-06-09 16:07:02

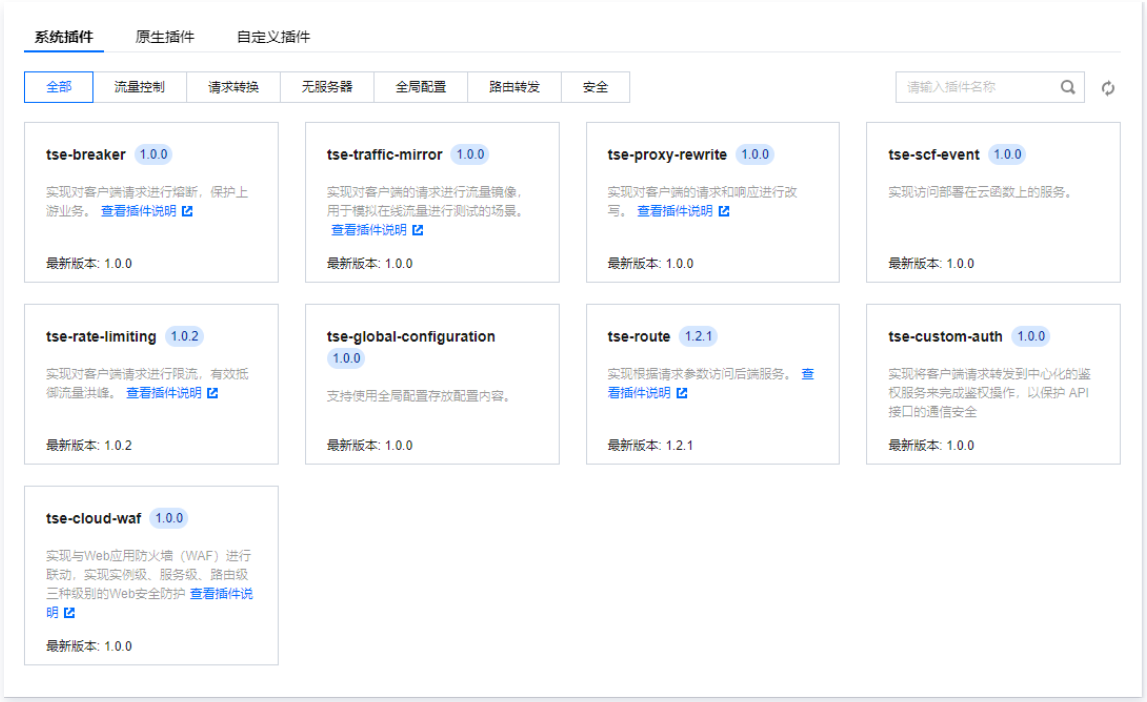
## 操作背景

限流主要是通过设置限流规则保护后端服务，防止瞬时流量过大造成服务和数据崩溃，导致服务不可用。当资源成为瓶颈时，需要对请求做限流，启动流控保护机制。TSF 云原生 API 网关支持**服务级**和**路由级**别的限流，本文介绍限流规则的配置方式。

## 前提条件

在配置限流策略前，请确保您的 tse-rate-limiting（流量控制）插件已升级到最新版本。如果需对限流阈值进行精细化配置，请确保您的 tse-global-configuration（全局配置）插件已升级到最新版本。

1. 登录 TSF 控制台。
2. 在左侧导航栏单击**云原生 API 网关** > **插件管理**，在页面上方选择好网关实例。
3. 选择**系统插件**页签，查看插件是否已更新到最新版本，若未更新，单击操作列的**安装最新版本**即可。




## 服务级限流

服务级限流对服务下所有路由均生效，如需对路由配置限流，您可以前往服务路由页配置。

### 步骤1：配置限流策略

1. 登录 TSF 控制台。
2. 在左侧导航栏单击**云原生 API 网关** > **服务路由**，在页面上方选择好网关实例。
3. 单击**服务**页签，单击服务名称，进入服务详情页面。
4. 在页面上方选择**限流策略**页签，单击**配置限流规则**。

| 参数 | 是否必选 | 说明 |
| --- | --- | --- |
| 是否启用 | 是 | 开启后，限流规则将会生效。 |
| 限流配置 | 是 | • 基础限流：根据请求数阈值进行流量控制。当该服务上的请求数量在设定的时间周期内大于阈值将触发限流。支持秒、分钟、小时、天、月、年六种时间维度。<br>• 参数限流：根据用户请求参数及指定条件进行流量控制。 |

| | | |
|---|---|---|
| | | ○ 过滤条件（可选）：仅对符合一定过滤条件（IP 或 Path）的请求才进行限流统计。<br>○ 限流条件：统计的限流参数到达指定阈值即触发限流，支持依据 Header、Cookie、Path、IP、Query 参数进行流量控制，支持设置固定限流阈值以及使用全局配置针对不同限流参数值进行精细化配置。 |
| 计数策略 | 是 | • 当选择默认 Redis 时，计数器保存在 Kong 实例自带的 Redis 服务节点实现分布式限流，支持300万个 Key。<br>• 存储当选择外部 Redis 时，可配置 Redis 访问地址（服务器地址、连接密码、超时时间），实现更大规模的分布式限流。 |
| 限流处理策略 | 是 | • 直接返回：触发限流直接返回 429。<br>• 自定义返回：触发限流后，自定义 HTTP 应答行为，需输入状态码、响应头和响应内容。<br>• 请求转发：触发限流后，跳转到指定页面，需输入请求转发地址。 |
| 请求排队 | 否 | 请求处理的最大排队时间，支持 0–15 秒。配置为 0，代表不延迟处理。配置大于 0，每隔 1 秒网关重试处理该请求，如果达到最大排队时间还未被处理，则请求被限流。 |
| 隐藏限流响应头 | 否 | 默认开启，隐藏后客户端将不会收到限流相关响应头。 |

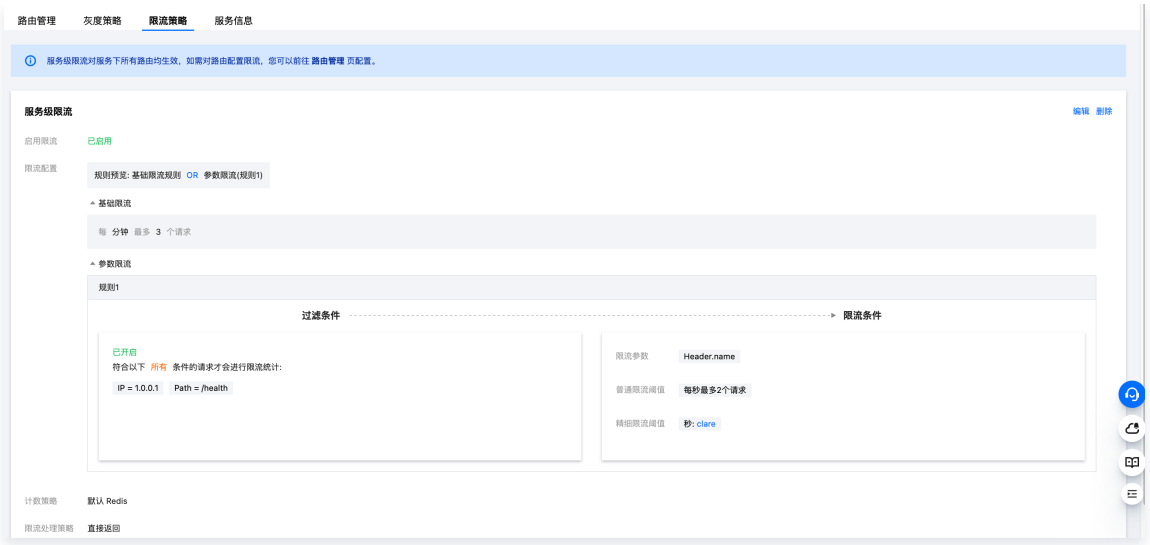



**说明：**

1. 精细限流阈值适用于不同限流参数有不同限流阈值的场景，例如，根据 Query 请求参数 appid 来进行限流，当 appid=1时，每分钟限流100，当appid=2时，每分钟限流200。
2. 精细限流阈值仅在配置了普通限流阈值后才支持配置，普通限流阈值会作为未匹配到指定请求参数的默认限流阈值。
3. 精细限流阈值使用全局配置定义，使用 JSON 格式描述，请按照限流 key:限流阈值格式配置，使用下划线(_)按顺序连接全部限流参数值作为限流 key。例如，当限流依据为 Query 参数 appid，当 appid=1时，限流阈值为 100，appid=2时，限流阈值为 200，全局配置如下：

```
{
  "1": 100,
  "2": 200
}
```

5. 单击**确定**，完成规则创建。




## 步骤2：验证策略是否生效

发起请求触发限流规则，查看限流是否生效。

# 路由级限流

路由级限流仅对本路由生效，如需配置服务限流，您可以前往服务级限流策略页配置。

## 步骤1：配置限流策略

1. 登录 TSF 控制台 。
2. 在左侧导航栏单击**云原生 API 网关 > 服务路由**，在页面上方选择好网关实例。
3. 选择**服务**页签，单击目标服务名称，进入服务路由页面。
4. 单击路由级限流模块的**配置限流规则**。

| 参数 | 是否必选 | 说明 |
| --- | --- | --- |
| 是否启用 | 是 | 开启后，限流规则将会生效。 |
| 限流配置 | 是 | • 基础限流：根据请求数阈值进行流量控制。当该服务上的请求数量在设定的时间周期内大于阈值将触发限流。支持秒、分钟、小时、天、月、年六种时间维度。<br>• 参数限流：根据用户请求参数及指定条件进行流量控制。<br>  ○ 过滤条件（可选）：仅对符合一定过滤条件（IP 或 Path）的请求才进行限流统计。<br>  ○ 限流条件：统计的限流参数到达指定阈值即触发限流，支持依据 Header、Cookie、Path、IP、Query 参数进行流量控制，支持设置固定限流阈值以及使用全局配置针对不同限流参数值进行精细化配置。 |
| 计数策略 | 是 | • 当选择默认 Redis 时，计数器保存在 Kong 实例自带的 Redis 服务节点实现分布式限流，支持300万个 Key。<br>• 存储当选择外部 Redis 时，可配置 Redis 访问地址（服务器地址、连接密码、超时时间），实现更大规模的分布式限流。 |
| 限流处理策略 | 是 | • 直接返回：触发限流直接返回 429。<br>• 自定义返回：触发限流后，自定义 HTTP 应答行为，需输入状态码、响应头和响应内容。<br>• 请求转发：触发限流后，跳转到指定页面，需输入请求转发地址。 |
| 请求排队 | 否 | 请求处理的最大排队时间，支持 0-15 秒。配置为 0 ，代表不延迟处理。配置大于 0 ，每隔 1 秒网关重试处理该请求，如果达到最大排队时间还未被处理，则请求被限流。 |
| 隐藏限流响应头 | 否 | 默认开启，隐藏后客户端将不会收到限流相关响应头。 |

5. 单击**确定**，完成规则创建。

## 步骤2：验证策略是否生效

发起请求触发限流规则，查看限流是否生效。

# 健康检查

最近更新时间：2025-06-09 16:07:02

## 操作背景

云原生 API 网关提供健康检查机制，通过配置健康检查相关策略，网关会对服务节点进行“探活”、“探死”，当服务节点发生故障时，可将请求代理到健康的节点上，以增强网关服务的连续性。

> **注意：**
> 云原生 API 网关健康检查机制仅适用于 IP 列表类型的服务，K8S 服务建议使用容器健康检查机制。

## 健康检查的方式

云原生 API 网关提供两种健康检查的方式：主动检查和被动检查。

## 主动检查（active checks）

主动检查是指网关通过预设的探针类型，主动探测服务节点的存活性。当启用主动健康检查后，网关会按照预设的探测周期，向每个服务节点配置的路径发出 HTTP 或 HTTPS 请求， 网关将根据探测结果自动启用处于健康状态的节点，并禁用不健康的节点。

## 被动检查（passive checks）

被动检查也被称为断路器模式，该模式下网关会基于网关代理的（HTTP/HTTPS/TCP）进行检查，不会主动发起对节点的探测。当节点无响应时，被动健康检查将检测该目标并将其标记为不健康。

> **说明：**
> 由于该模式下不健康的节点无法收到请求，即使节点恢复健康，网关也无法探测到该节点恢复进而重新将该节点标记为健康，只能通过管理端 API 手动触发并通知健康检查器再次启用该节点。因此通常需要结合主动健康检查策略。

## 如何配置健康检查

新创建的 IP 列表类型的网关服务，可在服务列表页或服务详情页配置健康检查。




## 配置主动健康检查

在网关服务列表页面，单击健康检查列处的编辑按钮，进行健康检查配置。

- 被动健康检查：通过判断上游节点的响应状态确定节点健康状态。
- 主动健康检查：网关按照预设的探测周期，主动探测服务节点的存活性。
  - 请求路径：网关向目标节点发出主动探测请求时使用的路径，默认为“/”。
  - 健康检查间隔：两次对健康节点主动进行健康检查的间隔时间，支持 2-300 秒。
  - 异常检查间隔：两次对异常节点主动进行健康检查的间隔时间，支持 2-300 秒。
- 健康 HTTP 状态码：用于判断健康检查正常的 HTTP 状态码，返回上述 HTTP 状态码，节点将被标记为健康。健康 HTTP 状态码自动填充以下健康 HTTP 状态码：
  - 2xx：200, 201,202,204

版权所有：腾讯云计算（北京）有限责任公司

- 3xx：301，302，303，304, 307，308
- 4xx：400, 401, 403, 404，410, 411,429, 499

- 异常 HTTP 状态码：用于判断健康检查异常的 HTTP 状态码，返回上述 HTTP 状态码，节点将被标记为异常。异常 HTTP 状态码自动填充以下异常 HTTP 状态码：
  - 5xx：500，502，503，504
- 超时阈值：连续健康检查超时次数，超出该值认为该节点不健康，默认 10 次，可配置 1–10 次。
- 连续健康检查阈值：连续健康检查成功次数，超出该值认为该节点健康，默认 1 次，可配置 1–10 次。
- 连续异常阈值：连续健康检查失败次数，超出该值认为该节点异常，默认 10 次，可配置 1–10 次。

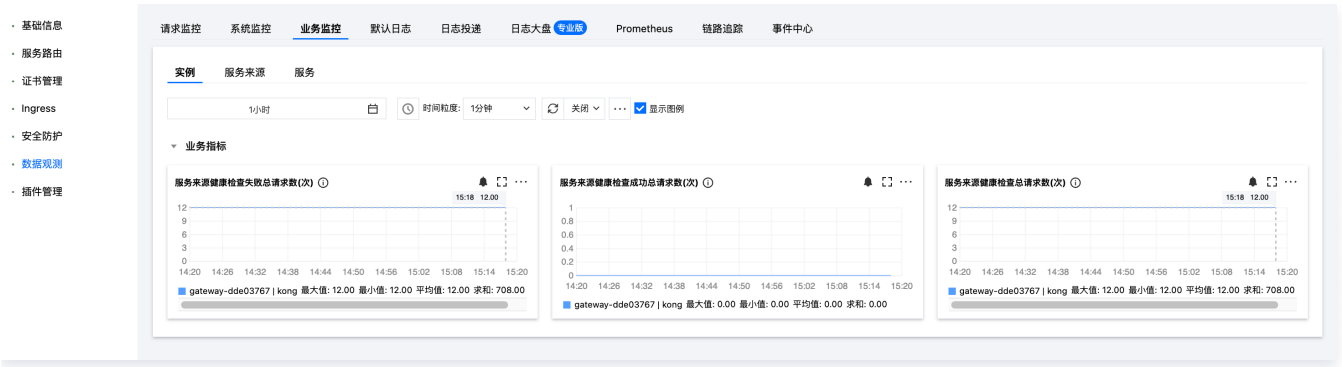
> **说明：**
> 首次开启健康检查，网关将根据健康 HTTP 状态码范围进行节点健康状态标记，请确保 HTTP 状态码配置正确。




## 监控健康检查状态

配置完健康检查后，在左侧导航栏选择**数据观测**，页面顶部选择**业务监控** > **业务指标**：

- 通过服务筛选，可查看该服务下的节点健康状况。
- 通过实例视图筛选，可查看该实例下所有服务的节点健康状况聚合情况。
- 支持为健康检查健康指标配置告警。




## 查看节点状态

在服务详情页面的节点信息模块，可以查看到该服务下各节点的健康状况。您可以在节点列表的操作栏手动修改节点健康状态。



# 全局配置

最近更新时间：2025-06-09 16:07:02

TSF 云原生 API 网关支持使用全局配置存放配置内容，用于配置限流规则或者灰度策略。

- 限流配置：使用全局配置定义不同参数值的限流阈值，以实现更精细化的流控。
- 灰度配置：用于配置灰度名单，当灰度策略的逻辑关系选择了**包含**或者**不包含**，参数值可选择全局配置。

## 操作步骤

### 新建全局配置

1. 登录 TSF 控制台 。
2. 在左侧导航栏单击**云原生 API 网关** > **服务路由**，在页面上方选择好网关实例。
3. 选择**全局配置**页签，单击**新建**，您可以直接输入配置内容，也可以下载模板填写后，导入控制台进行提交。
   - 限流配置：JSON 格式，用于配置精细化限流阈值。
     - 精细限流阈值适用于不同限流参数有不同限流阈值的场景，例如，根据 Query 请求参数 appid 来进行限流，当 appid=1时，每分钟限流100，当appid=2时，每分钟限流200。
     - 精细限流阈值仅在配置了普通限流阈值后才支持配置，普通限流阈值会作为未匹配到指定请求参数的默认限流阈值。
     - 精细限流阈值使用全局配置定义，使用 JSON 格式描述，请按照限流 key:限流阈值格式配置，使用下划线(_)按顺序连接全部限流参数值作为限流 key。例如，当限流依据为 Query 参数 appid，当 appid=1时，限流阈值为 100，appid=2时，限流阈值为 200，全局配置如下：

     ```
     {
       "1": 100,
       "2": 200
     }
     ```

   - 灰度配置：文本格式，用于配置灰度名单，支持英文逗号，英文分号，空格，换行符作为分隔符。配置灰度规则时，当逻辑关系选择了**包含**或者**不包含**，参数值可选择手动输入或者**全局配置**。
4. 单击**确定**，完成规则创建。




## 后续步骤

全局配置创建完成后，可用于配置限流规则或者灰度策略，具体操作步骤请参考：

- 配置限流规则
- 配置灰度策略

# 证书管理

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文档介绍如何在微服务平台 TSF 控制台上管理您的 SSL 证书，包括新建、删除、编辑证书等操作。

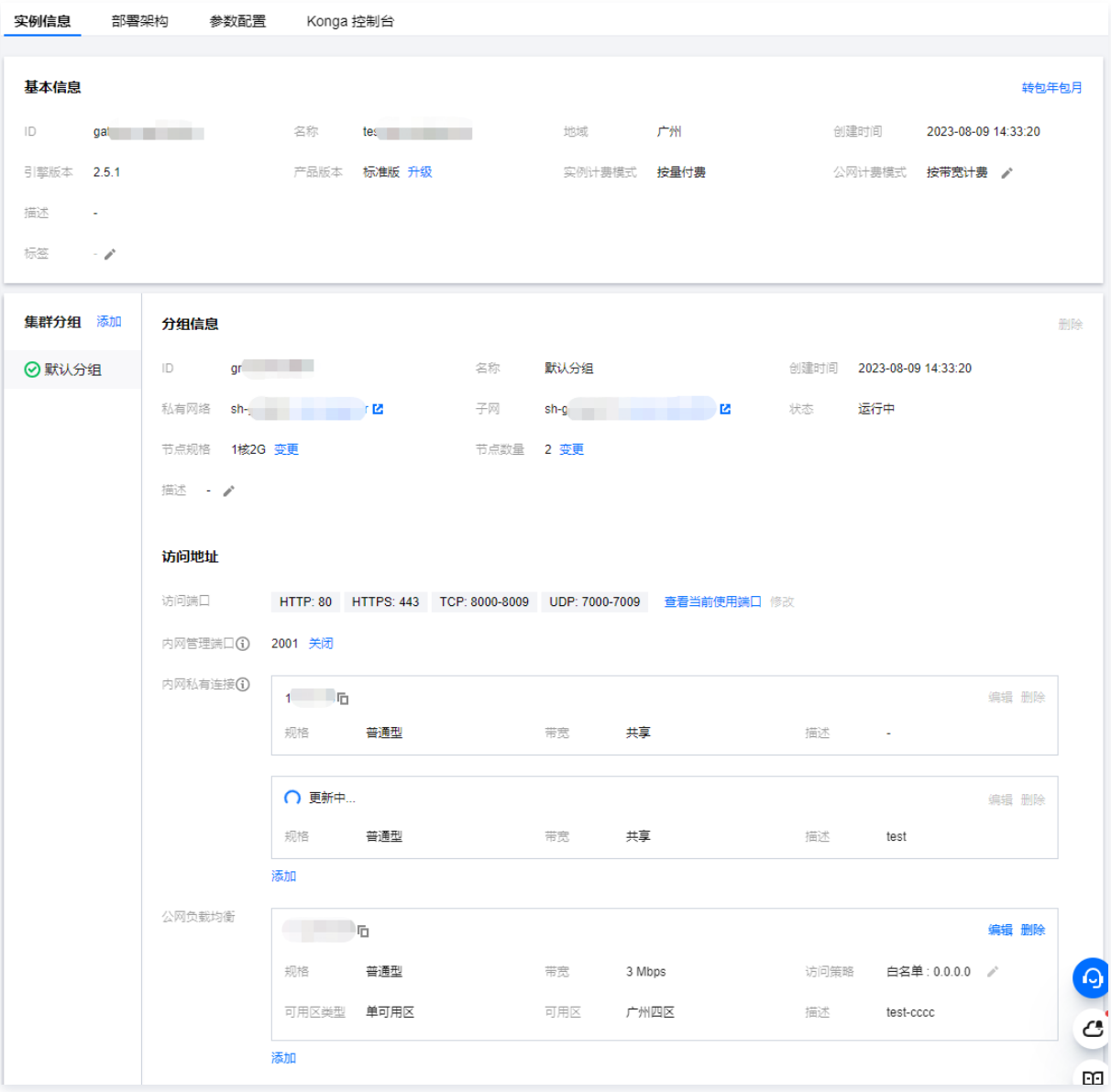
## 前提条件

1. 已经完成域名注册、实名认证、备案等流程，若您尚未完成，可以使用腾讯云提供的域名服务，查看指引 。
2. 已经拥有 SSL 证书，若您暂无证书，可以使用腾讯云提供的证书服务，查看指引 。

## 新建证书

### 步骤1：配置 DNS 解析

1. 前往 TSF 控制台，获取 Kong 公网访问地址 IP。




2. 根据 快速添加域名解析 指引，将您的域名解析至 Kong 公网代理地址中的 IP。

### 步骤2：新建证书

1. 在左侧导航栏选择**证书管理**，单击**新建**。
2. 填写证书名称，选择证书并填写您的域名。**仅支持 SVR 类型证书。**




3. 单击**确定**，完成证书新建，返回证书列表查看创建好的证书。

## 步骤3：验证证书是否生效

访问 `https://域名` 验证是否生效。

## 更新证书

在证书过期前，您可以在 TSE 控制台上更新证书。

1. 在证书列表页面，单击您要更新的证书操作栏的**更新证书**。




2. 在弹窗中更新证书后，单击**确定**，完成更新。




## 编辑证书

**您可以在 TSE 控制台上修改证书的证书名称和绑定的域名。**

**1. 在证书列表页面，单击您要更新的证书操作栏的编辑。**




**2. 在弹窗中输入证书名称和绑定域名，单击确定，完成修改。**



# 删除证书

> ⚠ **注意：**
> 删除后，该证书绑定的域名将无法使用 HTTPS 访问网关。

1. 在证书列表页面，单击您要更新的证书操作栏的**删除**。
2. 在弹窗中单击**确定**，完成删除。

# 域名管理

最近更新时间：2025-06-09 16:07:02

本文将快速引导您如何在云原生 API 网关中，使用自定义的域名访问 API 网关。云原生 API 网关，提供了 IP 访问的方式，并没有提供域名接入点。因此您需要在自己的域名 DNS 添加一个 A 记录，指向云原生 API 网关提供的 IP。

## 概述

您需要依次完成以下步骤：

1. 在 DNS中 添加 A 记录，绑定云原生 API 网关提供的 IP。
2. 绑定证书（如果有使用 HTTPS 访问网关的需要）。

## 操作步骤

### 步骤1：在 DNS 中添加 A 记录，绑定云原生 API 网关提供的 IP

网关提供的 IP（以公网 IP 为例），在**实例列表** > **实例信息** >**网络配置** > **公网负载均衡**中可以看到网关提供的 IP。把域名跟此公网 IP 绑定即可。




### 步骤2：绑定证书（如果有 HTTPS 访问的需要）

绑定证书步骤请参见 证书管理 。

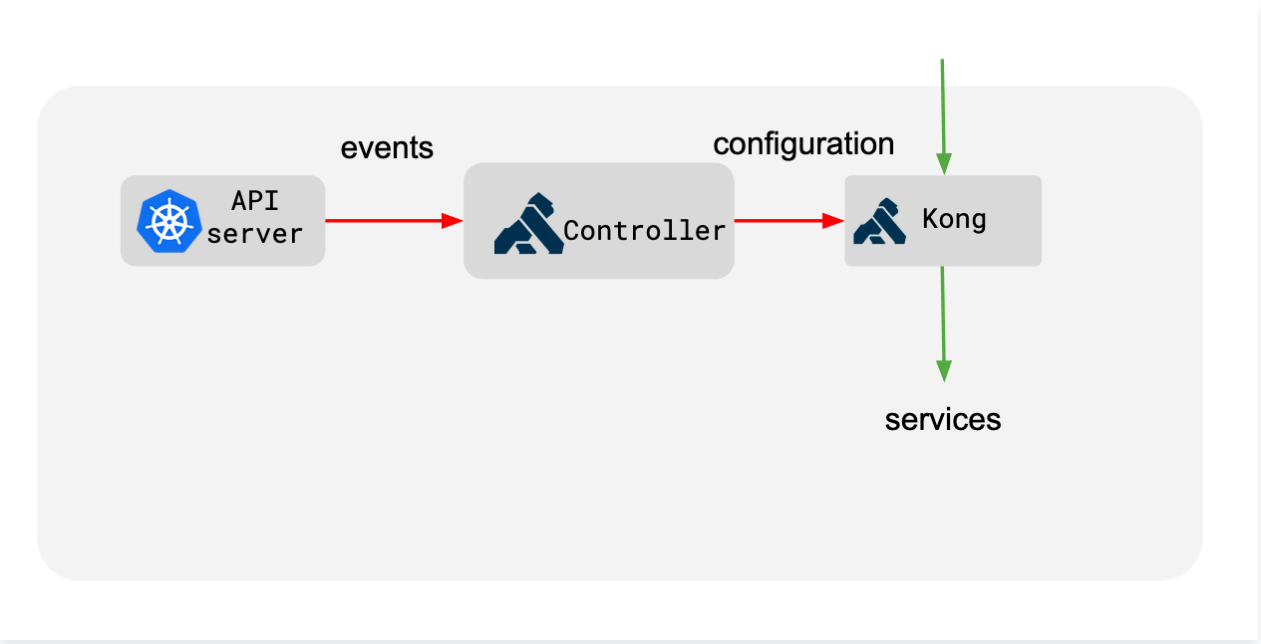
> **注意：**
> 如果没有绑定证书，使用 HTTPS 协议，访问自定义域名的时候，会出现证书跟自定义域名不一致的情况。如果您是使用 curl 工具进行测试，使用加上 `-k` 的参数跳过对服务端证书的校验。在生产的环境，使用 HTTPS 需要绑定证书。

# Ingress
# Kong Ingress介绍

最近更新时间：2025-06-09 16:07:02

## Kong Ingress Controller 介绍

Kong 网关一般通过 Admin API 进行资源管理，可管理的资源包括 Service、Route、Plugin、Consumer 等，在定义资源规则后，Kong 将按照这些配置规则进行对上游服务的请求进行路由分发和控制。
在 Kubernetes 集群环境下，Admin API 方式不是很适应 Kubernetes 声明式管理方式。所以 Kong 在 Kubernetes 集群环境下推出Kong Ingress Controller，涵盖了原 Admin API 的各个方面。因此，Kong Ingress Controller 的作用不仅仅是代理进入 Kubernetes 集群的流量，也可以实现插件配置、负载平衡、健康检查等 Kong 提供的所有功能。
下图展示了它的工作原理：



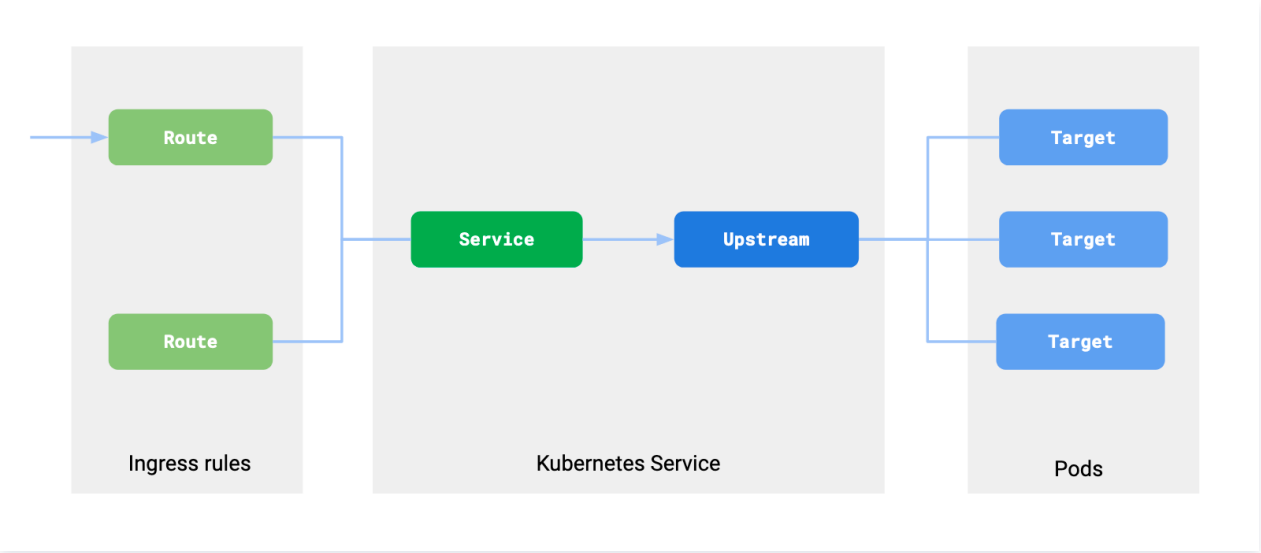

当您安装了 Kong Ingress Controller，Controller 管理器将监听 Kubernetes 集群内部发生的变化，并更新 Kong 以响应这些变化，从而能够正确代理所有流量。与此同时，Kong 也会根据 Controller 同步的信息进行动态更新，以响应 Kubernetes 集群内发生的扩展、配置更改和故障等变化。

## Kubernetes 资源与 Kong 资源对应关系

Kong Ingress Controller 旨在将您配置在 Kubernetes 集群中的资源同步到 Kong ，并生成对应的 Kong 资源以便能够使用 Kong 原有机制进行网关操作。两者资源的对应关系如下：

- Kubernetes Service：对应 Kong Service 和 Kong Upstream
- Kubernetes Pod：对应 Kong Target
- Kubernetes Ingress：对应 Kong Route

版权所有：腾讯云计算（北京）有限责任公司




## Kubernetes 版本与 Kong Ingress Controller 版本对应关系

| Ingress 版本 | 1.3.4 | 2.5.0 | 2.7.0 |
| --- | --- | --- | --- |
| Kubernetes 1.14 | ✔ | – | – |
| Kubernetes 1.16 | ✔ | – | – |
| Kubernetes 1.17 | – | – | – |
| Kubernetes 1.18 | ✔ | ✔ | – |
| Kubernetes 1.19 | – | – | – |
| Kubernetes 1.20 | ✔ | ✔ | – |
| Kubernetes 1.21 | – | – | – |
| Kubernetes 1.22 | – | ✔ | ✔ |
| Kubernetes 1.23 | – | – | – |
| Kubernetes 1.24 | – | ✔ | ✔ |
| Kubernetes 1.25 | – | – | – |
| Kubernetes 1.26 | – | – | ✔ |
| Kubernetes 1.27 | – | – | – |
| Kubernetes 1.28 | – | – | ✔ |

版权所有：腾讯云计算（北京）有限责任公司

# 使用 Kong Ingress Controller

最近更新时间：2025-06-09 16:07:02

## 操作场景

Kong 通过 Ingress Controller 实现为 Kubernetes Service 配置插件、健康检查、负载均衡等功能。通过整合 Kong Ingress Controller，Kong 可直接关联到 Kubernetes 整个生命周期，Ingress Controller 将监听容器集群变化，并更新 Kong 的配置，以便能正确代理所有的流量，免去人工管理的困扰。

## 前提条件

- 已创建云原生 API 网关实例 。
- 已购买腾讯云容器服务，包括 TKE 标准集群 或 TKE Serverless 集群 。

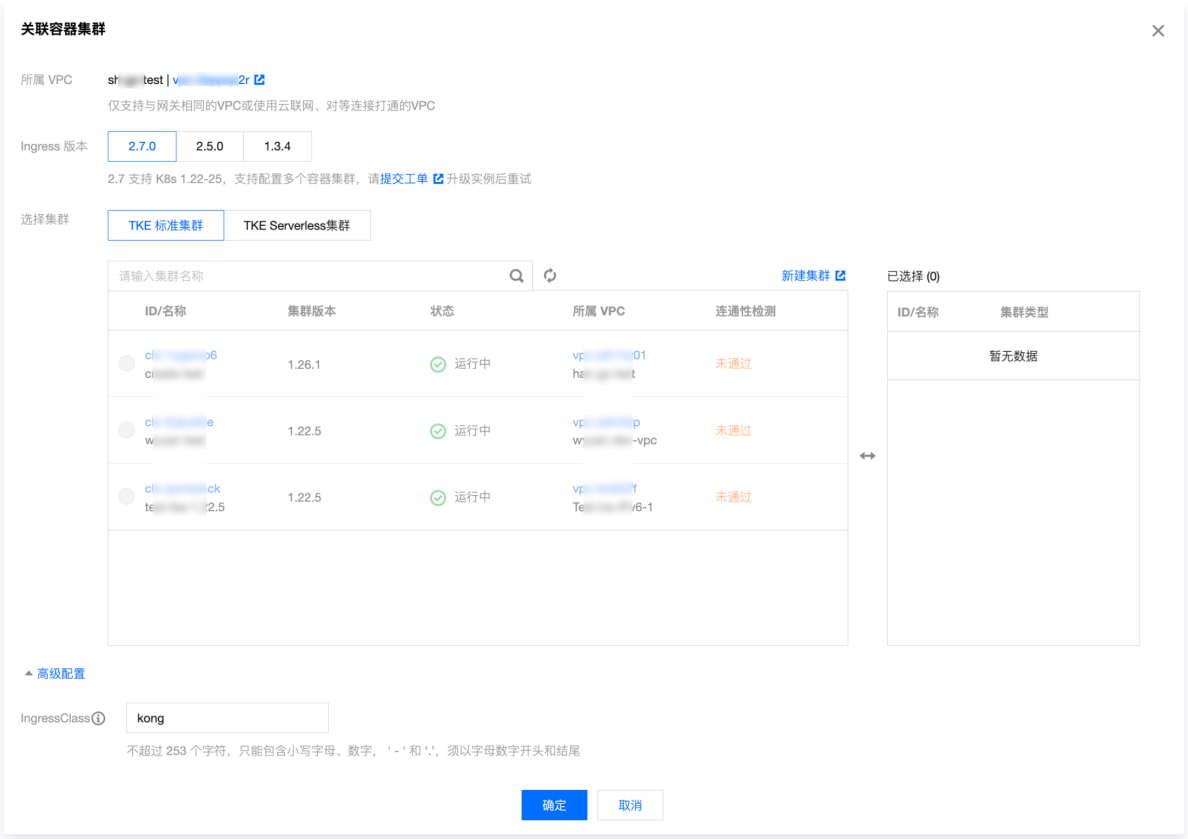
## 操作步骤

### 步骤1：启用 Kong Ingress Controller

1. 登录 TSF 控制台 。
2. 在左侧导航栏单击**云原生 API 网关** > **实例详情** > **Ingress**，在页面上方选择网关实例。
3. 未关联容器集群时，页面提示需关联容器集群。

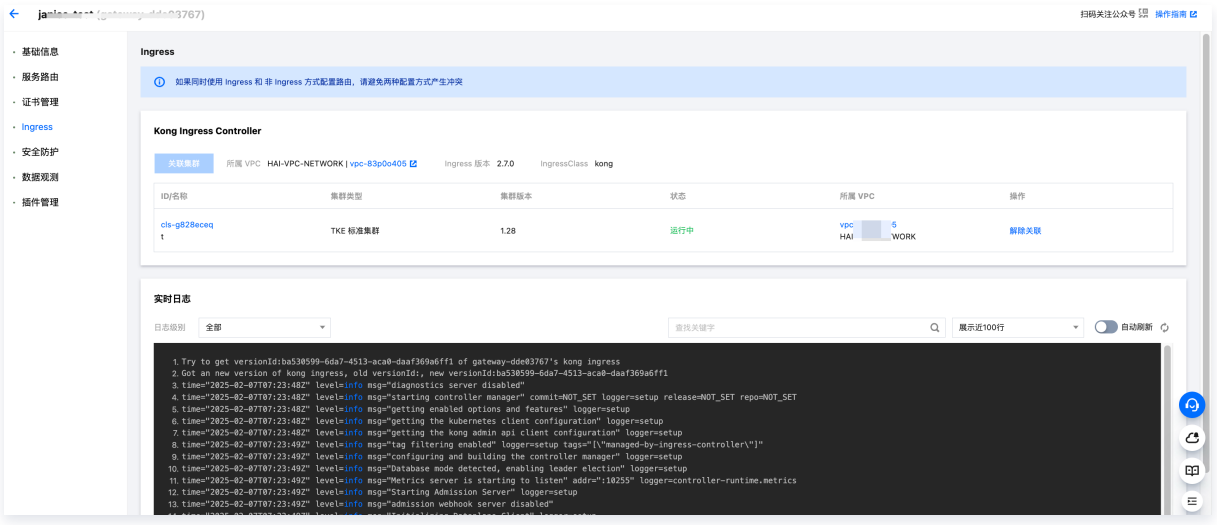



4. 单击**立即关联容器集群**，选择对接集群类型和集群信息，单击**确定**，即启用 Kong Ingress Controller。




> **说明：**
>
> - 请选择与网关实例网络连通的容器集群，包括在相同 VPC，使用云联网或对等连接打通的 VPC。
> - Ingress 版本支持 2.7.0，2.5.0 和 1.3.4，仅 Ingress 2.7.0 版本支持关联多容器集群。
> - IngressClass 默认为 kong，支持配置自定义 IngressClass，用来标识网关实例。

5. 确认已展示选择的容器集群信息。




6. 创建对应 Ingress 规则，请配置 Kubernetes Ingress 资源中的 ingress.class 为步骤 4 中配置的 IngressClass 名称。

> **注意：**
>
> 对于 apiVersion 为 v1beta1 和 v1 版本的 Ingress 资源，配置方式有所不同，请参考如下示例进行配置。有关不同版本的详细区别，请参见 Kong 官方文档说明 。

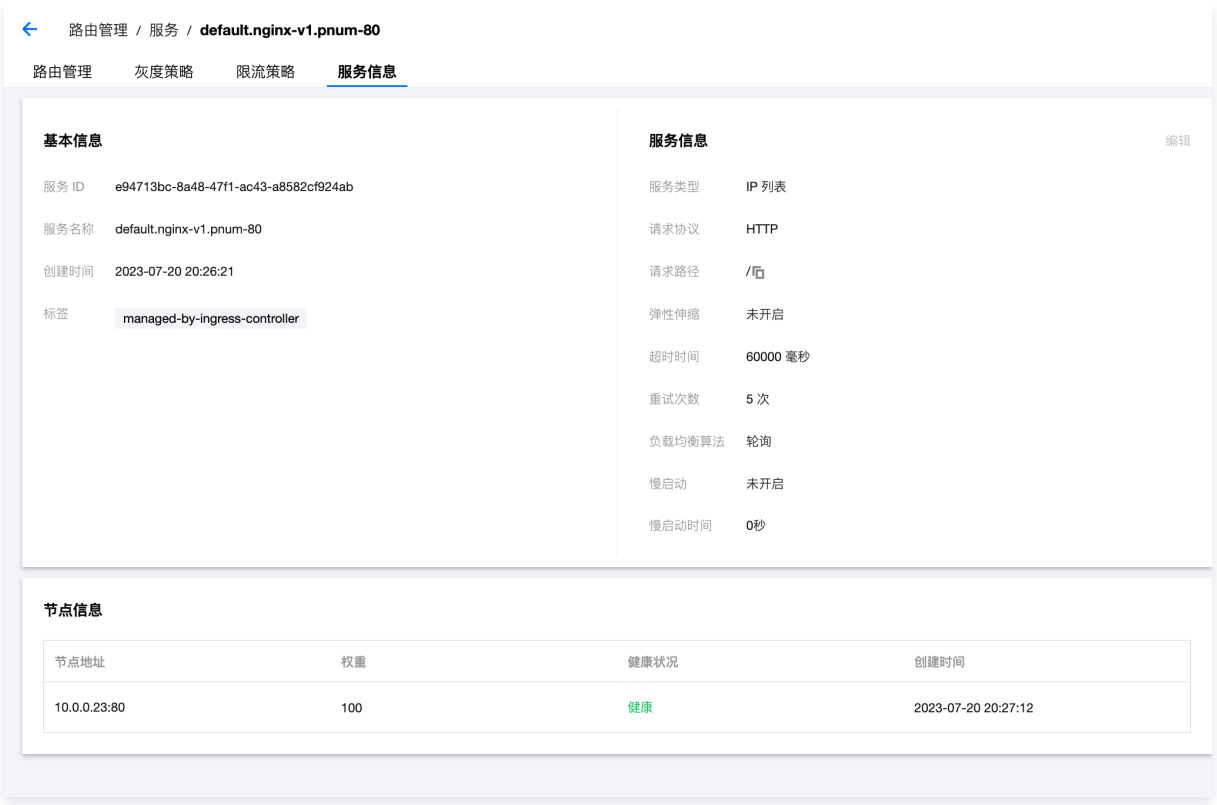
- 对于 apiVersion 为 v1beta1 版本的 Ingress 资源，请参考如下示例进行配置：

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: demo
  annotations:
    kubernetes.io/ingress.class: kong
spec:
  rules:
  - http:
      paths:
      - path: /tse
        backend:
          serviceName: nginx
          servicePort: 80
```

- 对于 apiVersion 为 v1 版本的 Ingress 资源，请参考如下示例进行配置：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: demo-v1
  annotations:
    konghq.com/plugins: "httpbin-auth"
spec:
  ingressClassName: kong
  rules:
  - http:
      paths:
      - path: /demo-v1
        pathType: Prefix
        backend:
          service:
           name: nginx
           port:
             number: 80
```

7. 进入服务列表页面，查看是否已生成对应的服务。
8. 若已生成对应的服务，单击服务的名称，进入服务详情页面，单击**服务信息**页签，查看是否有对应的节点信息。




## 步骤2：停止 Kong Ingress Controller

1. 在 Kong 实例详情页，查看基本信息 > Kong Ingress Controller 卡片。




2. 单击**解除关联** 操作，确认后解除集群关联。



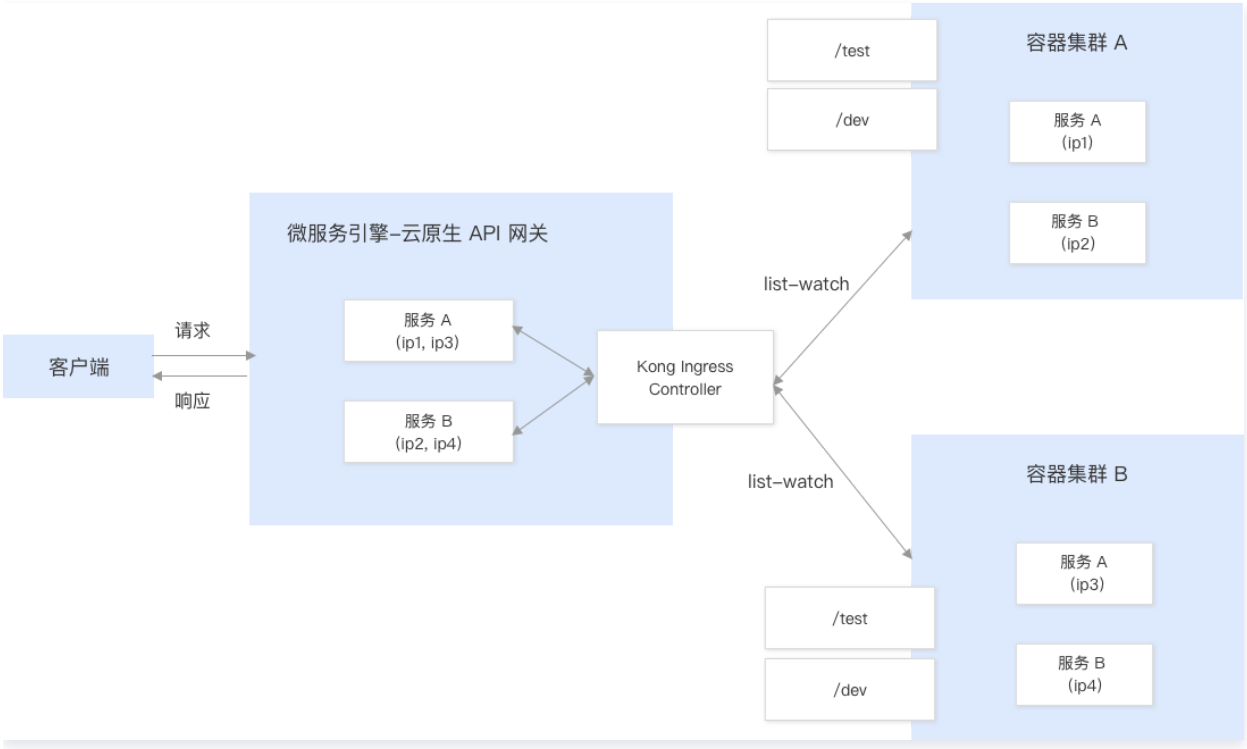

> **注意：**
> 停止后，Kong 不再监听容器集群变化，资源变更无法同步到网关实例中，通过网关访问容器服务可能出现异常。

# 使用云原生 API 网关管理多容器集群

最近更新时间：2025-06-09 16:07:02

## 操作场景

在生产系统中，为了保证服务的高可用，容器集群通常使用对等部署的方式，两个容器集群部署相同的服务，当一个容器集群出现故障时，请求能访问对等的容器集群上的服务，从而保证服务的高可用。云原生 API 网关提供针对该场景的解决方案，支持在开启 Kong Ingress Controller 时关联多个容器集群，网关将同步容器集群 Ingress 上的信息以生成网关资源。当任意容器集群修改资源定义（Service，Ingress，Plugin）时，实时同步更新所有其他容器集群。




如图所示，云原生 API 网关关联了容器集群 A 和 B，Ingress Controller 将自动同步集群 A 和 B 上的资源：

1. Kubernetes Service：不同名服务将分别生成 Kong Service 和 Upstream，同名服务将进行 Pod IP 合并，展示在服务的节点信息中。
2. Kong Plugin：生成对应的插件。
3. Ingress：生成对应的路由。

## 集群故障处理

1. 当集群 A 的 Pod 发生故障时，K8S 机制将自动剔除该节点，网关节点也将自动删除该节点。
2. 当集群 A 的 Api Server 无法连接时，网关将根据健康检查机制自动转发到健康的节点，网关将根据健康检查机制自动转发到健康的节点，以保证服务的连续性。开启健康检查请参考 配置健康检查 。

## 前提条件

- 已创建云原生 API 网关实例 。
- 已购买腾讯云容器服务，包括 TKE 标准集群 或 TKE Serverless 集群 。

## 操作步骤

### 步骤1：创建容器资源

1. 登录 容器服务控制台 。
2. 创建并部署容器服务，创建方法可参见 创建集群 。

> **注意：**
> 相同命名空间、服务名称、端口号及服务端口名称的服务才会进行 Endpoint 合并，请在Service的 ports 中完整定义 protocol、port、targetPort 和 name。具体可参见以下示例。

示例：

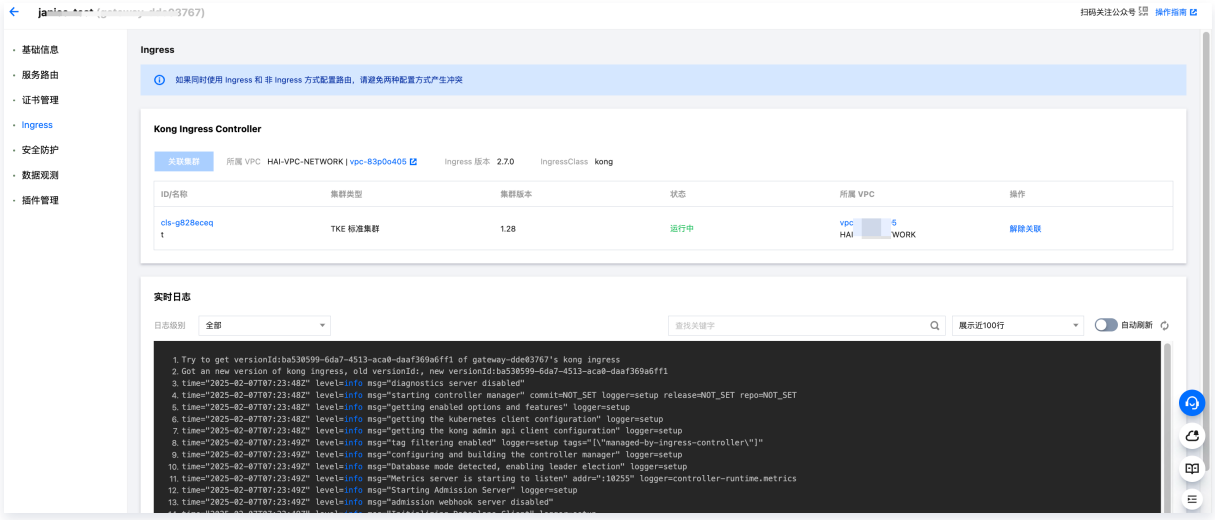
```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
      name: http-mvc
    - protocol: TCP
      port: 443
      targetPort: 8443
      name: https-mvc
```

3. 参见 Ingress 创建 Ingress。

> **说明：**
> 部署 Ingress 后，Ingress 会在容器集群间进行同步创建和更新。

## 步骤3：关联容器集群

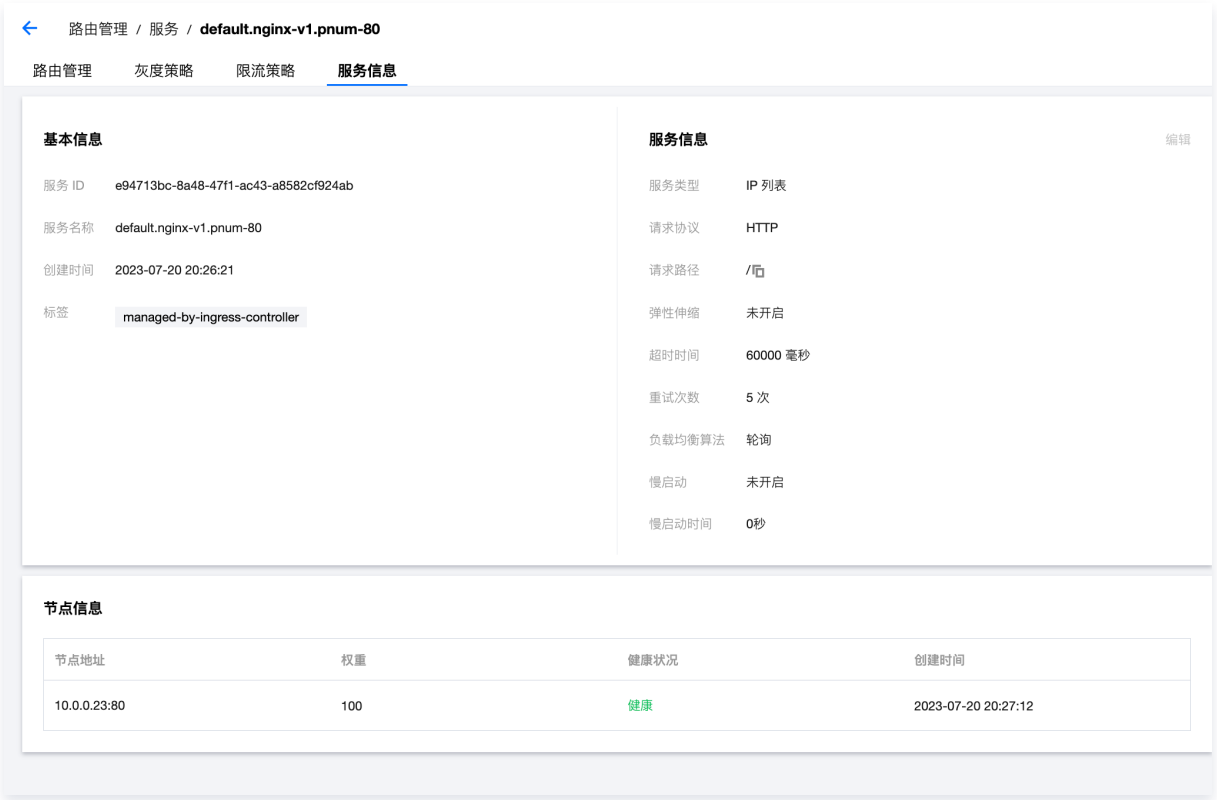
1. 登录 TSF 控制台 。
2. 在左侧导航栏单击**云原生 API 网关** > **Ingress**，在页面上方选择网关实例。
3. 未关联容器集群时，页面提示需关联容器集群。
4. 单击**立即关联容器集群**，选择对接集群类型和集群信息，单击**确定**，即启用 Kong Ingress Controller。
5. 确认集群绑定成功，状态为**运行中**。




## 步骤4：验证服务信息

1. 在左侧导航栏单击**云原生 API 网关** > **路由管理**，在页面上方选择网关实例。

2. 单击**服务页**签，确认从容器集群同步过来的服务显示正确。
3. 单击服务名称，进入服务详情页面，单击**服务信息** 页签，确认节点信息包含来自多个容器集群的 Endpoint。



# 灰度配置 Ingress

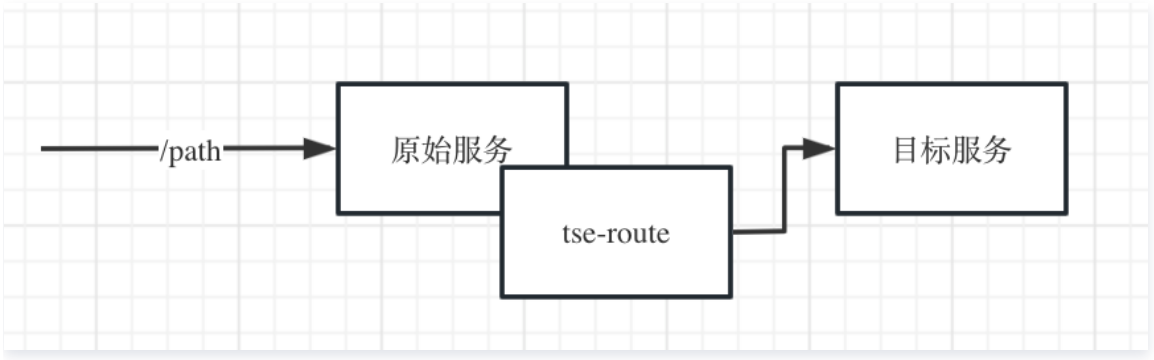
最近更新时间：2025-06-09 16:07:02

TSF 云原生 API 网关支持基于 Method 、特定位置（Header，Query、Body、Cookie）参数、系统参数（domain，clientIP，httpScheme，clientUa）的灰度发布能力。
通过灰度条件和目标服务进行配置。路由规则的多个条件为逻辑与（AND）关系，即一个路由规则内的条件都满足了，才会转发到对应的后端服务。
后端服务通过 Kong Upstream 来定义，绑定容器集群后，Kong Ingress Controller 从 K8S Service 中同步过来自动生成资源。
参数支持表达式运算：

- 支持<，<=, >, >=, ==, != 运算符。
- 支持存在与正则表达式
- 支持包含和不包含

## 实现原理

tse-route 插件为 TSE 提供的用于进行参数路由的插件。通过将 tse-route 插件绑定在原始服务上，通过参数条件路由到目标服务（灰度服务）以进行灰度发布。




tse-route 插件介绍可参考：实现规则路由 。

## 操作步骤

### 步骤1：创建Ingress

前提：已在 k8s 创建好正式服务（nginx）和灰度服务（demo1，demo2）。
创建 Ingress 定义转发规则：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    konghq.com/strip-path: 'true'
  name: demo
  namespace: default
  spec:
  ingressClassName: kong
  rules:
  - http:
      paths:
      - backend:
          service:
            name: nginx
            port:
```

```
            number: 80
      path: /nginx
      pathType: Prefix
    - backend:
        service:
          name: demo1
          port:
            number: 80
      path: /demo1
      pathType: Prefix
    - backend:
        service:
          name: demo2
          port:
            number: 80
      path: /demo2
      pathType: Prefix
```

## 步骤2：定义 TSE Route 插件的 KongPlugin 注解

1. 插件配置字段：

| 配置字段 | 配置说明 | 备注 |
| --- | --- | --- |
| rules-weight | 规则优先级，支持 0-100，数值越大优先级越高。 | / |
| rules-upstream-name | 目标服务名称。 | / |
| rules-upstream-weight | 目标服务流量百分比，默认 100。 | 所有 upstream 的 weight 加起来等于100 |
| rules-condition | 灰度规则的条件说明，支持 header 参数，query 参数，system 参数，cookie 参数，path 和 method。 | / |
| description | 插件说明。 | / |

2. 表达式写法
   - 大于>

```
    - value: '1'
      operator: gt
      key: 'version'
      global_config_id:
      delimiter:
```

   - 小于<

```
    - value: '1'
      operator: lt
      key: 'version'
      global_config_id:
      delimiter:
```

   - 等于==

```
  - value: '1'
    operator: eq
    key: 'version'
    global_config_id:
    delimiter:
```

- 不等于!=

```
  - value: '1'
    operator: ne
    key: 'version'
    global_config_id:
    delimiter:
```

- 大于等于>=

```
  - value: '1'
    operator: ge
    key: 'version'
    global_config_id:
    delimiter:
```

- 小于等于<=

```
  - value: '1'
    operator: le
    key: 'version'
    global_config_id:
    delimiter:
```

- 存在

```
  - operator: exists
    key: 'version'
    global_config_id:
    delimiter:
```

- 正则表达式

```
  - value: '1'
    operator: regex
    key: 'version'
    global_config_id:
    delimiter:
```

**场景一：基于 Header 的灰度发布**

使用 KongPlugin 注解，定义插件配置，condition 配置 header 参数的转发条件。
例如：请求 Header 为 version=v1 时，转发到目标灰度服务（demo1），其他情况访问正式服务（nginx）。

```
apiVersion: configuration.konghq.com/v1
```

```
kind: KongPlugin
metadata:
  name: my-test-route
config:
  description: test
  rules:
  - enabled: true
    condition:
      cookie:
      system:
      path:
      body:
      method:
      header:
      - global_config_id:
        key: version
        value: '1'
        delimiter:
        operator: eq
      query:
    weight: 22
    upstream:
    - weight: 100
      name: demo1.default.80.svc
plugin: tse-route
```

**场景二：基于Query的灰度发布**

**使用 KongPlugin 注解，定义插件配置，condition 配置 query 参数的转发条件。**

**例如：请求 Query 参数 age <3 0 时，转发到目标灰度服务（demo1），其他情况访问正式服务（nginx）。**

```
apiVersion: configuration.konghq.com/v1
kind: KongPlugin
metadata:
  name: my-test-route
config:
  description: test
  rules:
  - enabled: true
    condition:
      cookie:
      system:
      path:
      body:
      method:
      header:
      query:
      - global_config_id:
        key: age
        value: '30'
        delimiter:
        operator: lt
    weight: 22
    upstream:
    - weight: 100
      name: demo1.default.80.svc
```

版权所有：腾讯云计算（北京）有限责任公司

```
plugin: tse-route
```

**场景三：基于权重的灰度发布**

使用 KongPlugin 注解，定义插件配置，配置不同后端服务百分比的转发条件。

例如：配置灰度服务 demo1的百分比为 20%，灰度服务 demo2的百分比为 30%，剩余流量（50%）访问正式服务 nginx。

```
apiVersion: configuration.konghq.com/v1
kind: KongPlugin
metadata:
  name: my-test-route
config:
  description: test
  rules:
  - enabled: true
    condition:
      cookie:
      system:
      path:
      body:
      method:
      header:
      query:
    weight: 10
    upstream:
    - weight: 20
      name: demo1.default.80.svc
    - weight: 30
      name: demo2.default.80.svc
    - weight: 50
      name: nginx.default.80.svc
plugin: tse-route
```

## 步骤3：将插件绑定在服务上

为正式服务 Nginx 添加注解，将插件绑定到该正式服务上。

```
kubectl apply -f tseroute.yaml
kubectl annotate service nginx konghq.com/plugins=my-test-route
```

## 步骤4：访问正式服务

访问正式服务，将根据条件依次匹配，满足匹配条件则按照条件转发到对应的灰度服务。

```
claire@CLAREZZHANG-MB0 ~ % curl -i 'http://1.15.161.241/hello?
uid=/custokjhjhjkkj&whiteList=qatest'
HTTP/1.1 200 OK
Accept-Ranges: bytes
Connection: keep-alive
Content-Length: 615
Content-Type: text/html; charset=UTF-8
Date: Wed, 01 Mar 2023 13:47:53 GMT
Etag: "6398a011-267"
Last-Modified: Tue, 13 Dec 2022 15:53:53 GMT
Server: nginx/1.23.3
```

```
Via: kong/2.5.1
X-Kong-Proxy-Latency: 7
X-Kong-Route-Rule: 100
X-Kong-Route-Upstream: nginx.default.80.svc
X-Kong-Upstream-Latency: 2
```

# 安全防护
# 使用 Web 安全防火墙
# Web 安全防护概述

最近更新时间：2025-06-09 16:07:02

## 什么是 Web 安全防火墙?

Web 应用防火墙（Web Application Firewall，WAF）能够帮助腾讯云内及云外用户应对 Web 攻击、入侵、漏洞利用、挂马、篡改、后门、爬虫等网站及 Web 业务安全防护问题。企业通过部署腾讯云 WAF 服务，将 Web 攻击威胁压力转移到腾讯云 WAF 防护集群节点，分钟级获取腾讯 Web 业务防护能力，为网站及 Web 业务安全运营保驾护航。
WAF 支持两种类型：SaaS 型和负载均衡型，这两种类型的 WAF 都支持接入防护，对比如下：

| WAF 类型 | 负载均衡型 WAF（推荐） | SaaS 型 WAF |
| --- | --- | --- |
| 是否需要网关控制台操作 | 需要。 | 不需要。 |
| 接入方式 | 支持**域名防护**和**对象防护**两种方式，也可同时使用。 | 参考 接入SaaS型WAF 操作指南，源站地址填写一个网关公网负载均衡的公网 IP。 |
| WAF 到网关的流量路径 | 旁路对接。流量先到网关，再到 WAF，旁路送审。 | 流量先到 WAF，再到网关。 |
| WAF 到网关流量类型 | 内网。 | 公网。 |
| 如何选择 | • 旁路对接，业务转发与安全防护分离，最大限度减少安全防护对业务的影响。<br>• 使用内网对接，更省流量。 | • 只能绑定一个公网负载均衡的公网 IP。<br>• 如需使用网页防篡改和数据防泄漏功能，仅 SaaS 型 WAF 可支持。 |

云原生网关推荐使用负载均衡型 WAF，可将网关收到的 HTTP/HTTPS 流量镜像到 WAF 集群，WAF 进行旁路威胁检测和清洗，将用户请求的可信状态同步到 云原生 API 网关进行威胁拦截或放行，以保护您的业务安全。通过旁路对接的方式，具备如下优势：

- 无感知接入，域名接入 WAF 无需调整现有网络架构。
- 网站业务转发和安全防护分离，保障网站业务安全、稳定可靠。

## 负载均衡型 WAF 接入方式

负载均衡型 WAF 提供两种接入方式：

### 方式一：域名接入

通过添加域名的方式与网关资源进行绑定，实现对经过网关的 HTTP 或 HTTPS 流量进行检测和拦截，需要在网关侧进行：

1. 添加接入域名
2. 配置需要防护的资源

### 方式二：对象接入

通过 WAF 资产中心的实例管理模块，与网关实例进行关联，实现对象维度的接入，网关实例的所有流量都进行防护。
云原生对象接入支持基于云原生网关实例对象维度接入防护，将网关实例 ID 的 Web 业务流量镜像，通过协商协议转发到 WAF 集群接入防护。接入后，自动生成基于网关实例 ID 对象的防护集合，对未配置域名接入的 Web 流量启动防护，并允许客户自定义修改防护策略规则。

**两种接入方式的对比如下：**

| 对比项 | 域名接入 | 对象接入 |
| --- | --- | --- |
| 是否需要配置域名 | 需要 | 不需要 |

| | | |
|---|---|---|
| 防护粒度 | 资源级（支持路由级和服务级） | 实例级 |
| 接入复杂度 | 较复杂 | 简单 |
| 网关与 WAF 流量方式 | 旁路对接 | |

**两种相关防护生效顺序如下：**

对象接入以网关实例为维度进行安全防护接入，如您需要更精细的防护，可使用域名接入，通过配置防护域名，精准实现资源防护。您也可以同时配置两种防护策略，相关防护生效顺序如下：

- 精准域名接入防护：优先级最高，命中后安全防护策略优先生效。
- 对象接入防护：没有命中任何域名防护策略后的流量，继续执行对象默认防护策略。

> **说明：**
> - 对象接入：仅企业版及以上版本支持 WAF 对象接入。
> - 域名接入：不同 WAF 版本支持域名数量不同，详细参考 WAF套餐与版本说明 。

## 使用场景

### 场景一：实例全防护

**场景说明**：需要访问该网关实例的所有请求都进行 WAF 防护，存量路由和后续新增路由不用单独配置。适用于安全性要求极高的生产场景。
**使用方式**：开启对象接入防护。

### 场景二：仅部分服务需要防护

**场景说明**：仅访问某些后端服务的请求需要接入防护，其他请求不需要接入防护，适用于测试场景或安全性要求较高的生产场景。
**使用方式**：对需要防护的服务开启域名防护。

### 场景三：仅部分路由需要防护

**场景说明**：仅访问某些路由需要接入防护，其他请求不需要接入防护，适用于测试场景或安全性要求较低的生产场景。
**使用方式**：对需要防护的路由开启域名防护。

### 场景四：开启了服务级防护，需要关闭服务下部分路由的防护

**场景说明**：已开启服务级防护，新增了测试路由需要取消防护进行测试。
**使用方式**：对需要测试的路由关闭防护，此时服务防护将由全部开启变为部分开启状态，待测试完成后，可选择性开启路由防护，此时服务防护会再次变为全部开启。

# 域名防护

最近更新时间：2025-06-09 16:07:02

本文介绍云原生 API 网关如何结合 WAF 通过添加域名的方式与云原生 API 网关的资源（包括服务、路由）进行绑定，实现对经过云原生 API 的流量进行检测和拦截。
域名防护提供服务级、路由级别的防护，分别适用于不同粒度的防护，您可根据网关业务特性和 API 使用情况分别配置，在节约成本的同时保障 API 安全。对应的防护的区别为：

| 防护类型 | 防护说明 | 状态转换 |
| --- | --- | --- |
| 服务级防护 | 访问该服务下所有路由的请求都将接入 WAF 默认规则的防护引擎，如您需要针对具体服务开启WAF防护，可使用该粒度防护。 | 默认关闭。<br>• 服务下全部路由开启：自动变为全部开启<br>• 服务下部分路由关闭：自动变为部分开启 |
| 路由级防护 | 仅访问该路由的请求都将接入 WAF 默认规则的防护引擎，如您只需要对具体路由开启WAF防护，可使用该粒度防护。 | 默认关闭。<br>• 服务防护开启：自动变为全部开启 |

**说明：**
不同 WAF 版本支持域名数量不同，详细参考 WAF套餐与版本说明 。

## 前提条件

在开启 WAF 前，确保您的 tse-global-configuration 插件和 tse-cloud-waf 插件已升级到最新版本。
1. 登录 TSF 云原生API网关控制台 。
2. 在左侧导航栏单击**云原生 API 网关** > **插件管理**，在页面上方选择好网关实例。
3. 选择系统插件页签，查看插件是否已更新到最新版本，若未更新，单击操作列的安装最新版本即可。

## 操作步骤

### 步骤 1：配置 Web 应用防火墙

1. 登录 Web 应用防火墙控制台 ，在左侧导航栏中，选择**资产中心** > **接入管理** > **域名接入**。
2. 在域名接入页面，单击**添加域名**，填写相关配置参数，单击确定即可。




| 参数 | 说明 |
| --- | --- |
| 所属实例 | 负载均衡型。 |
| 域名 | 在域名输入框中添加需要防护的域名。 |

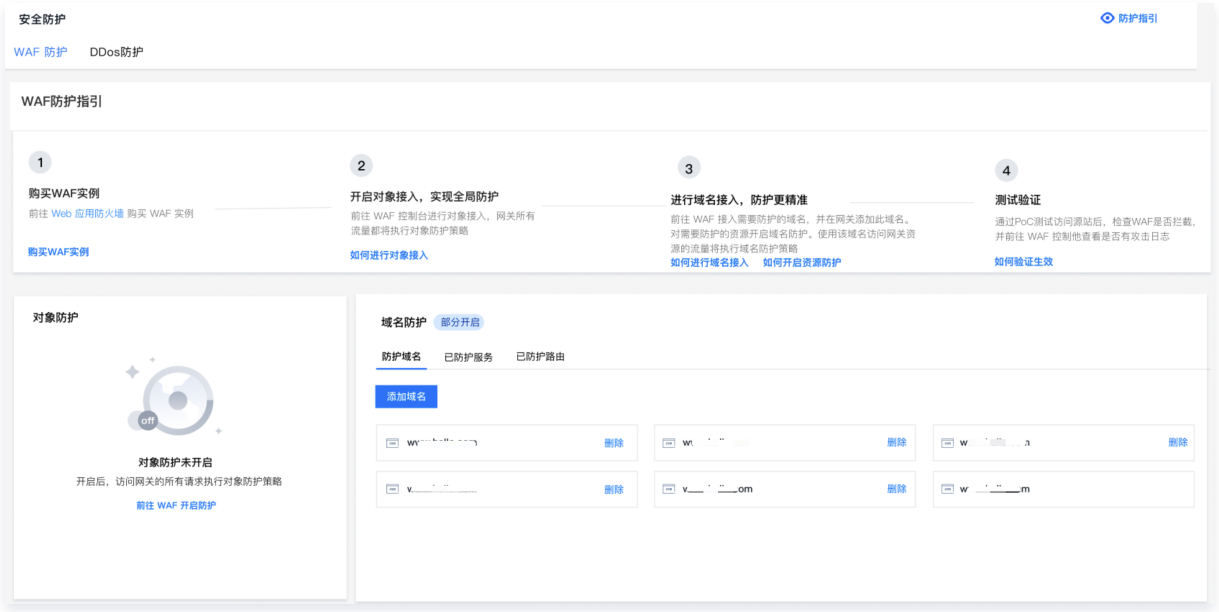
| | |
|---|---|
| 流量来源 | 选择云原生 API 网关。 |
| 代理情况 | 选择“是”，WAF 将通过 XFF 字段获取客户真实 IP 地址作为源地址，勾选可能存在源 IP 被伪造的风险。 |
| 国内地域 | 选择需要防护的地域。 |

3. 单击**确定**，即可返回域名接入。在域名接入可以查看到防护域名和网关实例 ID、名称等信息。

## 步骤 2：添加防护域名

1. 登录 云原生 API 网关控制台，在左侧导航栏选择**安全防护** > **WAF 防护**，在防护域名 Tab，单击添加域名，选择或填写 步骤 1 中已经接入 WAF 的域名，支持选择在证书管理中已添加证书对应的域名，也支持自行输入域名进行添加。
2. 单击**确定**，确认防护域名已添加完成。

> **注意：**
> 您需要确保防护域名已完成 WAF 接入，否则将无法将对应域名的请求进行送审。




## 步骤 3：开启资源防护

云原生 API 网关支持服务级和路由级粒度的防护，

### 开启服务级防护

1. 进入防护服务 Tab，默认展示当前实例下所有服务。
2. 选择需要进行 WAF 接入的服务，单击**开启防护**，在开启 WAF 防护的弹窗中，单击确定，开启该服务的 WAF 防护。

> **说明：**
> - 服务级防护开启后，该服务以及服务下所有的路由都对应开启防护。
> - 如将服务下部分路由防护关闭，服务防护状态和实例防护状态都将变为部分开启。

### 开启路由级防护

1. 进入防护路由 Tab，默认展示当前实例下所有路由。
2. 选择需要进行 WAF 接入的路由，单击**开启防护**，在开启 WAF 防护的弹窗中，单击**确定**，开启该路由的 WAF 防护。

### 关闭服务级防护

1. 进入防护服务 Tab，默认展示当前实例下所有服务。
2. 选择需要关闭 WAF 接入的服务，单击**关闭防护**，在关闭 WAF 防护的弹窗中，单击确定，关闭该服务的 WAF 防护。

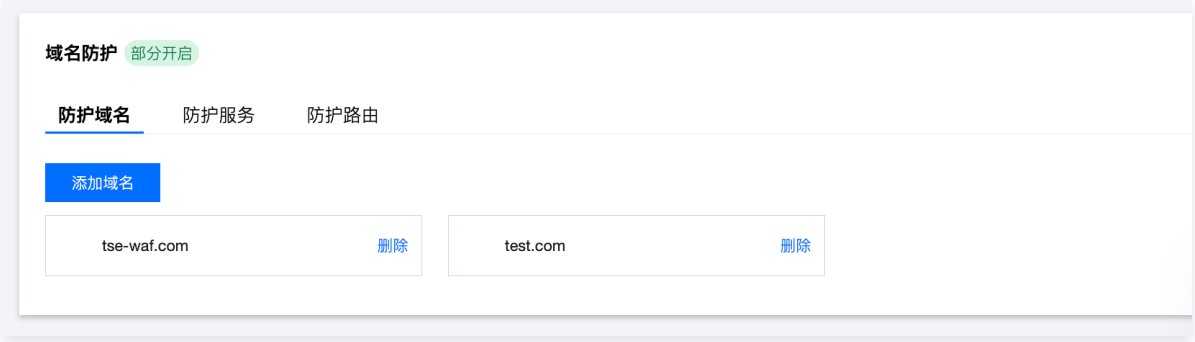
> **注意：**
> 服务级防护关闭后，该服务以及服务下所有的路由都对应关闭防护。

### 关闭路由级防护

1. 进入防护路由 Tab，默认展示当前实例下所有路由。
2. 选择需要进行 WAF 接入的路由，单击**关闭防护**，在关闭 WAF 防护的弹窗中，单击确定，关闭该路由的 WAF 防护。

## 步骤4: 测试验证

1. 登录 云原生 API 网关 控制台，在左侧导航栏中，选择安全防护。
2. 确认域名防护状态为部分开启或全部开启。




3. 在浏览器中输入网址 `http://<网关域名或IP>/?test=alert(123)` 并访问，浏览器返回阻断页面，说明 Web 应用防火墙防护功能正常。



# 对象防护

最近更新时间：2025-06-09 16:07:02

本文介绍云原生 API 网关如何结合 WAF 通过对象防护的方式实现网关安全防护。对象防护开启后，所有访问网关的请求都默认进行防护。

> **说明：**
> 仅企业版及以上版本支持 WAF 对象接入。

## 前提条件

在开启 WAF 前，确保您的 tse-global-configuration 插件和 tse-cloud-waf 插件已升级到最新版本。

1. 登录 云原生 API 网关控制台 。
2. 在左侧导航栏单击**云原生 API 网关** > **插件管理**，在页面上方选择好网关实例。
3. 选择系统插件页签，查看插件是否已更新到最新版本，若未更新，单击操作列的安装最新版本即可。

## 操作步骤

1. 登录 Web 应用防火墙控制台，在左侧导航栏中，选择**资产中心** > **接入管理**。
2. 在对象接入列表，选择需要进行对象接入的云原生 API 网关实例，开启 WAF 开关。




3. 登录云原生 API 网关控制台，在左侧导航中选择**安全防护**，进入 WAF 防护页签，确认对象防护已开启。




4. 在浏览器中输入网址 `http://<网关域名或IP>/?test=alert(123)` 并访问，浏览器返回阻断页面，说明 Web 应用防火墙防护功能正常。



# 使用 DDos 防护

最近更新时间：2025-06-09 16:07:02

## 操作背景

DDoS 防护（Anti-DDoS）具有全面、高效、专业的 DDoS 防护能力，为企业组织提供 DDoS 高防包、DDoS 高防 IP 等多种 DDoS 解决方案，应对 DDoS 攻击问题。通过充足、优质的 DDoS 防护资源，结合持续进化的“自研+AI 智能识别”清洗算法，保障用户业务的稳定、安全运行。
云原生 API 网关公网负载均衡默认提供基础 DDos 防护服务，如需更高级别防护，可购买 DDos 高防产品，并将其绑定云原生 API 网关，即可享受更安全的服务。

## 操作步骤

### 步骤 1：购买 DDos 防护实例

网关默认为所有已添加的公网负载均衡提供最高不超过 2Gbps 的基础 DDos 防护，如需更高级的使用 DDos 防护，可购买 DDos 高防包或高防 IP。DDos 根据不同场景提供了多种防护解决方案，请参考 DDos 防护解决方案对比 进行选择。

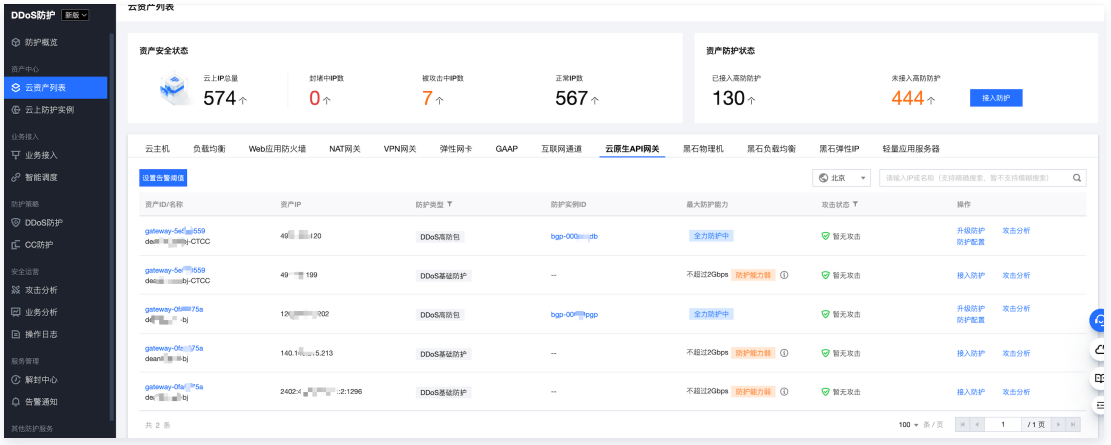
### 步骤 2：防护实例绑定网关资源

#### 场景一：使用高防 IP

可参考 DDos 高防 IP 操作指南 进行业务接入。

#### 场景二：使用高防包

1. 进入 DDos 防护控制台，点击云资产列表， 选择**云原生API网关**，确认地域。
2. 在资产 IP 中，选择需要进行 DDos 防护的公网负载均衡IP，点击右侧的**接入防护**。
3. 在接入防护对话框中，选择需要绑定的 DDos 高防包，点击确定。
4. 确认在资产列表中，防护类型已经更新为 DDos 高防包，最大防护能力为全力防护中。




5. 在 云原生 API 网关控制台，点击实例 ID，进入实例详情页，选择安全防护，进入DDos 防护页签，确认当前高防包关联资源。



### 步骤 3：查看/配置防护策略

可参考 DDos 防护配置 进行防护策略查看和配置。

# 结合 WAF 和 DDos实现安全防护

最近更新时间：2025-06-09 16:07:02

## 应用场景

云原生 API 网关可结合 Web 应用防火墙与 DDoS 高防包联动，为用户提供全方位的安全防护。

- Web 应用防火墙提供实时防护能力，可有效拦截 Web 攻击，保障用户业务的数据和信息安全。
- DDoS 高防包可以提供上百 Gbps 的 DDoS 防护能力，轻松应对 DDoS 攻击，保障业务稳定运行。

本文介绍云原生 API 网关如何结合 WAF 和 DDoS 高防包，实现网关安全防护。

## 操作步骤

> **说明：**
> 步骤1和步骤2没有严格的先后次序，您可以按照您的使用习惯完成配置。

### 步骤1：配置 Web 应用防火墙

云原生 API 网关支持域名接入和对象接入两种方式。对象接入以网关实例为维度进行安全防护接入，如您需要更精细的防护，可使用域名接入，通过配置防护域名，精准实现资源防护。您也可以同时配置两种防护策略，相关防护生效顺序如下：

- 精准域名接入防护：优先级最高，命中后安全防护策略优先生效。
- 对象接入防护：没有命中任何域名防护策略后的流量，继续执行对象默认防护策略。

> **说明：**
> - 对象接入：仅企业版及以上版本支持 WAF 对象接入。
> - 域名接入：不同 WAF 版本支持域名数量不同，详细参见 WAF 套餐与版本说明 。

#### 对象接入

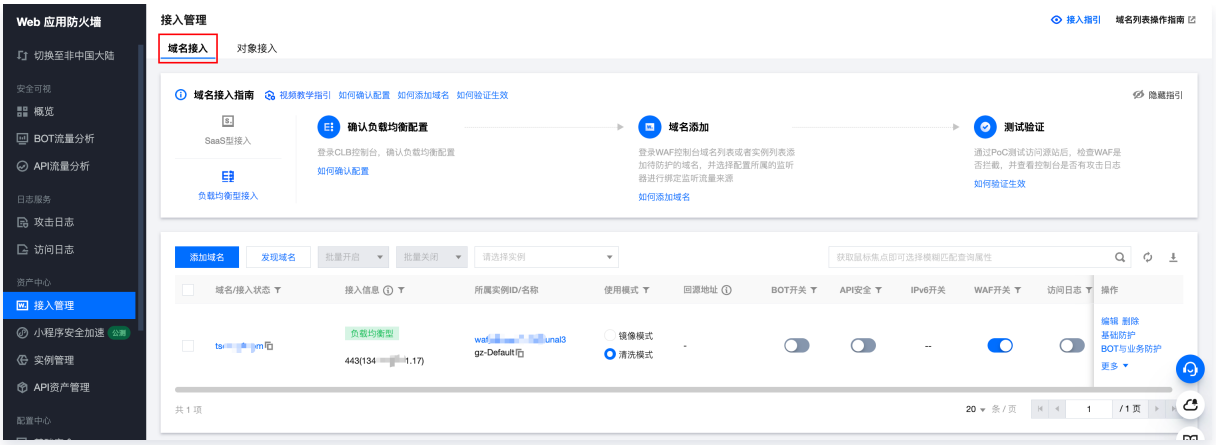
1. 登录 Web 应用防火墙控制台，在左侧导航中，选择**接入管理**。




2. 在**对象接入**页签，选择需要进行对象接入的云原生 API 网关实例，开启 WAF 开关。

#### 域名接入

1. 登录 Web 应用防火墙控制台，在左侧导航中，选择**接入管理**。

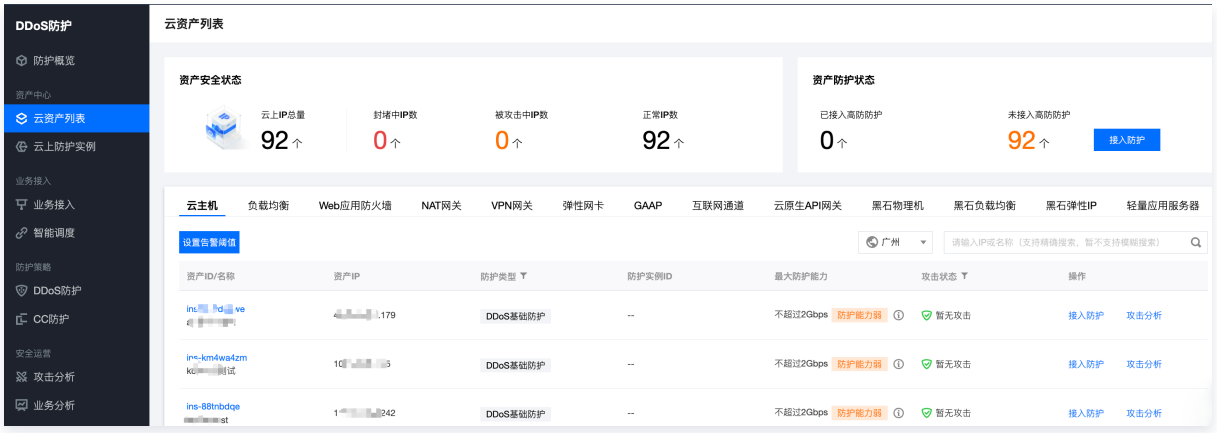



2. 在**域名接入**页签，单击**添加域名**，填写相关参数，单击**确定**。
   - 所属实例：选择负载均衡型。
   - 域名：输入需要防护的域名。
   - 流量来源：选择云原生 API 网关。
   - 代理情况：选择“是”，WAF 将通过 XFF 字段获取客户真实 IP 地址作为源地址，勾选可能存在源 IP 被伪造的风险。
   - 国内地域：选择需要防护的地域。




## 步骤2：配置 DDos 高防包

1. 登录 DDoS 防护控制台，在左侧导航栏中，选择**云资产列表**。
2. 在云资产列表页面，选择需要接入防护的云原生 API 网关实例，单击操作列的**接入防护**。




3. 在接入防护对话框中，选择需要绑定的 DDos 高防包，点击**确定**。
4. 确认在资产列表中，防护类型已经更新为 DDos 高防包，最大防护能力为全力防护中。

## 步骤3：配置云原生 API 网关

> **说明：**
> 如您使用对象接入，此步骤可忽略。如您使用域名接入，请参考如下步骤添加域名。

1. 登录 云原生 API 网关控制台，在左侧导航栏中，选择**安全防护**。
2. 进入 **WAF 防护**页签，在域名防护模块，选择**添加域名**，填写需要进行防护的域名。




3. 选择需要防护的服务或路由开启防护。




## 步骤4：测试验证

1. 登录 云原生 API 网关 控制台，在左侧导航中，选择安全防护。
2. 进入 DDos 防护 Tab，确认当前网关已关联防护实例。




3. 进入 WAF 防护 Tab，确认防护状态已开启。

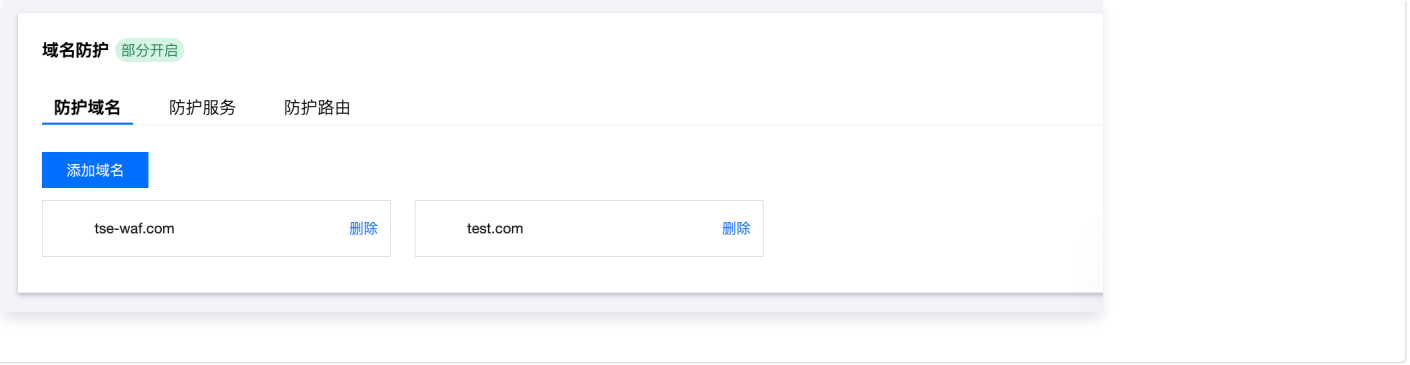
**对象接入**

在对象防护模块，确认对象防护已开启。




**域名接入**

确认域名防护状态为部分开启或全部开启。




4. 在浏览器中输入网址 `http://<网关域名或IP>/?test=alert(123)` 并访问，浏览器返回阻断页面，说明 Web 应用防火墙防护功能正常。



# 监控告警
# 查看默认监控

最近更新时间：2025-06-09 16:07:02

## 操作场景

云原生 API 网关 Kong 对运行的网关实例提供了多项监控指标，用以监测实例运行情况，例如：4xx请求数、5xx请求数等业务指标，CPU 利用率、内存使用率等系统指标。
您可以根据这些指标实时了解网关实例的运行状况，针对可能存在的风险及时处理，保障系统的稳定运行。本文为您介绍通过 TSF 控制台查看网关实例监控的操作。

## 支持监控指标及含义

### 请求监控

**实例/节点**

| 指标名 | 指标含义 |
| --- | --- |
| 总请求数(Count) | 总请求数。按照所选择的时间粒度统计求和。 |
| 请求平均时延(ms) | 请求平均时延，按照所选择的时间粒度统计求平均值。 |
| 请求最大时延(ms) | 请求最大时延，按照所选择的时间粒度统计求最大值。 |
| 成功请求数(Count) | 云原生 API 网关返回为1xx,2xx,3xx的请求数量，按照所选择的时间粒度统计求和。 |
| 4xx请求数(Count) | 客户端发送到云原生 API 网关的是请求非法请求，如鉴权不通过或者超过限流值的错误个数，按照所选择的时间粒度统计求和。 |
| 5xx请求数(Count) | 云原生 API 网关将消息转发到后端服务，后端服务返回大于等于400错误状态码的个数，按照所选择的时间粒度统计求和。 |
| 404请求数(Count) | 请求后端服务失败，请求所希望的资源未被在后端服务器上发现，此类错误的个数的统计，按照所选择的时间粒度统计求和。 |
| 429请求数(Count) | 请求后端服务失败，请求被限流，此类错误的个数的统计，按照所选择的时间粒度统计求和。 |
| 502请求数(Count) | API 网关尝试执行后端请求时，从后端服务器接收到无效的响应（通常连接服务失败），此类错误的个数的统计，按照所选择的时间粒度统计求和。 |
| 504请求数(Count) | API 网关尝试执行后端请求时，后端机器不可达，此类错误的个数的统计，按照所选择的时间粒度统计求和。 |

**路由**

| 指标名 | 指标含义 |
| --- | --- |
| 总请求数(Count) | 总请求数。按照所选择的时间粒度统计求和。 |
| 请求平均时延(ms) | 请求平均时延，按照所选择的时间粒度统计求平均值。 |
| 请求最大时延(ms) | 请求最大时延，按照所选择的时间粒度统计求最大值。 |
| 网关直接返回的请求数(次) | 网关直接返回的请求数，按照所选择的时间粒度统计求和。 |
| 网关最大时延(ms) | 网关最大时延，按照所选择的时间粒度统计求最大值。 |

版权所有：腾讯云计算（北京）有限责任公司

| | |
|---|---|
| 2xx请求数(次) | 返回为2xx 的请求数量，按照所选择的时间粒度统计求和。 |
| 3xx请求数(次) | 返回为3xx 的请求数量，按照所选择的时间粒度统计求和。 |
| 4xx请求数(Count) | 客户端发送到云原生 API 网关的是请求非法请求，如鉴权不通过或者超过限流值的错误个数，按照所选择的时间粒度统计求和。 |
| 5xx请求数(Count) | 云原生 API 网关将消息转发到后端服务，后端服务返回大于等于400错误状态码的个数，按照所选择的时间粒度统计求和。 |
| 404请求数(Count) | 请求后端服务失败，请求所希望的资源未被在后端服务器上发现，此类错误的个数的统计，按照所选择的时间粒度统计求和。 |
| 429请求数(Count) | 请求后端服务失败，请求被限流，此类错误的个数的统计，按照所选择的时间粒度统计求和。 |
| 499请求数(Count) | 云原生 API 网关将消息转发到后端服务，后端服务返回499的请求个数，按照所选择的时间粒度统计求和。 |
| 502请求数(Count) | API 网关尝试执行后端请求时，从后端服务器接收到无效的响应（通常连接服务失败），此类错误的个数的统计，按照所选择的时间粒度统计求和。 |
| 504请求数(Count) | API 网关尝试执行后端请求时，后端机器不可达，此类错误的个数的统计，按照所选择的时间粒度统计求和。 |
| 网关转到后端的请求数(次) | 网关转到后端的请求数，按照所选择的时间粒度统计求和。 |
| 后端最大时延(ms) | 后端最大时延，按照所选择的时间粒度统计求最大值。 |
| 后端2xx请求数(次) | 后端服务返回 2xx 的请求数量，按照所选择的时间粒度统计求和。 |
| 后端3xx请求数(次) | 后端服务返回 3xx 的请求数量，按照所选择的时间粒度统计求和。 |
| 后端4xx请求数(次) | 后端服务返回 4xx 的请求数量，按照所选择的时间粒度统计求和。 |
| 后端5xx请求数(次) | 后端服务返回 5xx 的请求数量，按照所选择的时间粒度统计求和。 |
| 后端404请求数(次) | 后端服务返回 404 的请求数量，按照所选择的时间粒度统计求和。 |
| 后端429请求数(次) | 后端服务返回 429 的请求数量，按照所选择的时间粒度统计求和。 |
| 后端499请求数(次) | 后端服务返回 499 的请求数量，按照所选择的时间粒度统计求和。 |
| 后端502请求数(次) | 后端服务返回 502 的请求数量，按照所选择的时间粒度统计求和。 |
| 后端504请求数(次) | 后端服务返回 504 的请求数量，按照所选择的时间粒度统计求和。 |

## 系统监控

### 实例/节点监控指标

| 指标名 | 指标含义 |
|---|---|
| CPU 利用率（%） | 云原生 API 网关集群的 CPU 使用率，按照所选择的时间粒度统计求平均值。 |
| 内存使用率（%） | 云原生 API 网关集群的内存使用率，按照所选择的时间粒度统计求平均值。 |
| 入带宽流量（Mbps） | 云原生 API 网关集群的入口带宽流量，按照所选择的时间粒度统计求平均值。 |
| 出带宽流量（Mbps） | 云原生 API 网关集群的出口带宽流量，按照所选择的时间粒度统计求平均值。 |
| TCP 入连接数（个） | 云原生 API 网关集群的 TCP 连接数，按照所选择的时间粒度统计求平均值。 |

版权所有：腾讯云计算（北京）有限责任公司

**负载均衡监控指标**

| 指标名 | 指标含义 |
| --- | --- |
| 客户端到 LB 的活跃连接数(Count) | 在统计粒度内，从客户端到负载均衡的活跃连接数。 |
| 客户端到 LB 的非活跃连接数(Count) | 在统计粒度内，从客户端到负载均衡的非活跃连接数。 |
| 客户端到 LB 的并发连接数(Count) | 在统计粒度内，从客户端到负载均衡的并发连接数。 |
| 客户端到 LB 的新建连接数(Count) | 在统计粒度内，从客户端到负载均衡的新建连接数。 |
| 新建连接数利用率（%） | 在统计粒度内，从客户端到负载均衡的新建连接数相比负载均衡规格的新建连接数性能上限的利用率。 |
| 入流量(MiBytes) | 在统计粒度内，客户端流入到负载均衡的流量。 |
| 出流量(MiBytes) | 在统计粒度内，负载均衡流出到客户端的流量。 |
| 入包量(Count/s) | 在统计粒度内，客户端向负载均衡每秒发送的数据包数量。 |
| 出包量(Count/s) | 在统计粒度内，负载均衡向客户端每秒发送的数据包数量。 |
| 入带宽(Mbps) | 在统计粒度内，客户端流入到负载均衡所用的带宽。 |
| 出带宽(Mbps) | 在统计粒度内，负载均衡流出到客户端所用的带宽。 |
| 入带宽利用率(%) | 在统计粒度内，客户端通过外网访问负载均衡所用的带宽利用率。 |
| 出带宽利用率(%) | 在统计粒度内，负载均衡访问外网所用的带宽使用率。 |
| 负载均衡到网关节点的入带宽(Mbps) | 在统计粒度内，负载均衡流入到网关节点所用的带宽。 |
| 负载均衡到网关节点的出流量（MB） | 在统计粒度内，网关节点流出到负载均衡所用的流量。 |
| 丢弃连接数(Count) | 在统计粒度内，负载均衡丢弃的连接数。 |
| 丢弃入带宽(bps) | 在统计粒度内，客户端通过外网访问负载均衡时丢弃的带宽。 |
| 丢弃出带宽(bps) | 在统计粒度内，负载均衡访问外网时丢弃的带宽。 |
| 丢弃流入数据包(Count/s) | 在统计粒度内，客户端通过外网访问负载均衡时丢弃的数据包。 |
| 丢弃流出数据包(Count/s) | 在统计粒度内，负载均衡访问外网时丢弃的数据包。 |
| 丢弃 QPS(Count) | 在统计粒度内，负载均衡丢弃的请求数。 |
| 平均请求时间(ms) | 在统计粒度内，负载均衡的平均请求时间。 |
| 最大请求时间(ms) | 在统计粒度内，负载均衡的最大请求时间。 |
| 平均响应时间(ms) | 在统计粒度内，负载均衡的平均响应时间。 |
| 最大响应时间(ms) | 在统计粒度内，负载均衡的最大响应时间。 |
| 响应超时个数(Count/min) | 在统计粒度内，负载均衡响应超时的个数。 |

版权所有：腾讯云计算（北京）有限责任公司

| | |
|---|---|
| 每分钟成功请求数(Count/min) | 在统计粒度内，负载均衡每分钟的成功请求数。 |
| 每秒请求数(Count/s) | 在统计粒度内，负载均衡每秒钟的请求数。 |
| QPS 利用率(%) | 在统计粒度内，负载均衡的 QPS 相比负载均衡规格的 QPS 性能上限的利用率。 |
| CLB 返回的 3xx 状态码(Count/min) | 在统计粒度内，负载均衡返回 3xx 状态码的个数（负载均衡和网关节点返回码之和）。 |
| CLB 返回的 4xx 状态码(Count/min) | 在统计粒度内，负载均衡返回 4xx 状态码的个数（负载均衡和网关节点返回码之和）。 |
| CLB 返回的 5xx 状态码(Count/min) | 在统计粒度内，负载均衡返回 5xx 状态码的个数（负载均衡和网关节点返回码之和）。 |
| CLB 返回的 404 状态码(Count/min) | 在统计粒度内，负载均衡返回 404 状态码的个数（负载均衡和网关节点返回码之和）。 |
| CLB 返回的 499 状态码(Count/min) | 在统计粒度内，负载均衡返回 499 状态码的个数（负载均衡和网关节点返回码之和）。 |
| CLB 返回的 502 状态码(Count/min) | 在统计粒度内，负载均衡返回 502 状态码的个数（负载均衡和网关节点返回码之和）。 |
| CLB 返回的 503 状态码(Count/min) | 在统计粒度内，负载均衡返回 503 状态码的个数（负载均衡和网关节点返回码之和）。 |
| CLB 返回的 504 状态码(Count/min) | 在统计粒度内，负载均衡返回 504 状态码的个数（负载均衡和网关节点返回码之和）。 |
| 2xx 状态码(Count/min) | 在统计粒度内，网关节点返回 2xx 状态码的个数。 |
| 3xx 状态码(Count/min) | 在统计粒度内，网关节点返回 3xx 状态码的个数。 |
| 4xx 状态码(Count/min) | 在统计粒度内，网关节点返回 4xx 状态码的个数。 |
| 5xx 状态码(Count/min) | 在统计粒度内，网关节点返回 5xx 状态码的个数。 |
| 404 状态码(Count/min) | 在统计粒度内，网关节点返回 404 状态码的个数。 |
| 499 状态码(Count/min) | 在统计粒度内，网关节点返回 499 状态码的个数。 |
| 502 状态码(Count/min) | 在统计粒度内，网关节点返回 502 状态码的个数。 |
| 503 状态码(Count/min) | 在统计粒度内，网关节点返回 503 状态码的个数。 |
| 504 状态码(Count/min) | 在统计粒度内，网关节点返回 504 状态码的个数。 |

## 业务监控

### 实例

| 指标名 | 指标含义 |
|---|---|
| 服务来源健康检查失败总请求数(Count) | 云原生 API 网关集群服务来源健康检查失败总请求数。 |
| 服务来源健康检查成功总请求数(Count) | 云原生 API 网关集群服务来源健康检查成功总请求数。 |
| 服务来源健康检查总请求数(Count) | 云原生 API 网关集群服务来源健康检查总请求数。 |

### 服务来源

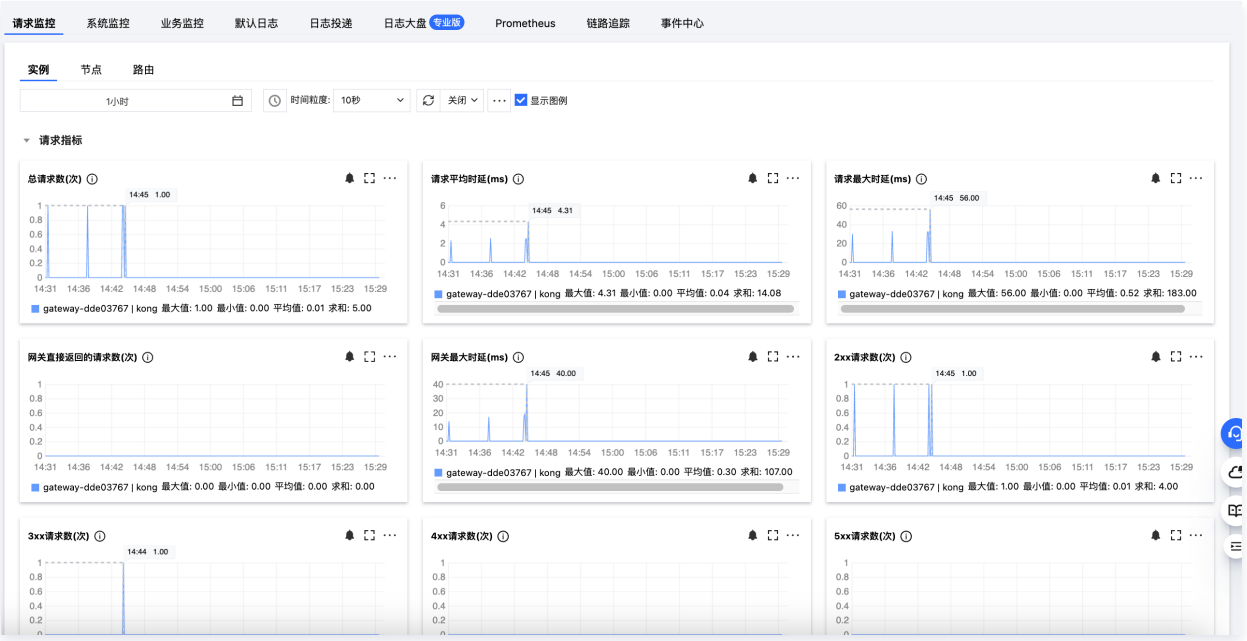
| 指标名 | 指标含义 |
| --- | --- |
| 云原生网关注册中心探测失败请求数(Count) | 云原生网关注册中心探测失败请求数 |
| 云原生网关注册中心探测请求成功数(Count) | 云原生网关注册中心探测请求成功数 |
| 云原生网关注册中心探测总请求数(Count) | 云原生网关注册中心探测总请求数 |

**服务**

| 指标名 | 指标含义 |
| --- | --- |
| 云原生网关服务上游节点总数(Count) | 云原生网关服务上游节点总数 |
| 云原生网关服务上游健康节点总数(Count) | 云原生网关服务上游健康节点总数 |
| 网关服务上游异常节点数(Count) | 网关服务上游异常节点数 |
| 云原生网关服务上游异常节点百分比(%) | 云原生网关服务上游异常节点百分比 |

## 查看监控入口

1. 登录 TSF 控制台 。
2. 在左侧导航栏单击**云原生 API 网关 > 实例详情页 > 数据观测**，在页面上方选择好网关实例。
3. 设置要查看的视图和时间范围，即可查看以下监控指标信息。



# 使用 Prometheus 监控

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何使用 Prometheus 获取云原生 API 网关 的监控数据。TSF 支持关联腾讯云 Prometheus 和配置自建 Prometheus 两种方式。

- 关联腾讯云 Prometheus：关联 Prometheus 实例后，网关 将自动上报监控数据到已关联的 Prometheus。
- 配置自建 Prometheus：将 网关 节点信息配置在您的 Prometheus Server 上，进行数据采集。

## 前置条件

- 已购买 云原生 API 网关实例，详情请参见 操作文档 。
- 已拥有腾讯云 Prometheus 实例 或自建 Prometheus 实例。

## 方式1：关联腾讯云 Prometheus

1. 登录 微服务平台 TSF 控制台 。
2. 在左侧导航栏单击**云原生 API 网关** > **数据观测**，在页面上方选择好网关实例。
3. 单击 **Prometheus** 页签，在**关联腾讯云 Prometheus** 模块点击编辑按钮。
4. 选择您要关联的腾讯云 Prometheus 实例（仅支持关联与实例在同一 VPC 下的 Prometheus 实例）。
5. 单击**确认**，完成关联。

## 方式2：配置自建 Prometheus

### 步骤1：开启 Prometheus 插件

使用 Prometheus 监控，需要在 云原生 API 网关开启普罗米修斯（Prometheus）插件，您可以根据需要选择以下两种特性场景：

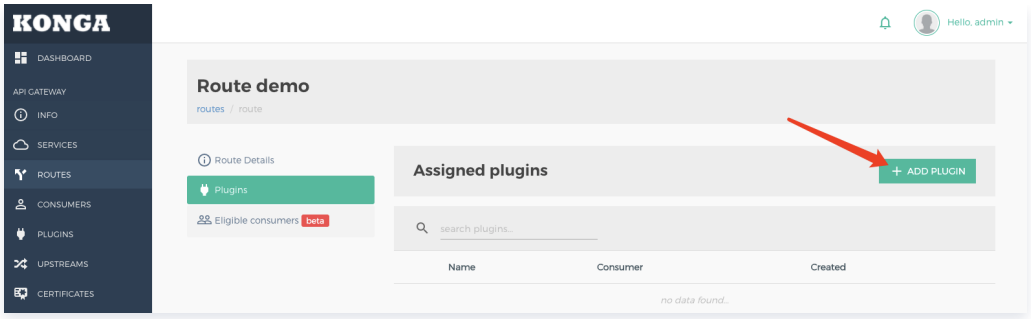
- 针对指定 API 开启 Prometheus 插件
- 开启全局 Prometheus 插件

#### 场景1： 针对指定 API 开启 Prometheus 插件

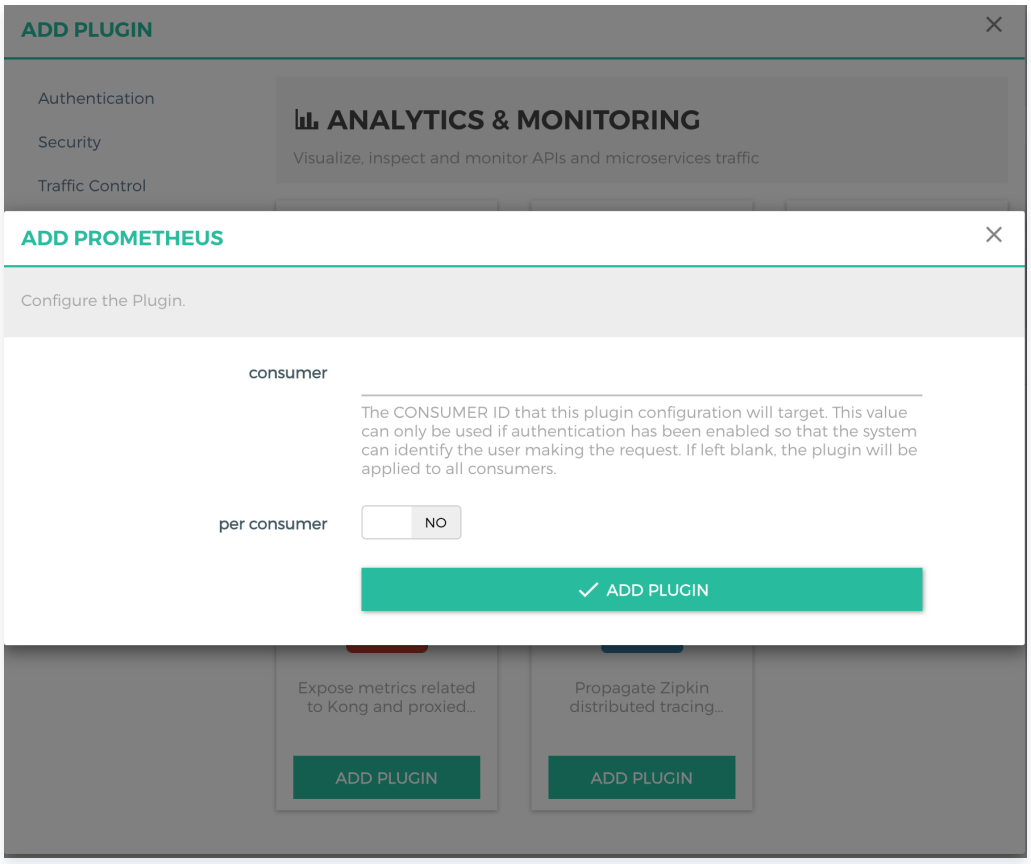
1. 登录 TSF 控制台，进入需要配置 Prometheus 插件的网关实例详情页，在 **Konga 控制台**页面查看管理控制台登录方式。




2. 登录 Konga 管理控制台，进入需要添加 Prometheus 插件的 **Route 详情页**，单击 **Add Plugin** 按钮添加插件，在 **Analytics & Monitoring** 分组下选择 **Prometheus** 插件。

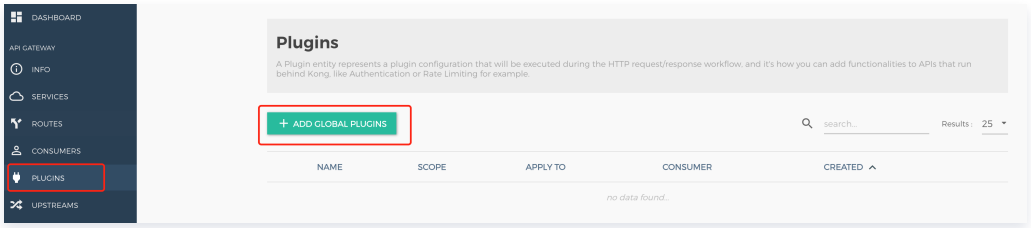



3. 添加 Prometheus 插件配置中，若您不需要区分 consumer ，请直接用默认配置。




## 场景2：开启全局 Prometheus 插件

1. 进入左侧导航栏的 PLUGINS 页面，单击 ADD GLOBAL PLUGINS 按钮，在 Analytics & Monitoring 分组下选择 Prometheus 插件。




2. 登录 TSF 控制台，进入需要配置限流插件的网关实例详情页，在**配置管理**页查看管理控制台登录方式。

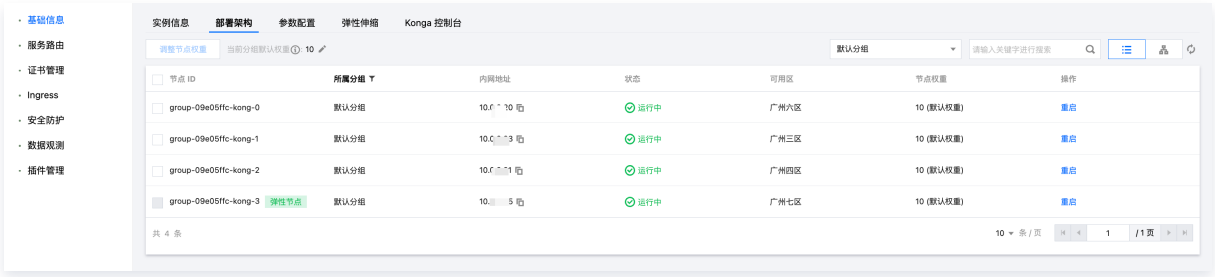
## 步骤2：配置数据拉取地址

1. 准备好**腾讯云**或**自建**的 Prometheus 实例。该步骤的 Prometheus 实例用于拉取云原生 API 网关 kong 的监控数据。

> **说明：**

> 关于如何购买腾讯云 Prometheus 实例，以及开通配套的 Grafana 实例，可以参见 Prometheus 监控服务 相关文档。

2. 登录 TSF 控制台，在**实例详情页 > 基础信息 > 部署架构** 页面查看 网关 实例的内网节点地址。




3. 若您使用 Prometheus agent，可以在 prometheus.yml 配置文件中的 scrape_configs.static_configs.targets 添加内网节点地址列表，注意端口需配置为 2100（2023年4月18日前创建的实例请配置端口为8100）。例如对于上图的内网地址列表，可以参照以下参数进行配置：

```
job_name: kong
honor_timestamps: true
metrics_path: /metrics
scheme: http
static_configs:
- targets:
  - 172.27.0.35:2100
  - 172.27.0.133:2100
  - 172.27.0.79:2100
  labels:
    instance: gateway-f3bdaaa7
relabel_configs:
  - source_labels: [__address__]
    target_label: node
```

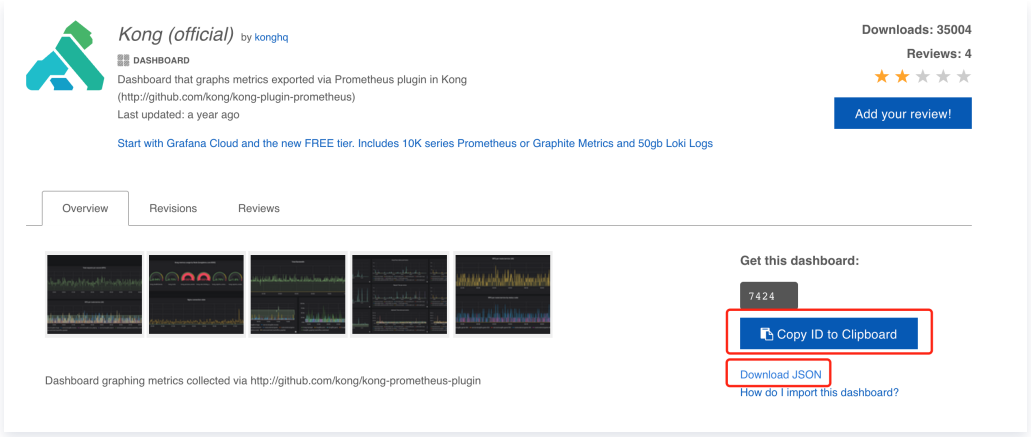
> **注意：**
> 1. 请配置 relabel_configs 信息，以保证指标上报准确。
> 2. 请确保 Prometheus Agent 与网关实例在相同 VPC，以保证网络连通性。
> 3. 请确保 Prometheus Agent Egress安全组放通 2100 端口（2023年4月18日前创建的实例请放通端口8100），以允许网关进行数据采集。

4. 若您使用自定义程序的拉取方式，可以通过云原生 API 网关 DescribeCloudNativeAPINodes 接口自动获取节点 IP 列表。

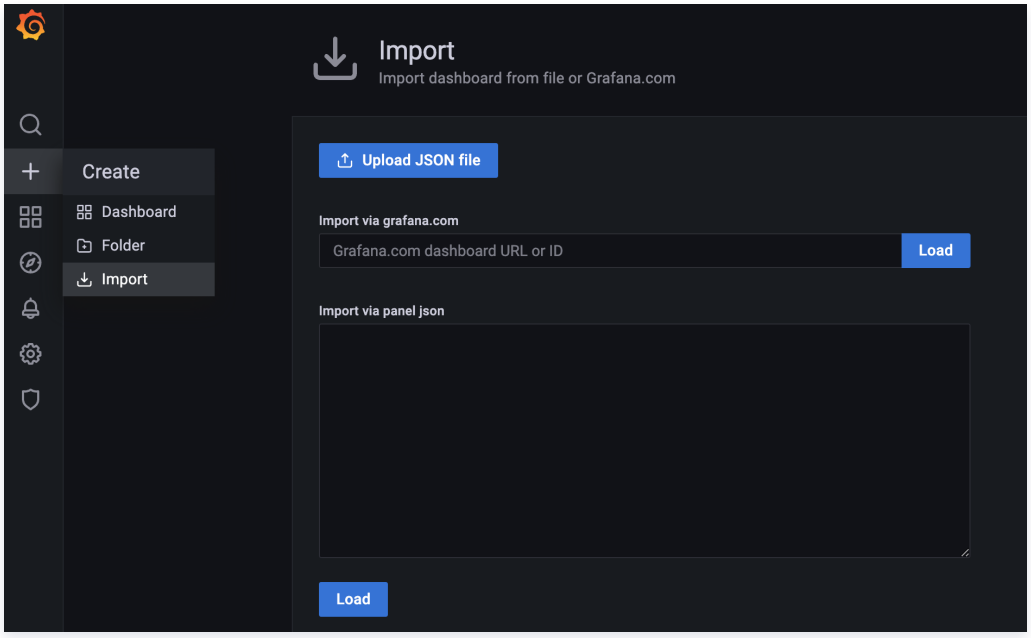
## 步骤3：在 Grafana 配置 网关 模板的 Dashboard（可选）

该步骤指导您在 Grafana 实例展示 Prometheus 采集到的 网关 监控数据。

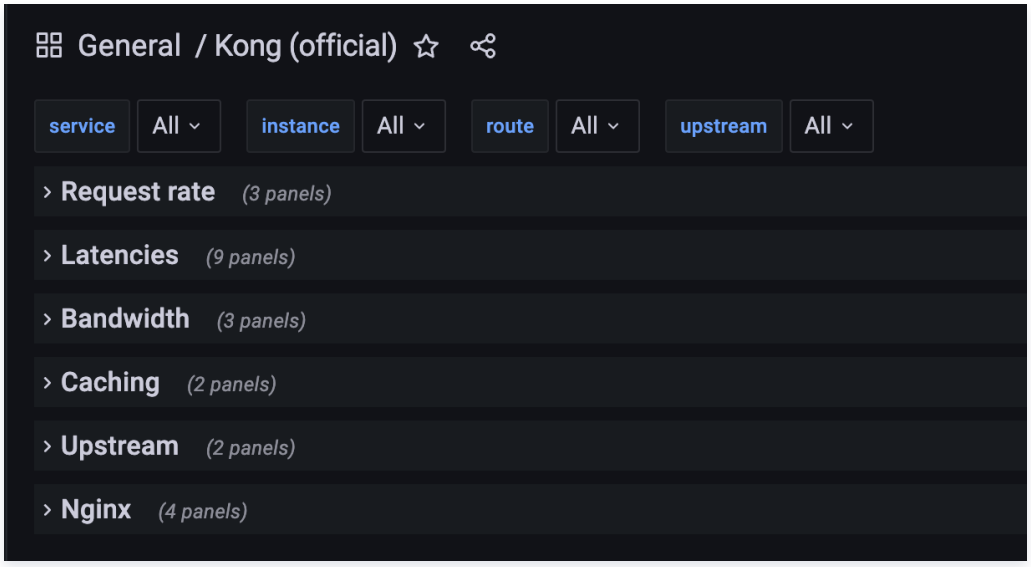
1. 前往 Grafana-Kong 官网，下载 Dashboard 配置 JSON 文件。




2. 在您的 Grafana 上导入 Kong 的 Dashboard 模板。




3. 导入 Kong 的 Dashboard 模板后，如果有数据上报到 Prometheus ，则 Grafana 上会展示类似下图的视图。




## 注意事项

- 开启 Prometheus 插件对于网关的数据流性能有影响，建议只为需要监控的特定的 API(Route) 开启 Prometheus 插件。

版权所有：腾讯云计算（北京）有限责任公司

- 在腾讯云上购买云监控 Prometheus 将产生费用，需要您自行承担。
- 为简单起见，上述操作依赖 Konga 控制台演示。您也可以调用 kong admin api 来绑定 Prometheus 插件，具体可以参见 Kong-Prometheus 插件文档 。关于如何开启 kong admin api ，请参见 开启 admin-api 并配置安全认证 。

## 参考

更多相关说明请参见 Kong Prometheus 插件官方文档 。

# 链路追踪

最近更新时间：2025-06-09 16:07:02

随着微服务架构的普及，一个完整的业务逻辑调用请求的背后可能牵涉后端几个、几十个甚至上百个服务接口，后台服务形成一种复杂的分布式网络。在该场景下，如何在请求发生异常时快速定义问题所在就成为了分布式场景下排障的关键问题。
云原生 API 网关支持链路追踪功能，使用者可以通过具体的服务、状态码、响应耗时等查询具体的调用过程，包括调用过程所需要的时间和运行情况。根据这些信息分析在分布式链路调用过程中每个环节的耗时和异常，如：

- 定位耗时较长的服务。
- 不合理的调用逻辑（如一次请求多次调用某服务，建议改为批量调用接口）。

## 开启链路追踪

1. 登录微服务平台 TSF 控制台。
2. 在左侧导航栏单击**云原生 API 网关** > **数据观测**，在页面上方选择好网关实例。
3. 单击 **链接追踪** 页签，在页面中间单击**立即开启**，开启链路追踪。
   - 协议：支持 OpenTelemetry (推荐)、SkyWalking、Zipkin、aeger 四种协议。网关默认使用收到请求时客户端指定的链路追踪协议，如客户端未指定，将使用控制台指定的协议。
   - API 业务系统：勾选后网关将使用该 APM 业务系统进行链路上报。
   - 采样率：取值范围 0-100，0 为不采集，100 为每个请求都采集，默认为 10，采样率配置过高可能会影响性能。

> **说明：**
> - 如需展示完整调用链路，请确保网关下游服务使用相同的 APM 业务系统进行上报。应用上报方式请参见 应用上报指南。
> - 链路追踪由 应用性能监控服务 提供，具体计费项查看 费用详情。

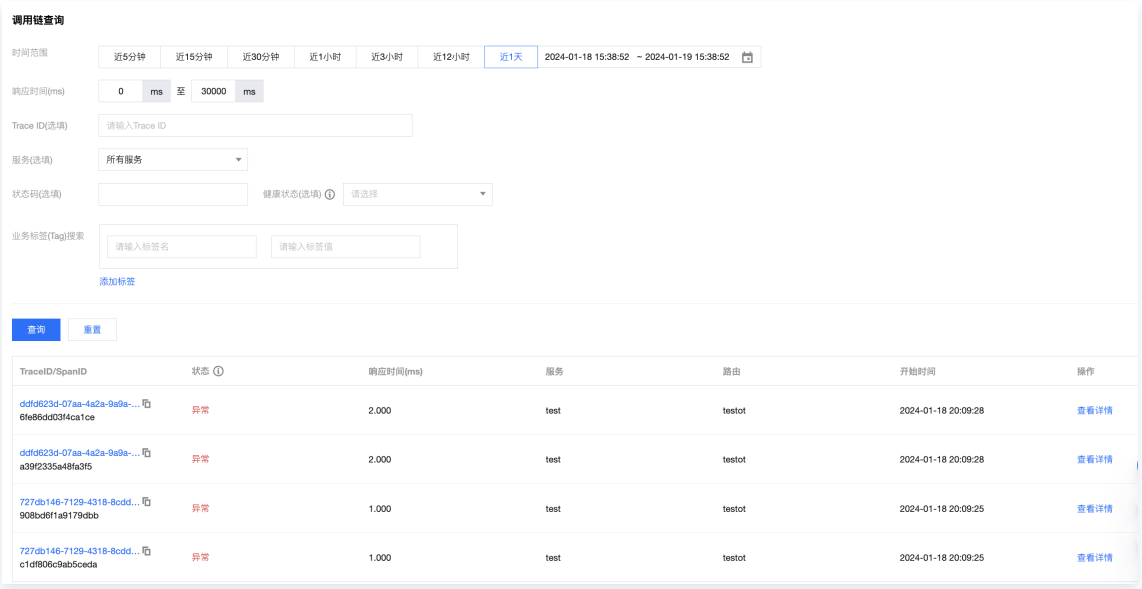



## 查询调用链详情

开启链路追踪后，在链路追踪页面，可以查看网关实例的调用链及详情。

1. 在调用链查询中，设置好查询条件，单击**查询**。
   - 时间范围：支持特定和自定义时间范围选择。特定时间范围包括：近5分钟、近15分钟、近30分钟、近1小时、近3小时、近12小时和近1天。

- 响应时间：设置响应时间范围。
- Trace ID（可选）：Trace ID。
- 服务：单击下拉框，在下拉框中选择要查询的服务，可以输入关键字进行搜索。
- 状态码（可选）：状态码。
- 健康情况（可选）：
  - 正常：根据响应时间和错误率评估，该调用正常。
  - 慢调用：该调用耗时执行超过2000ms。
  - 异常：该调用返回非 200 HTTP 状态码。
- 业务标签：网关用于链路追踪的标签。



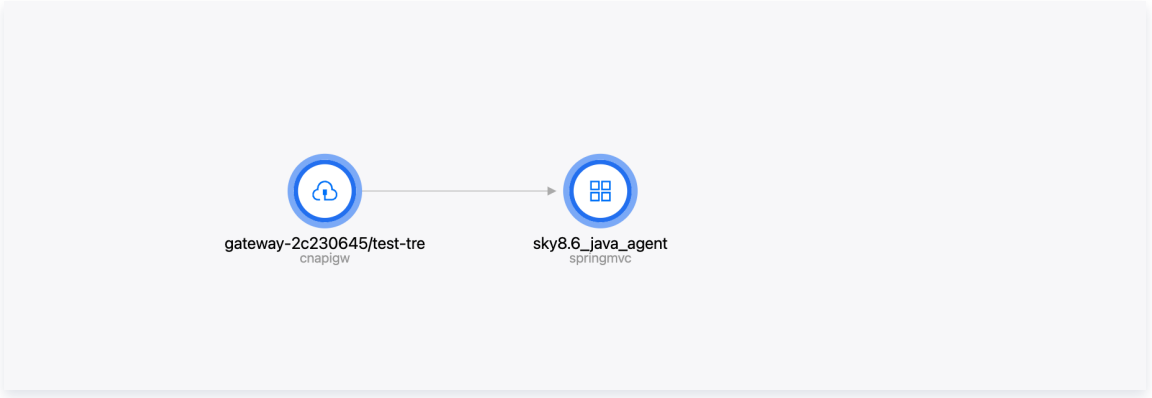

2. 单击调用链的“ID”或者操作列的**查看详情**，可查看调用链的详细信息，包含调用信息详情和各阶段耗时。




## 查看依赖拓扑图

开启链路追踪后，在服务基本信息页面，可以查看服务依赖拓扑图，根据服务拓扑图，您可以直观的了解服务的上下游组件与它们的调用关系，从而更快速地找出应用的瓶颈。

1. 在左侧导航栏选择**服务路由**，在页面上方选择好网关实例后，单击**服务**页签。
2. 单击要查看的服务名称，进入服务详情页面，在依赖拓扑模块，设置好时间范围后可以查看服务调用信息。



# 日志
# 查看默认日志

最近更新时间：2025-06-09 16:07:02

## 操作场景

云原生 API 网关默认为您提供网关实时日志服务和简单搜索能力，免费使用。
默认日志主要分为用户访问日志和网关错误日志。您可以通过查看云原生 API 网关的访问日志了解用户的请求相关信息，便于进行数据分析、审计、业务排障等或者查看，也可以查看云原生 API 网关的错误日志，以便排查问题。

- 访问日志（accessLog）记录了用户的请求相关信息，可用于进行数据分析、审计、业务排障等。
- 错误日志（errorLog）是网关内部错误日志，用于网关排障。

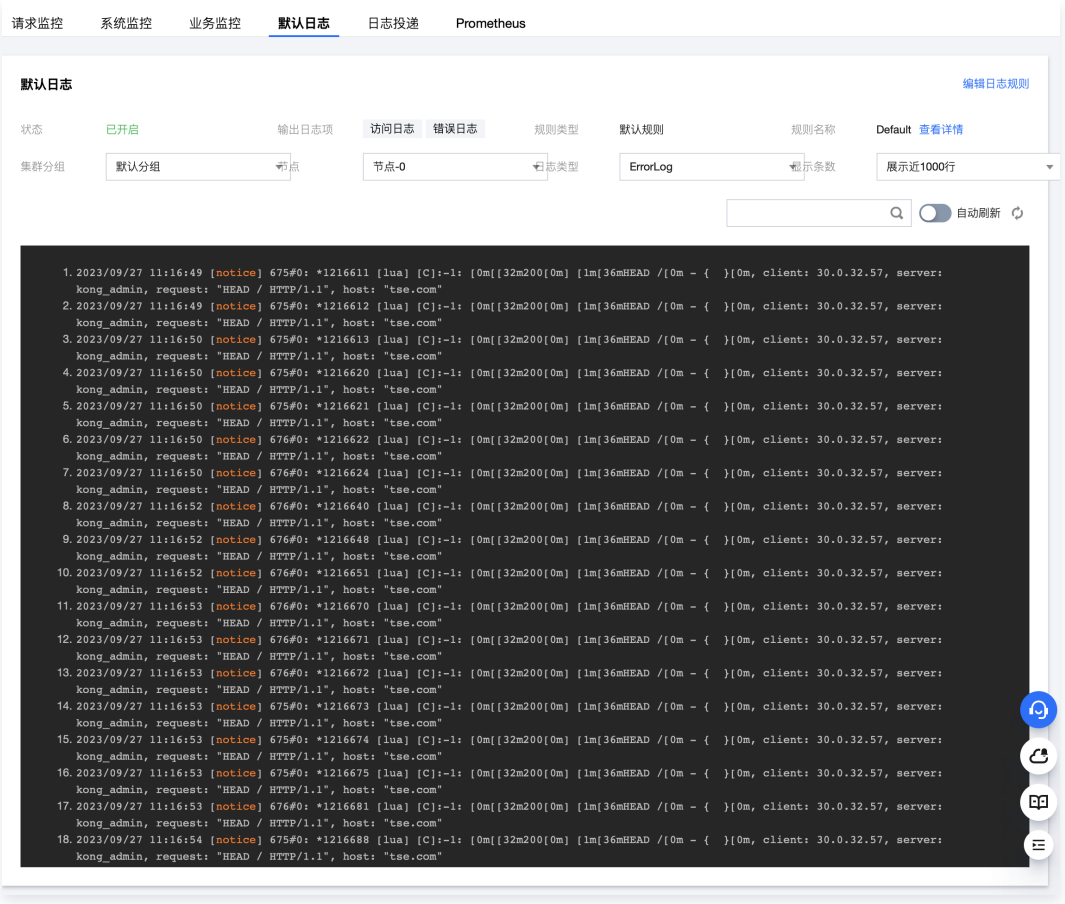
本文为您介绍云原生 API 网关默认日志功能的使用说明。

## 前提条件

已创建云原生 API 网关实例，具体操作请参见 新建云原生 API 网关 。

## 查看默认日志

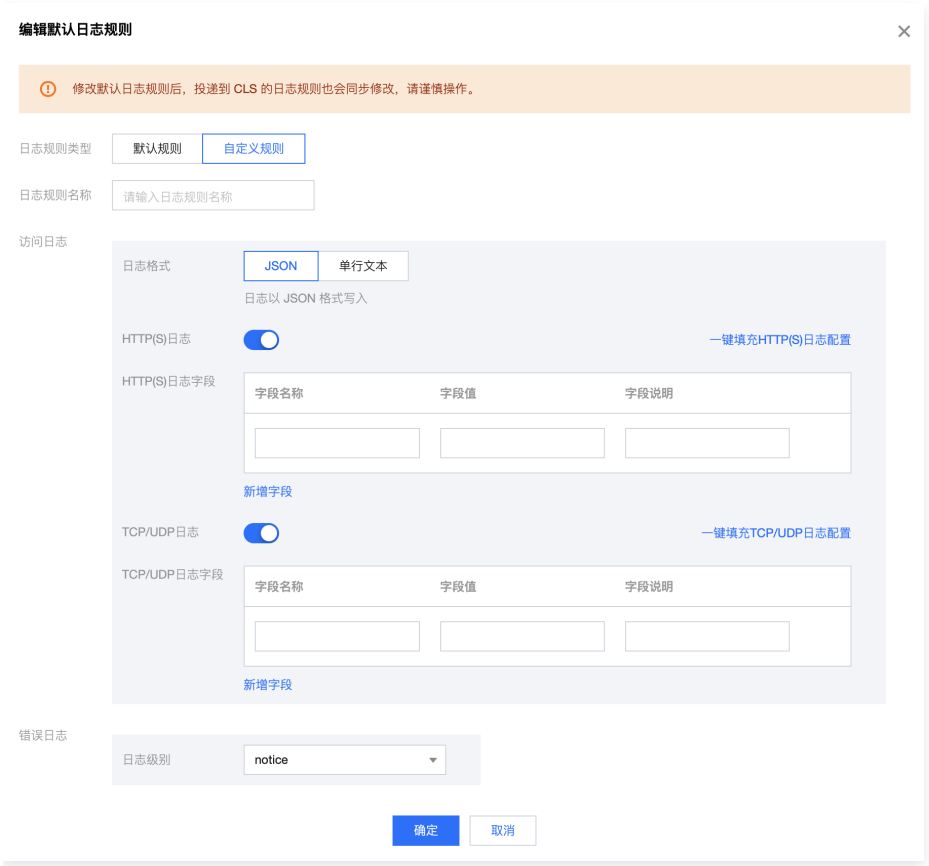
1. 登录 TSF 控制台 。
2. 在左侧导航栏单击**云原生 API 网关** > **数据观测**，在页面上方选择好网关实例，单击**默认日志**页面。
3. 设置好您要查看的日志，页面即可展示相关日志内容。 可以通过关键字查询相关日志。输入关键词查询，例如：“info”，注意日志检索区分大小写。




## 编辑默认日志规则

在默认日志页面，单击右上角的**编辑日志规则**，即可修改默认日志规则。您可以选择继续使用默认规则，也可以根据您的业务需求自定义日志规则。编辑默认日志规则时支持开启和关闭 HTTP 和 TCP 的日志输出。

> **注意：**
> 修改默认日志规则后，投递到 CLS 的日志规则也会同步修改，请谨慎操作。




## 日志字段

下表列出了云原生 API 网关支持的访问日志字段，您可以根据需要进行配置：

## HTTP/HTTPS 日志字段

| 字段 | 说明 |
| --- | --- |
| $remote_addr | 客户端地址。 |
| $status | HTTP 状态码。 |
| $remote_user | Basic authentication 提供的用户名。 |
| $time_local | 请求时间。 |
| $request | 完整的请求行。 |
| $body_bytes_sent | 发送给客户端的文件主体内容的大小。 |
| $request_method | 请求方法。 |
| $host | 请求携带 Host 请求头时为“Host”字段的值，未携带时为主机虚拟域名。 |
| $upstream_addr | 后端服务的 IP 地址。 |

| | |
|---|---|
| $upstream_status | 上游服务返回响应中的 HTTP 响应码。 |
| $upstream_response_time | 上游服务响应耗时（毫秒精度），包括网关向后端服务开始建立连接、接收数据、关闭连接的时间。 |
| $scheme | HTTP 或 HTTPS 协议。 |
| $url | 请求 URL。 |
| $request_length | 请求数据大小 bytes，包含请求行、请求头、请求体。 |
| $bytes_sent | 响应字节数。 |
| $http_referer | 页面来源，header Referer 引用页面 URL。 |
| $http_user_agent | 客户端代理信息。 |
| $request_time | 请求耗时，从接收请求开始到发送完响应数据的时间，包含接收请求数据、处理请求、返回响应数据的时间。 |

## TCP/UDP 日志字段

| 字段 | 说明 |
|---|---|
| $remote_addr | 客户端 IP 地址。 |
| $time_local | 请求时间，处理完请求打印日志的时间。 |
| $protocol | 四层协议 tcp 或 udp。 |
| $status | 连接状态，可选值200, 400, 403, 500, 502, 503。 |
| $bytes_sent | 响应字节数，网关发往客户端的字节数。 |
| $bytes_received | 接收字节数，网关从客户端接收到的字节数，包括响应头和响应体。 |
| $session_time | 会话持续时间（毫秒精度）。 |

## Nginx 变量

不支持的 Nginx 变量如下：

1. 如下变量：
   - $connection_time
   - $http3
   - $jwt_claim_
   - $jwt_header_
   - $jwt_payload
   - $memcached_key
   - $mqtt_preread_clientid
   - $mqtt_preread_username
   - $otel_parent_id
   - $otel_parent_sampled
   - $otel_span_id
   - $otel_trace_id
   - $proxy_protocol_tlv_

- $proxy_protocol_tlv_aws_vpce_id
- $proxy_protocol_tlv_azure_pel_id
- $proxy_protocol_tlv_gcp_conn_id
- $secure_link
- $secure_link_expires
- $session_log_binary_id
- $session_log_id
- $slice_range
- $ssl_alpn_protocol
- $ssl_curve
- $upstream_queue_time

2. geo 开头的变量。

# 日志投递至 CLS

最近更新时间：2025-06-09 16:07:02

## 操作场景

如您需要日志持久化存储，用于排障、审计等场景，建议开启 CLS 日志服务，将网关日志投递到 CLS。开启前请先确认您已开通 CLS 日志服务。日志服务由 CLS 日志服务 提供，会产生费用，具体计费项查看 费用详情 。

## 前提条件

已创建云原生 API 网关实例，具体操作请参见 新建云原生 API 网关 。

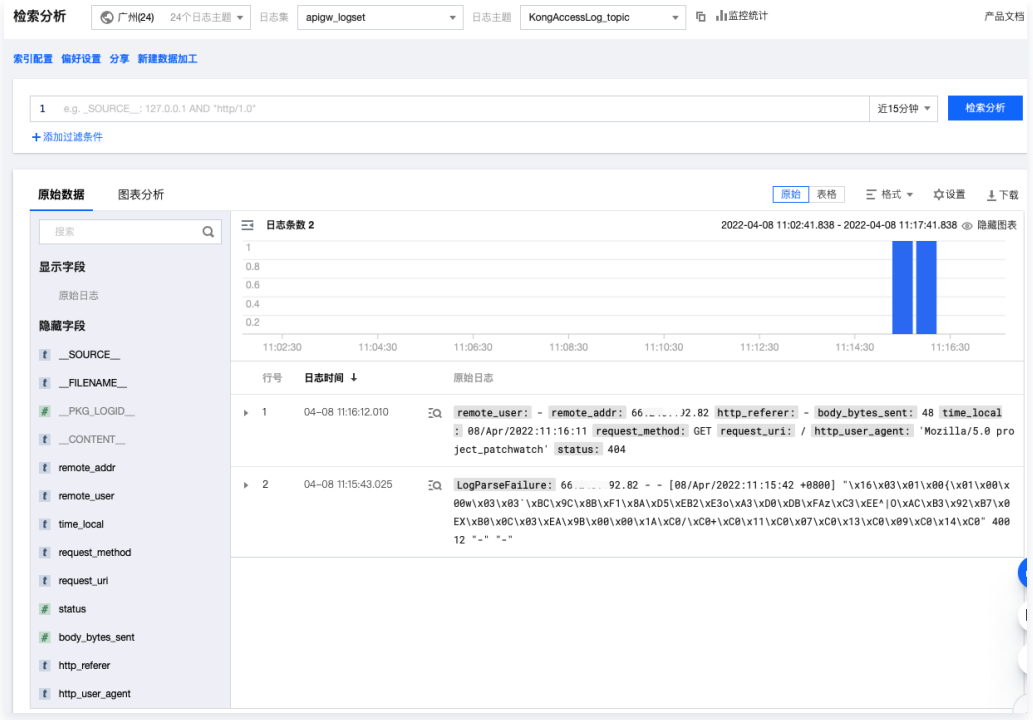
## 开启日志投递

1. 登录 TSF 控制台 。
2. 在左侧导航栏单击**云原生 API 网关** > **数据观测**，在页面上方选择好网关实例，单击**日志投递**页签。
3. 在 CLS 日志投递页面，单击**开启**，在弹窗中开启**投递日志**按钮，勾选要投递的日志项，支持自动创建和选择已有日志集和日志主题，单击**确定**后，即可开始投递日志。

> **注意：**
> 关闭日志投递不会删除日志集和日志主题，如需删除，请前往 CLS 控制台操作。



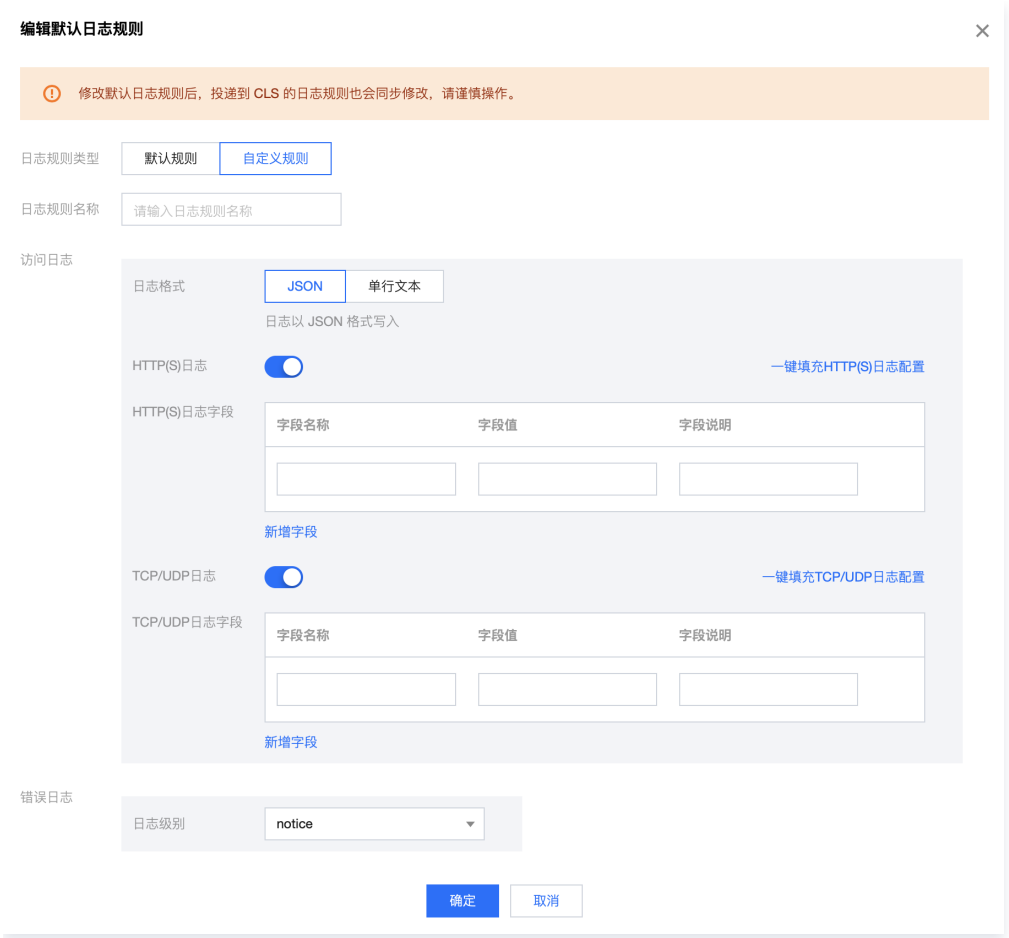

4. 配置完成后您可以单击**关联的日志主题**链接地址，自动跳转至网关投递的日志库查看日志。




## 编辑 CLS 日志规则

在日志投递页面，单击右上角的编辑日志规则，即可修改 CLS 日志规则。您可以选择继续使用默认规则，也可以根据您的业务需求自定义日志规则。

> **注意：**
> 投递到 CLS 的日志与默认日志使用相同的日志规则。修改 CLS 日志规则后，默认日志规则也会同步修改，请谨慎操作。




## 编辑 CLS 日志投递

**在日志投递页面，单击右上角的编辑日志投递，在弹窗中即可修改日志投递规则。**



# 上报日志到 Kafka

最近更新时间：2025-06-09 16:07:02

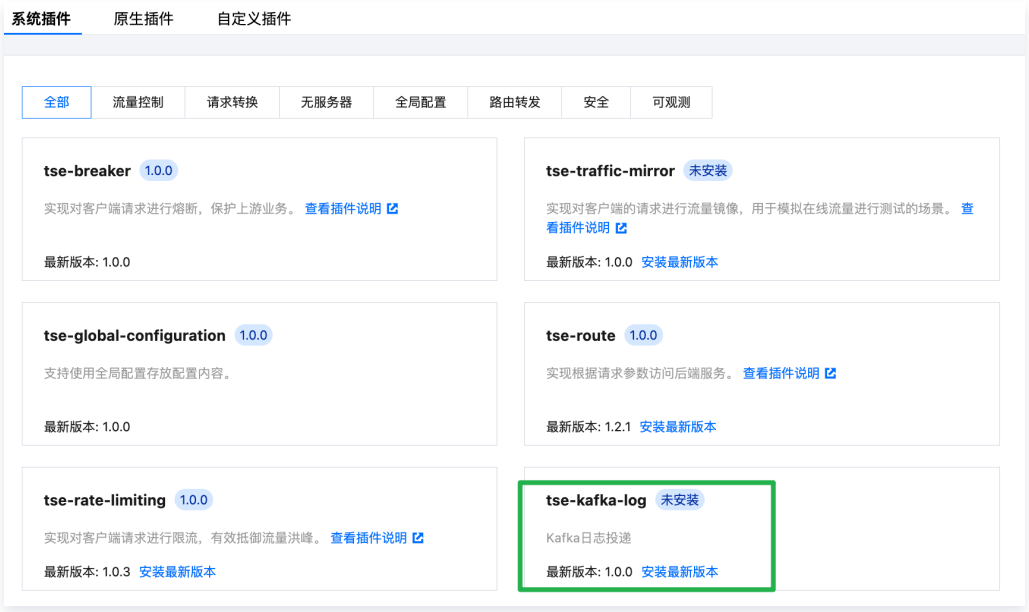
本文介绍云原生 API 网关 Kong 如何将请求日志上报到 Ckafka 或自建 Kafka 。

## 操作场景

提供自建的 Kafka 以及对应的 topic 或使用腾讯云 Ckafka 产品，根据下述操作，就可以将云原生 API 网关的流量日志落盘到该 Kafka 实例上。

## 操作步骤

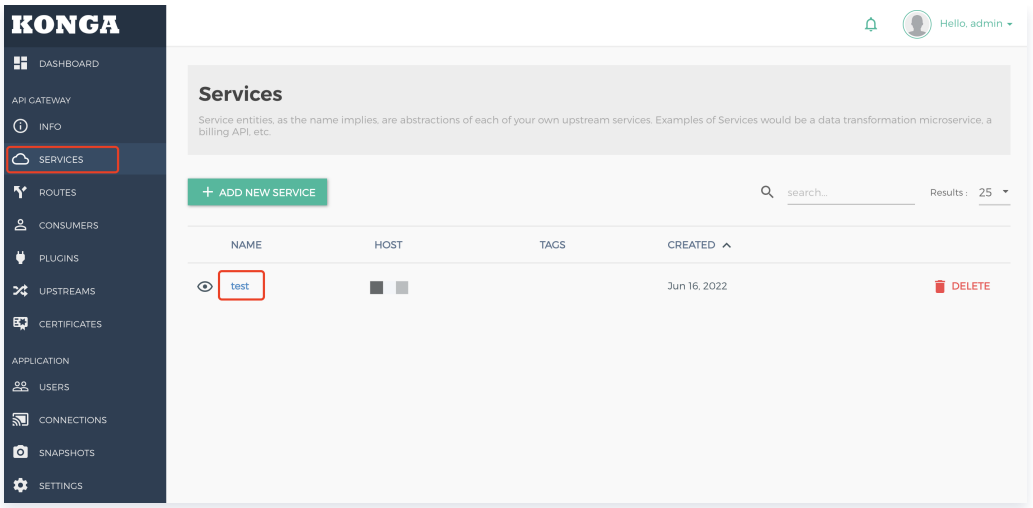
1. 安装最新版本 tse-kafka-log 插件。 微服务引擎 Kong 控制台 ，选择需要的网关实例，单击**插件管理 > 系统插件**。
   如果已经安装，可以忽略此步骤。



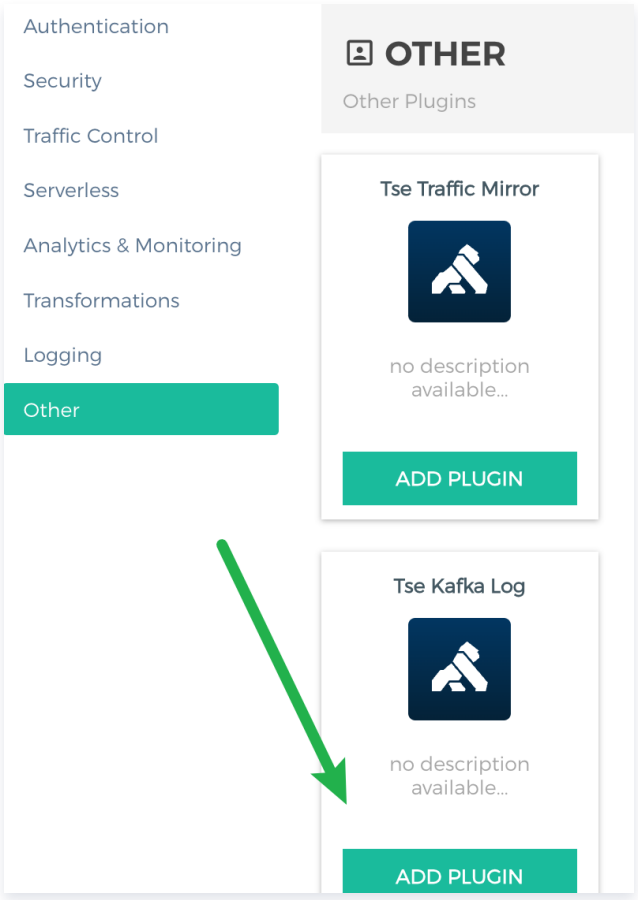

2. 进入云原生 API 网关的 Kong 实例的**基本信息 > Konga 控制台**，查看 Kong 管理控制台地址并进入。




3. 进入 Kong 管理控制台页面后，单击侧边导航栏 SERVICES，选择需要的 service 并进入详情页。

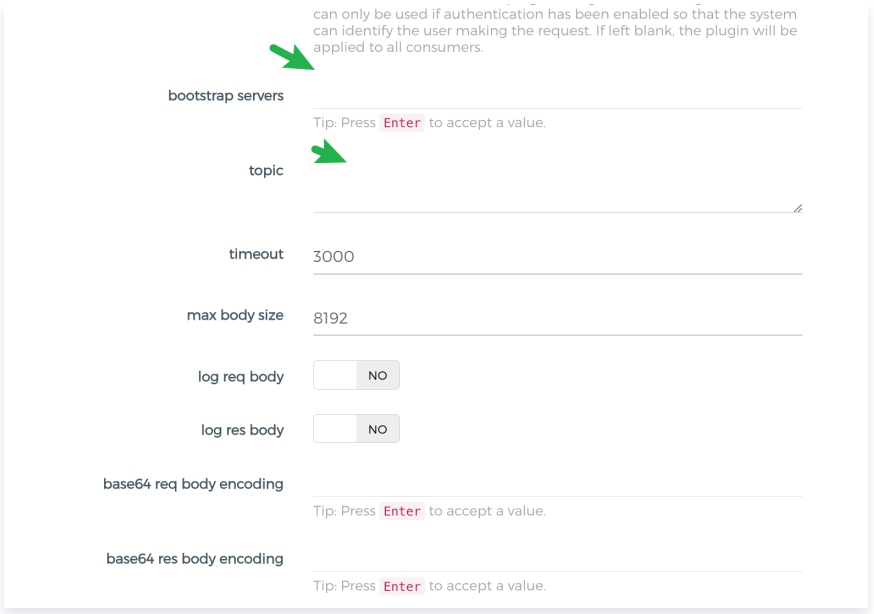



4. 选择 Plugins 并单击 **Add Plugin**，在 Other 栏中选择添加 Tse Kafka Log。




5. 填写 kafka server 地址以及 topic，其中 bootstrap servers 填写 kafka 的 server 地址，topic 是 kafka 的 topic 名字。

> **注意：**
> 1. **topic** 是填写 topic 的名字，而不是 ID。
> 2. 插件配置的详细字段说明，在步骤7中。




6. 该插件绑定 service 后，请求 service，消费 kafka 后可以查看消费到当前请求的日志内容（json 格式）。

```
root@VM_0_12_centos ~# curl www.mytest.com -H'myheader:test'
headers begin<br/>
host : www.f[redacted].net.cn<br/>
x-forwarded-host : www.mytest.com<br/>
connection : keep-alive<br/>
x-forwarded-proto : http<br/>
myheader : test<br/>
x-forwarded-for : [redacted].195.[redacted].84<br/>
user-agent : curl/7.29.0<br/>
accept : */*<br/>
x-real-ip : [redacted].195.[redacted].84<br/>
x-forwarded-path : /<br/>
x-forwarded-port : 8000<br/>
GET / HTTP/1.1
nil
{"ret":0,"data":{"miniAppStatus":3},"code":"new code"}
```

```
[leonardjin@VM-203-246-centos ~/kafka/kafka_2.11-1.1.1/bin]$ sh kafka-console-consumer.sh --topic cms --bootstrap-server 11.[redacted]:16286 --from-beginning
one

{"request_time":"0.269","status":200,"date_time":"2022-06-09 18:23:03","method":"GET","http_host":"www.mytest.com","upstream_addr":"203.[redacted].:80","upstream_response_time":"0.013","user_agent":"curl/7.29.0","machine":"kong-0","in_bytes":"93","out_bytes":"678","name":"mytest","uri":"/","remote_addr":"203.[redacted]","forwarded_ip":"203[redacted]","query_string":""}
```

7. tse-kafka-log 插件配置字段详细说明

> **注意：**
> 在 Konga 的页面中，配置项键的_会被隐藏掉，例如：bootstrap_servers 会显示为 bootstrap servers。

| 配置项键 | 是否必填 | 默认值 | 说明 |
| --- | --- | --- | --- |
| bootstrap_servers | 是 | 无 | 数组，一般的 bootstrap_servers 地址是9092端口，例如：10.0.90.16:9092 |
| topic | 是 | 无 | topic 的名字，而不是 ID |
| max_body_size | 否 | 8192 | 请求和响应的最大 body 大小，超过此大小的 body 会被截断 |

| | | | |
|---|---|---|---|
| log_req_body | 否 | false | 是否打印请求 body |
| log_res_body | 否 | false | 是否打印响应 body |
| base64_req_body_encoding | 否 | 无 | 数组。当请求 Header：content-encoding 符合其中一个，会把请求的 body进行 base64 encode |
| base64_res_body_encoding | 否 | 无 | 数组。当响应 Header：content-encoding 符合其中一个，会把响应的 body进行 base64 encode |
| log_req_headers | 否 | true | 是否打印请求 headers |
| log_res_headers | 否 | true | 是否打印响应 headers |
| exclude_req_headers | 否 | 无 | 数组。打印请求 headers 中，删除指定的 header |
| exclude_res_headers | 否 | 无 | 数组。打印响应 headers 中，删除指定的 header |
| skip_expr_in_access | 否 | 无 | 高级，在 access 阶段运行一个 lua 代码，判断是否跳过打印日志。return true 就是跳过 |
| skip_expr_in_body_filter | 否 | 无 | 高级，在 body filter 阶段运行一个lua代码，判断是否跳过打印日志。return true 就是跳过 |
| http_log_mode | 否 | default | default/close/custom。default 为内置的日志字段，close 为关闭日志，custom 为自定义日志 |
| http_log_custom_format | 否 | {} | 七层自定义的日志格式。例如{"cip": "$remote_addr", "host": "$host"}，跟 CLS 日志保持一致 |
| stream_log_mode | 否 | default | default/close/custom。default 为内置的日志字段，close 为关闭日志，custom 为自定义日志 |
| stream_log_custom_format | 否 | {} | 四层自定义的日志格式。例如{"cip": "$remote_addr", "protocol": "$protocol"}，跟 CLS日 志保持一致 |
| batch_size | 否 | 100 | 消息缓存队列长度，默认100 |
| retry_count | 否 | 3 | 消息发送重试次数，失败时候会重试 |
| flush_timeout | 否 | 2000 | 发送的超时时间，单位 ms |
| compressor | 否 | Lz4 | 消息压缩算法，默认用 Lz4 |

## 参考

更多相关说明请参见 Kong 插件文档 。

# 插件上报日志到 CLS

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何通过 CLS Log 插件，将包含详细请求、响应 Body 报文的日志上报到 CLS Topic 中。

> **注意：**
> 目前云原生 API 网关已经支持 Access Log 投递到 CLS，并且支持修改 Access Log 的日志格式，可以先评估投递到 CLS 的功能是否满足需求，详细可阅读 日志投递至 CLS 。再考虑是否用插件方式上报日志到 CLS，使用插件上报到 CLS，可以定制自己的日志上报格式和上报更多的字段。

## 前置条件

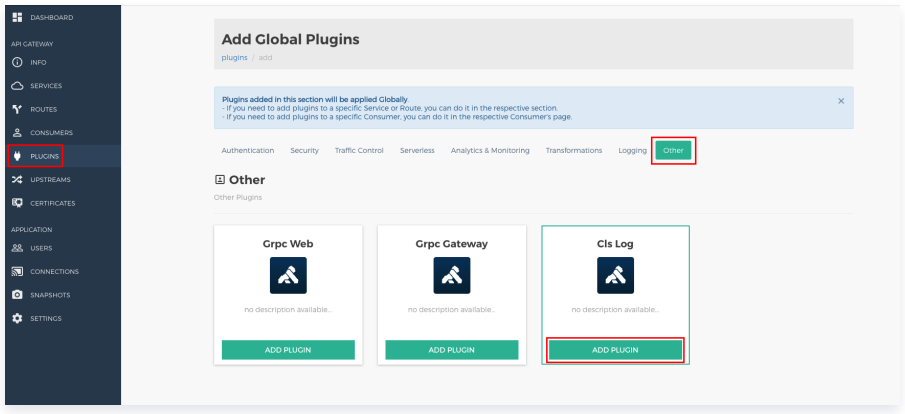
1. 已购买 Kong 网关实例，操作文档 。
2. 进入 TSF 控制台，并找到需要的实例。
3. 进入实例详情页后，找到**插件管理 > 自定义插件**。

## 操作步骤

1. 下载 CLS Log 插件安装包 cls_log.zip 。
2. 参考 插件管理，在网关实例插件配置中，上传安装 CLS Log 插件，并启用。




3. 登录 Konga 控制台，在 PLUGINS 菜单中添加 CLS Log 插件全局开启，参数配置请参见下文中插件参数部分。




4. 通过网关 IP 发起请求，在 CLS 控制台确认日志正确上报到对应的 CLS Topic 中，日志示例如下：

```
{
    "tries":"
[{\"balancer_latency\":0,\"port\":80,\"balancer_start\":1654569xxxxxx,\"ip\":\"172.0.0.2\"}]"
,
    "upstream_uri":"/10k",
    "querystring":"{}",
    "req_url":"http://172.0.0.3/test",
    "latency_kong":"1",
    "req_method":"GET",
    "resp_headers":"{\"x-kong-upstream-latency\":\"2\",\"date\":\"Tue, 07 Jun 2022 02:34:15
GMT\"}",
    "req_uri":"/test",
    "resp_status":"200",
    "request_time":"3",
    "req_body":"",
    "resp_body":"some response body",
    "latency_proxy":"2",
    "req_headers":"{\"host\":\"172.16.0.37\"}",
    "started_at":"1654569xxxxxx",
    "client_ip":"127.0.0.1",
    "upstream_response_time":"2",
    "resp_size":"10476",
    "route_id":"cca80823-3189-47fb-8a1f-21ebe7xxxxxx",
    "route_name": "",
    "service_id":"17caaef5-dc84-4c93-8148-e63ce1xxxxxx",
    "service_name": "",
    "consumer_id":"",
    "consumer_name": ""
}
```


## 插件参数介绍

| 参数 | 默认值 | 描述 |
| --- | --- | --- |
| cls_host | - | CLS API Host，例如：ap-beijing.cls.tencentyun.com。详情请参见 CLS doc |
| cls_topic | - | 日志上报的目标 CLS Topic |
| secret_id | - | 腾讯云账户 SecretID |
| secret_key | - | 腾讯云账户 SecretKey |
| log_req_body | false | 是否记录请求 Body 日志字段，开启后对网关性能有一定影响 |
| log_resp_body | false | 是否记录响应 Body 日志字段，开启后对网关性能有一定影响 |
| max_req_body_size | 10240 | 最大请求 Body 日志长度（单位字节），超出会被截断 |
| max_resp_body_size | 10240 | 最大响应 Body 日志长度（单位字节），超出会被截断 |
| bacth_size | 100 | 单次上报日志数量，一批日志大小不建议超过 8 MB。 |
| retry_count | 3 | 最大重试次数。 |

版权所有：腾讯云计算（北京）有限责任公司

| | | |
|---|---|---|
| mode | memory | memory: 通过 CLS SDK 从内存队列中异步上报日志。access_log: 通过 access_log 模块，配合 loglistener 采集上报，受磁盘性能影响，如需使用请工单联系开通。 |
| fallback_to_access_log | false | 当使用 memory 模式时，上报队列堆积时是否降级到 access_log 上报模式, 如需使用请工单联系开通。 |

## 注意事项

1. CLS Log 插件中配置的密钥对需拥有 `cls:UploadLog` 策略权限，详情请参见 CLS 自定义策略示例 中使用 API 上传数据部分。
2. 开启 CLS Log 日志插件会对网关性能有少量影响，影响程度和报文日志大小成正比。
3. CLS Log 插件中实现了缓冲队列、分批上报，失败重试等功能，能够在偶发性上报失败时进行重试，但由于缓冲队列大小有限，长时间上报链路故障时仍然有丢失日志的风险。
4. CLS Log 插件包含通用的基本日志字段，如需自定义日志字段，可以基于该插件二次开发。

# 查看日志大盘

最近更新时间：2025-06-09 16:07:02

## 操作场景

云原生 API 网关提供开箱即用的 CLS 日志大盘，如您开启了 CLS 日志投递，即可使用日志分析仪表盘对投递到 CLS 的访问日志和错误日志进行快速分析。（**仅专业版支持**）
日志大盘支持的场景包括：

- 状态码分析
- 请求量分析
- 请求耗时分析
- 错误日志分析
- 用户行为分析
- 业务分析

本文为您介绍云原生 API 网关日志大盘功能的使用说明。

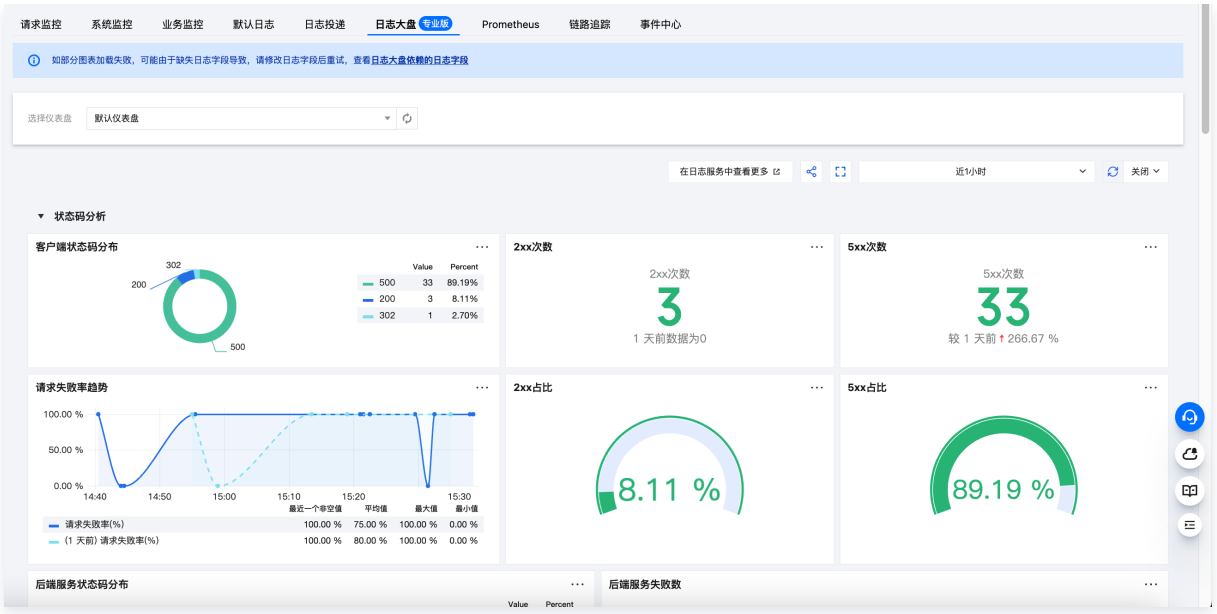
## 前提条件

1. 已创建云原生 API 网关实例，具体操作请参见 新建云原生 API 网关 。
2. 已开通 CLS 日志服务。
3. 已开启 CLS 日志投递，具体操作请参见 日志投递至 CLS 。
4. 日志格式为 JSON 格式，如当前格式为单行文本，请修改日志格式为 JSON 格式，具体操作请参见 编辑 CLS 日志规则 。
5. 由于大盘依赖部分日志字段进行展示，请将以下字段添加至日志规则：

| 字段名 | 字段值 |
| --- | --- |
| status | $status |
| ups_status | $upstream_status |
| client | $remote_addr |
| req_time | $request_time |
| ups_rsp_time | $upstream_response_time |
| referer | $http_referer |
| method | $request_method |
| host | $host |
| uri | $uri |
| ups_ip | $upstream_addr |

## 操作步骤

1. 登录 微服务平台控制台 ，在左侧导航栏单击云原生 API 网关 > 实例列表。
2. 在左侧导航栏单击**云原生 API 网关** > **数据观测**，在页面上方选择好网关实例，单击**日志大盘**页签。




# 支持指标及含义

## 状态码分析

| 指标名 | 指标含义 |
| --- | --- |
| 客户端状态码分布 | 网关返回到客户端的状态码分布。 |
| 2xx次数 | 客户端收到的状态码为 2xx 的请求次数。 |
| 5xx次数 | 客户端收到的状态码为 5xx 的请求次数。 |
| 请求失败率趋势 | 网关请求失败率的趋势。 |
| 2xx占比 | 客户端收到的状态码为 2xx 的请求占比。 |
| 5xx占比 | 客户端收到的状态码为 5xx 的请求占比。 |
| 后端服务状态码分布 | 后端服务返回网关的状态码分布。 |
| 后端服务失败数 | 后端服务失败（4xx和5xx）的请求次数。 |
| 后端服务失败率趋势 | 后端服务收到失败请求（4xx和5xx）的次数。 |

## 请求量分析

| 指标名 | 指标含义 |
| --- | --- |
| 总请求数 | 总请求数，按照所选择的时间粒度统计求和。 |
| 请求次数趋势 | 请求次数趋势。 |
| 请求地域分布 | 客户端 IP 所在地域分布。 |

## 请求耗时分析

| 指标名 | 指标含义 |
| --- | --- |
| 请求总耗时趋势 | 请求总耗时趋势。 |

| | |
|---|---|
| 后端响应耗时趋势 | 后端服务响应的耗时趋势。 |

### 错误日志分析

| 指标名 | 指标含义 |
|---|---|
| Error日志数 | 在所选择的时间范围内出现的错误日志总数。 |
| Error日志趋势 | 在所选择的时间范围内出现的错误日志总数趋势。 |
| Error日志 | 在所选择的时间范围内出现的错误日志详情。 |

### 用户行为分析

| 指标名 | 指标含义 |
|---|---|
| HTTP Referer 分布 | HTTP Referer 分布。 |
| 请求方法分布 | 请求方法分布。 |
| Host 分布 | Host 分布。 |

### 业务分析

| 指标名 | 指标含义 |
|---|---|
| TOP10 调用量 API | 在所选择的时间范围内调用量前 10 的 API。 |
| TOP10 失败率 API | 在所选择的时间范围内失败率前 10 的 API。 |
| TOP10 调用量服务 | 在所选择的时间范围内调用量前 10 的 后端服务。 |
| TOP10 失败率服务 | 在所选择的时间范围内失败率前 10 的 后端服务。 |
| TOP10 请求 Host | 在所选择的时间范围内数量前 10 的 请求 Host。 |

版权所有：腾讯云计算（北京）有限责任公司

# Terraform 管理

最近更新时间：2025-06-09 16:07:02

Terraform 是一个 IT 基础架构自动化编排工具，它的口号是“Write, Plan, and Create Infrastructure as Code”，是一个“基础设施即代码”工具，通过 Terraform 您可以创建、更新和版本控制的 TEM 中的资源。了解更多关于 Terraform，请参见 Terraform 指南 。

## 通过 Terraform 管理 TSF 云原生网关资源

| 资源类型 | 说明 |
| --- | --- |
| tencentcloud_tse_cngw_canary_rule | 云原生 API 网关灰度规则。 |
| tencentcloud_tse_cngw_certificate | 云原生 API 网关证书。 |
| tencentcloud_tse_cngw_gateway | 云原生 API 网关实例。 |
| tencentcloud_tse_cngw_group | 云原生 API 网关实例分组。 |
| tencentcloud_tse_cngw_route | 云原生 API 网关路由。 |
| tencentcloud_tse_cngw_route_rate_limit | 云原生 API 网关路由级限流。 |
| tencentcloud_tse_cngw_service | 云原生 API 网关服务。 |
| tencentcloud_tse_cngw_service_rate_limit | 云原生 API 网关服务级限流。 |
| tencentcloud_tse_waf_domains | 云原生 API 网关证书。 |
| tencentcloud_tse_waf_protection | 云原生 API 网关WAF 防护。 |

版权所有：腾讯云计算（北京）有限责任公司

# 参数配置

最近更新时间：2025-06-09 16:07:02

## 操作场景

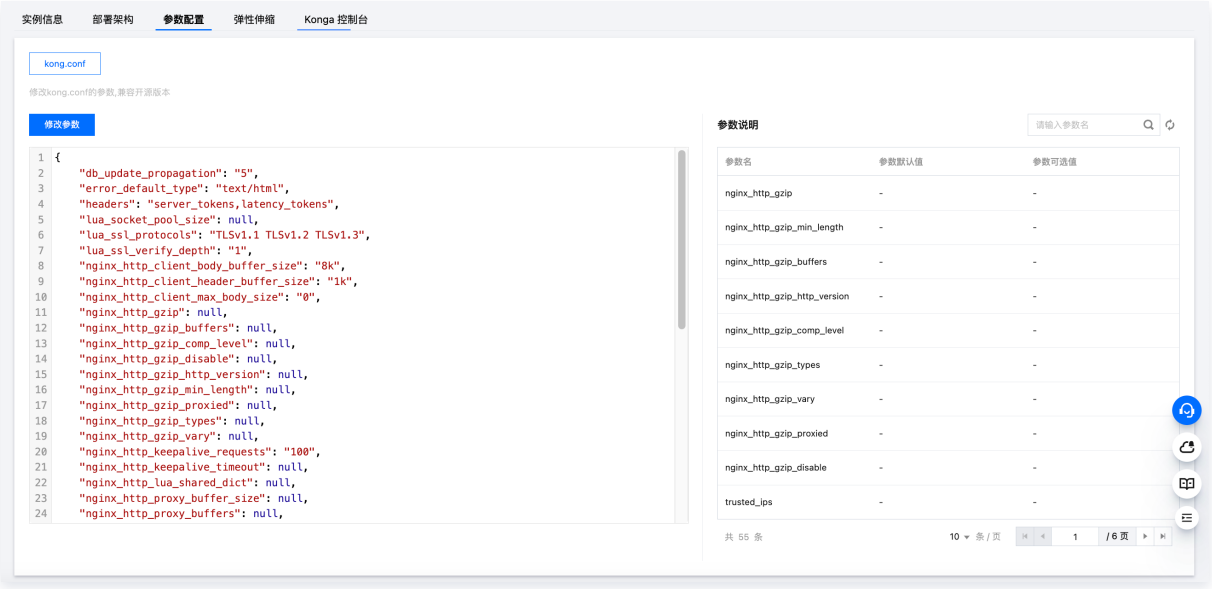
创建云原生 API 网关实例时，实例中的参数例如 `nginx_http_client_header_buffer_size` 、`nginx_http_client_max_body_size` 等采用默认配置，如果您的业务有特殊要求可以在 TSE 控制台修改对应的参数，优化网关性能。

## 前提条件

已创建云原生 API 网关实例 。

## 操作步骤

1. 登录 TSF 控制台 。
2. 在左侧导航栏选择**云原生 API 网关 > 实例列表**，单击目标实例的“ID”，进入网关实例基本信息页面。
3. 在**基础信息**，选择**参数配置**，进入参数配置页面。左侧以 JSON 格式展示所有支持修改的系统参数，右侧展示参数说明列表，您可通过参数名搜索，查看该参数支持的默认值和可选值。




4. 单击左上角的**修改参数**，编辑 JSON 内容，修改参数值为合适的内容。单击**保存**，实例将进行 kong reload 操作，单击**确定**，完成参数修改，修改后立即生效。

> **注意：**
> - 系统参数值均为字符串，修改需带英文双引号（""）。
> - 默认展示实例初始参数值，当前实例该参数未设置则展示为 null。
> - Access Log 日志格式的修改，在控制台的**可观测性 > 默认日志 > 编辑日志规则**中修改，请参见 查看默认日志 。

# 插件管理

最近更新时间：2025-06-09 16:07:02

## 操作场景

云原生 API 网关 Kong 提供插件管理功能，您可以在控制台上管理您的系统插件、原生插件、部署自定义插件等。本文介绍相关的插件定义和开发自定义插件并部署到云原生 API 网关 Kong 的操作步骤。

## 插件介绍

### 系统插件

**系统插件**是指云原生 API 网关在开源基础上提供的增强插件。

| 名称 | 类型 | 说明 |
| --- | --- | --- |
| tse-breaker | 流量控制 | Kong 熔断插件，查看插件说明 。 |
| tse-traffic-mirror | 流量控制 | 流量镜像插件，查看插件说明 。 |
| tse-rate-limiting | 流量控制 | 控制台服务与路由的限流策略能力，查看使用说明 。 |
| tse-proxy-rewrite | 请求转换 | 重写插件，查看插件说明 。 |
| tse-custom-auth | 安全 | 自定义鉴权的能力，查看插件说明 。 |
| tse-cloud-waf | 安全 | 控制台安全防护能力，查看使用说明 。 |
| tse-kafka-log | 可观测 | 上报日志到 Kafka，查看插件说明 。 |
| tse-prometheus | 可观测 | 控制台 Prometheus 监控能力，查看使用说明 。 |
| tse-trace | 可观测 | 控制台链路追踪能力，查看使用说明 。 |
| tse-global-configuration | 全局配置 | 控制台全局配置能力，查看使用说明 。 |
| tse-scf-web | 无服务器 | 控制台转发云函数类型服务能力，查看使用说明 。 |
| tse-scf-event | 无服务器 | |
| tse-route | 路由转发 | 建议使用控制台服务与路由的灰度策略能力，查看使用说明 。 |

### 原生插件

**原生插件**指开源 Kong 网关自带的插件。

| 名称 | 类型 | 说明 |
| --- | --- | --- |
| acl | 安全 | 通过使用任意 ACL 组名称将消费者列入白名单或将其列入黑名单来限制对 Service 或者 Route 的访问。此插件需要在服务或路由上启用身份验证插件。 |
| acme | 认证鉴权 | 允许 Kong 使用来自 Let’s Encrypt 或者其他 ACMEv2 服务的证书，并且支持自动续期。 |

| | | |
|---|---|---|
| aws-lambda | 无服务器 | 从 Kong 调用 AWS Lambda 函数。它可以与其他请求插件结合使用以保护，管理或扩展功能。 |
| azure-functions | 无服务器 | 从 Kong 调用 Azure上的无服务器函数。它可以与其他请求插件结合使用以保护，管理或扩展功能。 |
| basic-auth | 认证鉴权 | 使用用户名和密码保护将基础认证添加到 Service 或 Route 。 |
| bot-detection | 安全 | 保护服务或路由免受大多数常见机器人的攻击，并具有将自定义客户端列入白名单和列入黑名单的能力。 |
| correlation-id | 请求转换 | 使用通过 HTTP header 传输的唯一 ID 关联请求和响应。 |
| cors | 安全 | 通过启用此插件，轻松将跨源资源共享（CORS）添加到 Service, Route |
| datadog | 分析和监控 | 记录 Service 的指标，并路由到本地 Datadog 代理。 |
| file-log | 日志 | 将请求和响应数据写入磁盘上的日志文件中。不建议在生产中使用此插件。 |
| grpc-gateway | 其他 | 通过 HTTP REST 接口公开 gRPC 服务。它以 JSON 格式转换请求和响应，允许通过普通 HTTP 请求访问上游 gRPC 服务。 |
| grpc-web | 其他 | 通过 gRPC-web 协议访问 gRPC 服务。 |
| hmac-auth | 认证鉴权 | 将 HMAC 签名身份验证添加到服务或路由以确定传入请求的完整性。该插件将验证在Proxy-Authorization或Authorization header 中发送的数字签名（按此顺序）。 |
| http-log | 日志 | 将请求和响应数据转发到http服务器。 |
| ip-restriction | 安全 | 通过将 IP 地址列入白名单或列入黑名单来限制对 Service 或 Route 的访问，支持配置单个 IP，多个 IP 或 CIDR 范围，如10.10.10.0/24。 |
| jwt | 认证鉴权 | 验证包含 HS256 或 RS256 签名 JSON Web 令牌的请求。在请求中如果验证了令牌的签名，Kong 会将请求代理到您的上游服务，否则将丢弃该请求。 |
| key-auth | 认证鉴权 | 将密钥身份验证（有时也称为 API 密钥）添加到 Service 或 Route 。 |
| ldap-auth | 认证鉴权 | 使用用户名和密码保护将 LDAP 绑定身份验证添加到路由。 该插件将检查Proxy-Authorization和Authorization header 中的有效凭据（按此顺序）。 |
| loggly | 日志 | 通过 UDP 将请求和响应数据记录到 Loggly。 |
| oauth2 | 认证鉴权 | 使用授权码模式（Authorization Code Grant），客户端凭证模式（Client Credentials），简化授权模式（Implicit Grant）或密码模式（Resource Owner Password Credentials Grant）授予流添加 OAuth 2.0身份验证层。 |
| post-function | 无服务器 | 在访问阶段运行其他插件之后运行一段 Lua 代码。 |
| pre-function | 无服务器 | 在访问阶段运行其他插件之前运行一段 Lua 代码。 |
| prometheus | 分析和监控 | 以 Prometheus exposition 格式公开与 Kong 和代理上游服务相关的指标，Prometheus 服务器可以对这些指标进行抓取。 |
| proxy-cache | 流量控制 | 基于可配置的响应代码和内容类型以及请求方法来缓存响应实体。它可以缓存每个 Consumer 或每个 API。 |
| request-size-limiting | 流量控制 | 限制body大于特定大小（以兆字节为单位）的传入请求。 |
| request-termination | 流量控制 | 使用指定的状态代码和消息终止传入请求。 |

| | | |
|---|---|---|
| request-transformer | 请求转换 | 在将请求转发给上游服务之前，转换客户端发送的请求。 |
| response-ratelimiting | 流量控制 | 根据上游服务返回的自定义响应头限制可以发出的请求数。 |
| response-transformer | 请求转换 | 在将响应返回给客户端之前，转换上游服务发送响应。 |
| session | 认证鉴权 | 管理通过 Kong 代理的 API 的浏览器会话，它为会话数据存储、加密、续订、到期和浏览器 Cookie 提供配置和管理。 |
| statsd | 日志 | 为 Service Route 记录日志指标到一个 StatsD 服务器。 |
| syslog | 日志 | 将请求和响应数据转发到 syslog 服务器。 |
| tcp-log | 日志 | 将请求和响应数据转发到 TCP 服务器。 |
| udp-log | 日志 | 将请求和响应数据转发到 UDP 服务器。 |
| zipkin | 日志 | 传播 Zipkin 分布式跟踪 span，并向 Zipkin 服务器报告 span。 |

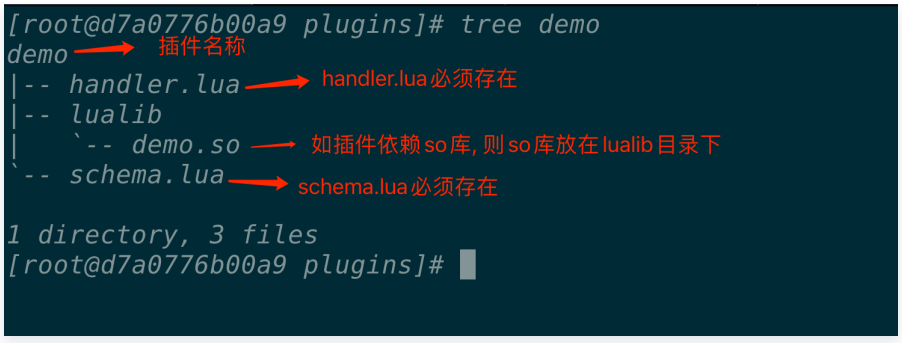
**自定义插件**

云原生 API 网关 Kong **提供自定义插件能力扩展**，您可按照 Kong 规定的数据格式，开发自己的定制插件，以实现对数据流的修改和深层次的集成。详细操作请参见使用自定义插件。

**准备自定义插件代码**

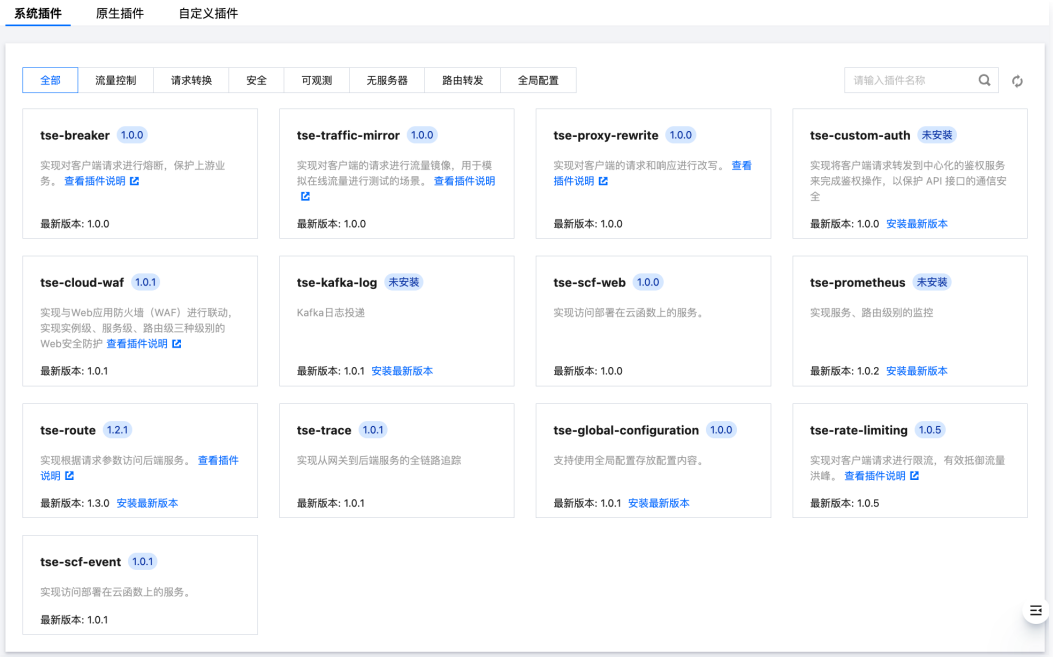
根据 kong lua 插件开发规范：Plugin Development - File Structure - v2.7.x | Kong Docs，自定义插件在保留 Kong 原生插件开发规范的情况下，同时还支持了so 库的能力，建议 so 库在 Centos7 的环境下编译。

插件目录结构可以参照如下：




## 进入插件管理入口

1. 登录 TSF 控制台。
2. 在左侧导航栏单击**云原生 API 网关 > 插件管理**，在页面上方选择好网关实例，可以查看系统插件、原生插件和自定义插件信息。




## 部署自定义插件

1. 参考 **插件介绍-自定义插件** 准备好插件代码。
2. 在插件管理页面，选择**自定义插件**页页签，单击**上传插件**。




3. 选择插件后，填写好插件版本号和版本描述信息，单击**确定**，完成上传。
4. 上传完插件后，单击**切换到此版本**，插件会自动部署在所有的 Kong 节点中。




5. 当您不再需要该插件时，可以单击插件右上角的**卸载**，卸载该插件。

# 弹性伸缩

最近更新时间：2025-06-09 16:07:02

## 操作场景

云原生 API 网关能够根据预先设定的弹性伸缩规则自动增加或减少节点数量，即扩容或缩容，从而提高资源利用率、降低资源成本。弹性伸缩支持指标规则和定时规则两种类型的弹性伸缩策略。（**仅专业版支持**）

- 指标规则：适用于突发流量和典型周期性流量的应用场景。由指标名称、阈值描述弹性扩缩容的触发条件、节点数量变化。
- 定时规则：适用于资源使用率有周期性规律的应用场景。达到设定的周期时间点后，TSF 云原生 API 网关将依据设定的策略自动调整实例节点数量。

## 前提条件

已创建云原生 API 网关实例。

## 操作步骤

### 步骤1：创建弹性伸缩策略

1. 登录 TSF 控制台。
2. 在左侧导航栏选择**云原生 API 网关** > **实例列表**，单击目标实例的“ID”，进入网关实例基本信息页面。
3. 在**基础信息**页选择**弹性伸缩**页签，单击**新建策略**。




4. 在新建弹性伸缩弹窗，填写策略信息：

**仅指标伸缩**

- 节点数范围：弹性伸缩的节点数范围，最少为分组当前节点数，最大为20个。
- 指标伸缩：满足一个条件即触发弹性伸缩。
  - 指标名称：弹性伸缩触发条件，支持为监控指标（CPU 使用率、内存使用率、QPS）配置阈值。
  - 阈值：触发弹性伸缩指标到达的阈值。
- 策略名称：输入弹性伸缩策略的名称。
- 描述：输入弹性伸缩策略的描述信息。




## 仅定时伸缩

- **节点数范围：弹性伸缩的节点数范围，最少为分组当前节点数，最大为20个。**
- **定时伸缩：如果多个规则触发时间相同，将按序执行，您可拖动调整顺序；若触发时间早于当前时间，则从下一个时间周期开始生效。**
  - **周期：执行定时弹性伸缩策略的周期。**
  - **节点数：弹性扩缩容生效后的节点数量。**
- **策略名称：输入弹性伸缩策略的名称。**
- **描述：输入弹性伸缩策略的描述信息。**

版权所有：腾讯云计算（北京）有限责任公司




### 指标伸缩与定时伸缩

- **节点数范围：弹性伸缩的节点数范围，最少为分组当前节点数，最大为20个。**
- **定时伸缩：如果多个规则触发时间相同，将按序执行，您可拖动调整顺序；若触发时间早于当前时间，则从下一个时间周期开始生效。**
  - **周期：执行定时弹性伸缩策略的周期。**
  - **节点数：弹性扩缩容生效后的节点数量。**
- **指标伸缩：满足一个条件即触发弹性伸缩。**
  - **指标名称：弹性伸缩触发条件，支持为监控指标（CPU 使用率、内存使用率）配置阈值。**
  - **阈值：触发弹性伸缩指标到达的阈值，**
- **策略名称：输入弹性伸缩策略的名称。**
- **描述：输入弹性伸缩策略的描述信息。**

版权所有：腾讯云计算（北京）有限责任公司

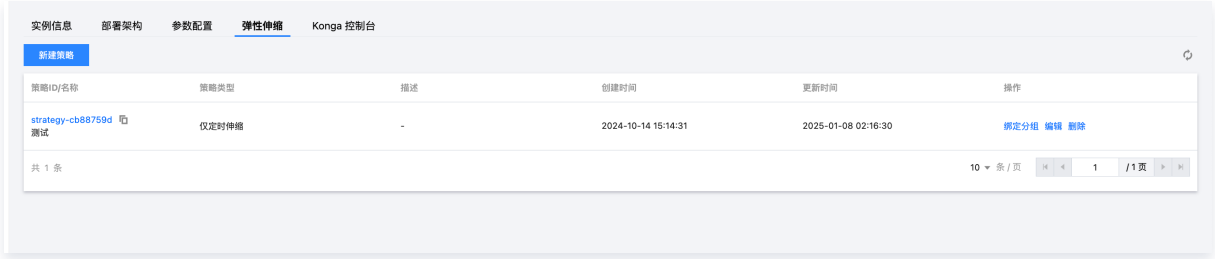



5. 单击**确定**，弹窗提示需绑定集群分组后才能生效，您可直接前往绑定分组，或稍后绑定。

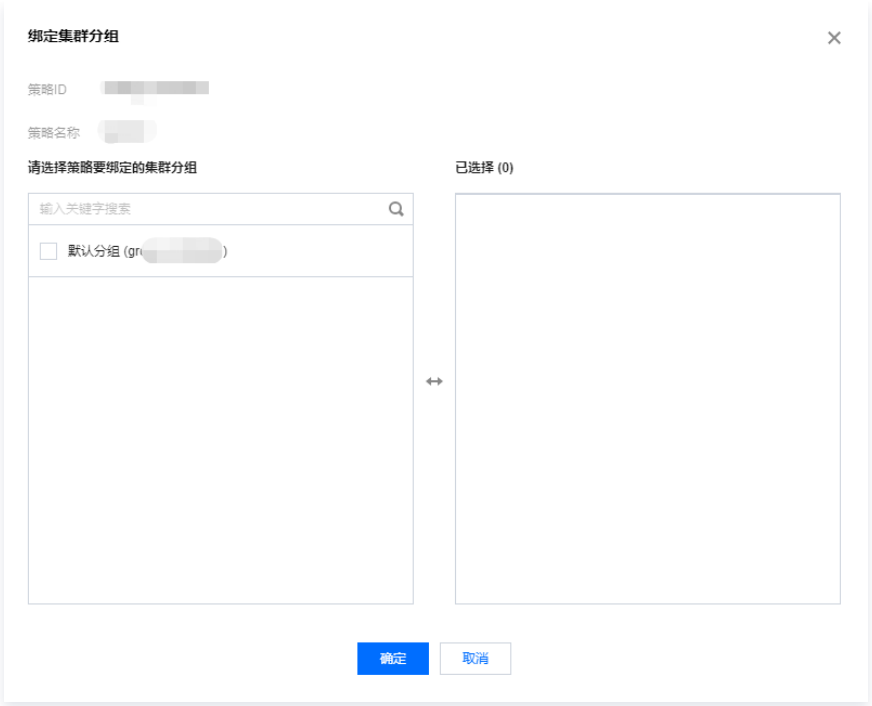
## 步骤2：绑定集群分组并生效

弹性伸缩策略需绑定集群分组后才能生效，支持一次绑定一个或多个集群分组。

1. 在弹性伸缩策略列表页， 单击目标策略右侧**绑定分组**。

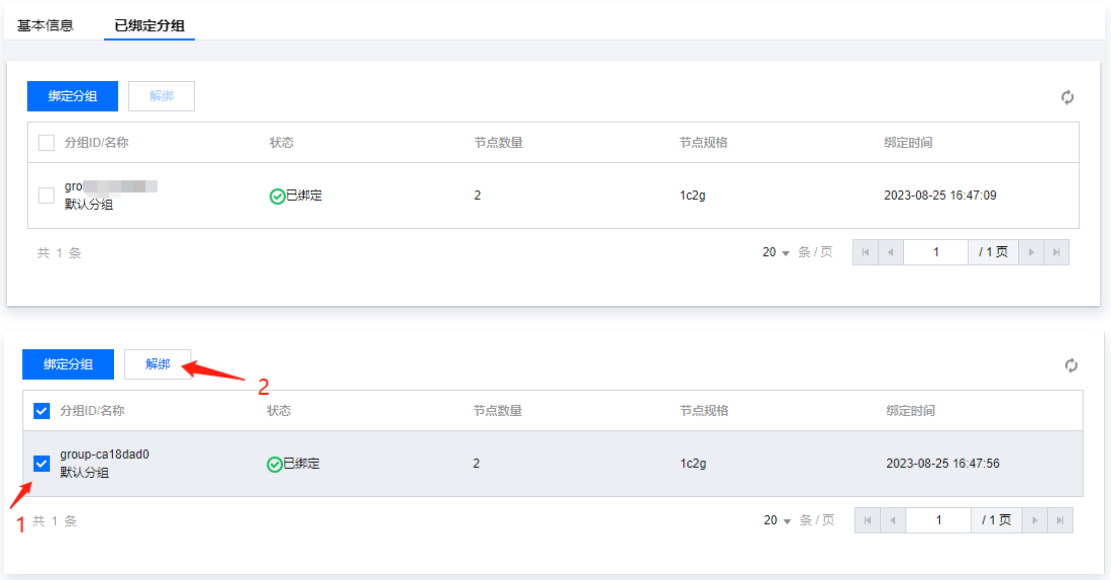



2. 在绑定集群分组弹窗中，选择一个或多个集群分组，单击**确定**，绑定后策略立即生效。




## 步骤3：查看策略已绑定的集群分组

1. 在弹性伸缩策略列表页，单击目标策略，进入策略详情页。
2. 单击已绑定分组，查看该策略已绑定的集群分组，您可在此绑定新的分组或解绑已有分组。

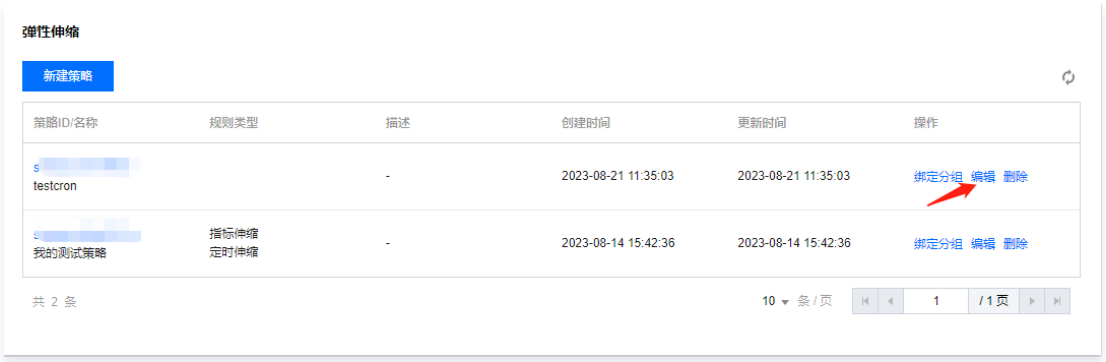






## 步骤4：编辑弹性伸缩策略

1. 在弹性伸缩策略列表页， 单击目标策略右侧**编辑**。




2. 在 **编辑弹性伸缩策略** 页面，修改策略触发条件、弹性伸缩范围、策略名称或描述。




3. 单击 **确定** ，完成编辑。

## 步骤5：删除弹性伸缩策略

1. 在弹性伸缩策略列表页， 单击目标策略右侧**删除**。



2. 在 **删除策略** 弹窗，二次确认是否要删除该策略。




3. 单击**确定**，删除策略。

> **注意：**
> 删除后，该策略立即失效，自动与之前绑定的分组解绑，请谨慎操作。

## 同时配置指标伸缩与定时伸缩

定时伸缩通过设置定时的方式触发网关节点的弹性伸缩，常用在周期性任务中。指标伸缩通过配置关键指标的阈值进行节点弹性伸缩，常用于防止突发的流量冲击等状况。在实际使用中，可结合两种方式进行配置，当云原生 API 网关感知到两种策略同时配置时，会结合两种策略内容进行节点数量调整。

当两种策略同时生效时，定时伸缩并不直接进行节点数调整，而是通过调整指标伸缩节点数范围的方式来进行伸缩判断与调整，调整原则为：

- 当前节点数 = 定时伸缩目标节点数，不进行调整。
- 当前节点数 < 定时伸缩目标节点数：
  - 定时伸缩目标节点数 > 指标伸缩最小节点数，调整指标伸缩最小节点数为定时弹性伸缩目标节点数。
  - 定时伸缩目标节点数 < 指标伸缩最大节点数，指标伸缩最大节点数不调整。
- 当前节点数 > 定时伸缩目标节点数：
  - 定时伸缩目标节点数 > 指标伸缩最小节点数，指标伸缩最小节点数不调整。
  - 定时伸缩目标节点数 < 指标伸缩最大节点数，指标伸缩最大节点数不调整。

节点数会根据指标伸缩节点数范围进行相应的调整，以实现节点弹性伸缩。

下表列出了当两种策略同时生效时，不同场景下节点扩缩容结果：

| 指标伸缩节点数（调整前） | 定时伸缩目标节点数 | 当前节点数 | 指标伸缩节点数（调整后） | 节点数（调整后） |
| --- | --- | --- | --- | --- |
| 最小：1　最大：10 | 5 | 5 | 最小：1　最大：10 | 5 |
| 最小：1　最大：10 | 7 | 5 | 最小：7　最大：10 | 7 |
| 最小：1　最大：10 | 3 | 5 | 最小：1　最大：10 | 5 |

# 事件中心

最近更新时间：2025-06-09 16:07:02

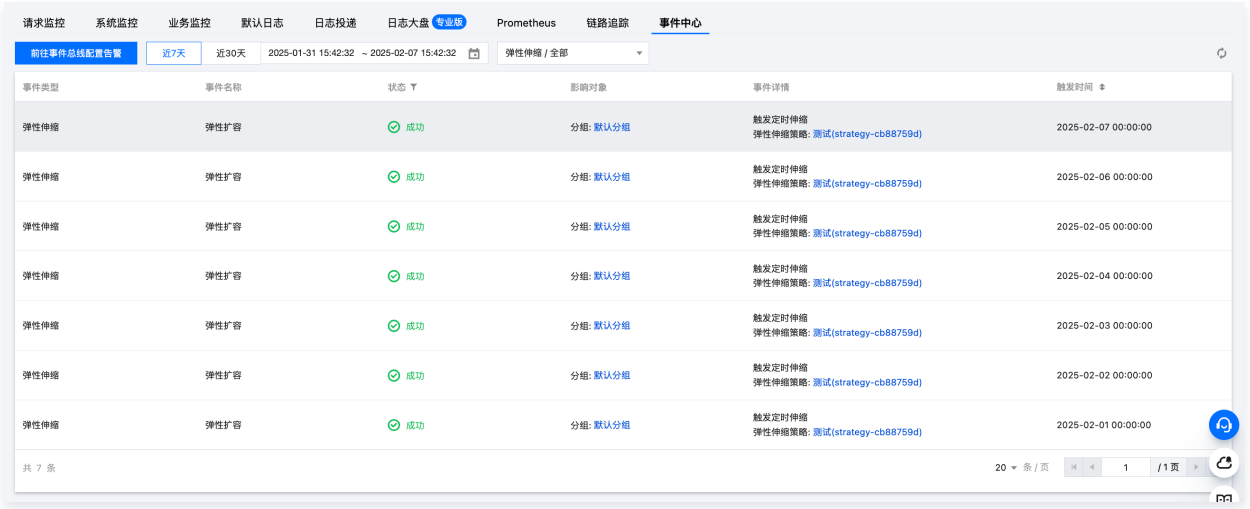
**事件中心将云原生 API 网关生成的事件信息进行统一管理、存储、分析和展示，您可以在事件中心查看事件详情，并为事件配置告警通知，以便及时发现问题并进行处理。**
**云原生 API 网关当前支持的事件有：**

| 事件类型 | 事件名称 | 事件详情 |
| --- | --- | --- |
| 弹性伸缩 | 弹性扩容 | 触发的策略类型、策略 ID(名称)、变更节点 IP |
| 弹性伸缩 | 弹性缩容 | 触发的策略类型、策略 ID(名称)、变更节点 IP |
| 弹性伸缩 | 绑定弹性伸缩策略 | 触发的策略类型、策略 ID(名称) |
| 弹性伸缩 | 解绑弹性伸缩策略 | 触发的策略类型、策略 ID(名称) |

## 查看事件详情

1. 登录 微服务平台控制台 ，在左侧导航栏单击**云原生 API 网关 > 数据观测**。
2. 在可观测性页面，进入**事件中心** Tab。
3. 在页面上方设置时间范围（支持近 7 天、近 30 天和自定义时间范围）。

| 字段 | 描述 |
| --- | --- |
| 事件类型 | 支持弹性伸缩。 |
| 事件名称 | 事件信息的概要描述，如弹性扩容、弹性缩容、绑定弹性伸缩策略、解绑弹性伸缩策略。 |
| 状态 | 该事件是否执行成功，支持成功、失败两种。 |
| 影响对象 | 弹性伸缩事件类型：分组 ID。 |
| 事件详情 | 事件明细信息，例如事件摘要、涉及的对象和事件内容等。 |
| 触发时间 | 事件触发时间，精确到秒。 |

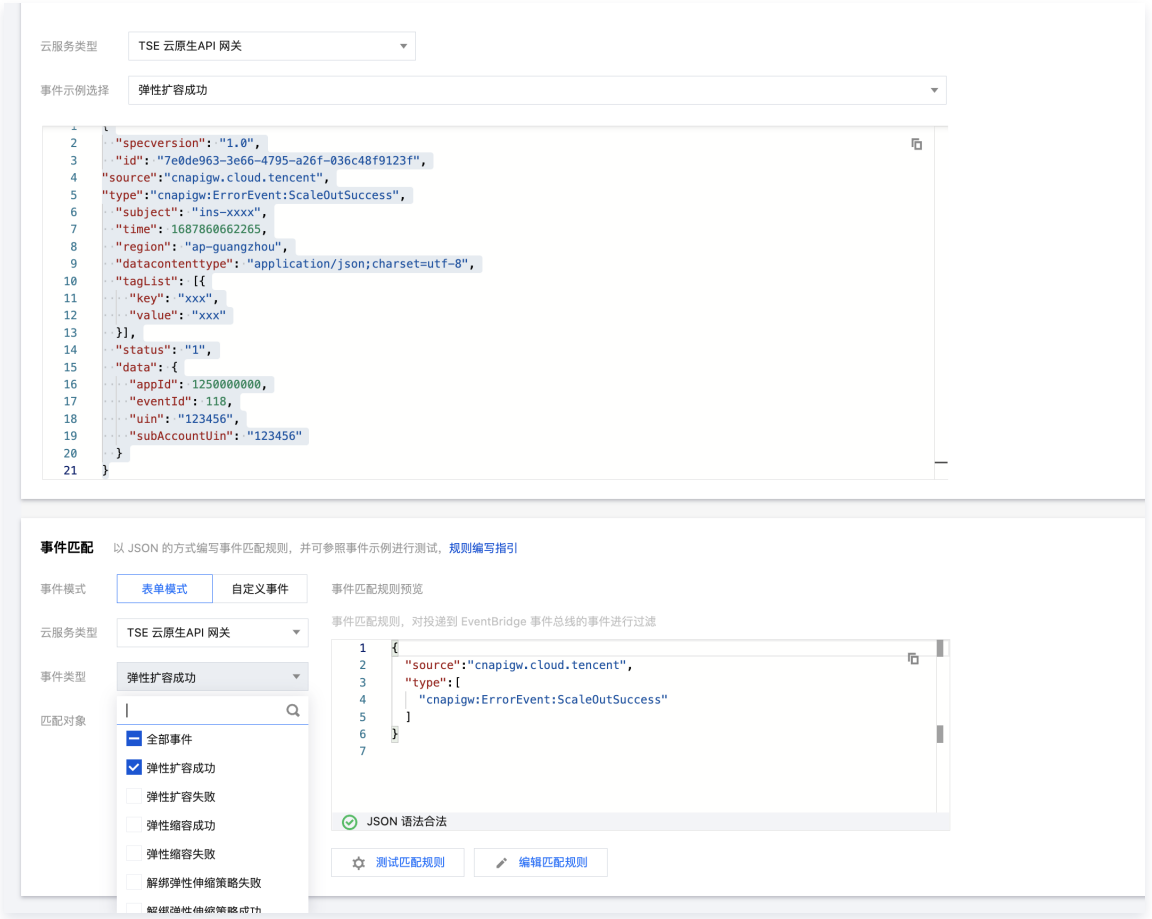



## 配置事件告警规则

1. 在**事件中心**页面，单击左上角的**前往事件总线配置告警**跳转至事件规则配置页面。
2. 单击**新建事件规则**，配置事件模式信息。

- 规则名称：填写事件规则名称，规则名称一旦创建无法修改。
- 事件示例：选择**默认模板**。
- 云服务类型：选择 **TSF 云原生 API 网关**。
- 事件类型：勾选您要配置告警规则的事件类型。

3. 单击**下一步**，填写事件目标信息。
   - 触发方式：选择**消息推送**。
   - 消息模板：根据事件类型，提供两种不同类型推送模板，您可以根据实际情况选择，对于告警事件，建议统一选择告警通知模板。
   - 用户通知：根据您的需要自定义选择通知方式和告警接收对象。
4. 单击**完成**，完成事件告警规则配置。



# 故障演练

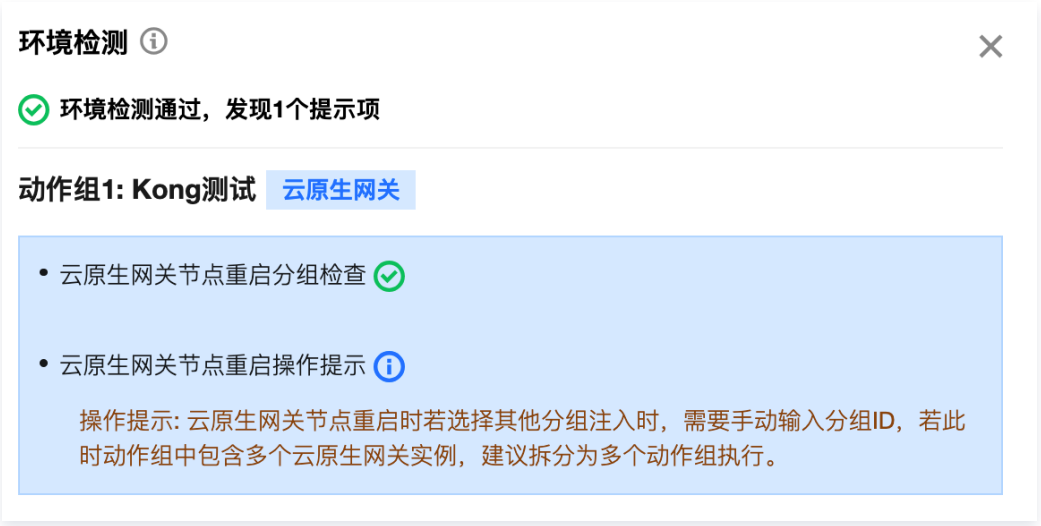
最近更新时间：2025-06-09 16:07:02

## 背景

云原生 Kong 网关是一个常用的 API 网关，用于管理和保护后端服务。为保证您的业务能够持续提供服务，云原生网关（Kong）分组（标准版、专业版）提供节点同城双可用区部署，可以在某些特殊场景下导致的可用区故障或节点临时宕机等非预期情况时，保护您的应用程序不受影响。为此混沌演练平台提供了针对云原生网关（Kong）分组节点重启的故障演练动作，您可使用混沌演练平台进行该故障演练，模拟可用区故障或节点临时宕机离线的故障场景，以此验证业务系统韧性，及时规避隐患，从而保证业务能够持续稳定地提供服务。

## 注意事项

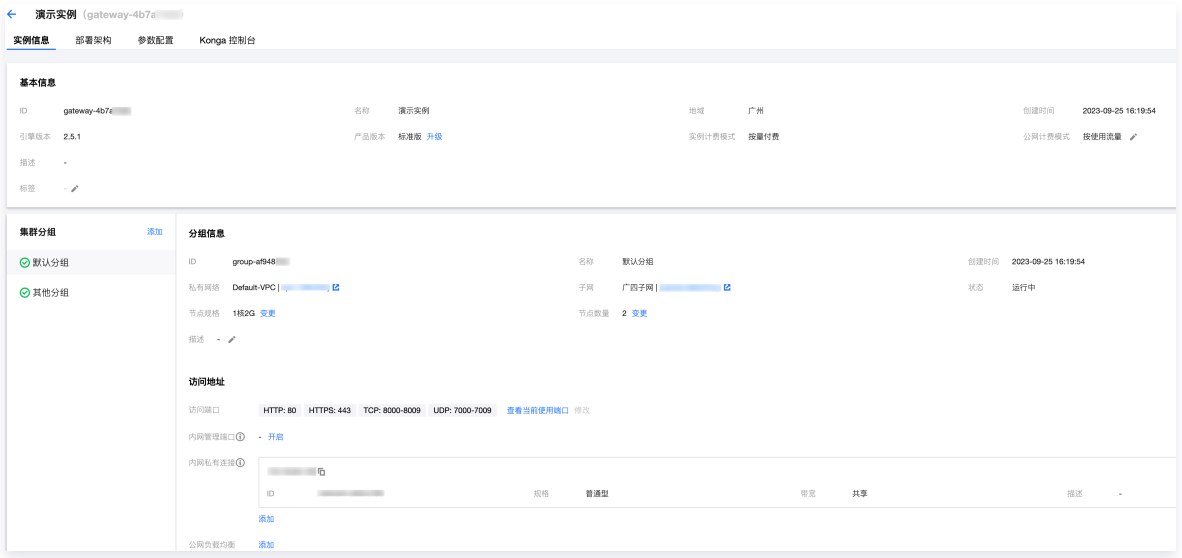
- 该故障动作支持按照**默认分组**以及**其他分组**（需指定分组 ID）进行故障注入。当选择其他分组指定分组时，如果需要对多个网关实例注入故障，建议拆分为一个动作组一个网关实例执行。
- 如果待注入的分组中的**节点数量小于2**，则会拒绝注入节点重启故障。
- 故障执行会从分组下的所有节点中**随机**选择一个节点进行重启，重启完成时间预计**3～5分钟**，在故障期间，可能会出现连接到重启节点的**长连接断开**，请业务做好重试连接机制。
- 环境检测会检查网关实例的分组状态、分组节点数量、注入分组类型等检查，若不通过，请按照提示进行调整或检查故障配置。




## 演练实施

### 步骤1：演练准备

1. 一个云原生 API 网关实例，且至少包含一个具有**2个节点**及以上的分组。

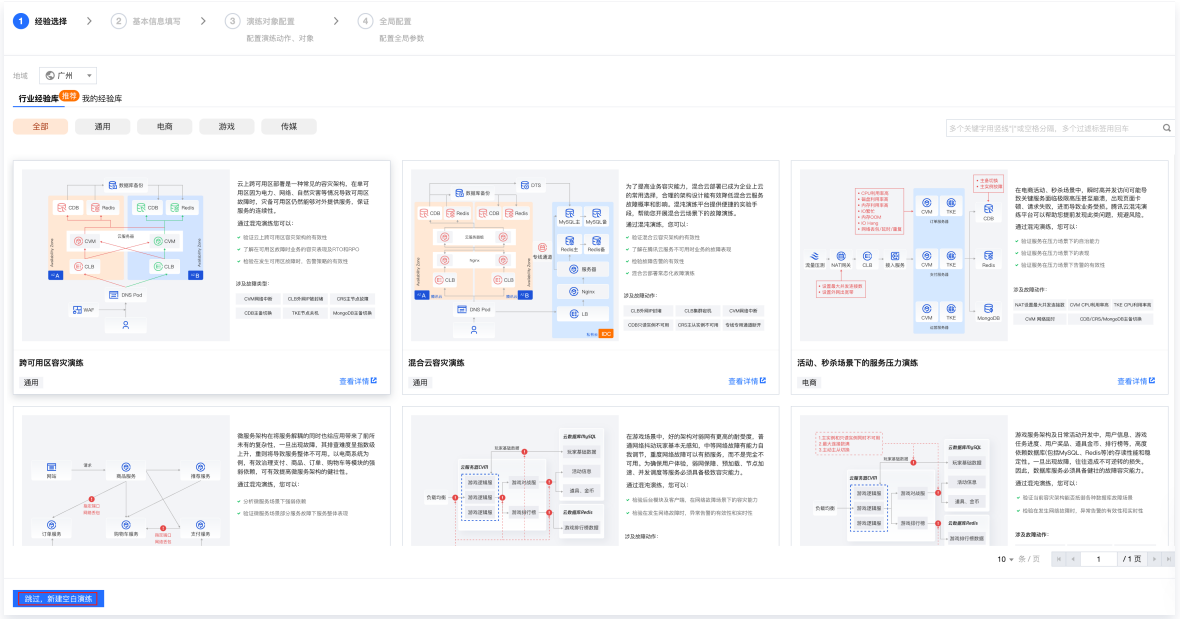



## 步骤2：演练编排

1. 登录 混沌演练控制台，进入**演练管理**页面，单击**新建演练**。




2. 单击左下角**跳过**，**新建空白演练**。



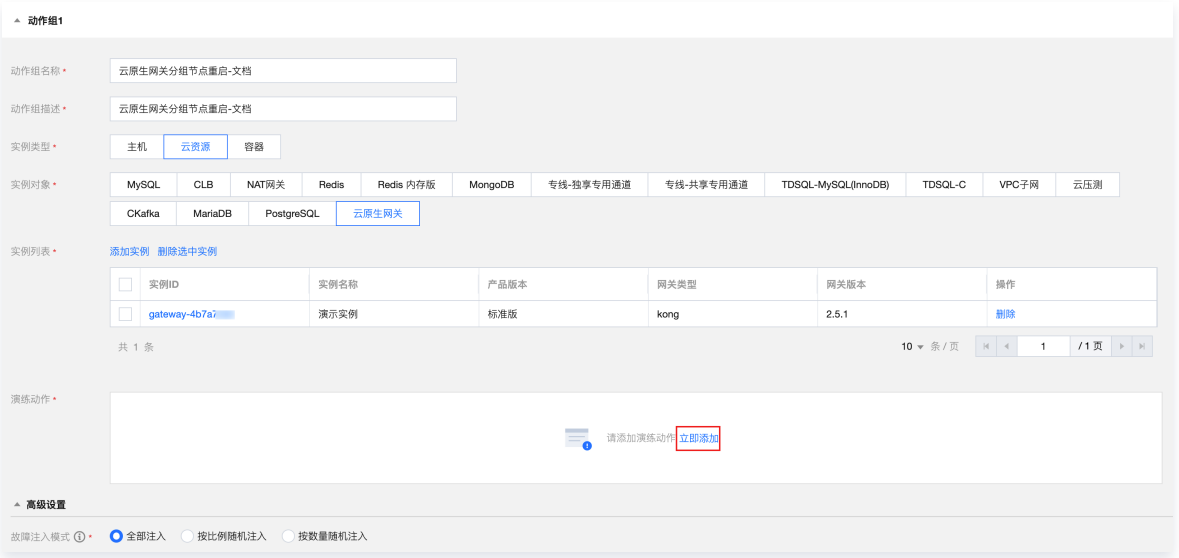

3. 填写演练基本信息，然后单击**下一步**。




4. 选择对象类型**云原生网关**，然后单击**添加实例**，添加需要进行演练的实例。



5. 选择实例之后，单击演练动作模块中**立即添加**。




6. 添加**网关节点重启(分组)**故障动作之后单击**下一步**。

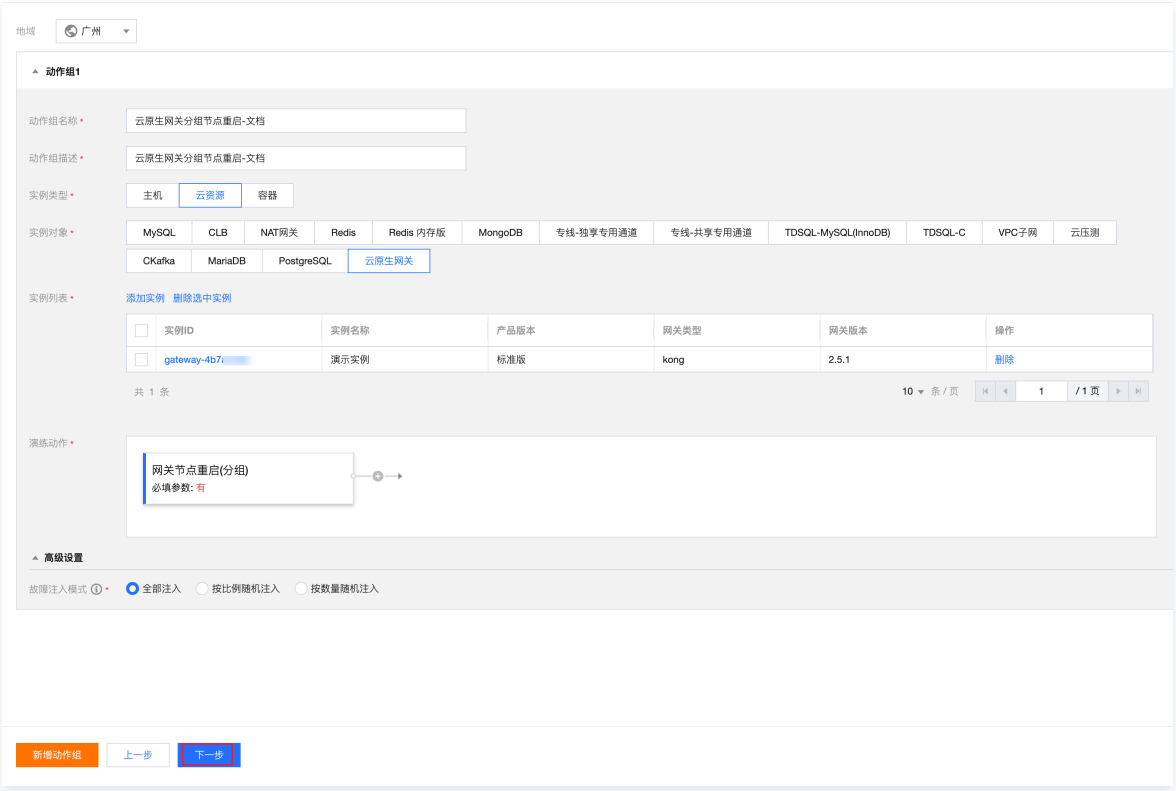



7. 配置动作参数。这里选择**默认分组**进行注入，然后单击**确认**。

> **注意：**
> 使用其他分组时，需要手动输入分组 ID。




8. 单击**下一步**，进入全局配置，详情请参见 **快速入门**。

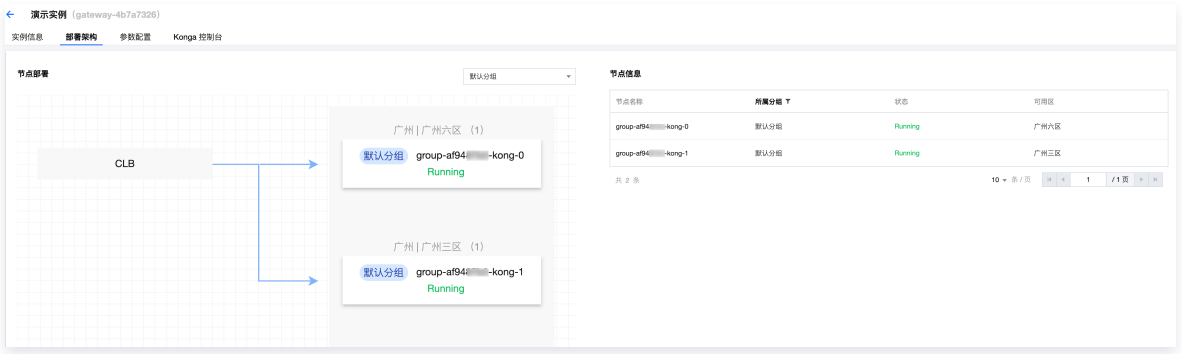



9. 确认无误后，单击**提交**。

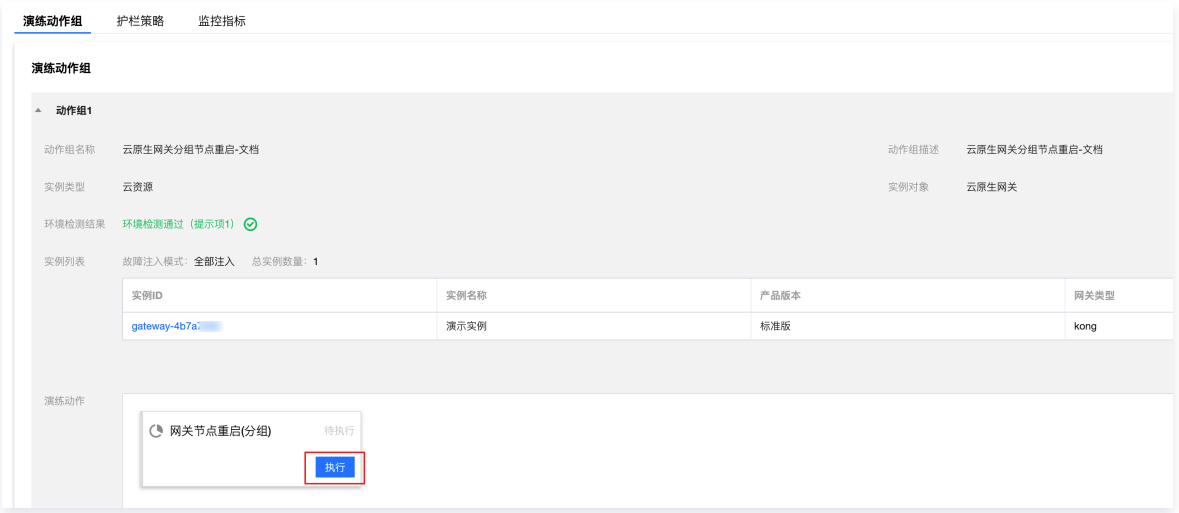



## 步骤3：执行演练

1. 可在 腾讯云微服务平台控制台 观察演练前实例分组节点信息数据。可知晓，演练前实例默认分组具有2个节点，分别位于广州六区和广州三区。



2. 由于演练为手动推进，所以需要手动执行故障动作。可单击**执行**启动故障注入。




3. 故障注入过程中，可在 腾讯云微服务平台控制台 对应实例的**部署架构**模块中观察到实例的对应分组节点的状态。

**故障中：**默认分组下广州三区的节点因重启下线 。




**故障完成：**默认分组下广州三区的节点因重新上线。




4. 单击**查看日志**查看故障演练日志。



版权所有：腾讯云计算（北京）有限责任公司

# 迁移指南
# 从自建 Kong 网关迁移到腾讯云

最近更新时间：2025-06-09 16:07:02

> **说明：**
> 目前云原生 API 网关仅对微服务场景（对接 TSF Nacos，TSF 北极星，TSF）支持开通白名单新购，其余场景**不支持新接入**。如有新购需求，请 提交工单 处理。

## 操作场景

本文介绍利用 Kong 官方的 **decK** 工具来完成自建 Kong 网关到腾讯云云原生 Kong 网关的配置迁移。

## 前置条件

- 已购买 Kong 网关实例，操作文档，并 配置 admin-api。
- 有自建的 Kong 网关，并配置了 **services**、**routes**、**plugins**。
- 终端机安装了 **decK** 工具，请前往 Kong 官方下载 decK 并安装到您的终端。

## 操作步骤

### 步骤1：利用 decK 工具导出自建的 Kong 网关的配置

1. 打开终端，运行如下命令导出自建 Kong 网关的配置。

```
# 注意将 kong-zj 替换为自建kong网关的ip地址或域名
deck --kong-addr http://kong-zj:8001/ dump
```

2. 检查生成的 `kong.yaml` 文件。

```
# 注意kong.yaml内容与实际情况会有所不同
cat kong.yaml
```

```
_format_version: "1.1"
plugins:
- config:
    per_consumer: false
  enabled: true
  name: prometheus
  protocols:
  - grpc
  - grpcs
  - http
  - https
services:
- connect_timeout: 60000
  host: 127.0.0.1
  name: admin-api
  path: /
  port: 8001
  protocol: http
  read_timeout: 60000
```

```
  retries: 5
  routes:
  - https_redirect_status_code: 426
    name: admin-api
    path_handling: v1
    paths:
    - /admin-api
    preserve_host: false
    protocols:
    - http
    - https
    regex_priority: 0
    request_buffering: true
    response_buffering: true
    strip_path: true
  write_timeout: 60000
- connect_timeout: 60000
  host: www.tencent.com
  name: tencent
  path: /
  port: 80
  protocol: http
  read_timeout: 60000
  retries: 5
  routes:
  - https_redirect_status_code: 426
    id: 040b0c62-4d31-4286-a595-1832e55bf568
    path_handling: v1
    paths:
    - /
    preserve_host: false
    protocols:
    - http
    - https
    regex_priority: 0
    request_buffering: true
    response_buffering: true
    strip_path: true
  write_timeout: 60000
```

## 步骤2：利用 decK 工具导入配置到云原生 API 网关 Kong

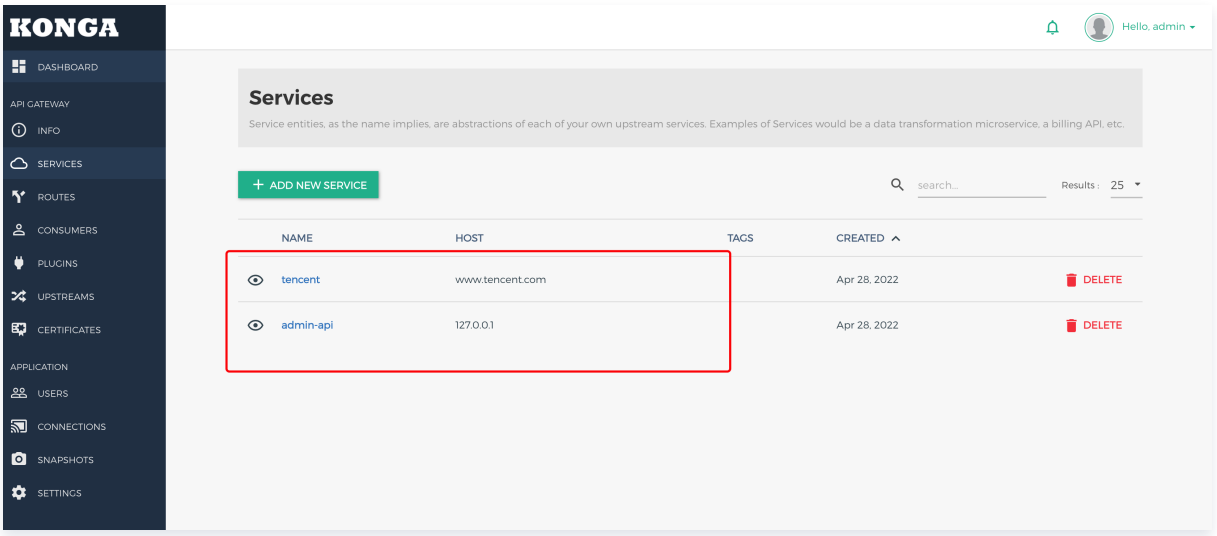
1. 打开终端，运行如下命令查看 `Kong.yaml` 中的配置与云原生 Kong 网关的配置差异性。

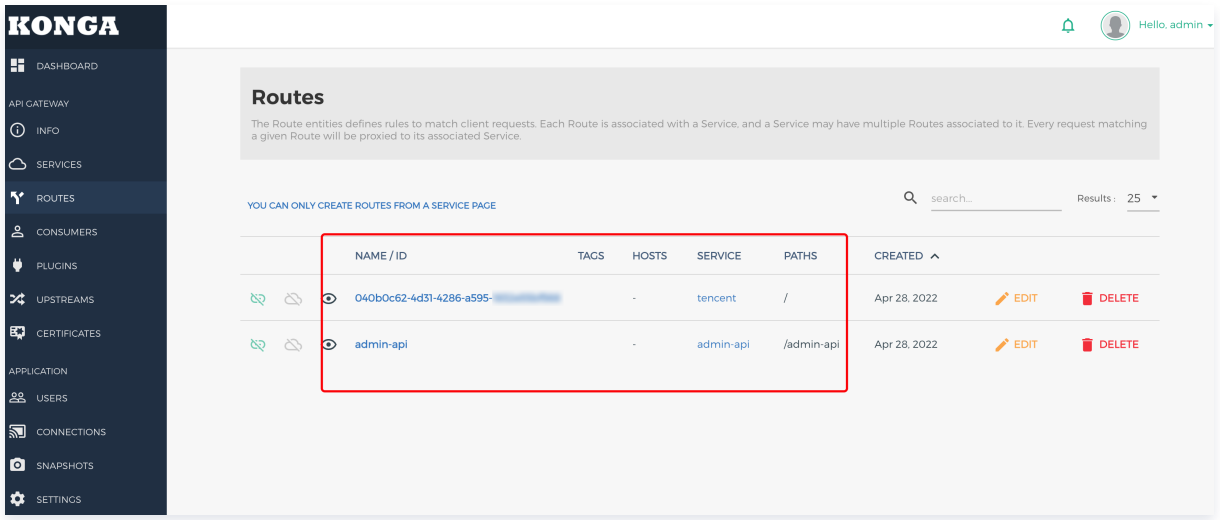
```
# 注意将 kong-tencent 替换为腾讯云云原生 API 网关Kong的代理地址
deck --kong-addr http://kong-tencent/admin-api diff
creating service tencent
creating route 040b0c62-4d31-4286-a595-1832e55bf568
creating plugin prometheus (global)
Summary:
  Created: 3
  Updated: 0
  Deleted: 0
```

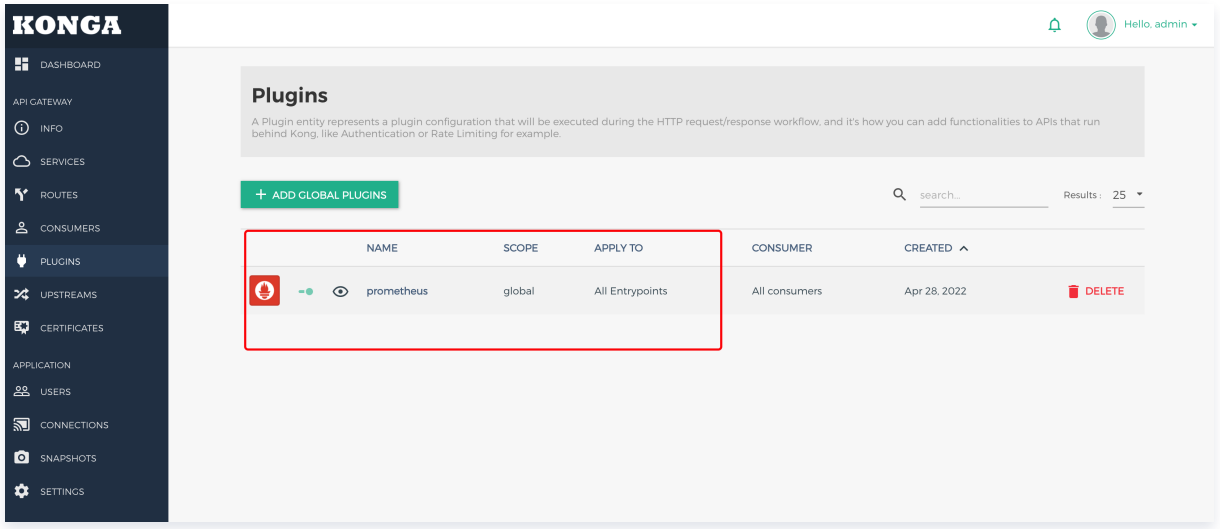
2. 运行如下命令将 `Kong.yaml` 中的配置导入腾讯云云原生 Kong 网关中。

```
# 注意 kong-tencent 替换为腾讯云云原生 API 网关Kong的代理地址
deck --kong-addr http://kong-tencent/admin-api sync
creating service tencent
creating route 040b0c62-4d31-4286-a595-1832e55bf568
creating plugin prometheus (global)
Summary:
  Created: 3
  Updated: 0
  Deleted: 0
```

3. 打开腾讯云云原生 Kong 网关的控制台查看配置是否导入成功。






## 注意事项

为了操作方便 admin-api 并未开启安全认证插件，强烈建议配置同步完成后为 admin-api 开启安全认证插件。

## 参考

更多相关说明请参见 decK 参考文档。

# 从 Spring Cloud Gateway 迁移到云原生 API 网关

最近更新时间：2025-06-09 16:07:02

本文介绍如何将服务从Spring Cloud Gateway迁移到云原生 API 网关。

## 步骤1：明确服务来源

如果您当前的服务来源为以下几种，可以直接迁移 SCG 配置。

- K8S 服务（使用 TKE 或 EKS）。
- 注册中心服务（使用 TSF 或自建的 Nacos/Consul/Polaris）。
- 未依赖任何服务发现机制，使用域名/固定 IP/IP列表。

如果您当前的服务来源为其他类型的注册中心，可以先执行相应的迁移操作将服务注册到 Nacos/Consul/Polaris，再进行 SCG 配置迁移。

## 步骤2：迁移 Spring Cloud Gateway 配置

SCG 相关功能配置可以在控制台进行相应操作完成：

1. **注册中心/K8S 集群**

云原生 API 网关通过服务来源关联用户的注册中心或K8s集群，添加对应类型的服务来源后，即可添加相应服务作为网关服务，可参考如下文档进行服务来源配置：服务来源 。

2. **服务配置**

Spring Cloud Gateway的 service 对应云原生 API 网关的服务，可参考如下文档进行服务配置：服务与路由 。

3. **路由配置**

Spring Cloud Gateway的 routes 对应云原生 API 网关的路由，可参考如下文档进行服务配置：服务与路由 。

4. **常用的 Predicate 配置**

- Host/Method/Path Route Predicate：云原生 API 网关支持根据Host，Method或Path进行路由匹配，可在创建路由时设置匹配条件。
- Header/Cookie/Query Route Predicate：云原生 API 网关支持根据上述参数进行路由，可通过灰度规则实现，可参考以下文档进行配置：灰度发布 。

5. **默认Filter**

Spring Cloud Gateway的 filter 对应云原生 API 网关的配置如下：

| GatewayFilter | 对应的云原生API网关能力 | 相关文档 |
| --- | --- | --- |
| AddRequestHeader, RemoveRequestHeader, AddRequestParameter, AddResponseHeader, SetResponseHeader | 使用 tse-proxy-rewrite 插件 | 使用重写插件改写请求或响应 |
| Hystrix, RequestRateLimiter | 限流能力 | 多维度、精细化限流 |
| PrefixPath | 默认根据前缀匹配方式进行路由匹配 | - |
| PreserveHostHeader, StripPath, Retry | 创建路由时支持保持 Host、StripPath 选项、支持设置重试次数 | 服务与路由 |
| SetPath, RewritePath | 使用 Request Transformer/Response Transformer 插件 | - |

6. **自定义Filter**

6.1 认证鉴权

云原生 API 网关支持主流认证鉴权方式，可参考以下文档进行配置。

- Basic Auth安全认证
- IP访问控制
- Key Auth 密钥认证
- 使用HMACAuth 认证访问
- 使用 ACL 访问控制
- 配置 JWT 认证鉴权

6.2 自定义Filter

云原生 API 网关支持使用自定义插件进行自定义逻辑处理，您需要编码将自定义Filter的逻辑通过自定义插件实现。详细可参见 自定义插件 。

## 步骤3：配置网关监控告警

云原生 API 网关支持查看监控告警，您可以配置相关监控信息掌握当前网关状况，请参考以下文档使用监控告警。

- 查看默认监控
- 使用 Prometheus 监控

## 步骤4：流量迁移

在完成上述准备过程后，可按照灰度方式逐步进行流量切换，最终完成流量迁移。
第一阶段：选取部分非核心业务场景配置服务和路由，修改访问地址，进行小规模验证。
第二阶段：逐步扩展至核心业务，对核心业务进行配置变更，并迁移部分核心场景流量。
第三阶段：待充分验证后，切换域名解析进行全量迁移。

# 实践教程
# 认证鉴权
# 开启 admin-api 并配置安全认证

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍 Kong 如何开启 admin-api 并为 admin-api 配置 Basic Auth 安全认证插件。

## 前置条件

已购买 Kong 网关实例，详情请参见 操作文档 。

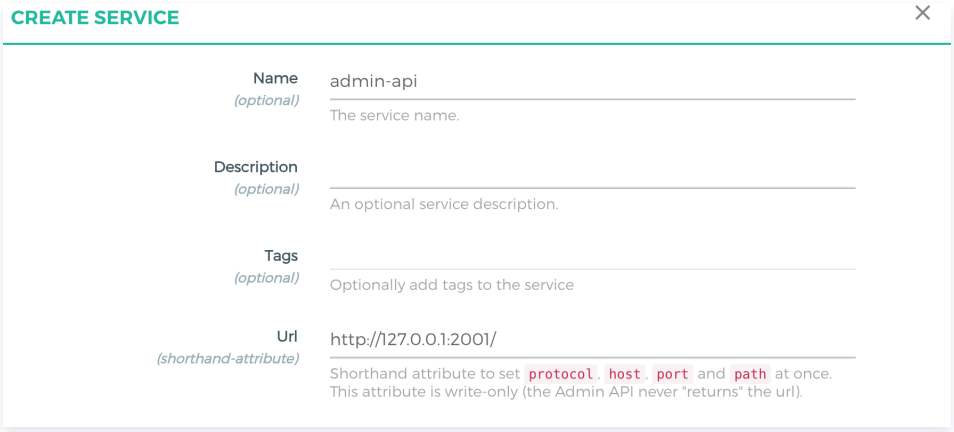
## 操作步骤

### 步骤1：在 Kong 的管理控制台上配置 admin api 的服务和路由

1. 登录 TSF 控制台，进入需要配置限流插件的 Kong 网关实例详情页，在 Konga 控制台页面查看管理控制台登录方式。

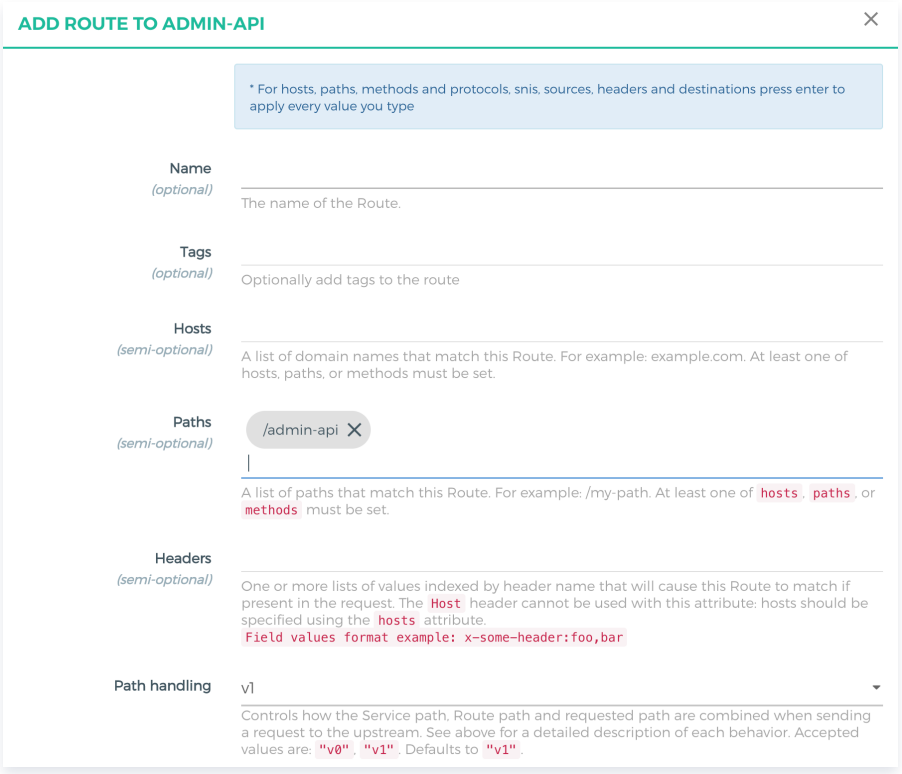



2. 单击**访问链接**进入**管理控制台**页面，在左侧导航栏单击 **SERVICES**，进入服务列表。
3. 单击 **ADD NEW SERVICE**，在弹出的对话框中填写内容，URL 的内容填写为 `http://127.0.0.1:2001/` ，创建一个服务。




4. 创建成功后，在服务列表中单击已创建好的服务，进入服务详情页面，在左侧导航栏单击 **Routes**，进入路由列表。

5. 单击 **ADD ROUTE**，在弹出的对话框中填写内容，**Path** 的内容填写为 `/admin-api` ，创建一个路由。



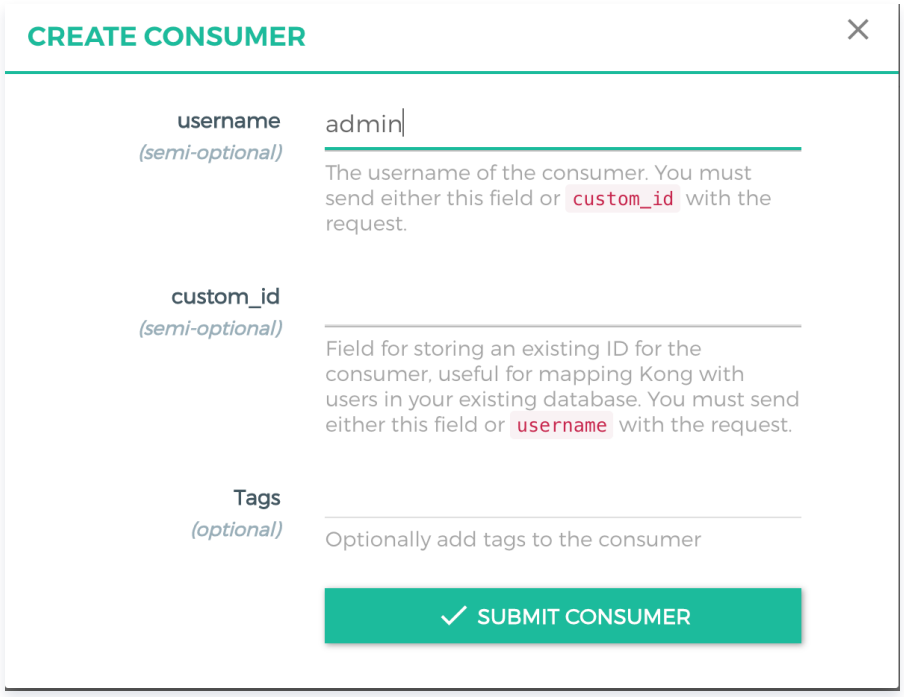

6. 打开浏览器通过公网地址访问 `admin api` ， `uri` 的内容填写为 `/admin-api` 。



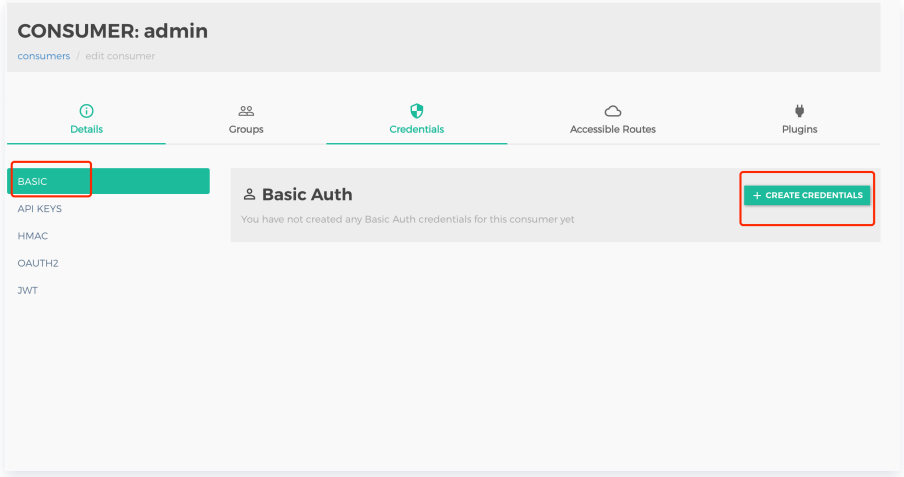

## 步骤2：为 admin api 配置 basic-auth 安全认证插件

1. 打开 Kong 的管理控制台，在左侧导航栏单击 **CONSUMERS**，进入消费者列表。

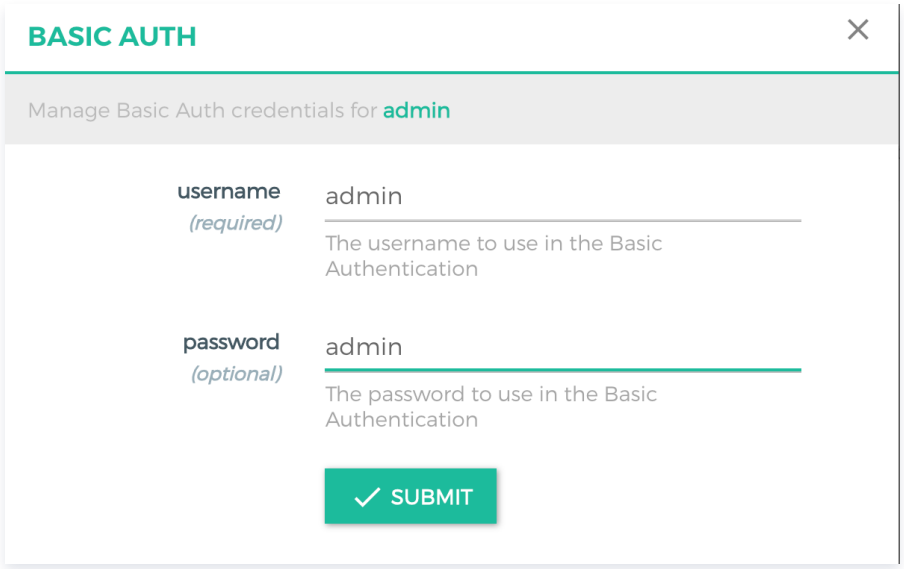
2. 单击 **CREATE CONSUMER**，在弹出的对话框中填写内容。



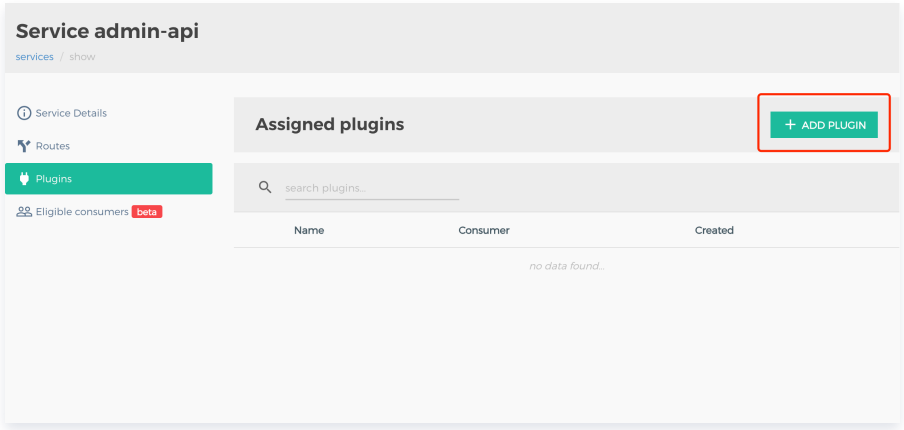

3. 单击 **Credentials** 左侧的选项选择 BASIC，并单击 **CREATE CREDENTIALS**。

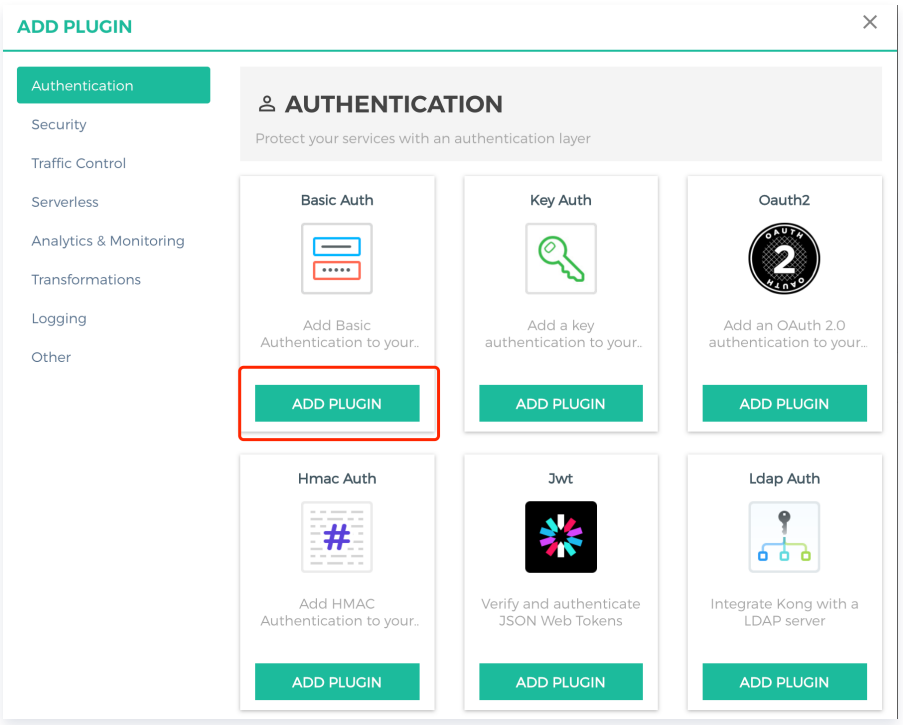


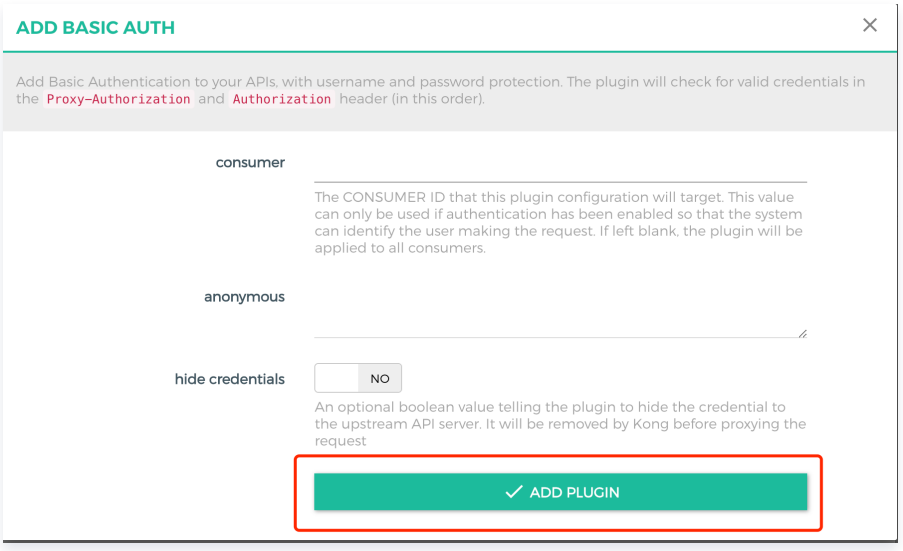
在弹出的对话框中填写内容，创建一个消费者。



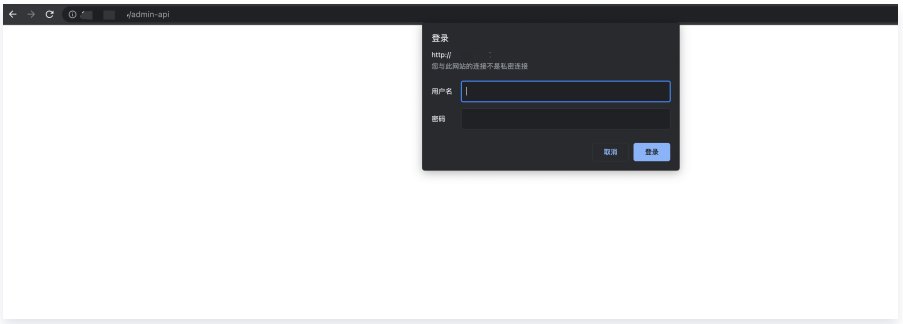

4. 在左侧导航栏中依次单击 **SERVICES** > **admin-api** > **Plugins** > **ADD Plugin**，选择 **Basic Auth** 插件并完成添加。






5. 打开浏览器通过公网地址访问 admin api，`uri` 的内容填写为 `/admin-api`，输入之前配置好的用户名和密码即可访问。



## 参考

更多说明请参见 Kong API Lookback 。

# 使用 IP 访问控制

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何在 Kong 云原生 API 网关上通过 Kong IP Restriction 插件实现 IP 访问控制，包括以下两种场景：

- 配置 IP 黑/白名单列表进行访问控制
- 配置 IP 黑/白名单 CIDR 段进行访问控制

## 前置条件

- 已购买 Kong 网关实例，操作文档。
- 配置了后端（Service）以及路由（Route）。

## 操作步骤

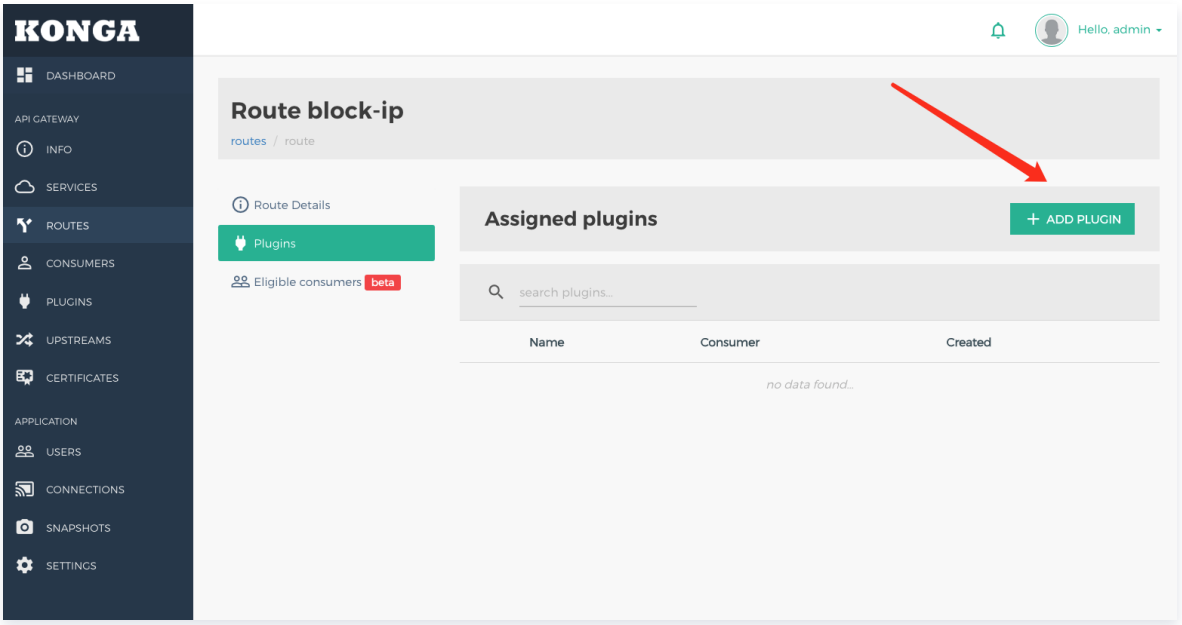
### 场景一：配置 IP 黑/白名单列表进行访问控制

本场景操作步骤以黑名单 IP 列表为例，指导如何拒绝某个（些）IP 访问，白名单配置与之类似。

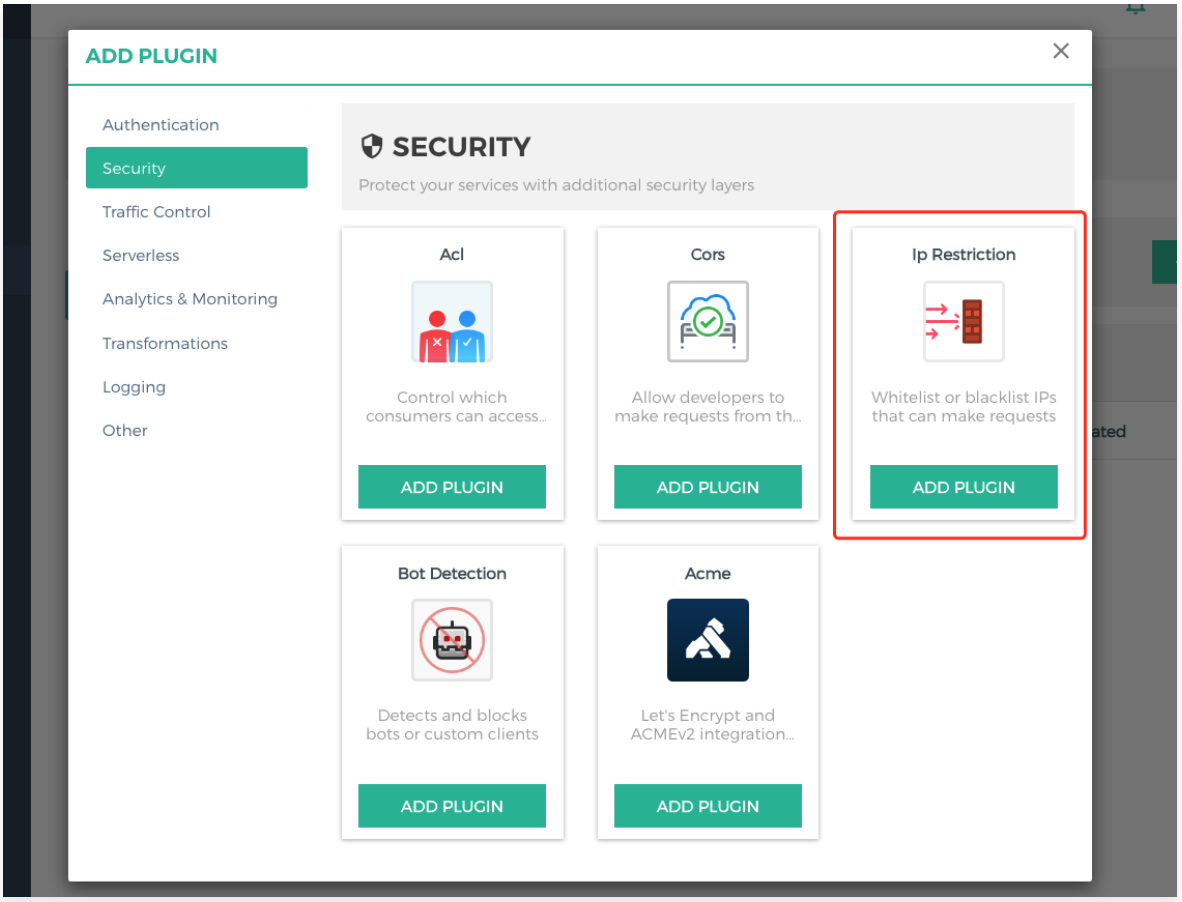
1. 登录 TSF 控制台，进入需要配置限流插件的 Kong 网关实例详情页，在 Konga 控制台页面查看管理控制台登录方式。



2. 登录 Konga 管理控制台，进入需要 IP 限制的 **Route 详情页**，单击 **Add Plugin** 按钮创建插件。




3. 在插件市场的 Security 分组中选择 IP Restriction 插件，单击 Add Plugin。

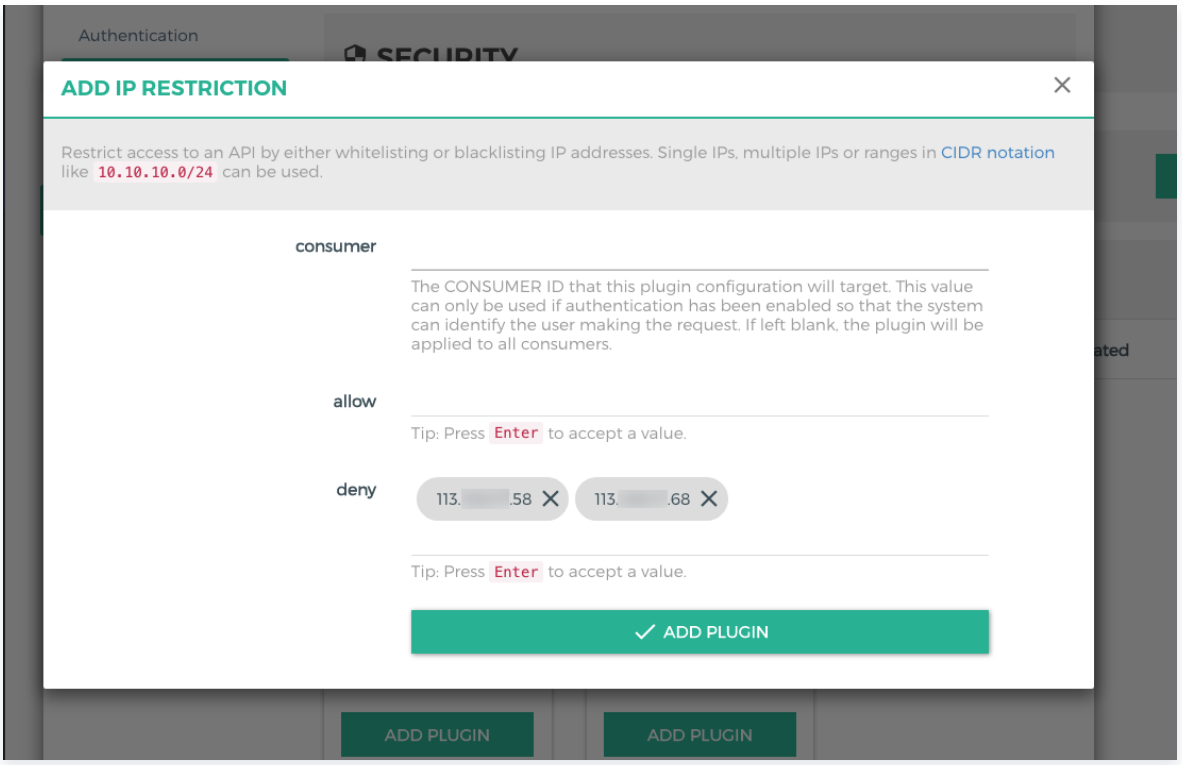



4. 在插件配置中输入需要配置限制的 IP，点击 enter，并保存。

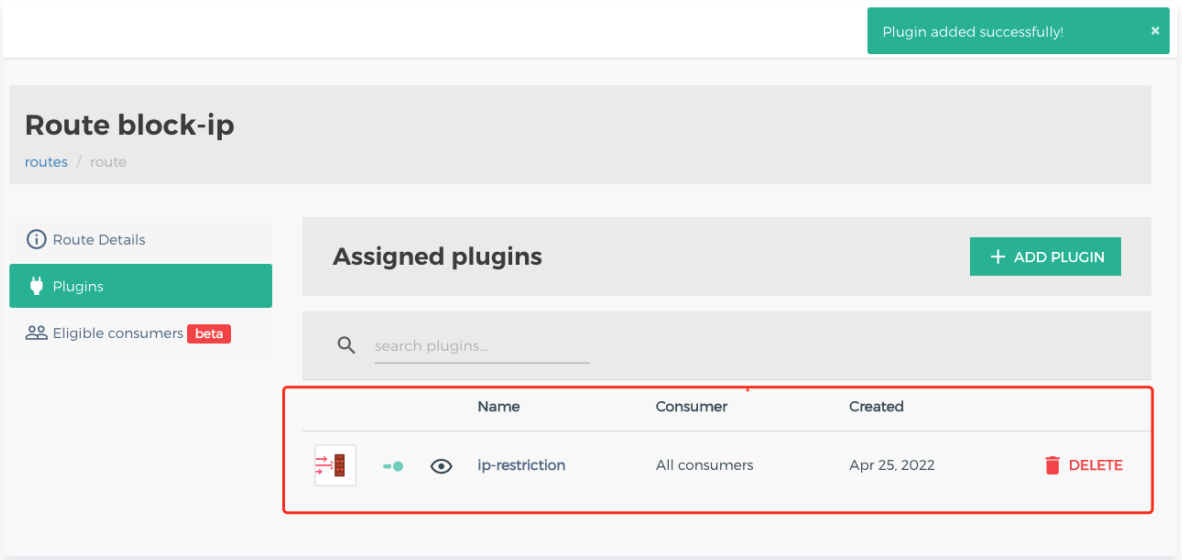
- `allow`：填写允许访问的IP，多个 IP 需要填写多个。
- `deny`：填写拒绝访问的IP，多个 IP 需要填写多个。
- `consumer`：填写需要应用访问控制的 Consumer ID，如空缺，表示该 IP 访问控制应用于所有 consumer。

> ⚠ **注意**

> **allow 和deny 两者至少配置一项。**




5. 返回 route 页面，确定该 route 已经绑定的创建的插件。




6. 发起 API 请求，该 IP 访问被限制。

```
HTTP/1.1 403 Forbidden
Connection: keep-alive
Content-Length: 48
Content-Type: application/json; charset=utf-8
Date: Mon, 25 Apr 2022 02:57:32 GMT
X-Kong-Response-Latency: 1

{
  "message":"Your IP address is not allowed"
```

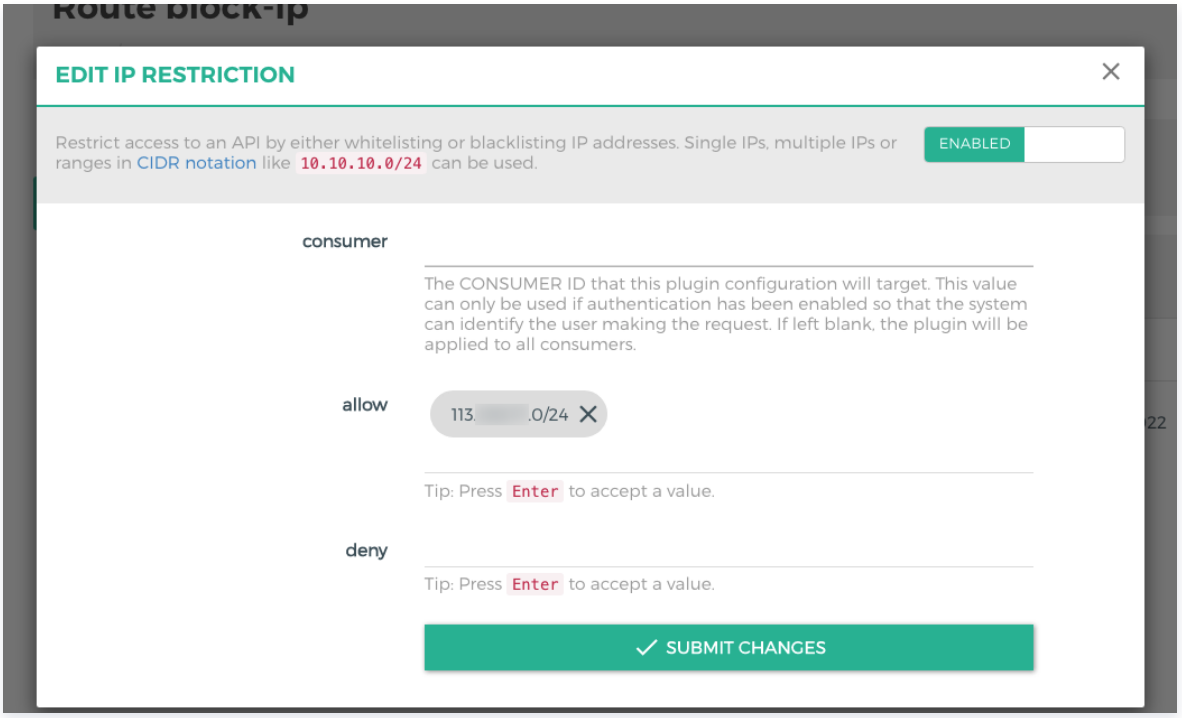
```
}
```

## 场景二：配置 IP 黑/白名单 CIDR 段进行访问控制

本场景操作步骤以白名单 IP CIDR 段为例，指导如何允许某个 IP 段的请求访问，黑名单配置与之类似。

1. 进入需要 IP 限制的 Route 详情页，单击 Add Plugin 按钮创建插件。
2. 在插件配置中填写以下配置，点击 enter 键，并保存。
   - `allow`：填写允许访问的 IP CIDR 段。
   - `deny`：填写拒绝访问的 IP CIDR 段。
   - `consumer`：填写需要应用访问控制的 Consumer ID，如空缺，表示该IP访问控制应用于所有 consumer。

> **注意**
> allow 和 deny 两者至少配置一项。




3. 发起 API 请求，使用不在该 CIDR 段的IP访问，请求被拒绝。

```
HTTP/1.1 403 Forbidden
Date: Mon, 25 Apr 2022 03:06:58 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Content-Length: 48
X-Kong-Response-Latency: 14

{
  "message":"Your IP address is not allowed"
}
```

## 注意事项

当对一个 IP 同时应用了 allow 和 deny ，该 IP 会被限制访问。

## 参考

更多相关说明请参见 Kong 插件文档 。

# 使用 Key Auth 密钥认证

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何在 Kong 云原生 API 网关上通过 Key Authentication 密钥身份认证插件实现认证访问。

## 前置条件

- 已购买 Kong 网关实例，详情请参见 操作文档 。
- 配置了后端（Service）以及路由（Route）。

## 操作步骤

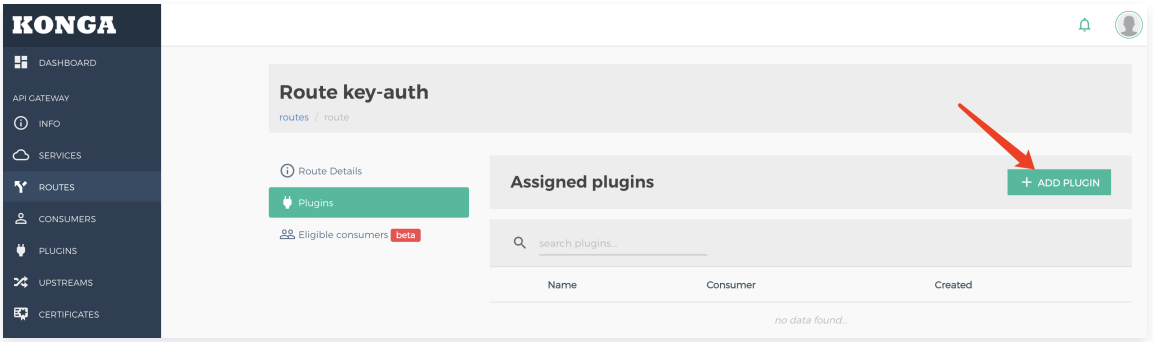
本场景操作步骤以配置 Route 插件为例，指导如何实现密钥身份认证访问。

### 步骤1：配置 Key Authentication 密钥身份认证插件

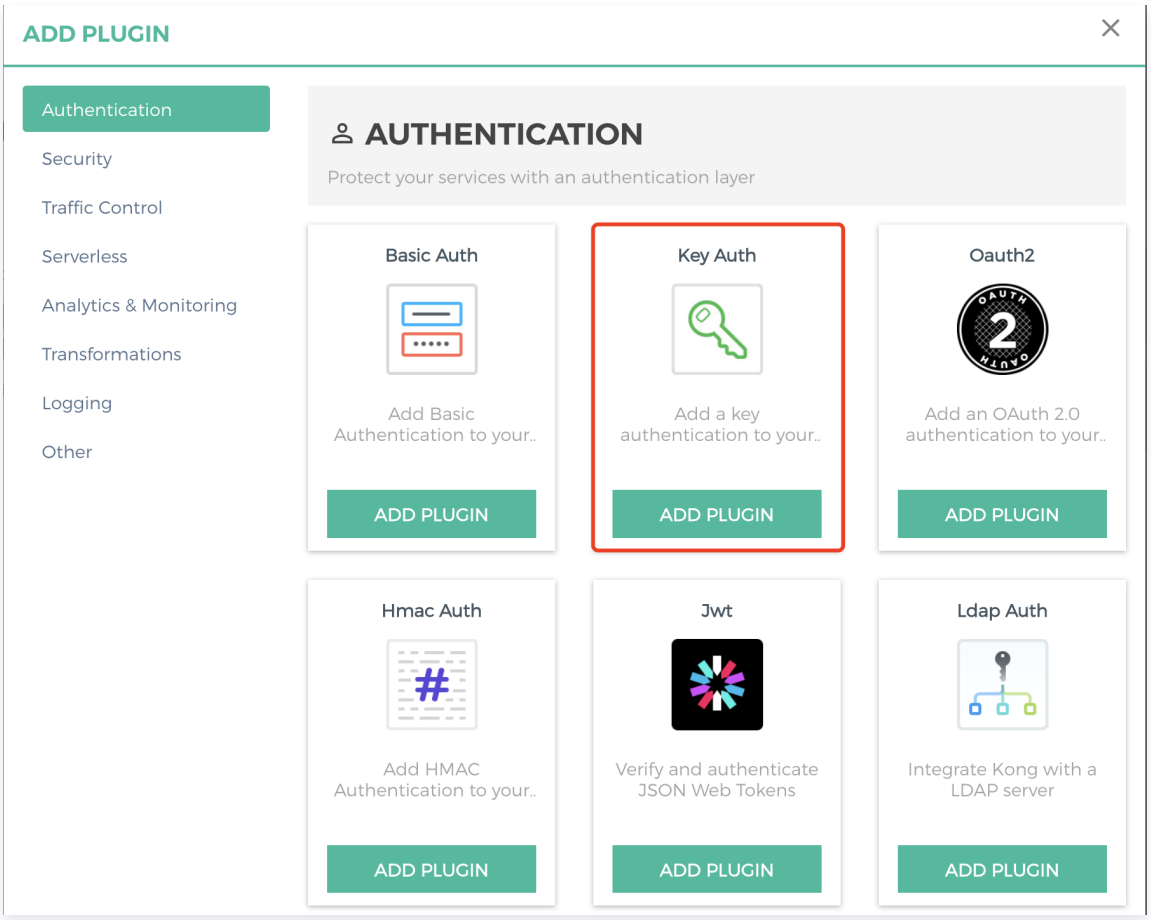
1. 登录 TSF 控制台，进入需要配置限流插件的 Kong 网关实例详情页，在 Konga 控制台页面查看管理控制台登录方式。




2. 登录 Konga 管理控制台，进入需要密钥身份认证访问的 **Route 详情页**，单击 **Add Plugin** 添加插件。



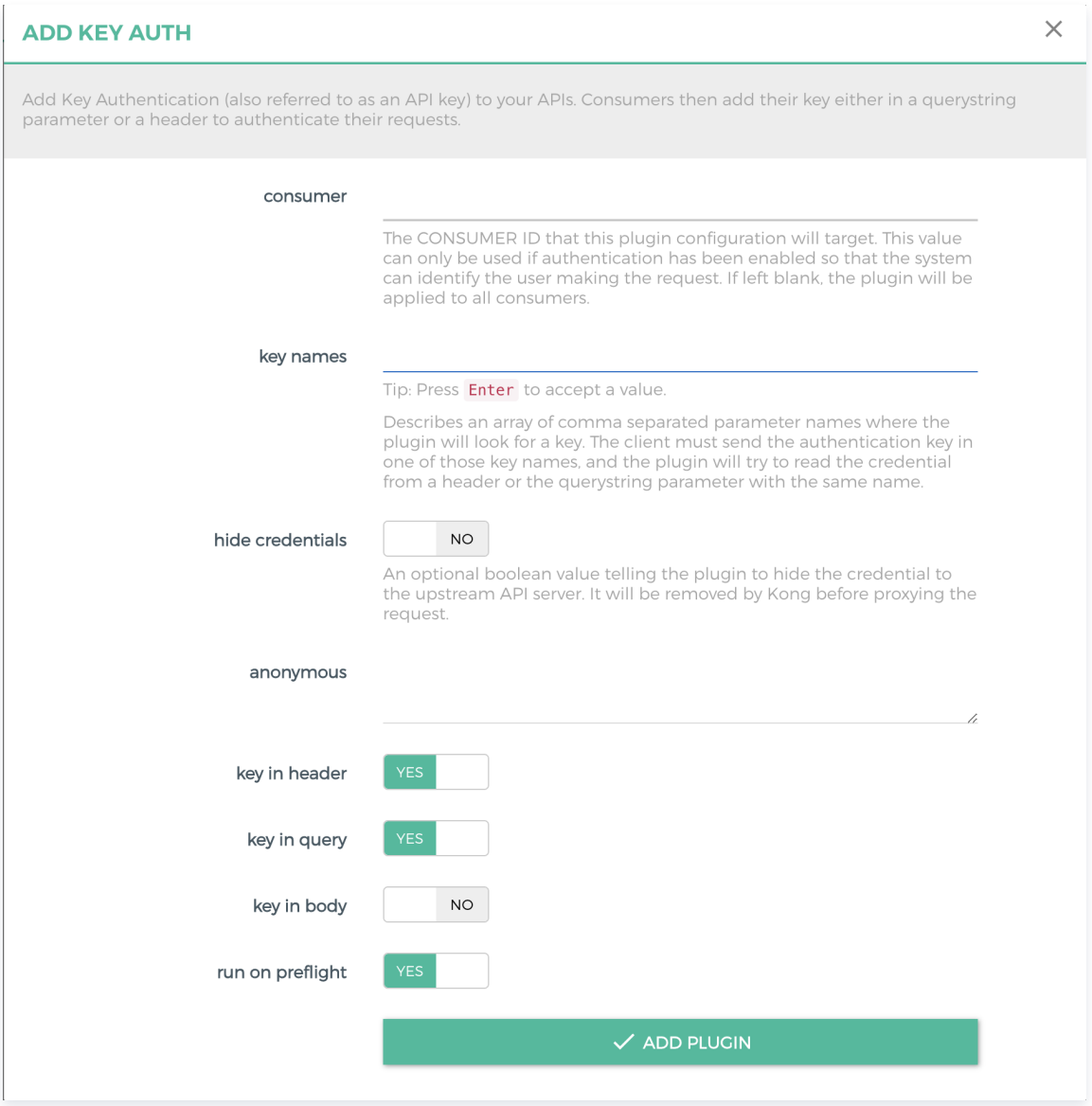

3. 在插件市场的 Authentication 分组中选择 **Key Auth** 插件，单击 **Add Plugin**。



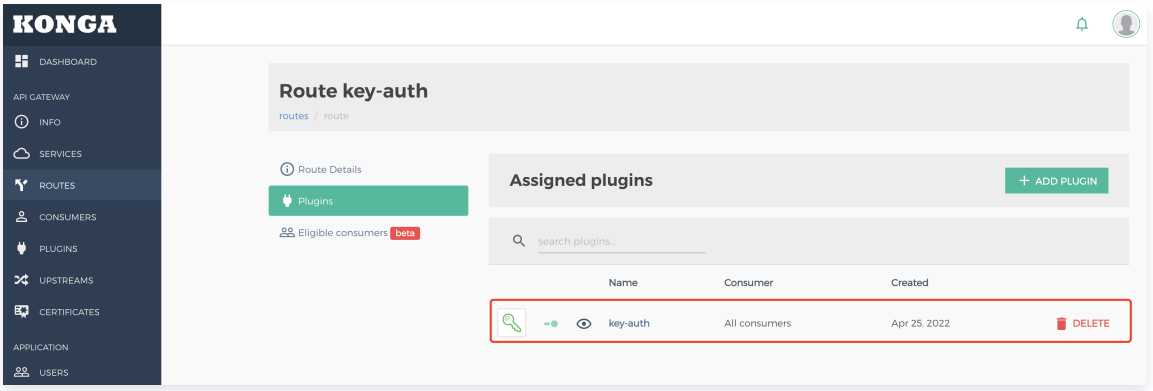

4. 设置密钥身份认证插件参数，并保存。
   - `key names`：插件将要查找键的参数名称数组（默认为 apikey）。客户端必须以其中一个密钥名称发送身份验证密钥，插件将尝试从 header 或具有相同名称的查询参数读取凭据。

     > **注意**
     > 键名只能包含[a-z]，[A-Z]，[0-9]，[_]和[-]。由于 NGINX 默认值中的附加限制，不允许在标题中使用下划线。

   - `hide credentials`：是否隐藏来自上游服务的凭据。
   - `anonymous`：如果身份验证失败，则用作“匿名”使用者的可选字符串（使用者 uuid）值。如果为空（默认），则请求将失败，并且身份验证失败4xx。请注意，此值必须引用 Kong 内部的 Consumer id 属性，而不是其 custom_id。
   - `key in header`：如果启用，插件将要查找键的参数放在 header。
   - `key in query`：如果启用，插件将要查找键的参数放在 query。
   - `key in body`：如果启用，插件将读取请求正文（如果所述请求有一个并且支持其MIME类型）并尝试在其中找到密钥。支持的 MIME 类型是 application/www-form-urlencoded，application/json 和 multipart/form-data。
   - `run on preflight`：插件是否应在 OPTIONS 预检请求上运行。

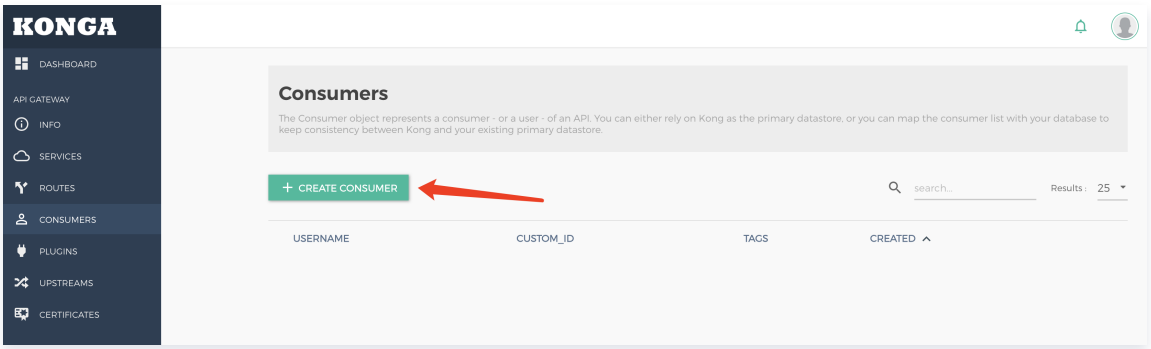



5. 返回 route 页面，确定该 route 已经绑定的创建的插件。

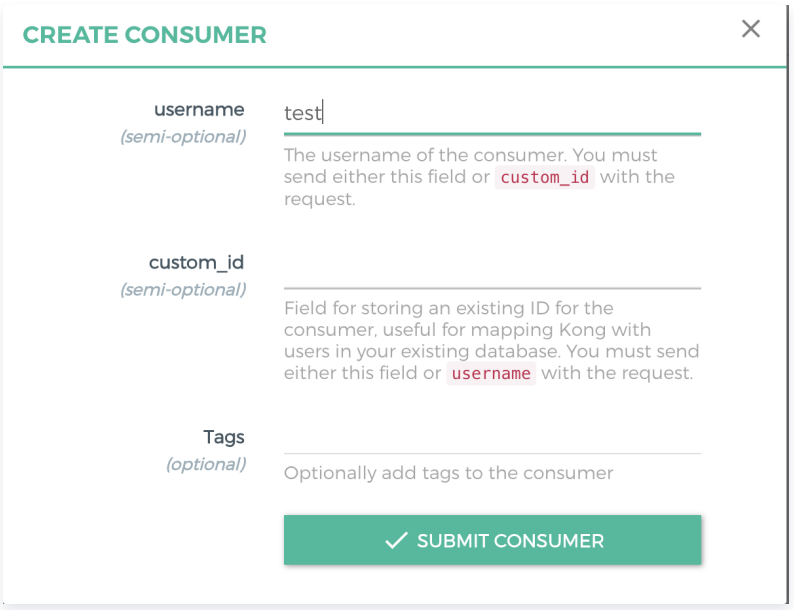



## 步骤2：创建 Consumer

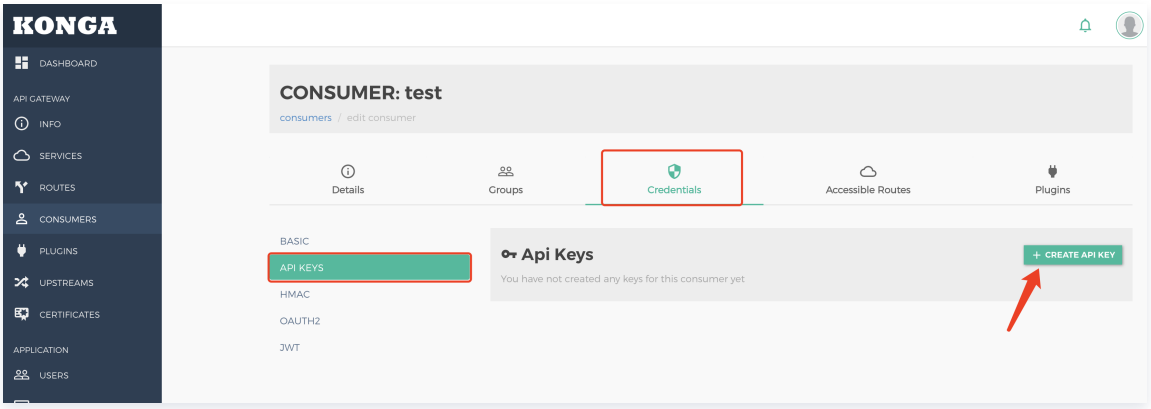
1. 进入 CONSUMERS 页面创建 Consumer。



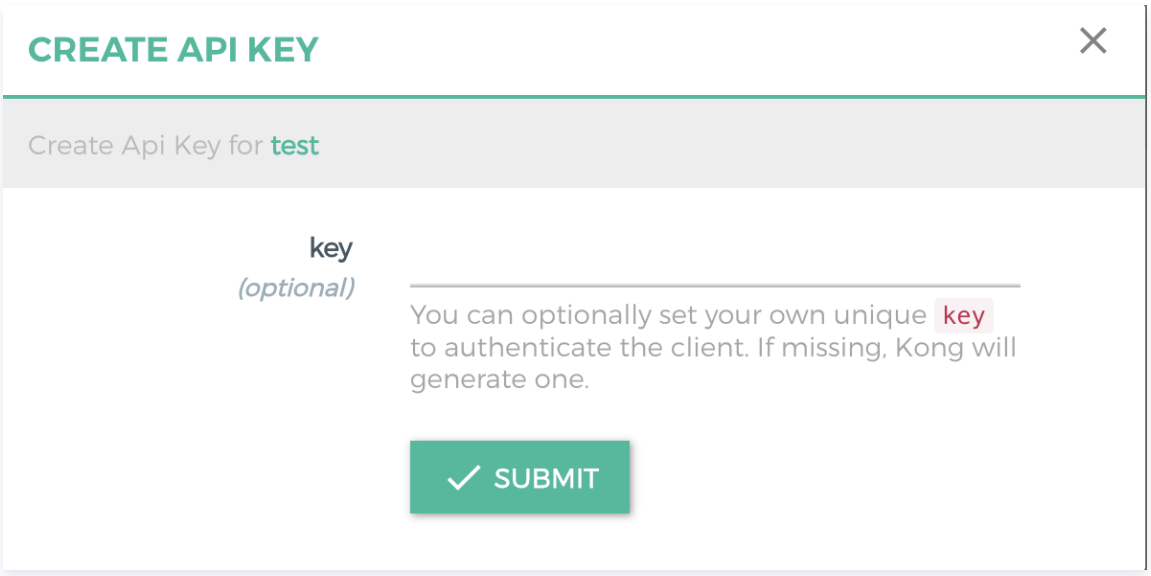

- `username`：用户（应用）名，必须指定此字段或 custom_id。
- `custom_id`：用于将使用者映射到另一个数据库的自定义标识符。必须指定此字段或username。
- `Tags`：标签。



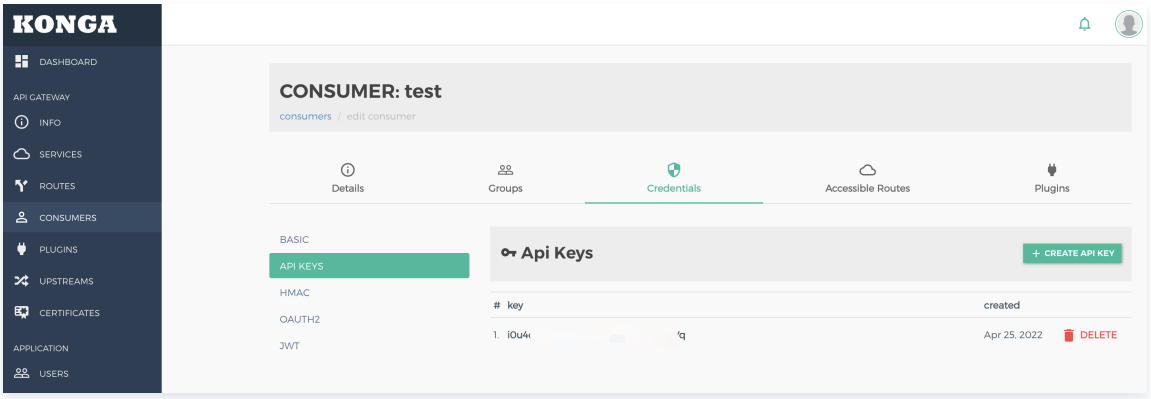

2. 为 Consumer 创建 api key 凭证。




`key`：key 值，默认为空会自动生成 key。




3. 查看生成的 key。




## 步骤3：发起 API 请求

1. 未携带 apikey 会返回报错：`curl http://kong:8000/{proxy path}/`。

```
HTTP/1.1 401 Unauthorized
Server:
Date: Mon, 25 Apr 2022 14:16:10 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 45
Connection: keep-alive
WWW-Authenticate: Key realm="kong"
X-Kong-Response-Latency: 1

{

   "message":"No API key found in request"
 }
```

2. 正确携带 apikey 则放通请求：`curl http://kong:8000/{proxy path}?apikey=<some_key>`。

```
HTTP/1.1 200 OK
Server:
Date: Mon, 25 Apr 2022 14:19:19 GMT
```

```
Content-Type: text/html; charset=UTF-8
Content-Length: 6
Connection: keep-alive
X-Kong-Upstream-Latency: 5
X-Kong-Proxy-Latency: 0
Via: kong/2.4.1

ok
```

## 注意事项

Key Auth 不支持针对不同 consumer 使用不同的 Api Key，如需针对单个用户进行 Key Auth 认证，需配合 post function 插件一起使用，自定义增加识别用户 ID 的处理逻辑。

## 参考

更多相关说明请参见 Kong 密钥认证插件官方文档 。

# 使用 ACL 访问控制

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何在 Kong 云原生 API 网关上通过 Kong ACL 插件实现访问控制，它主要包括以下两种场景：

- 配置允许指定用户访问。
- 配置拒绝指定用户访问。

## 前置条件

1. 已购买 Kong 网关实例，操作文档。
2. 配置了后端（Service）以及路由（Route）。
3. 配置了消费者（Consumer）并开启认证。

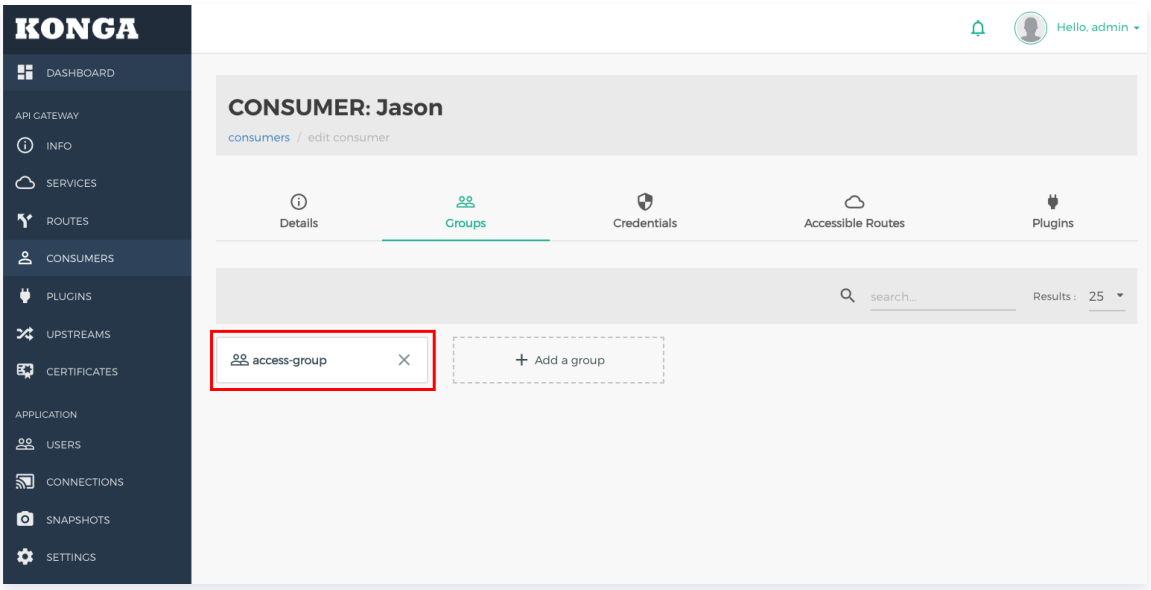
## 操作步骤

> **注意**
> 本操作步骤以允许指定用户访问的配置为例，拒绝指定用户访问的配置与之类似。

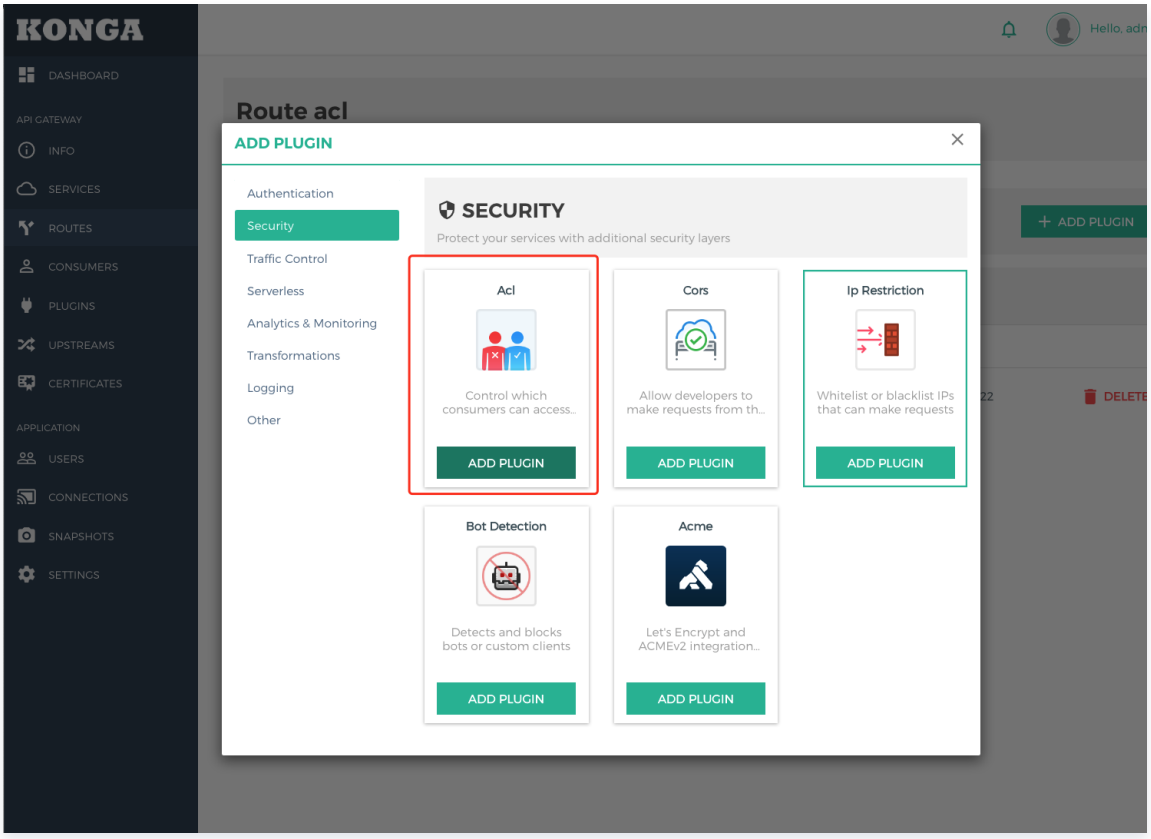
1. 登录 TSF 控制台，进入需要配置限流插件的 Kong 网关实例详情页，在 Konga 控制台页面查看管理控制台登录方式。




2. 登录到 Konga 控制台，进入 Consumer 详情页，选择需要配置访问控制的用户（例如 Jason），单击 Groups Tab，为该用户分配 Group（如 access-group）。

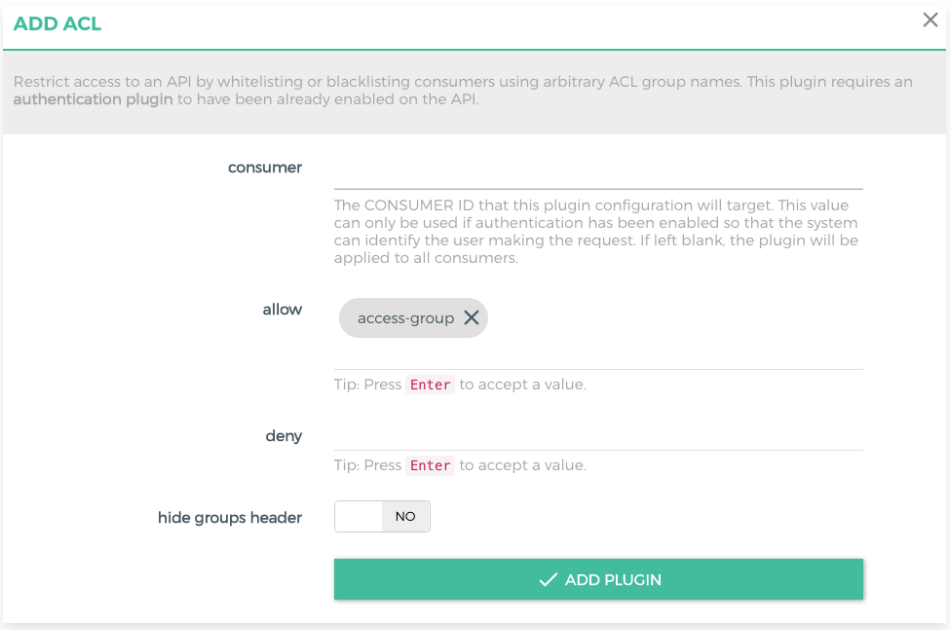



3. 进入选择的 Route，单击 **ADD PLUGIN**，在插件市场的 Security 分组中选择 Acl 插件，单击 **Add Plugin**。

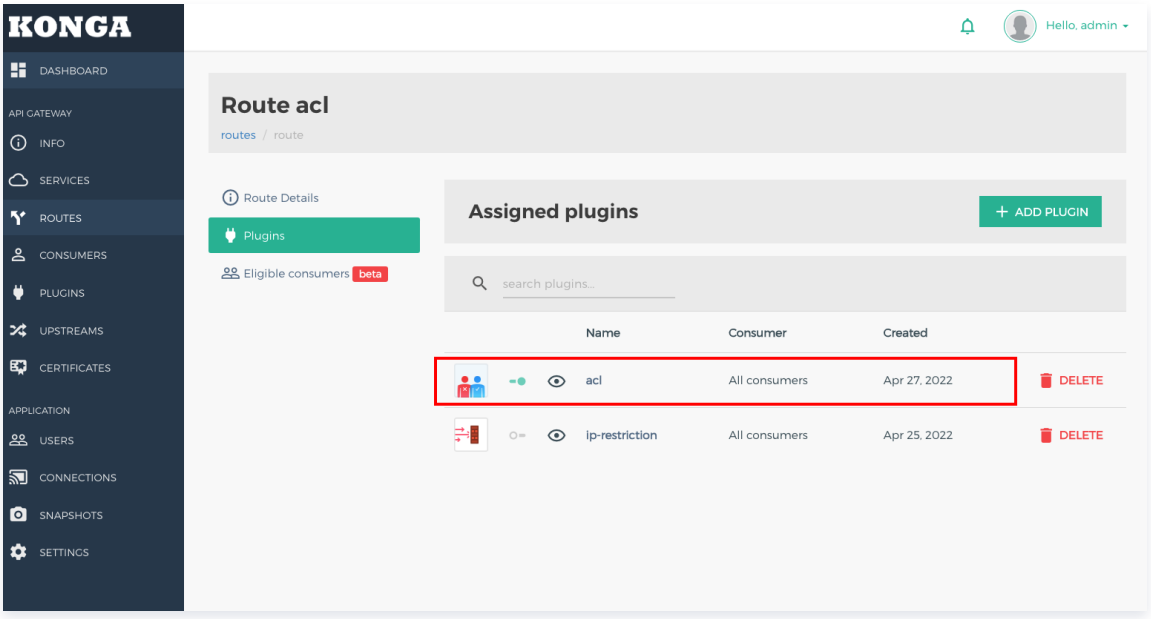



4. 插件配置中填写需要配置允许访问的 Group 信息，单击 enter 键，并保存。
   - `allow`：填写允许访问的 Group，多个 Group 需要填写多个。
   - `deny`：填写拒绝访问的 Group，多个 Group 需要填写多个。
   - `consumer`：填写需要应用访问控制的 Consumer ID，如空缺，表示该 IP 访问控制应用于所有 consumer。

> **注意**
> **allow 和 deny 两者至少配置一项。**




5. 返回 route 页面，确定该 route 已经绑定的创建的插件。




6. 使用允许访问用户组内的用户（如 Jason）的凭证发起 API 请求，该请求被允许。

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 13
Content-Type: text/plain; charset=UTF-8
Date: Wed, 27 Apr 2022 06:46:38 GMT
Server: Cowboy
Vary: Origin
X-Kong-Proxy-Latency: 10
X-Kong-Upstream-Latency: 1775

{"ok"}
```

7. 使用不在该用户组的用户（如Tom）的凭证发起请求，该请求被拒绝。

```
HTTP/1.1 403 Forbidden
Connection: keep-alive
Content-Length: 49
Content-Type: application/json; charset=utf-8
Date: Wed, 27 Apr 2022 06:44:55 GMT
Server: kong/2.4.1
X-Kong-Response-Latency: 10

{
 "message":"You cannot consume this service"
}
```

## 相关说明

更多相关说明请参见 Kong 插件 。

# 使用 HMAC Auth 认证访问

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何在 Kong 云原生 API 网关上通过 HMAC Authentication 认证插件实现认证访问。

## 前置条件

- 已购买 Kong 网关实例，操作文档。
- 配置了后端（Service）以及路由（Route）。

## 操作步骤

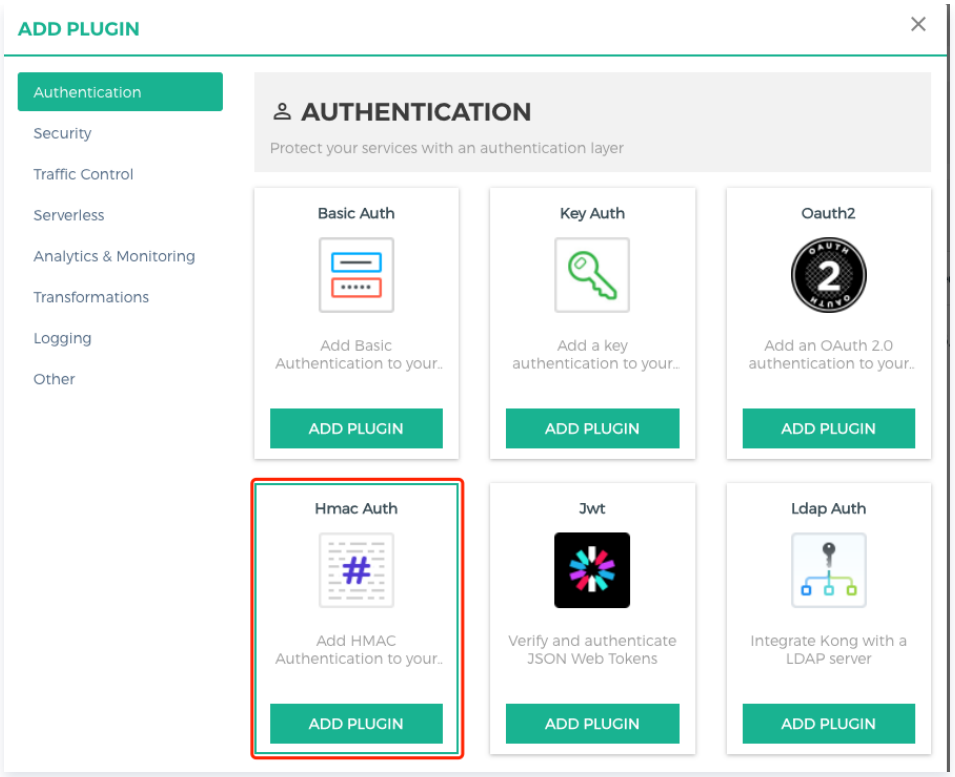
本场景操作步骤以配置 Route 插件为例，指导如何实现 HMac Auth 认证访问。

### 步骤1：配置 HMAC Authentication 认证插件

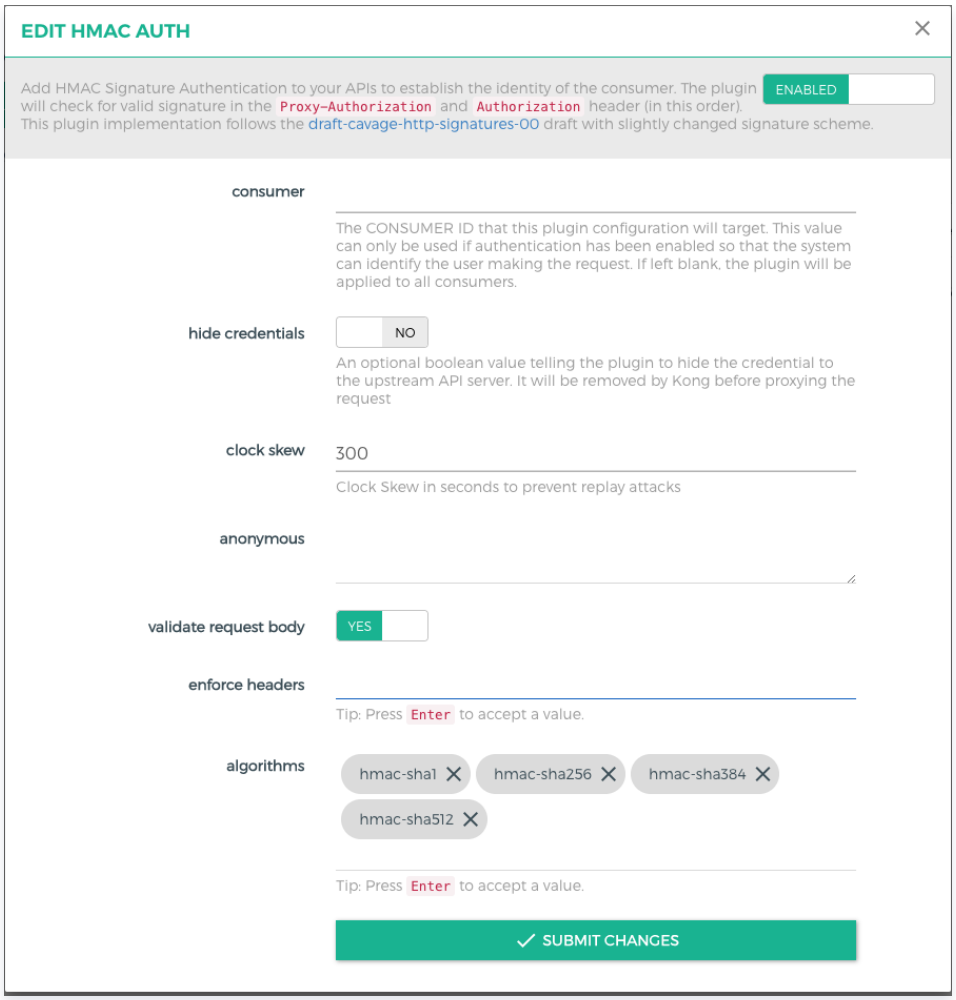
1. 登录 TSF 控制台，进入需要配置限流插件的 Kong 网关实例详情页，在 Konga 控制台页面查看管理控制台登录方式。




2. 登录 Konga 管理控制台，进入需要限流的 **Route 详情页**，单击 **Add Plugin** 按钮创建插件，在 **Authentication** 分组下选择 **Hmac Auth** 插件。

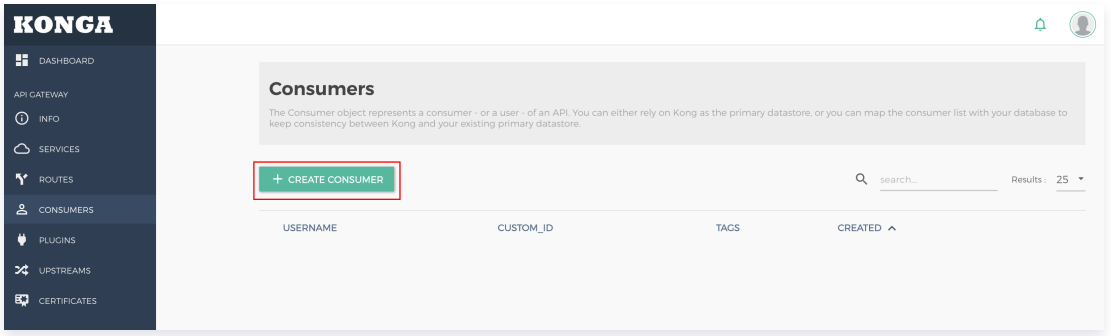



3. 设置密钥身份认证插件参数，单击 enter 键，并保存。
   - `consumer`：填写需要应用访问控制的 Consumer ID，如空缺，表示该 IP 访问控制应用于所有 consumer。
   - `hide credentials`：是否隐藏来自上游服务的凭据。
   - `anonymous`：如果身份验证失败，则用作“匿名”使用者的可选字符串（使用者 uuid）值。如果为空（默认），则请求将失败，并且身份验证失败4xx。请注意，此值必须引用 Kong 内部的 Consumer ID 属性，而不是其 custom_id。
   - `clock skew`：时钟偏移在几秒钟内以防止重放攻击。
   - `validate request body`：是否启用 body 验证。
   - `enforce headers`：客户端至少应用于 HTTP 签名创建的 header 列表。
   - `algorithms`：用户想要支持的 HMAC 摘要算法列表。允许的值为 hmac-sha1，hmac-sha256，hmac-sha384 和 hmac-sha512，默认全部支持。

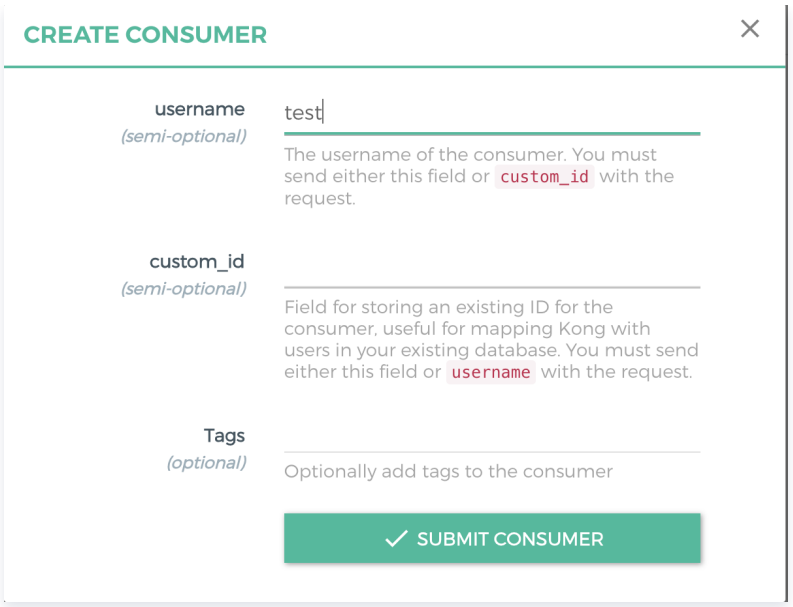



## 步骤2：创建 Consumer

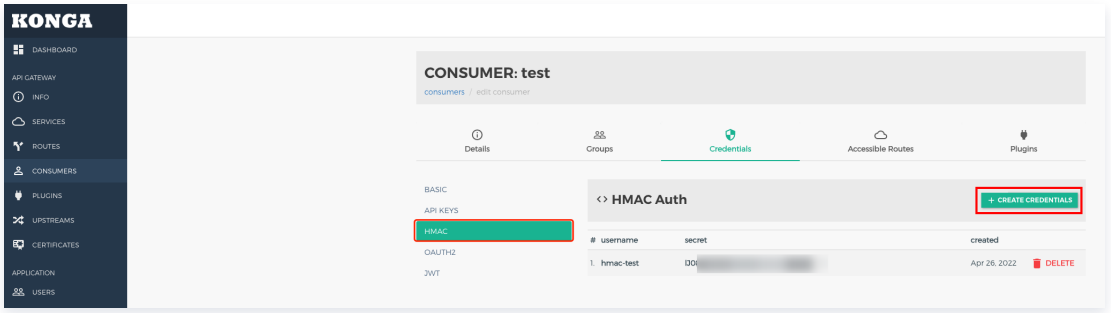
1. 进入 CONSUMERS 页面创建 Consumer。



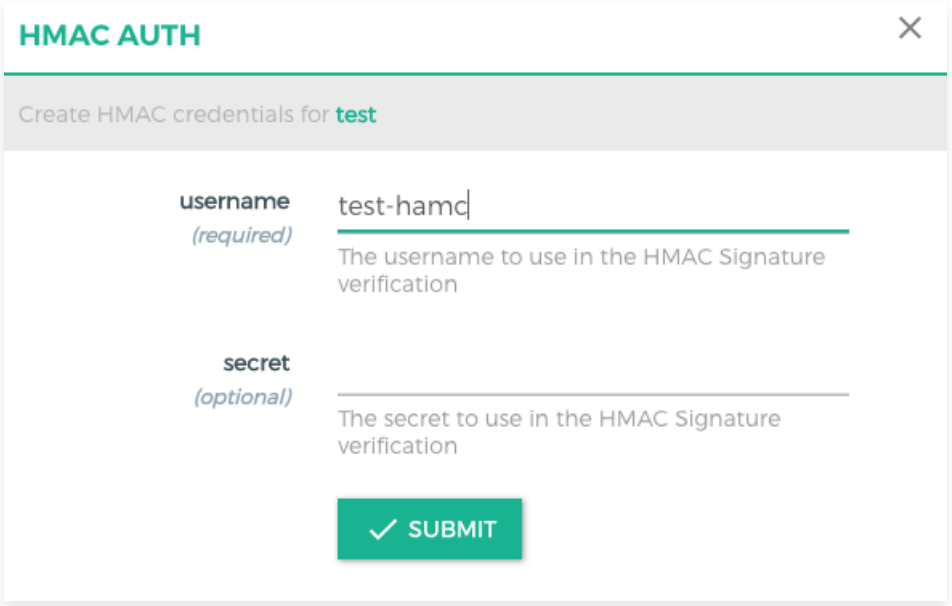

- `username`：用户（应用）名，必须指定此字段或 custom_id。
- `custom_id`：用于将使用者映射到另一个数据库的自定义标识符。必须指定此字段或 username。
- `Tags`：标签。




2. 为 Consumer 创建 HMAC Auth 凭证。




- `username`：在 HMAC 签名验证中使用的用户名。
- `secret`：secret 值，默认为空会自动生成 secret。




## 步骤3：签名认证方案

期望客户端使用以下参数化发送 Authorization 或 Proxy-Authorization header：

例如：

```
Authorization: hmac username="username", algorithm="hmac-sha1", headers="x-date digest",
signature="Base64(HMAC-SHA1(signing_str, secret))"
```

### 签名参数

- `username`：凭证的 username。
- `algorithm`：用于创建签名的数字签名算法。
- `headers`：用于对请求进行签名的 HTTP 标头名称列表，由单个空格字符分隔。
- `signature`：客户端生成的 Base64 编码数字签名。

### 签名字符串构造

为了生成使用密钥签名的字符串，客户端必须按照它们出现的顺序获取 headers 指定的每个 HTTP 标头的值。

1. 如果标题名称不是 request-line，则附加小写标题名称，后跟 ASCII 冒号：和 ASCII 空格 。
2. 如果标题名称是 request-line，则附加 HTTP 请求行（ASCII 格式），否则追加标题值。
3. 如果 value 不是最后一个值，则附加 ASCII 换行符\n。字符串绝不能包含尾随的 ASCII 换行符。
4. 服务器和请求客户端应与 NTP 同步，并且应使用 X-Date 或 Date header 发送有效日期（使用 GMT 格式, 例如 Mon, 19 Mar 2018 12:08:40 GMT）。

### 正文验证 Body Validation

用户可以将 config.validate_request_body 设置为 true 以验证请求正文。如果启用，插件将计算请求正文的 SHA-256 HMAC 摘要，并将其与 Digest header 的值进行匹配。摘要 header 需要采用以下格式：

```
Digest: SHA-256=base64(sha256(<body>))
```

如果没有请求主体，则应将 Digest 设置为0长度的主体的摘要。

> **注意：**
> 为了创建请求主体的摘要，插件需要将其保留在内存中，这可能会在处理大型主体（几个MB）或高请求并发期间对工作者的 Lua VM 造成压力。

## 步骤4：API 请求 Demo

> **说明：**
> 以下 Demo 仅供参考，针对 POST 请求和 GET 请求，请依据实际使用场景进行修改。

### Python版本

```python
# -*- coding: utf-8 -*-
import base64
import datetime
import hashlib
import hmac
import json
import requests
from urllib.parse import urlparse, urlencode

#username
Username = 'xxx'
#secret
Secret = 'xxxx'

# 访问地址
```

```
Url = 'http://test.com/'
HTTPMethod = 'POST'  # method
Accept = 'application/json'
ContentType = 'application/json'

urlInfo = urlparse(Url)
Host = urlInfo.hostname
Path = urlInfo.path

Digest = ''
GMT_FORMAT = '%a, %d %b %Y %H:%M:%S GMT'
xDate = datetime.datetime.utcnow().strftime(GMT_FORMAT)

body_json = ''

# 修改 body 内容
if HTTPMethod == 'POST' :
    # 填写实际请求body
    body = { "arg1": "a", "arg2": "中文" }
    body_json = json.dumps(body)

body_digest = hashlib.sha256(body_json.encode()).digest()
Digest = "SHA-256=" + base64.b64encode(body_digest).decode()

# 获取签名串
signing_str = 'x-date: %s\ndigest: %s' % (
    xDate, Digest)

# 计算签名
sign = hmac.new(Secret.encode(), msg=signing_str.encode(), digestmod=hashlib.sha1).digest()
sign = base64.b64encode(sign).decode()
auth = "hmac username=\"" + Username + "\", algorithm=\"hmac-sha1\", headers=\"x-date digest\",
signature=\""
sign = auth + sign + "\""

# 发送请求
headers = {
    'Host': Host,
    'Accept': Accept,
    'Content-Type': ContentType,
    'x-date': xDate,
    'Authorization': sign,
    'Digest': Digest
}

if HTTPMethod == 'GET' :
    ret = requests.get(Url, headers=headers)
if HTTPMethod == 'POST' :
    ret = requests.post(Url, headers=headers, data=(body_json))

print(ret.headers)
print(ret.text)
```

**Java版本**

```
package org.example;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.text.SimpleDateFormat;
import java.util.Base64;
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;

public class Main {
    public static void main(String[] args) throws Exception {
        String username = "xxxx";
        String secret = "xxxx";
        String url = "http://www.test.com/";
        String httpMethod = "POST";
        String accept = "application/json";
        String contentType = "application/json";

        URI uri = new URI(url);
        String host = uri.getHost();

        String digest = "";
        SimpleDateFormat format = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z",
Locale.US);
        format.setTimeZone(TimeZone.getTimeZone("GMT"));
        String xDate = format.format(new Date());
        String body = "";

        if (httpMethod.equals("POST")) {
            body = "arg1=a&arg2=中文";
        }
        byte[] bodyDigest = java.security.MessageDigest.getInstance("SHA-
256").digest(body.getBytes(StandardCharsets.UTF_8));
        digest = "SHA-256=" + Base64.getEncoder().encodeToString(bodyDigest);
        System.out.println(digest);

        String signingStr = "x-date: " + xDate + "\ndigest: " + digest;
        Mac mac = Mac.getInstance("HmacSHA1");
        mac.init(new SecretKeySpec(secret.getBytes(StandardCharsets.UTF_8), "HmacSHA1"));
        byte[] signBytes = mac.doFinal(signingStr.getBytes(StandardCharsets.UTF_8));
        String sign = Base64.getEncoder().encodeToString(signBytes);
        String auth = "hmac username=\"" + username + "\", algorithm=\"hmac-sha1\", headers=\"x-
date digest\", signature=\"";
```

```
        sign = auth + sign + "\"";
        HttpResponse response = null;
        CloseableHttpClient httpClient = HttpClients.createDefault();
        if (httpMethod.equals("POST")) {
            HttpPost httpPost = new HttpPost(url);
            httpPost.setHeader("Host", host);
            httpPost.setHeader("Accept", accept);
            httpPost.setHeader("Content-Type", contentType);
            httpPost.setHeader("x-date", xDate);
            httpPost.setHeader("Authorization", sign);
            if (!digest.isEmpty()) {
                httpPost.setHeader("Digest", digest);
            }
            StringEntity entity = new StringEntity("arg1=a&arg2=中文", "UTF-8");
            httpPost.setEntity(entity);
            response = httpClient.execute(httpPost);
        } else if (httpMethod.equals("GET")) {
            HttpGet httpGet = new HttpGet(url);
            httpGet.setHeader("Host", host);
            httpGet.setHeader("Accept", accept);
            httpGet.setHeader("Content-Type", contentType);
            httpGet.setHeader("x-date", xDate);
            httpGet.setHeader("Authorization", sign);
            if (!digest.isEmpty()) {
                httpGet.setHeader("Digest", digest);
            }
            response = httpClient.execute(httpGet);
        }

        HttpEntity responseEntity = response.getEntity();
        System.out.println(EntityUtils.toString(responseEntity));
    }
}
```

## 相关说明

更多相关说明请参见 Kong 插件 。

# 使用 Bot Detection 插件拒绝爬虫请求

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何在 Kong 云原生 API 网关上通过 Kong Bot Detection 插件屏蔽爬虫访问。
Bot Detection插件通过检查HTTP请求中User Agent字段，识别发起请求的用户代理软件信息，拒绝爬虫请求。该插件内置了一些基础的校验规则用于校验请求，您可以参见 爬虫示例 定义爬虫请求。

## 前置条件

- 已购买 Kong 网关实例，详情请参见 实例管理 。
- 配置了后端（Service）以及路由（Route）。

## 插件配置

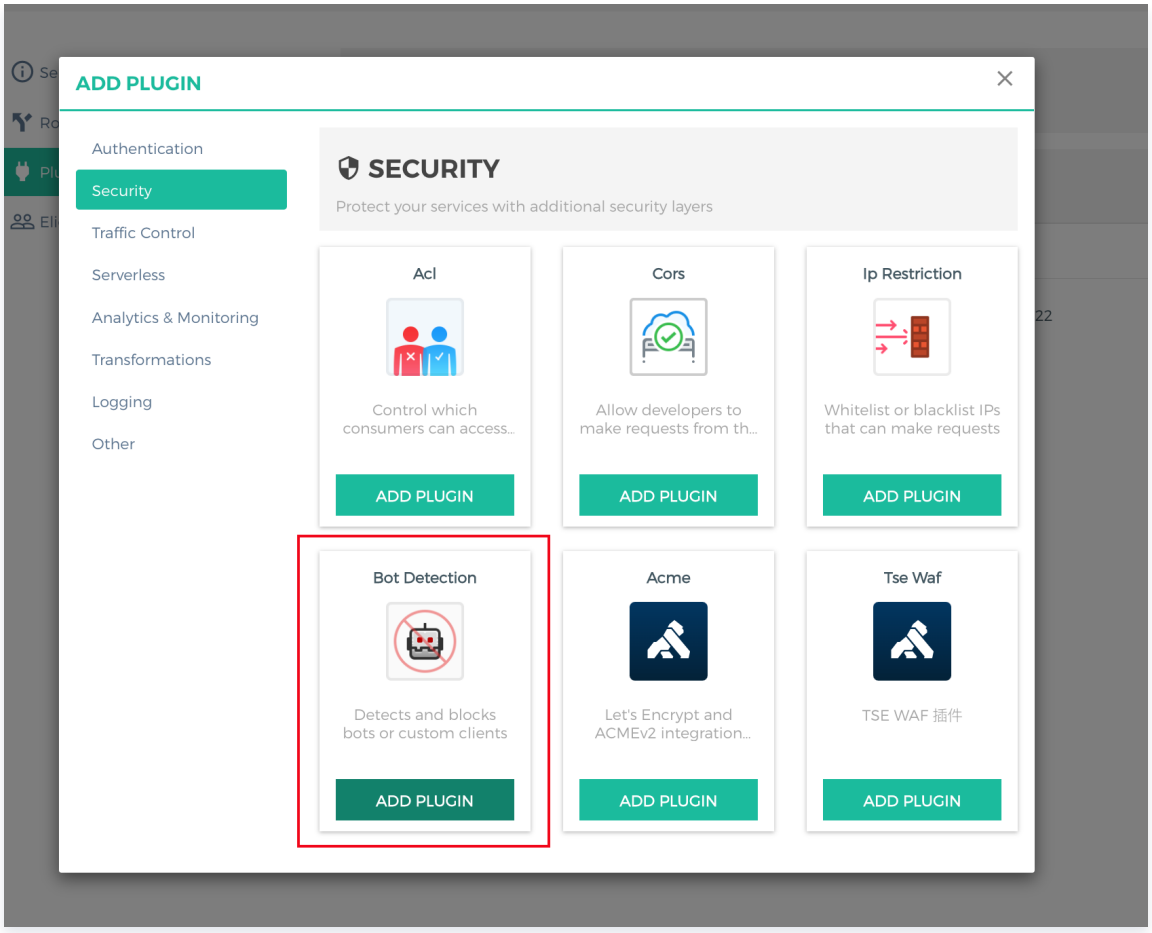
| 字段名称 | 字段说明 |
| --- | --- |
| allow | 使用正则表达式定义允许放行的 User-Agent 列表。 |
| deny | 使用正则表达式定义拒绝放行的 User-Agent 列表。 |

## 操作步骤

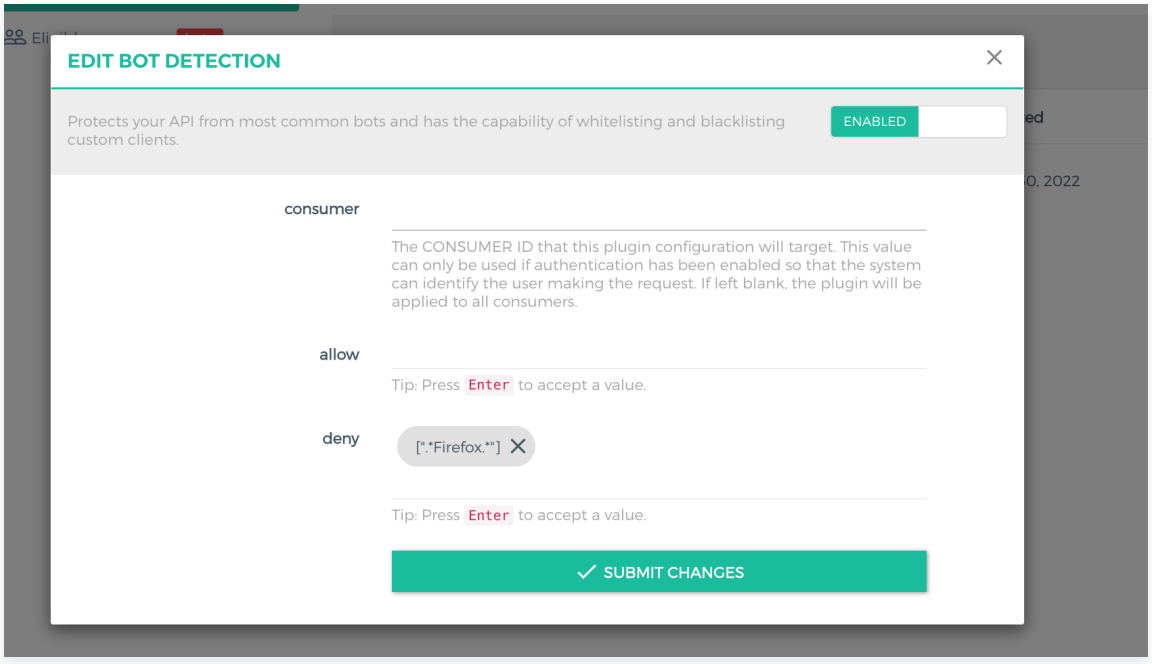
1. 登录 TSF 控制台，进入需要配置限流插件的 Kong 网关实例详情页，在 Konga 控制台页面查看管理控制台登录方式。



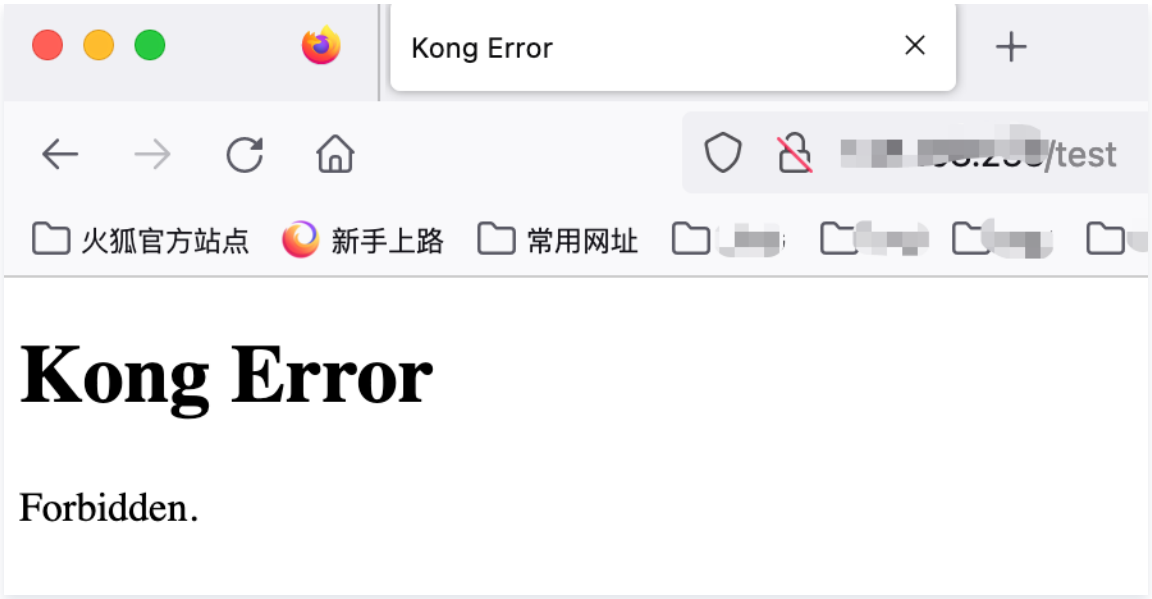

2. 登录 Konga 管理控制台，进入需要反爬虫的 Route 详情页，单击 **Add Plugin**，在插件市场的 Security 分组中选择 Bot Detection 插件，单击 **Add Plugin**。




3. 插件配置中使用正则表达式填写需要配置限制的代理信息，例如，拒绝来自 Firefox 的爬虫请求，在 deny 中配置 `[".*Firefox.*"]`，单击 enter 键。




4. 使用火狐浏览器发起 API 请求，由于请求头带有 `User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:106.0) Gecko/20100101 Firefox/106.0`，请求被拒绝。




## 参考

更多相关说明请参见 Kong Bot Detection 插件官方文档 。

# 配置 JWT 认证鉴权

最近更新时间：2025-06-09 16:07:02

## 操作场景

JWT（JSON Web Token）是现代 Web 服务中经常用到的认证和授权方式。Kong 能够验证使用 HS256 或 RS256 签名的 JWT 的请求。您可以为消费者配置 JWT 凭据（公钥和私钥），并用这些凭据签署他们的 JWT。Kong 支持通过 Query、Cookie、Header 传递 JWT Token。如果 JWT 的签名通过验证，Kong 将把请求代理到上游服务，否则将丢弃请求。
JWT 常见的三种签名算法为 HS256、RS256 和 ES256，区别在于消息签名与签名验证需要的 key 不同。

- HS256 使用同一个 secret_key 进行签名和验证，适用于集中式认证场景，签名和验证都必须由可信方进行。
- RS256 是使用 RSA 私钥进行签名，使用 RSA 公钥进行验证，因此可以将验证委托给其他应用。
- ES256 使用私钥签名，公钥验证，但是拥有更短的签名长度。

本文介绍如何在 Kong 云原生 API 网关上通过 JWT 插件实现下述常见 JWT 认证鉴权场景：

- 对使用 HS256 签名的请求进行认证鉴权。
- 对使用 RS256 签名的请求进行认证鉴权。

## 前置条件

- 已购买 Kong 网关实例，详情请参见 实例管理 。
- 配置了后端（Service）以及路由（Route）。

## 插件配置

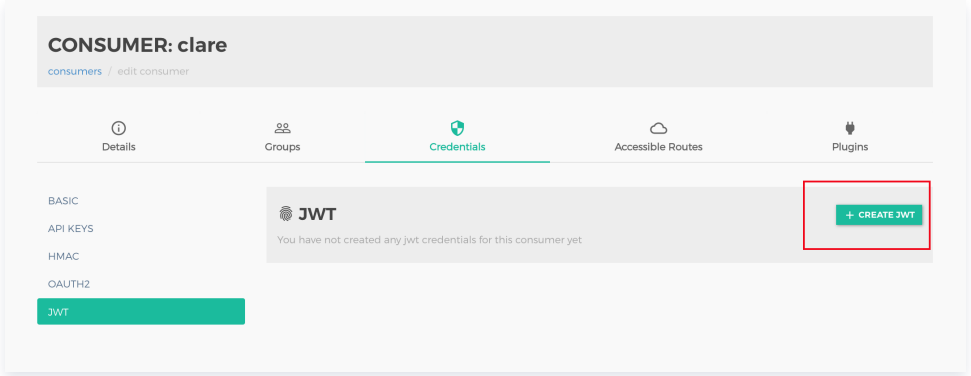
| 字段名称 | 字段说明 |
| --- | --- |
| uri param names | 用于获取 JWT Token 的 Query 参数列表，当 JWT Token 配置在 URI 时，设置此属性。 |
| cookies names | 用于获取 JWT Token 的 Cookie 参数列表，当 JWT Token 配置在Cookie时，设置此属性。 |
| key claim name | Credential 的 key 值，对应的 JWT Claim 的内容，默认为 iss。 |
| secret is base64 | Credential 的 secret 是否进行 base64 编码。 |
| claims to verify | Kong 进行校验的 Claims，支持 exp 或 nbf。 |
| anonymous | 认证失败时，判断是否作为匿名Consumer。默认为空，表明请求失败时返回4xx。 |
| run on preflight | 插件是否应在 OPTIONS 预检请求上运行（并尝试进行身份验证）。如果设置为 false，则将始终允许 OPTIONS 请求。 |
| maximum expiration | JWT Token 过期时间，支持0-31536000（365天）。当设置该值时，需要在 claims to verify 中指定 exp。默认值为0，表示无限期。当 Token 过期时请求将被拒绝，返回 HTTP 403。请注意，配置该值时应考虑潜在的时钟偏差。 |
| header names | 用于获取 JWT Token 的 Header 参数列表。当 JWT Token 配置在 URI 时，设置此属性。 |

## 操作步骤

### 场景一：对使用 HS256 签名的请求进行认证鉴权

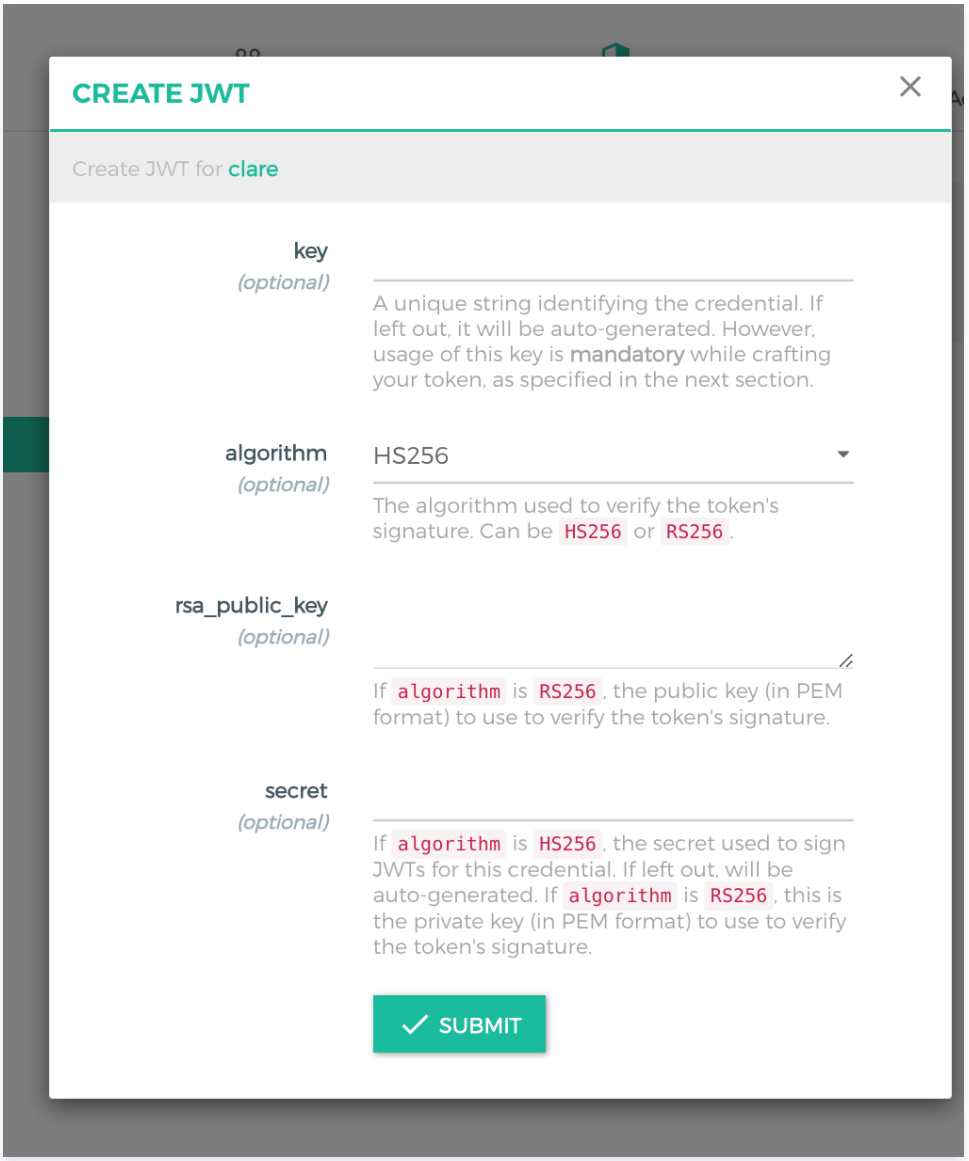
#### 步骤1：创建 Consumer

1. 登录 TSF 控制台，进入需要配置限流插件的 Kong 网关实例详情页，在 Konga 控制台页面查看管理控制台登录方式。
2. 登录到 Konga 控制台，进入 Consumer 详情页，选择需要配置访问控制的用户（如clare），单击 **Credentials Tab**，选择 JWT，单击 **创建 JWT**，为该用户创建一个 JWT Token 作为访问凭证。

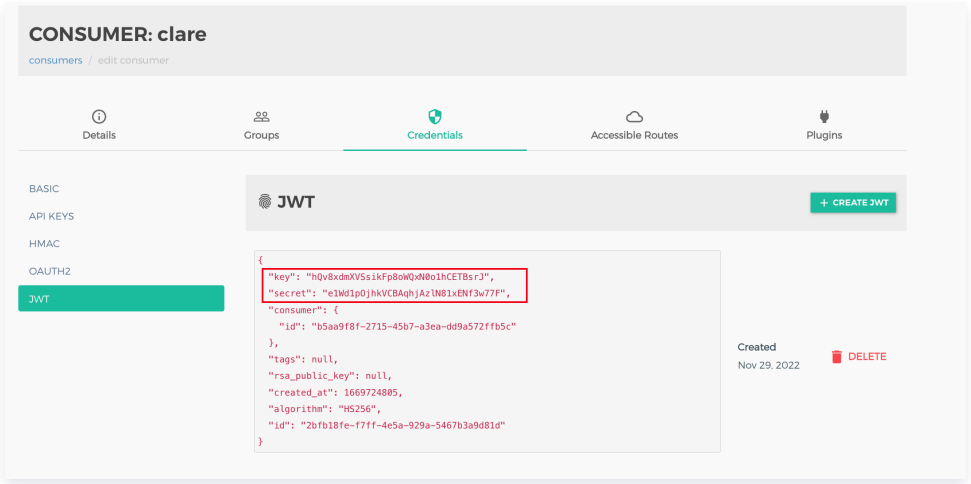



3. 配置 JWT 相关参数。

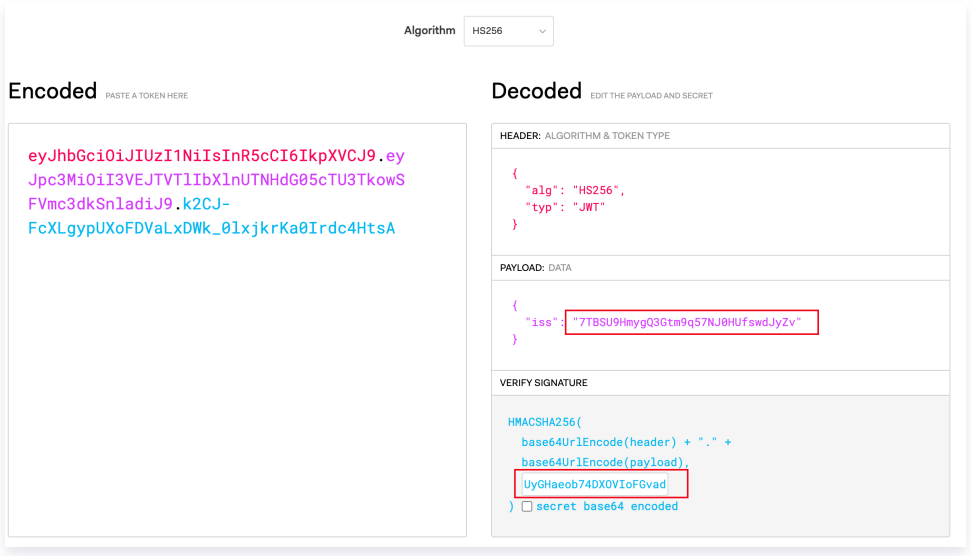
| 配置项 | 描述 |
| --- | --- |
| key | 对应 JWT Claims 的 issuer，即签发人。留空则自动生成。 |
| algorithm | JWT Token 签名所使用的加密算法，支持 HS256 和 RS256。本场景设置为 HS256。 |
| secret | 当签名算法为 HS256 时，设置 secret，用于验证签名，留空则自动生成。 |



4. 查看生成的用户凭证。

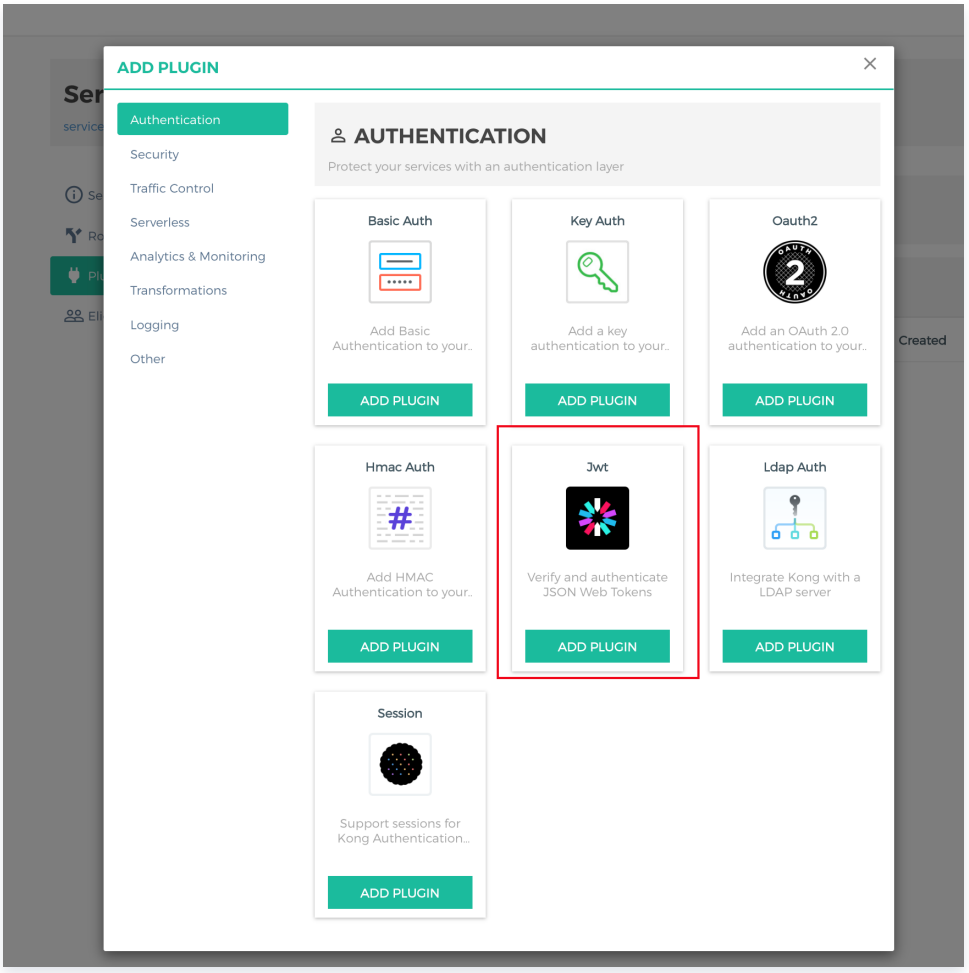



5. 使用程序生成对应的 JWT Token。

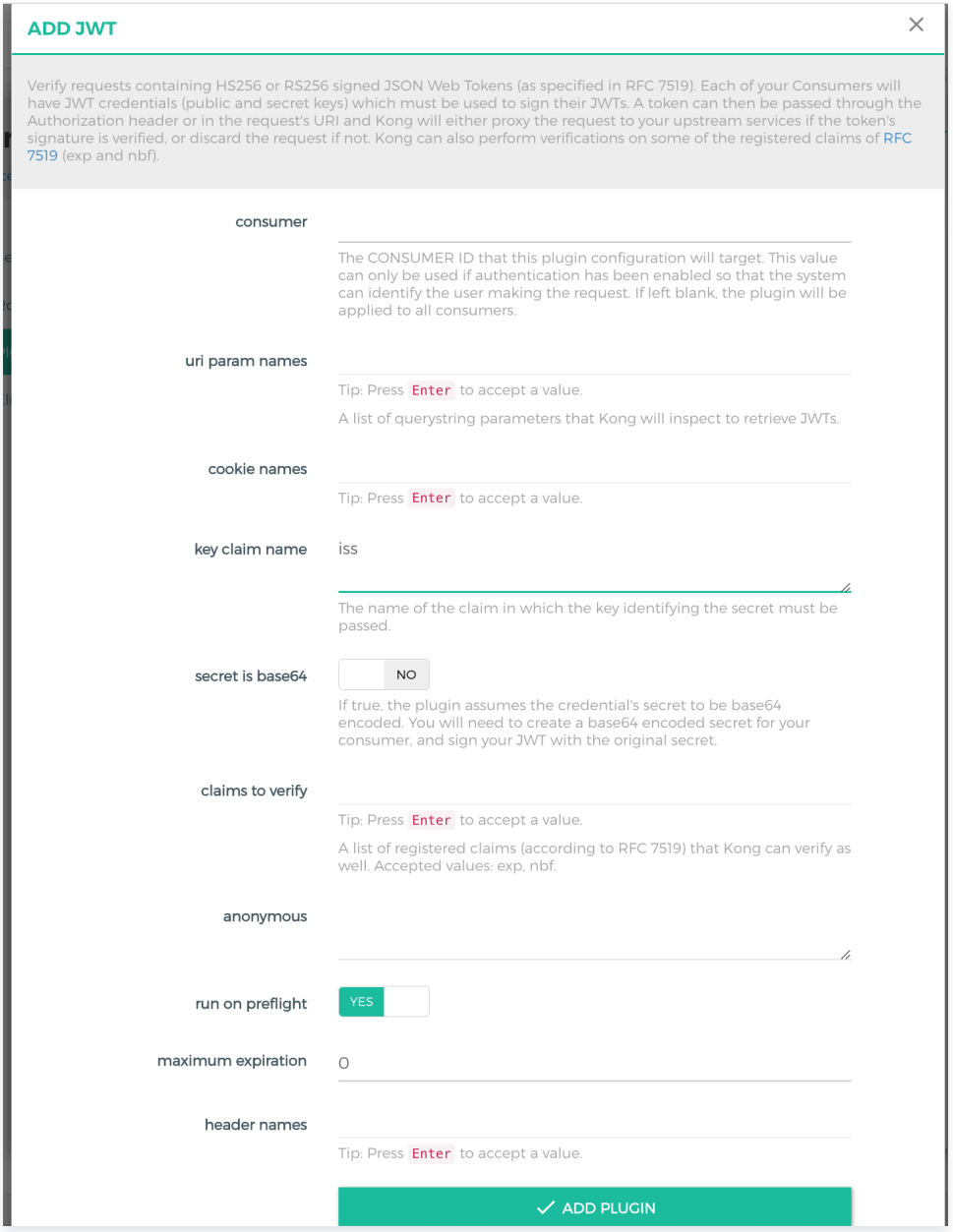



## 步骤2：绑定JWT插件

1. 进入需要认证的 Service，单击 **ADD PLUGIN**，在插件市场的 Authentication 分组中选择 Jwt 插件，单击 **Add Plugin**。



2. 填写 JWT Token 校验的信息，单击 **ADD PLUGIN**。




### 步骤3：测试请求

1. 不携带 JWT Token 请求服务，该请求被拒绝，返回401。

```
curl -i xxxxxxx/test

HTTP/1.1 401 Unauthorized
Connection: keep-alive
Content-Length: 26
Content-Type: application/json; charset=utf-8
Date: Tue, 29 Nov 2022 12:55:33 GMT
Server: kong/2.5.1
X-Kong-Response-Latency: 23

{"message":"Unauthorized"}
```

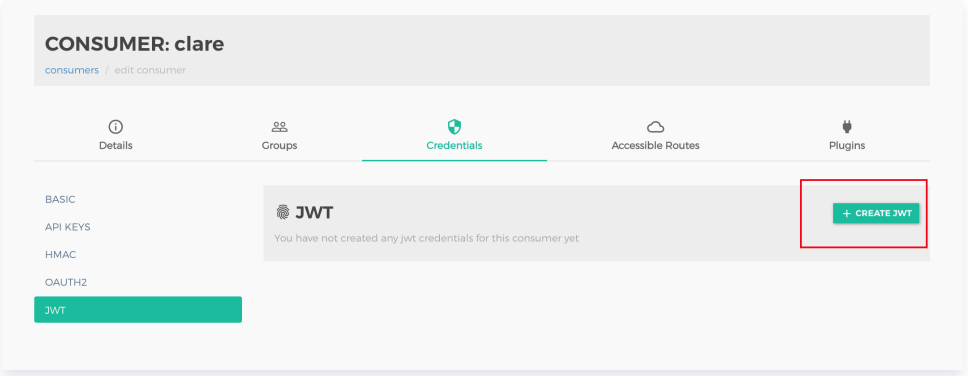
2. 携带 JWT Token 请求服务，请求成功。

```
curl -i 'http://xxxxxx/test' \
--header 'Authorization: Bearer
eyJhbGciOixxxxxxxxxxI6IkpXVCJ9.eyJpc3MiOiJoUXY4eGRtWxxxxxxxxxxxzFoQ0VUQnNySiJ9.APz7Kx9eIiV1Cx
AJUVt4i4-gvsJ56TtPxxxxxxK67VQ'

HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: application/json; charset=utf-8
Date: Tue, 29 Nov 2022 12:57:38 GMT
Server: apigw/1.0.15
Vary: Accept-Encoding
Via: kong/2.5.1
X-Api-Id: api-1nxxxxkuc
X-Api-Requestid: ab873xxxxd68cac394ddc208
X-Kong-Proxy-Latency: 7
X-Kong-Upstream-Latency: 6
Content-Length: 11

hello Kong
```

## 场景二：对使用 RS256 签名的请求进行认证鉴权

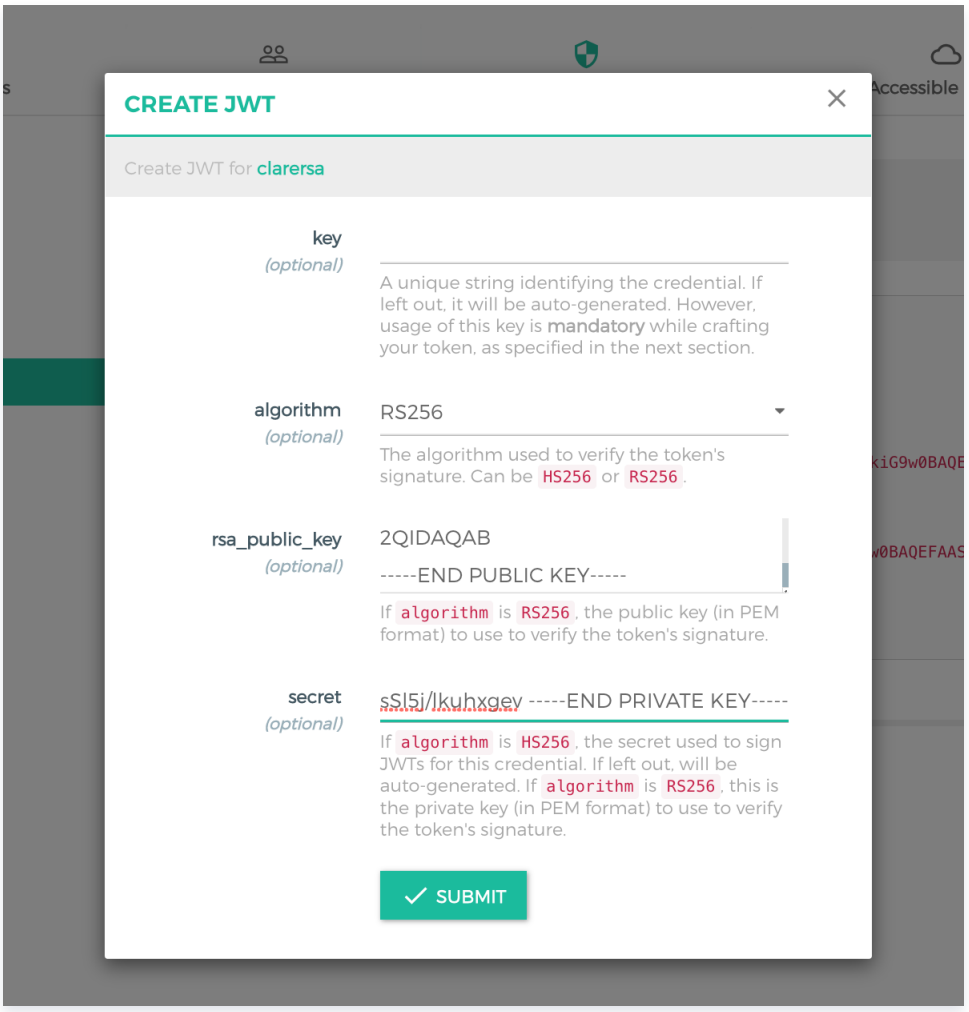
### 步骤1：创建 Consumer

1. 登录 TSF 控制台，进入需要配置限流插件的 Kong 网关实例详情页，在配置管理页查看管理控制台登录方式。
2. 登录到 Konga 控制台，进入 Consumer 详情页，选择需要配置访问控制的用户（如clarersa），单击 **Credentials Tab**，选择 JWT，单击 **创建 JWT**，为该用户创建一个 JWT Token 作为访问凭证。

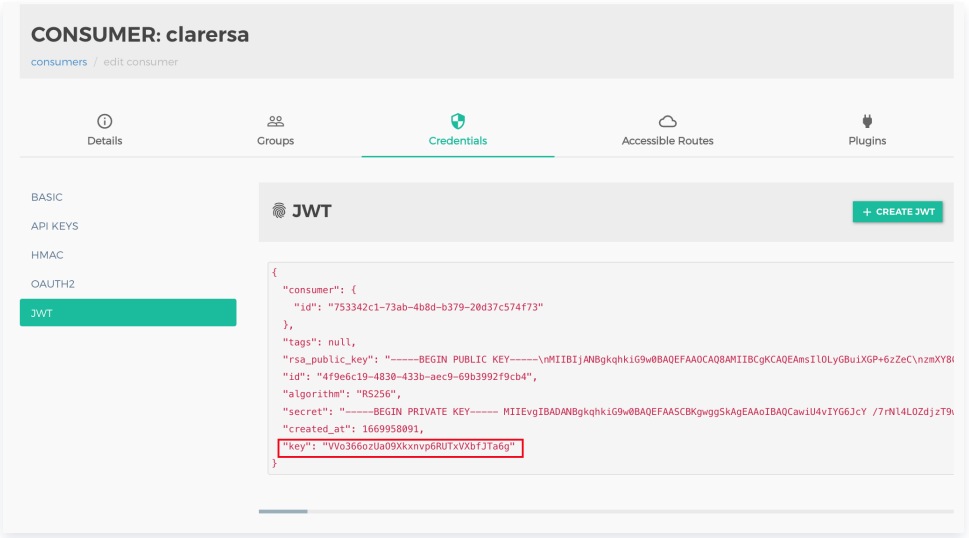



3. 配置JWT相关参数。

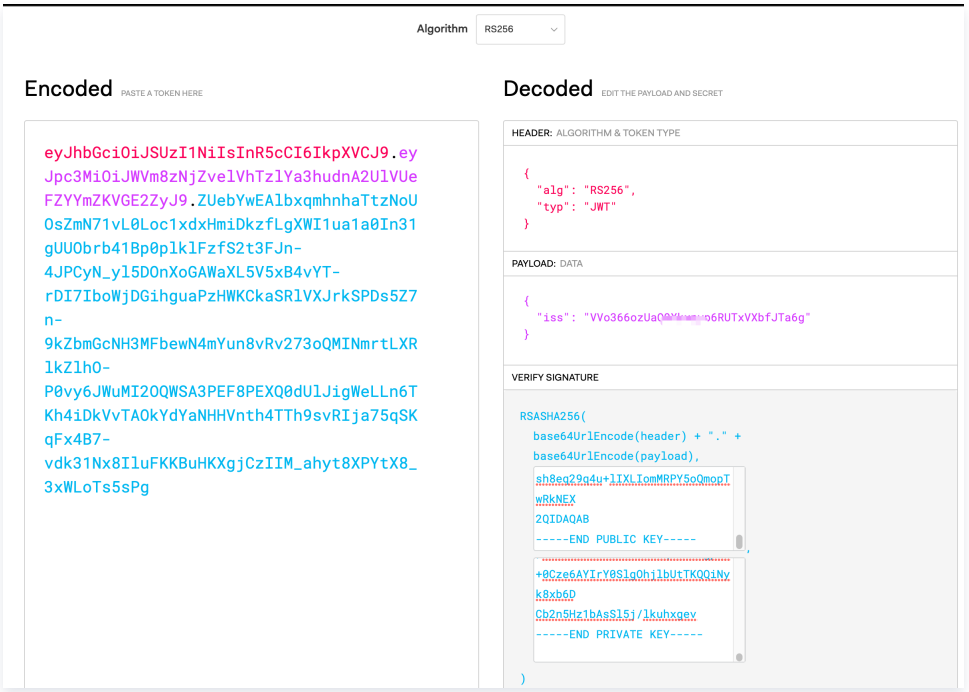
| 配置项 | 描述 |
| --- | --- |
| key | 对应 JWT Claims 的 issuer，即签发人。留空则自动生成。 |
| algorithm | JWT Token 签名所使用的加密算法，支持 HS256 和 RS256。本场景设置为 RS256。 |
| rsa_public_key | 当签名算法为 RS256 时，设置公钥（PEM 格式），用于验证签名。 |
| secret | 当签名算法为 RS256 时，设置私钥（PEM 格式），用于验证签名。 |


CREATE JWT
Create JWT for clarersa
key
(optional)
A unique string identifying the credential. If left out, it will be auto-generated. However, usage of this key is mandatory while crafting your token, as specified in the next section.
algorithm
(optional)
RS256
The algorithm used to verify the token's signature. Can be HS256 or RS256.
rsa_public_key
(optional)
2QIDAQAB
-----END PUBLIC KEY-----
If algorithm is RS256, the public key (in PEM format) to use to verify the token's signature.
secret
(optional)
sSI5j/lkuhxgev -----END PRIVATE KEY-----
If algorithm is HS256, the secret used to sign JWTs for this credential. If left out, will be auto-generated. If algorithm is RS256, this is the private key (in PEM format) to use to verify the token's signature.
SUBMIT

4. 查看生成的用户凭证。

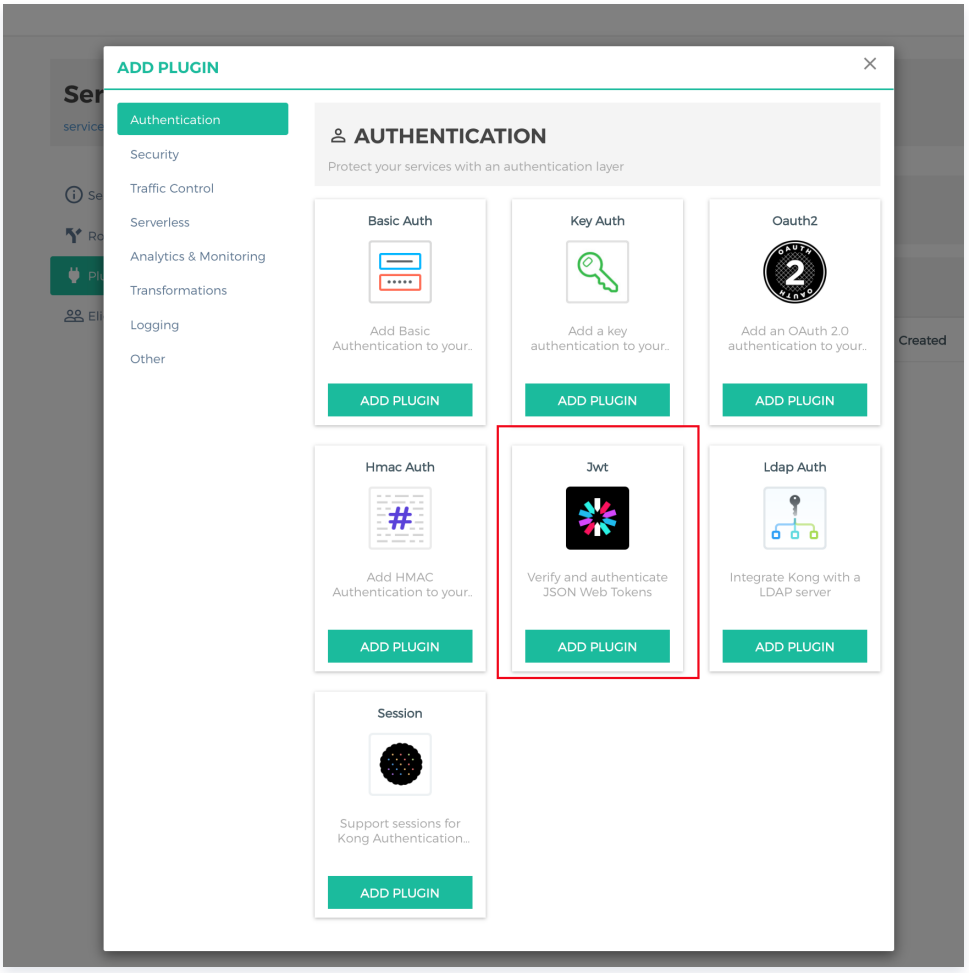

CONSUMER: clarersa
consumers / edit consumer
Details
Groups
Credentials
Accessible Routes
Plugins
BASIC
API KEYS
HMAC
OAUTH2
JWT
JWT
+ CREATE JWT
"consumer": {
"id": "753342c1-73ab-4b8d-b379-20d37c574f73"
},
"tags": null,
"id": "4f9e6c19-4830-433b-aec9-69b3992f9cb4",
"algorithm": "RS256",
"created_at": 1669958091,
"key": "VVo366ozUaO9XkxnvpGRUTxVXbfJTa6g"

5. 生成对应的 JWT Token。

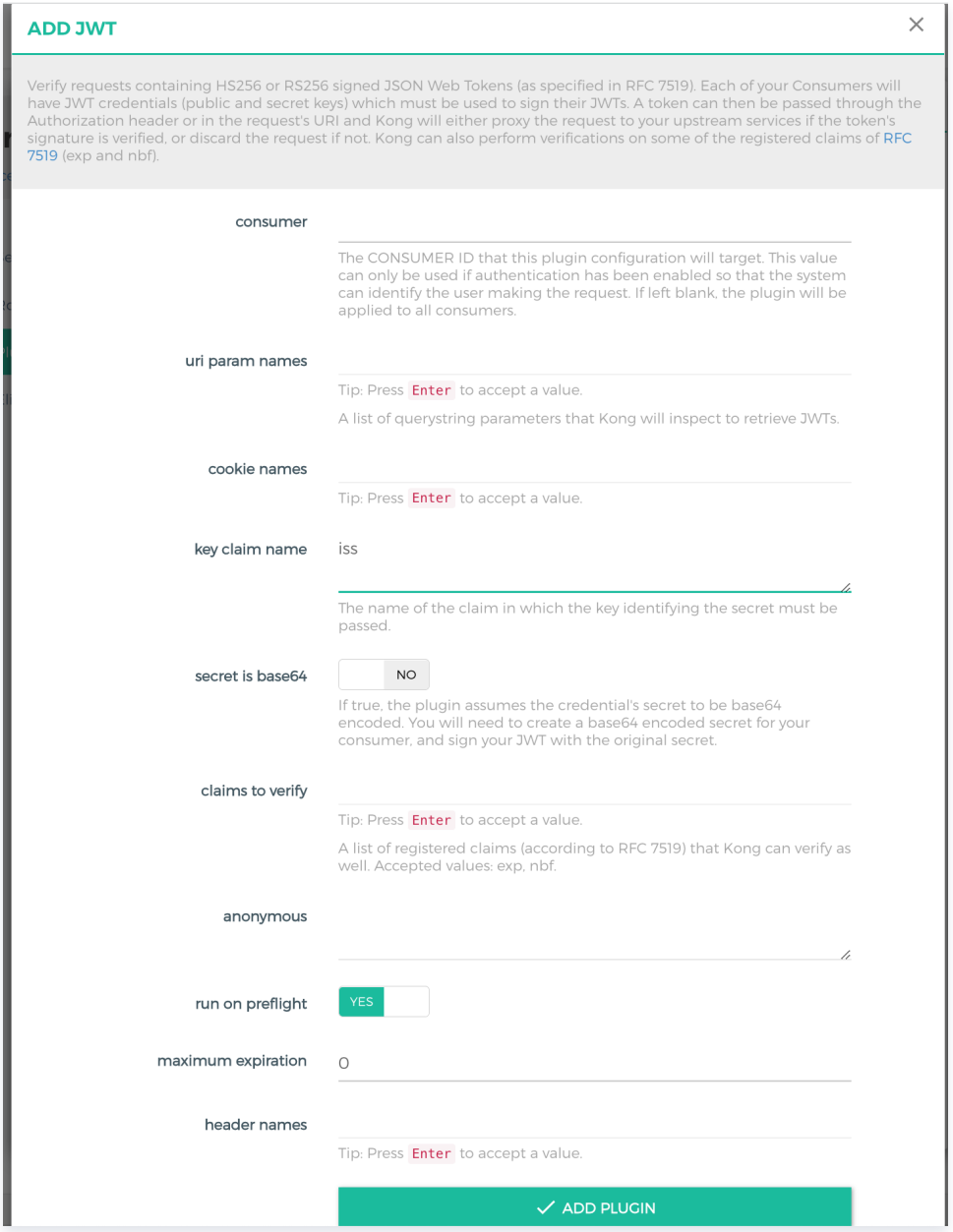



## 步骤2：绑定 JWT 插件

1. 进入需要认证的 Service，单击 **ADD PLUGIN**，在插件市场的 Authentication 分组中选择 Jwt 插件，单击 **Add Plugin**。



2. 填写 JWT Token 校验的信息，单击 **ADD PLUGIN**。




### 步骤3：测试请求

1. 不携带 JWT Token 请求服务，该请求被拒绝，返回401。

```
curl -i xxxxxxx/testrsa

HTTP/1.1 401 Unauthorized
Connection: keep-alive
Content-Length: 26
Content-Type: application/json; charset=utf-8
Date: Tue, 29 Nov 2022 12:55:33 GMT
Server: kong/2.5.1
X-Kong-Response-Latency: 23

{"message":"Unauthorized"}
```

2. 携带 JWT Token 请求服务，请求成功。

```
curl -i 'http://xxxxxx/testrsa' \
--header 'Authorization: Bearer
eyJhbGciOixxxxxxxxxxI6IkpXVCJ9.eyJpc3MiOiJoUXY4eGRtWxxxxxxxxxxxzFoQ0VUQnNySiJ9.APz7Kx9eIiV1Cx
AJUVt4i4-gvsJ56TtPxxxxxxK67VQ'

HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: application/json; charset=utf-8
Date: Tue, 29 Nov 2022 12:57:38 GMT
Server: apigw/1.0.15
Vary: Accept-Encoding
Via: kong/2.5.1
X-Api-Id: api-1nxxxxkuc
X-Api-Requestid: ab873xxxxd68cac394ddc208
X-Kong-Proxy-Latency: 7
X-Kong-Upstream-Latency: 6
Content-Length: 11

hello Kong
```

## 参考

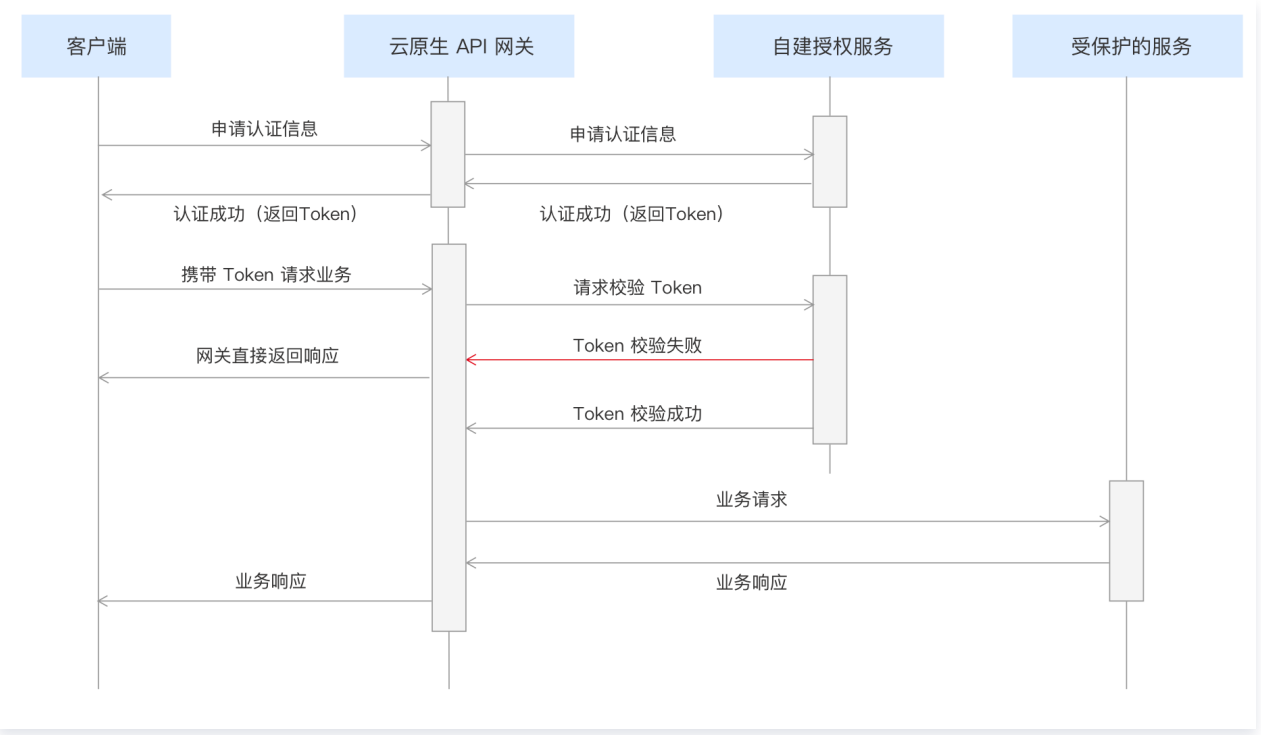
更多相关说明请参见 Kong JWT 插件官方文章 。

# 配置自定义鉴权

最近更新时间：2025-06-09 16:07:02

## 功能介绍

云原生网关提供自定义鉴权的能力，方便实现在网关入口进行统一鉴权。当客户端请求携带的凭证信息（Token）为自定义的格式时，网关可根据配置将请求转发到中心化的鉴权服务来完成鉴权操作，并根据鉴权结果转发/拒绝请求，以保护 API 接口的通信安全。
云原生网关自定义鉴权流程如下：




云原生网关自定义鉴权提供如下能力：

- 支持配置在网关服务或路由上，以针对特定路由或特定服务下所有路由进行自定义鉴权。
- 支持配置鉴权服务不可用策略，包括直接放通和直接拒绝。
- 支持配置路由白名单，简化使用。即服务下某些白名单路由将不进行鉴权，其余路由进行鉴权。
- 支持配置鉴权结果返回策略，包括通过HTTP 状态码标识和通过响应头标识。
- 鉴权请求支持携带 Header 和 body。
- 鉴权响应支持保留 Header 和 body（通过响应头传递）。
- 支持未携带Token请求时跳转到重定向地址进行 token 获取。

## 插件配置说明

| 字段名称 | 字段含义 | 说明 |
| --- | --- | --- |
| description | 插件描述 | - |
| auth service id | 鉴权服务 ID | 可在 控制台 > 路由管理 > 服务信息中获取 |
| auth service path | 鉴权 API 路径 | - |
| token key name | 存放 Token 的请求头名称 | - |
| route id whitelist | 路由白名单，路由 ID 列表不进行鉴权。 | - |

| | | |
|---|---|---|
| timeout | 超时时间：鉴权服务超时时间，默认 10 秒。 | - |
| auth server unavailable policy | 鉴权服务不可用策略，当鉴权服务不可用时，网关处理客户端请求的策略。支持直接放通（allow）和直接拒绝（deny）两种。 | • 直接放通（allow）：当鉴权服务不可用时，网关直接放通客户端请求到后端服务。<br>• 直接拒绝（deny）：当鉴权服务不可用时，网关拒绝客户端请求，返回403状态码，返回 Header 标识鉴权服务不可用。 |
| redirect path | 请求未携带 Token 时的重定向路径 | 网关返回 302 进行重定向，重定向 URL 拼接规则为：`{请求scheme}://{请求Host}/{redirect_path}` |
| allow request body | 鉴权请求中是否允许携带Body | - |
| allow request headers | 鉴权请求允许携带的 Header 列表 | - |
| auth result identifier | 鉴权结果返回策略，支持通过HTTP状态码标识（HTTP_status_code）或通过响应头（response_header）返回两种 | • HTTP 状态码标识（HTTP_status_code）：鉴权服务返回 200 时，鉴权成功；其余均认为鉴权失败。<br>• 通过响应头返回（response_header）：鉴权服务器到网关返回 HTTP 200 ,通过自定义响应头标识鉴权结果，当指定响应对应值为 true 时，鉴权成功，响应头对应值为 false 或空时，鉴权失败。 |
| auth result response header | 鉴权结果自定义响应头 | - |
| auth body header | 鉴权响应 Body 所属响应头 | 支持将鉴权服务返回 Body 存放在自定义响应头中传回客户端。 |
| allow response headers | 鉴权响应中允许携带的 Header 列表 | - |

# 使用场景

## 前提步骤

1. 创建需要访问的后端服务 nginx 和路由。

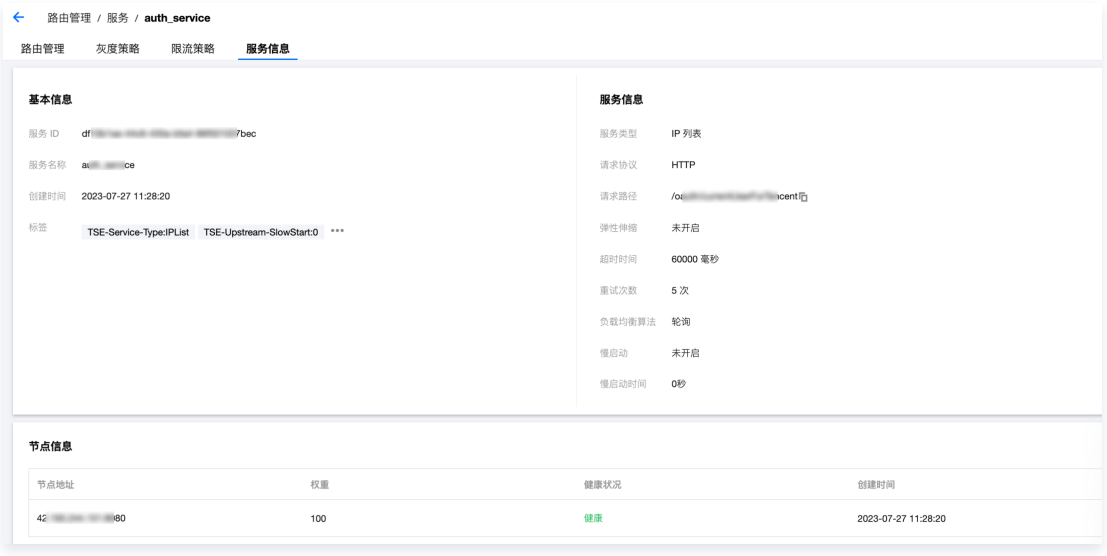





2. 新建鉴权服务 auth_service，根据鉴权服务信息，填写服务类型、地址列表、请求协议。

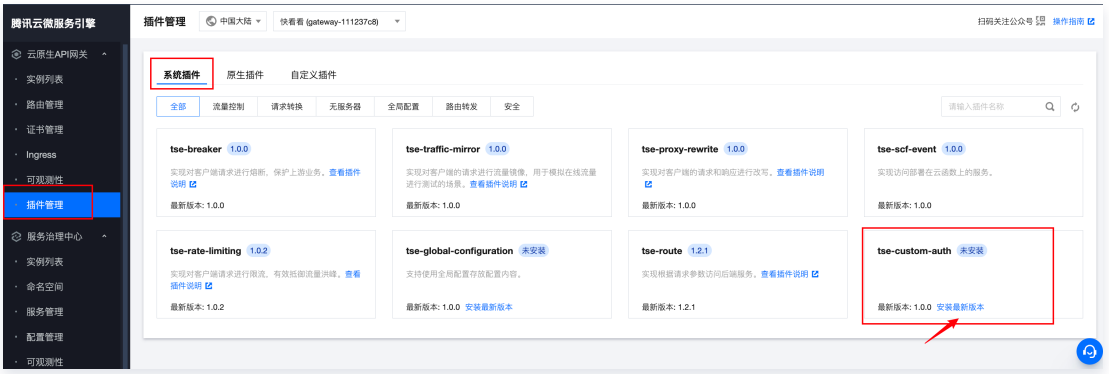



**单击确定，在服务信息Tab 查看鉴权服务的服务信息。**



3. 安装自定义鉴权插件

进入**控制台 > 插件管理**，选择**系统插件 > tse-custom-auth**，点击**安装最新版本**即可安装。

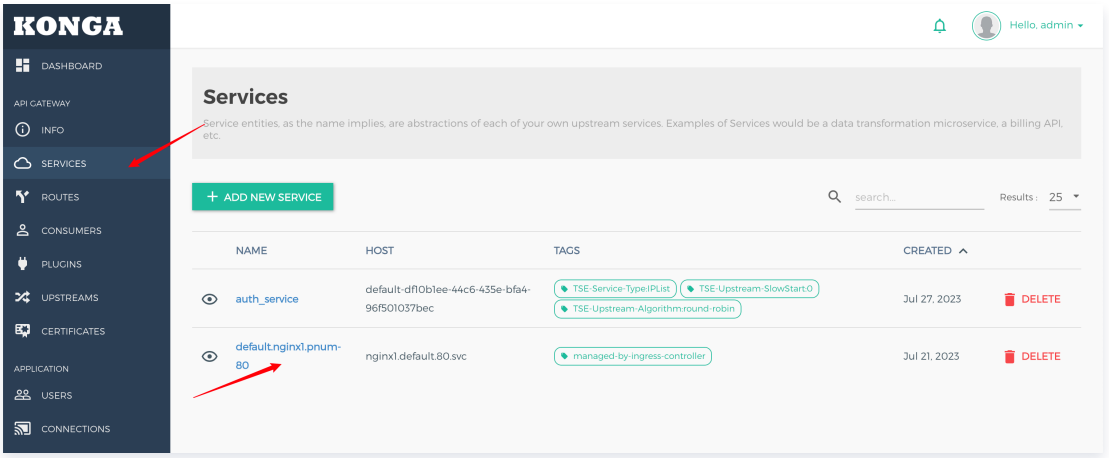



## 场景一：鉴权结果通过 HTTP 状态码标识

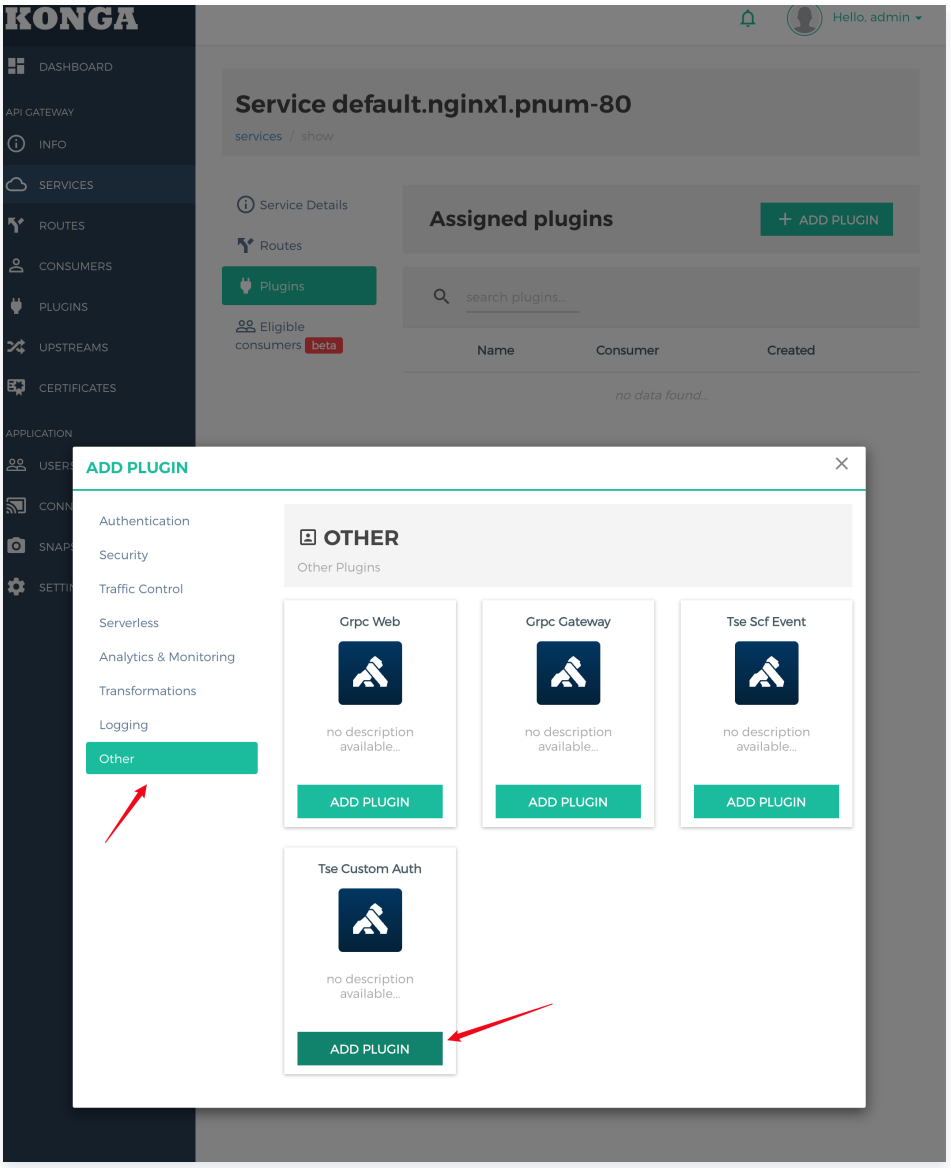
**场景说明：**当您的鉴权服务器使用HTTP状态码标识鉴权结果时，可配置 auth result identifier 为 HTTP status code，网关将根据鉴权服务器返回的状态码判定鉴权结果，当状态码为 200 时认为鉴权成功，网关将转发请求到后端服务；其余情况则认为鉴权失败，拒绝请求。

**操作步骤：**

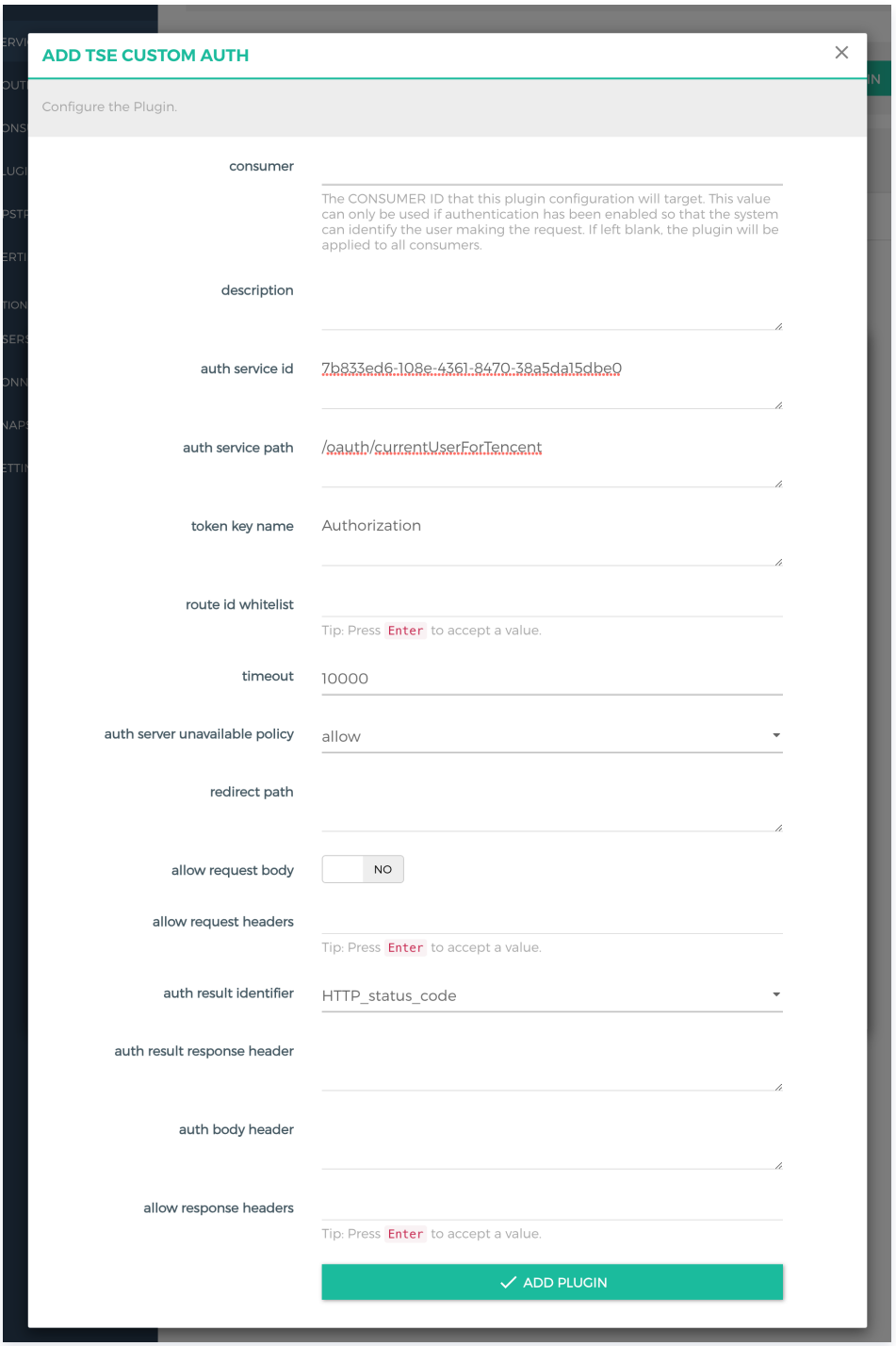
1. 登录 Konga，单击 nginx 服务进入详情。



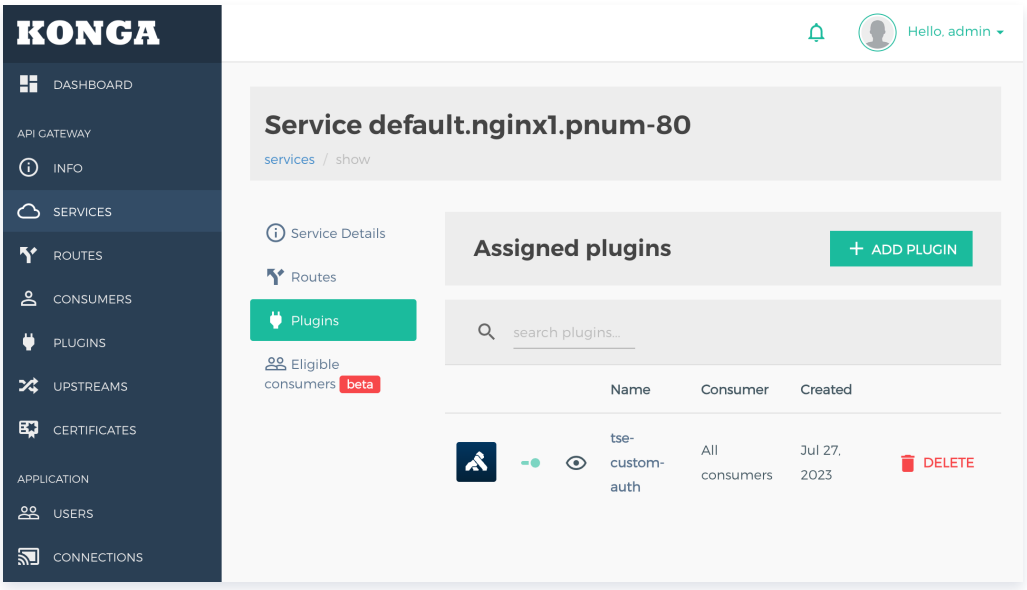

2. 单击 **Plugins > +ADD Plugin**添加自定义鉴权插件



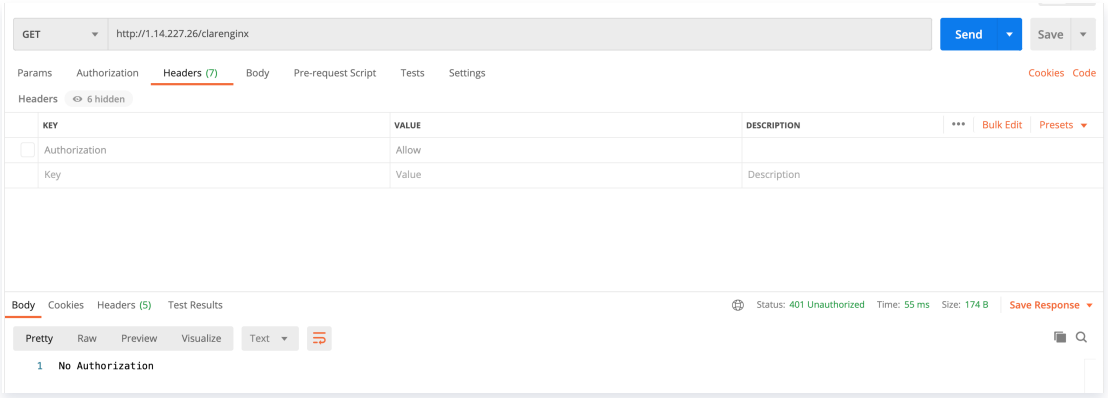

3. 填写插件配置，点击 ADD PLUGIN，设置**鉴权结果返回策略**为 HTTP 状态码。



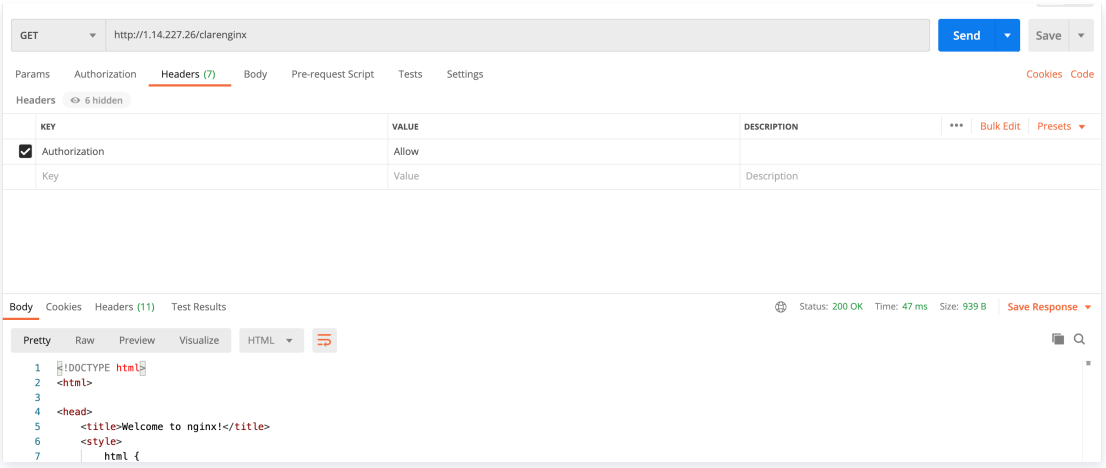

4. 确认插件已绑定后端服务。



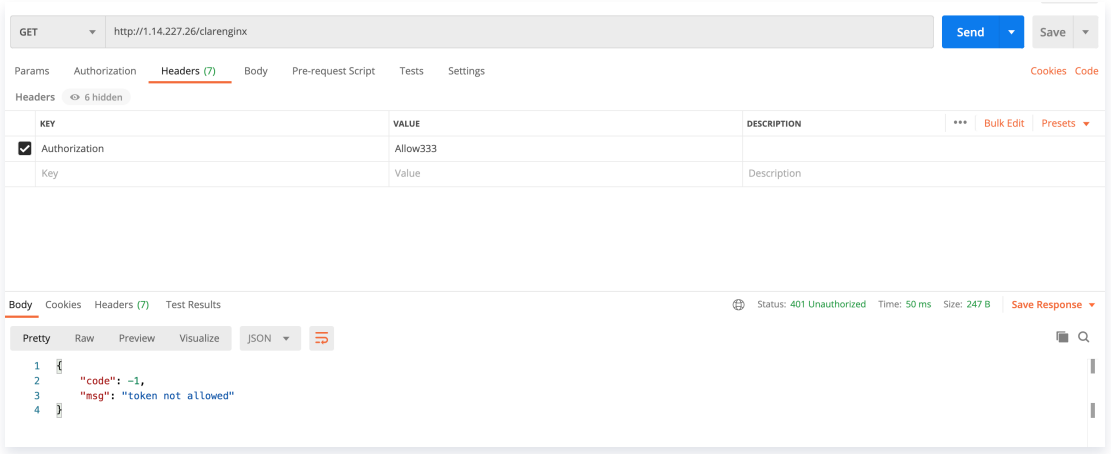

5. 未携带Token访问后端服务，访问被拒绝，返回 401。




6. 携带正确的Token访问后端服务，成功访问后端服务。




7. 携带错误的Token访问后端服务，访问被拒绝，返回 401。

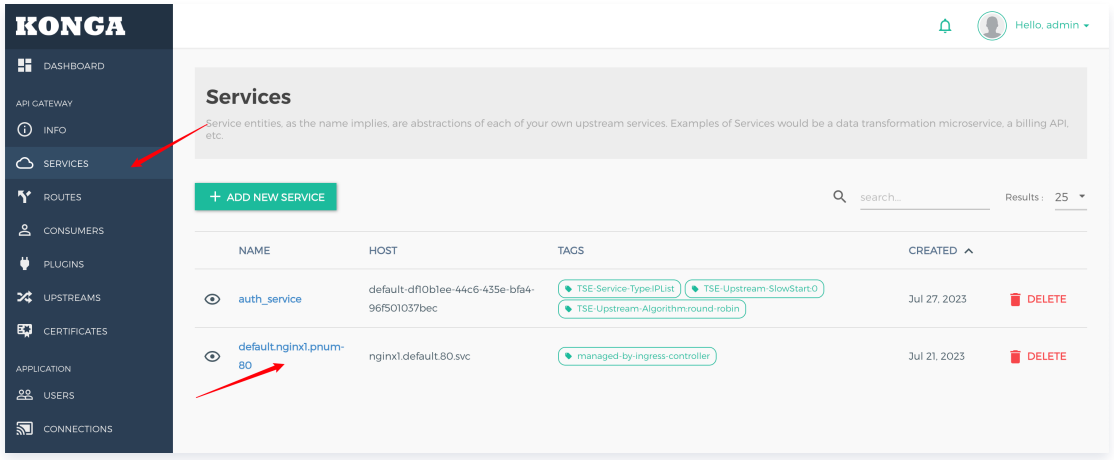



## 场景二：鉴权结果通过响应头返回失败信息

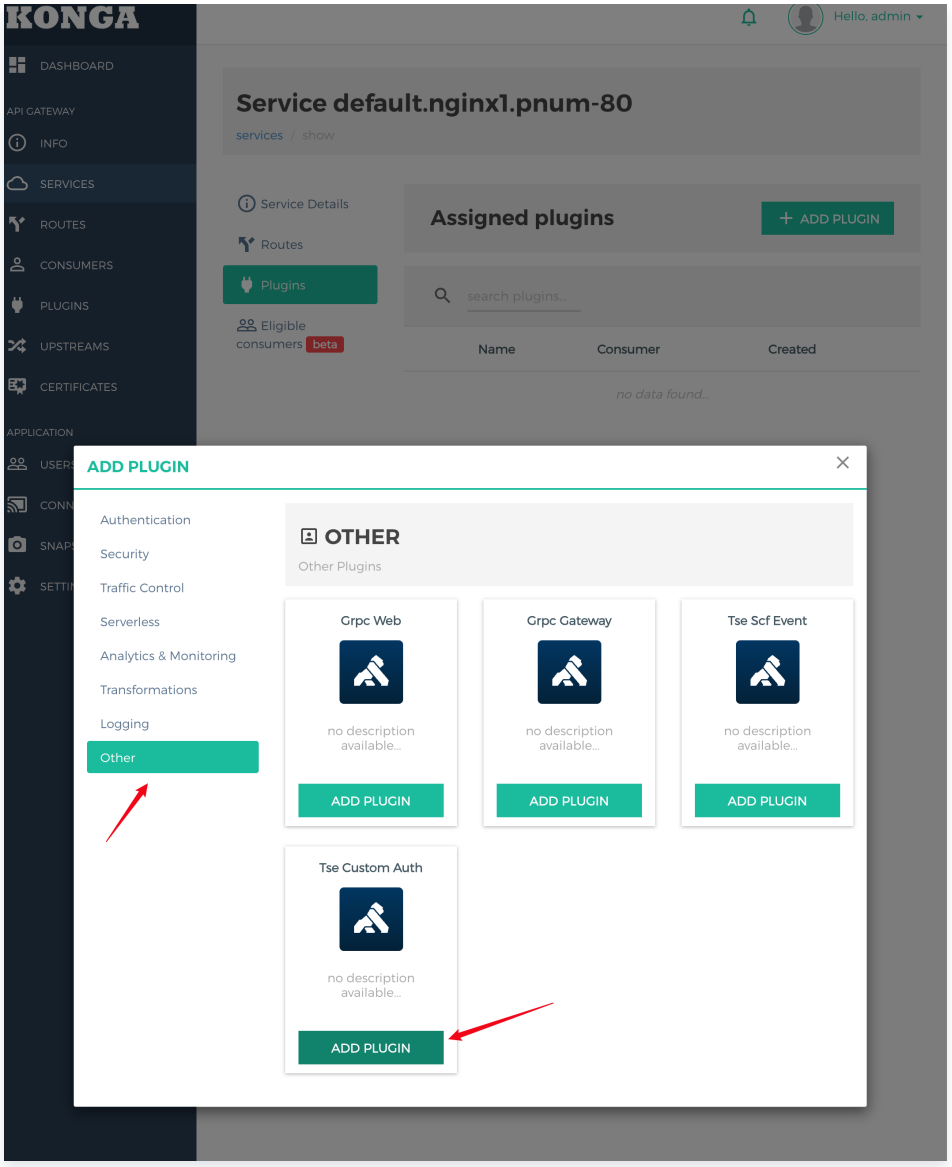
**场景说明：**当您的客户端始终接受 200 状态码，需要通过响应头来标识鉴权结果时，可配置 auth result identifier 为 response header，并定义一个存放鉴权结果的 Header 头。网关将尝试获取该响应头的值判定鉴权结果，当 Header 值为 true 时认为鉴权成功，网关将转发请求到后端服务；其余情况则认为鉴权失败，拒绝请求。

**操作步骤：**

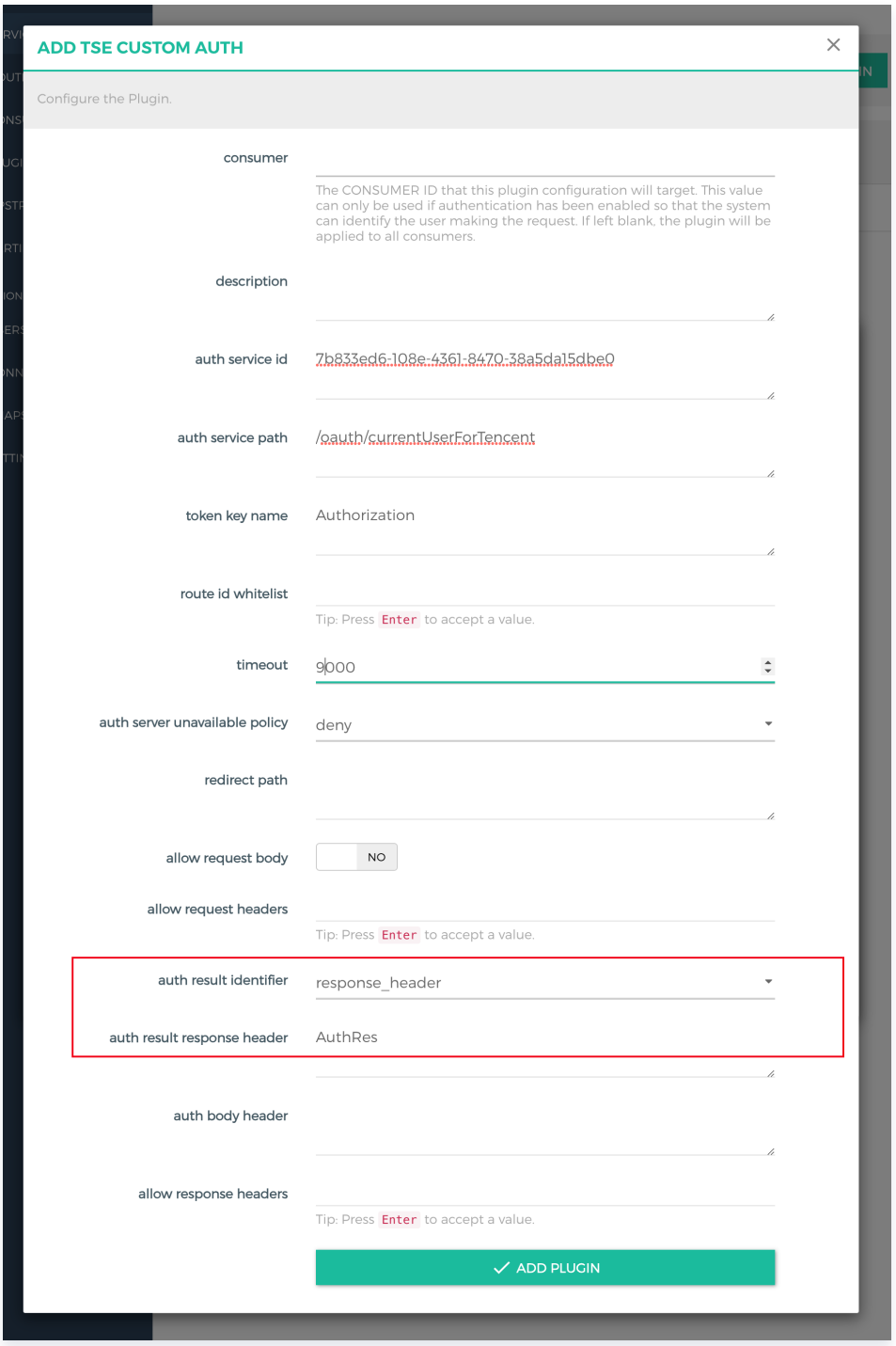
1. 登录 Konga，单击 nginx 服务进入详情。



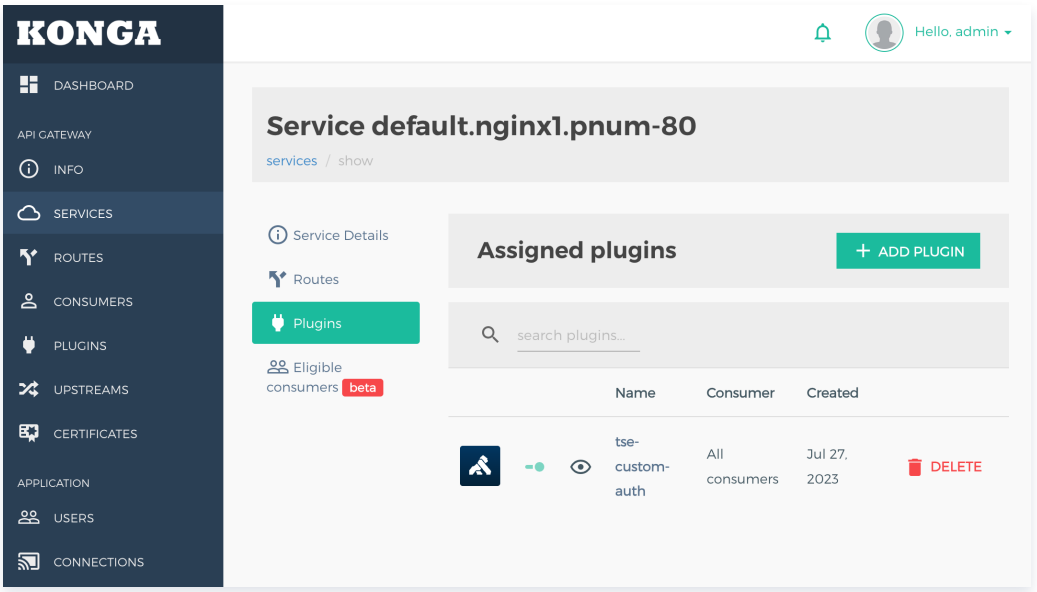

2. 单击 **Plugins** > **+ADD Plugin** 添加自定义鉴权插件。



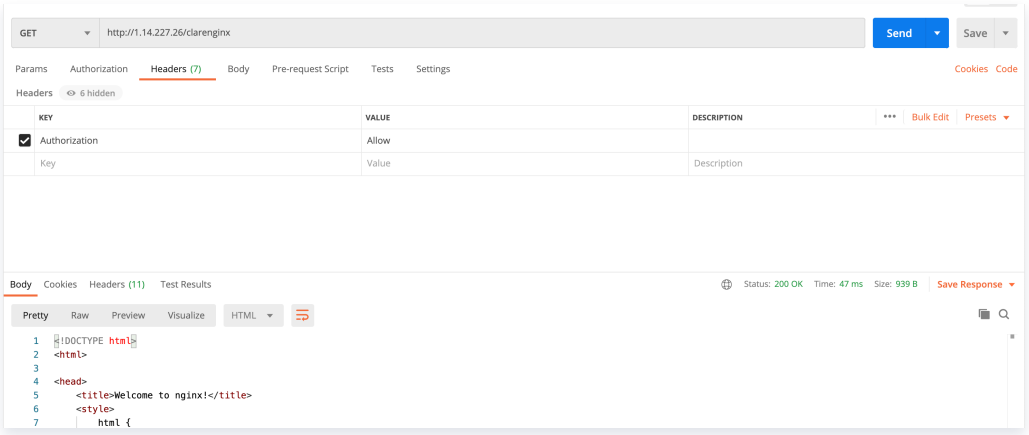

3. 填写插件配置，单击 ADD PLUGIN，设置**鉴权结果返回策略**为响应头，并定义鉴权结果所在的响应头 Header 为 AuthRes。



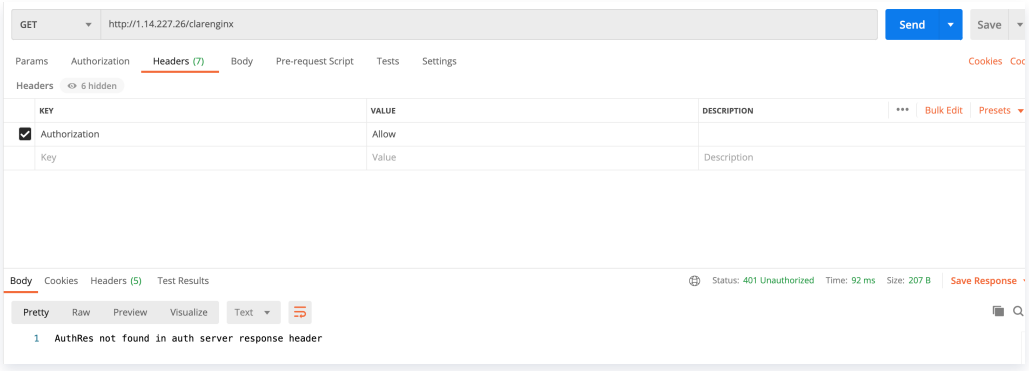

4. 确认插件已绑定后端服务。




5. 鉴权服务返回响应头 AuthRes=true 时，鉴权通过。




6. 鉴权服务未返回响应头 AuthRes时，鉴权失败。

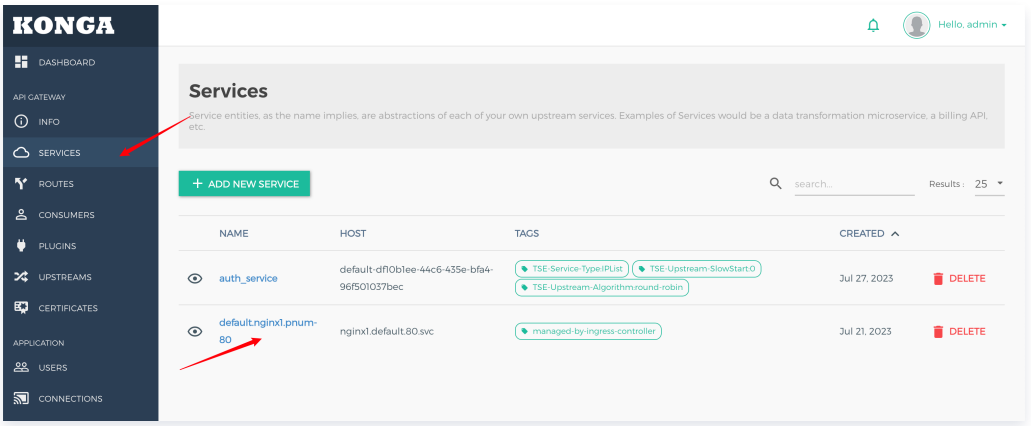



## 场景三：鉴权服务不可用场景

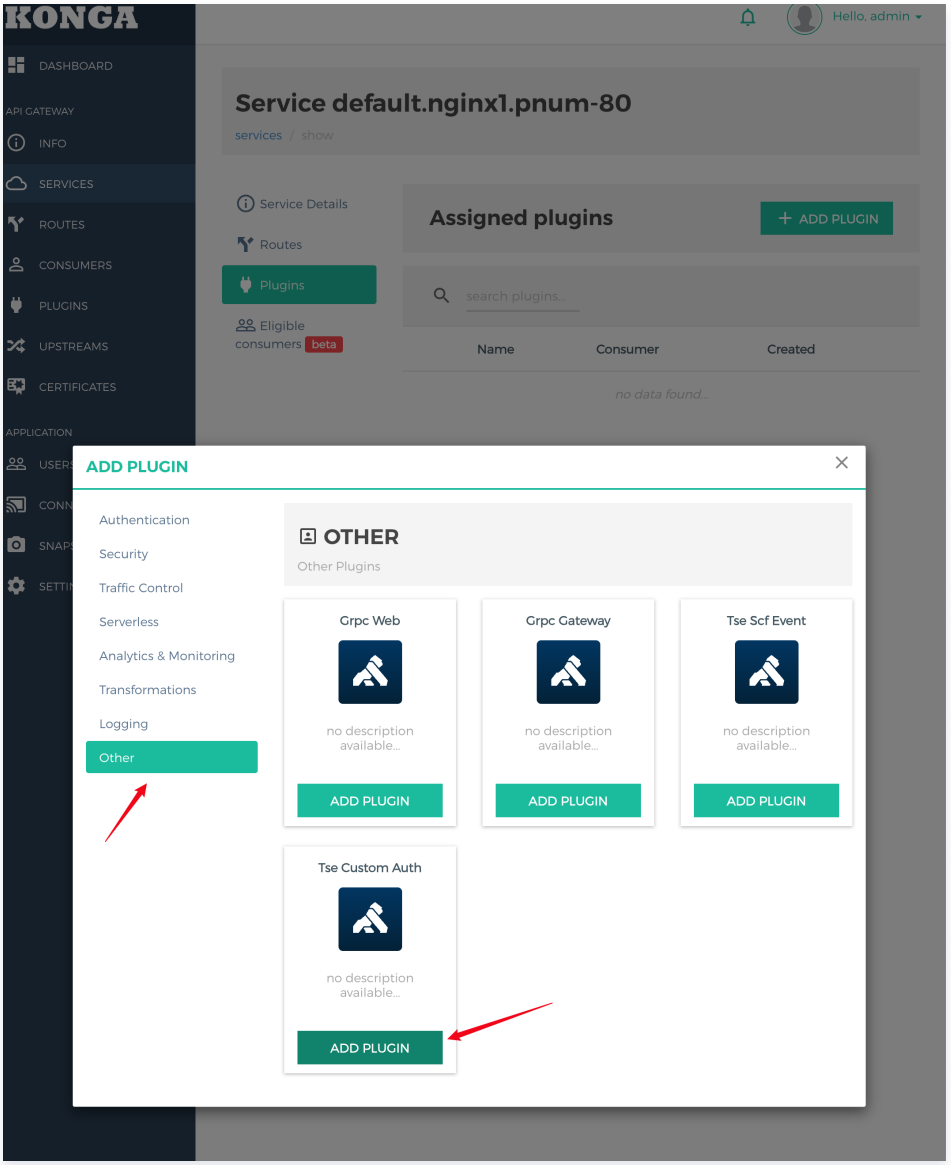
**场景说明：**当鉴权服务不可用（没有回包）时，网关会根据您的配置对请求进行全部放通/全部拒绝。

**操作步骤**

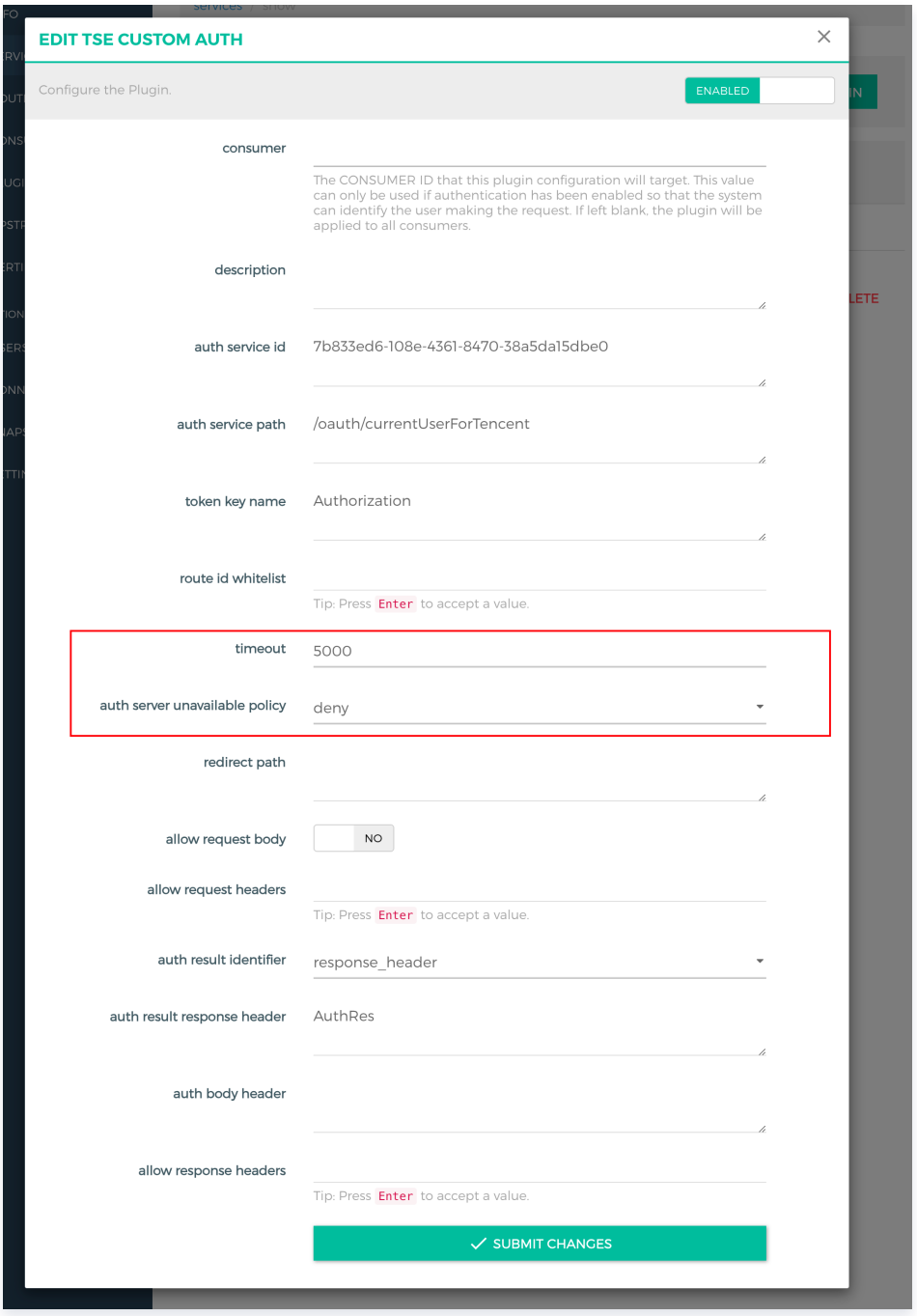
1. 登录 Konga，点击 nginx 服务进入详情。



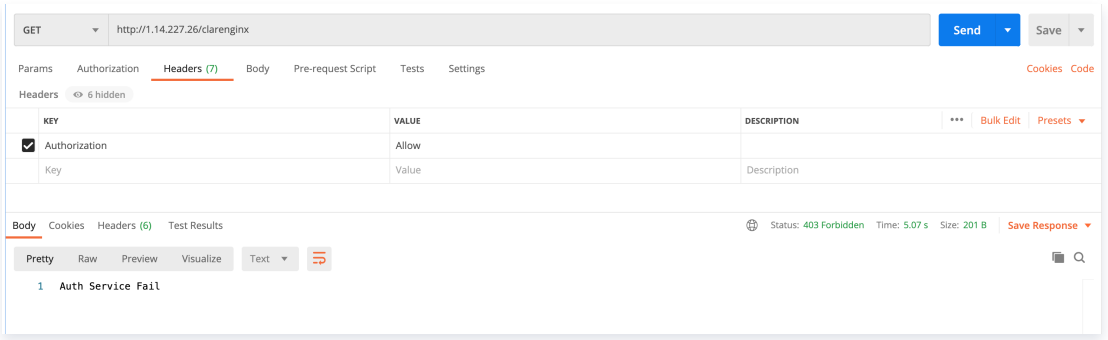

2. 点击 **Plugins** > **+ADD Plugin** 添加自定义鉴权插件。

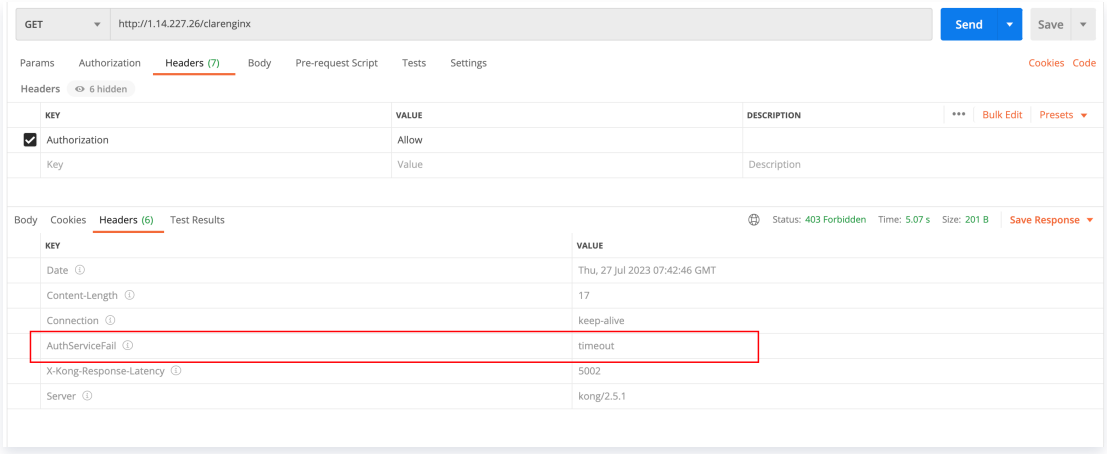



3. 填写插件配置，点击 ADD PLUGIN，设置鉴权服务不可用策略为直接拒绝，配置超时时间为 5000（即 5 秒）。

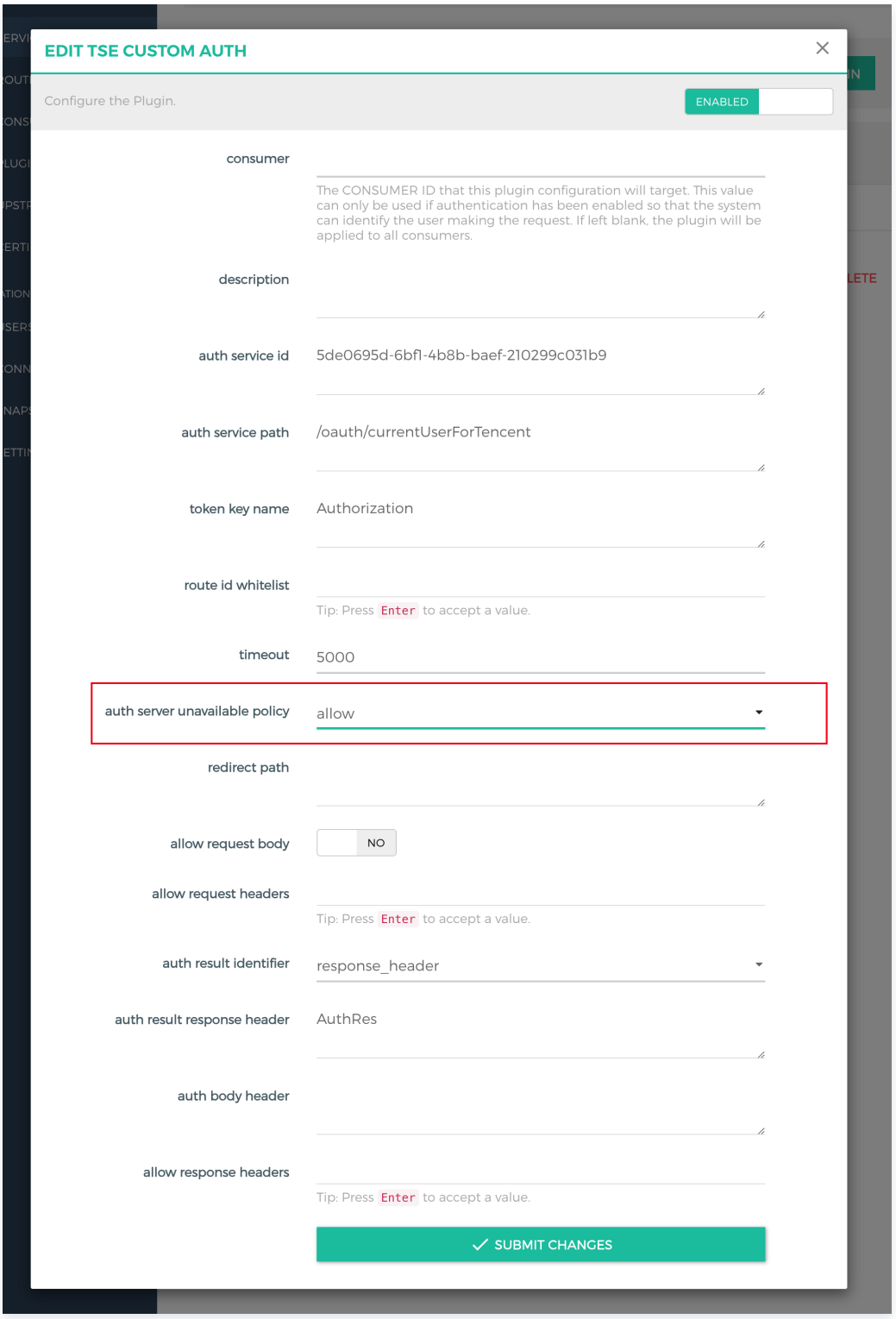



4. 模拟鉴权服务故障，携带 Token 请求服务，请求在达到超时时间后拒绝，返回的 Header 中标注失败原因。

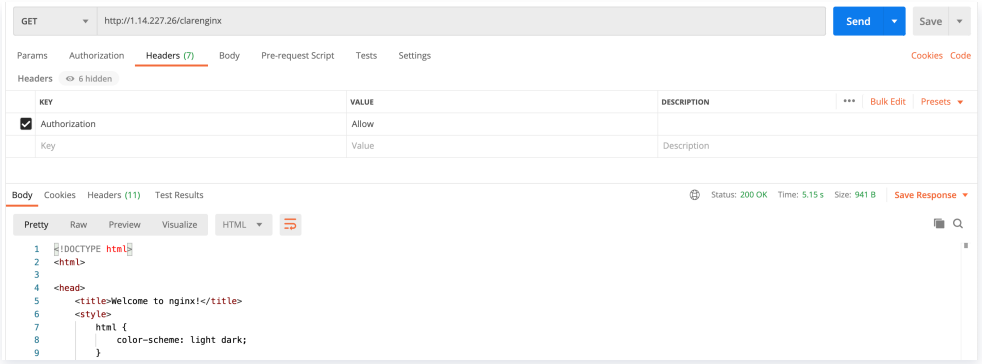





5. 修改插件配置，设置鉴权服务不可用策略为直接放通。




6. 模拟鉴权服务故障，携带 Token 请求服务，请求在达到超时时间后认为鉴权服务无响应，直接放通请求。




# Ingress 配置方式

1. 定义 KongPlugin 资源，可参考如下示例：

```
apiVersion: configuration.konghq.com/v1
kind: KongPlugin
metadata:
  name: my-test-auth
config:
  timeout: 10000
  allow_request_body: true
  token_key_name: Authorization
  auth_result_identifier: response_header
  auth_result_response_header: AuthResponse
  auth_service_path: /
  auth_body_header: AuthBody
  redirect_path: null
  allow_response_headers:
    - myRespHeader1
    - myRespHeader2
    - myRespHeader3
  allow_request_headers:
    - myReqHeader
  auth_service_id: 73efd5bb-7468-43bc-9ca6-03f118169ca2
  route_id_whitelist: null
  description: null
  auth_server_unavailable_policy: allow
plugin: tse-custom-auth
```

2. 将插件绑定在 service 或 route 上。

```
kubectl apply -f tseauth.yaml
kubectl annotate service nginx konghq.com/plugins=my-test-auth
```

版权所有：腾讯云计算（北京）有限责任公司

# 流量控制
# 使用缓存加速访问

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何在 Kong 云原生 API 网关上通过缓存（Proxy Cache）插件实现下述常见缓存场景：

- 针对指定 API 开启缓存插件。
- 通过 Admin API 查看指定缓存。
- 通过 Admin API 删除指定缓存。

## 前置条件

- 已购买 Kong 网关实例，详情请参见 操作文档 。
- 配置了后端（Service）以及路由（Route）。

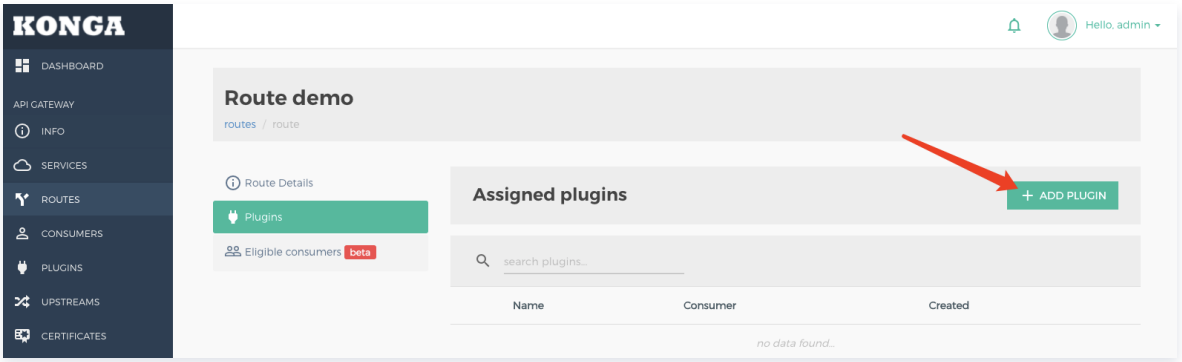
## 操作步骤

### 场景一：针对指定 API 开启缓存插件

1. 登录 TSF 控制台，进入需要配置限流插件的 Kong 网关实例详情页，在 Konga 控制台页面查看管理控制台登录方式。

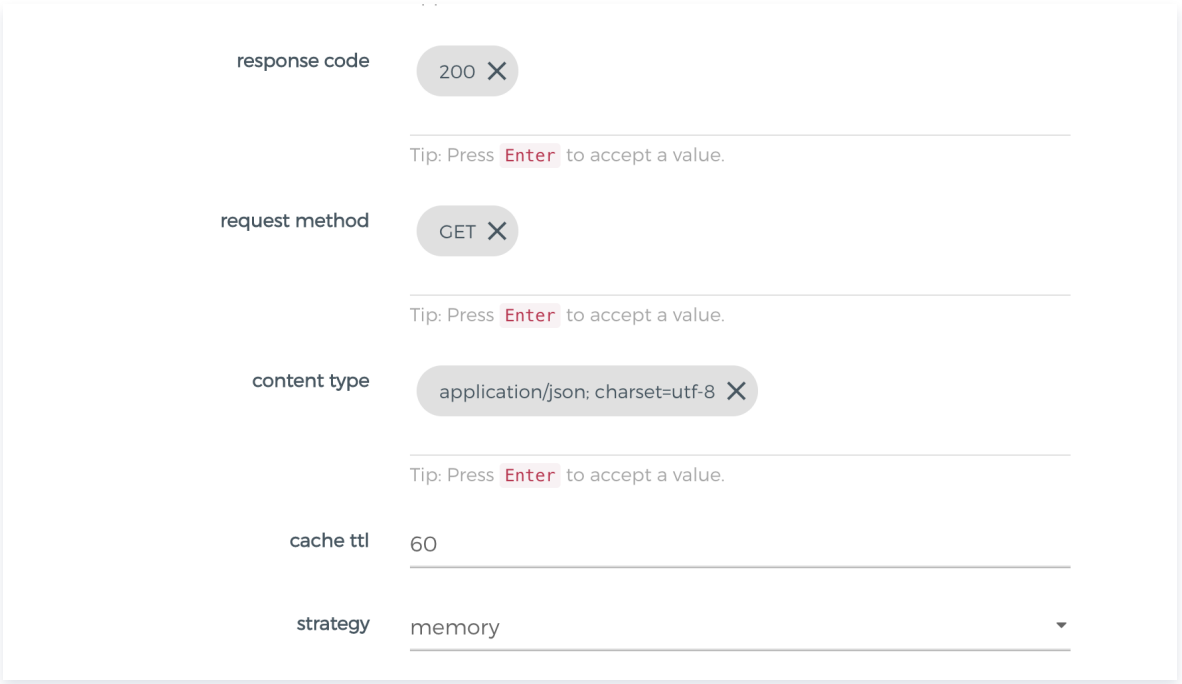



2. 登录 Konga 管理控制台，进入需要限流的 **Route 详情页**，单击 **Add Plugin** 按钮创建插件，在 **Traffic Control** 分组下选择 **Proxy Cache** 插件。

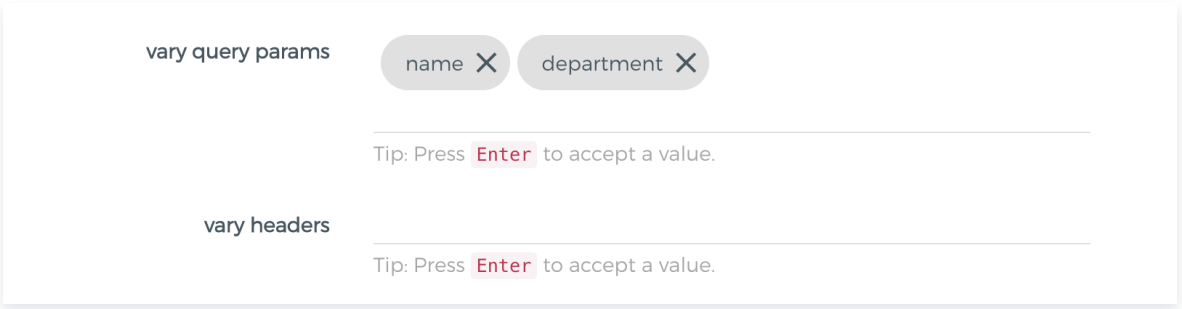



3. 在插件配置中填写以下配置，点击 enter键，并保存。
   - `response code`：200 // 允许缓存的后端响应码列表。建议输入正常响应码（例如 200），避免异常响应内容被缓存。

- `request method`：GET // 允许缓存的请求方法。
- `content type`：application/json; charset=utf-8 // 允许缓存的后端响应 Content-Type，注意该值必须和响应头完全匹配，例如 "application/json" 和 "application/json; charset=utf-8" 是不匹配的。
- `cache ttl`：60 // 缓存时长，单位秒。




4. 如果需要根据参数维度缓存，则需要额外配置。
   - `vary query params`：name, department // 用于缓存 key 的 query 参数名称列表，默认为空。
   - `vary headers`：x-cache-header // 用于缓存 key 的 header 参数名称列表，默认为空。




5. 发起 API 请求，验证缓存生效。
   - 首次请求 API URL `GET /cache/it`，缓存不存在，会请求到后端，响应头 `X-Cache-Status` 为 `Miss`

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 335
X-Cache-Key: 6fb1b6a4440980c758b0eff31177a41d
X-Cache-Status: Miss
Date: Tue, 26 Apr 2022 16:04:19 GMT
X-Kong-Upstream-Latency: 948
X-Kong-Proxy-Latency: 321
Via: kong/2.4.1
```

   - 缓存 TTL 之内再次请求 `GET /cache/it`，会命中缓存，响应头 `X-Cache-Status` 为 `Hit`

```
HTTP/1.1 200 OK
Server: openresty
Content-Type: application/json; charset=utf-8
X-Cache-Key: 6fb1b6a4440980c758b0eff31177a41d
Age: 2
X-Cache-Status: Hit
Date: Tue, 26 Apr 2022 16:09:23 GMT
Vary: Accept-Encoding
Content-Length: 335
X-Kong-Upstream-Latency: 0
X-Kong-Proxy-Latency: 0
Via: kong/2.4.1
```

- 当 `vary query params` 配置了 `name` 参数：
  - 首次请求 `GET /cache/it?name=john`，缓存不存在，会请求到后端，响应头 `X-Cache-Status` 为 `Miss`。
  - 缓存 TTL 之内再次请求 `GET /cache/it?name=john`，会命中缓存，响应头 `X-Cache-Status` 为 `Hit`。
  - 缓存 TTL 之携带不同 name 参数请求 `GET /cache/it?name=tom`，由于 name 参数不同，响应头 `X-Cache-Status` 为 `Miss`。

## 场景二：通过 Admin API 查看指定缓存

1. 通过 Proxy 代理方式开放 admin api（建议开启安全认证）。
2. 调用 Proxy Cache API `GET /proxy-cache/:cache_key` 查询特定 Cache Key 的缓存内容：
   - Cache Key 从 API 请求的响应头 `X-Cache-Key` 获取。
   - 当 Cache 不存在时，返回 `404`。
   - 当 Cache 存在时，返回 `200`，响应 Body 如下：

```
{
  "ttl": 60,
  "req_body": "",
  "headers": {
    "Date": "Wed, 27 Apr 2022 03:09:39 GMT",
    "X-Cache-Status": "Miss",
    "ETag": "W/\"14f-3sRRcm7jenKggj3qhgagNHIwiqw\"",
    "Vary": "Accept-Encoding",
    "connection": "keep-alive",
    "set-cookie": "sails.sid=s%3AT_YQH4rJgm2TYpJ5o6Ql1nH-
vlA6pR19.JR3bgC6WfzWO1OqDPwzaNn%2FXX6PcGdg2vTaT6d1BbfQ; Path=/; HttpOnly",
    "content-length": "335",
    "content-type": "application/json; charset=utf-8",
    "X-Cache-Key": "6fb1b6a4440980c758b0eff31177a41d"
  },
  "body": "<cached response body>",
  "status": 200,
  "timestamp": 1651028979,
  "version": 1,
  "body_len": 335
}
```

## 注意事项

- 缓存内容包含响应 Body 和响应 Headers（不缓存 Hop-By-Hop headers）。

- 如需在客户端中通过 `Cache-Control` 请求头对缓存行为进行动态控制，则需要在插件中启用 `cache-control` 配置项。
- `X-Cache-Status` 响应头说明：
  - `Miss`：请求匹配缓存策略，但未命中缓存。
  - `Hit`：请求命中缓存。
  - `Bypass`：请求和缓存插件配置规则不匹配。
  - `Refresh`：缓存未过期，但由于请求头中 `Cache-Control` 原因请求绕过了缓存。

## 参考

更多相关说明请参见 Kong Proxy Cache 插件文档 。

# 使用熔断保护后端

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何在 Kong 云原生 API 网关上通过 Kong 熔断插件实现下述常见熔断场景：

- 根据后端响应时间进行熔断。
- 根据错误码进行熔断。

## 前置条件

- 已购买 Kong 网关实例，详情请参见 实例管理 。
- 配置了后端（Service）以及路由（Route）。

## 插件说明

- 插件名称：TSE BREAKER
- 分类：Traffic Control
- 概述：实现对客户端的请求进行熔断，从而保护上游业务服务。
- 具体能力：

1.1 支持响应时间模式和错误码模式。针对不同模式配置熔断开启条件，错误条件支持响应时间模式和错误码模式。

a. 响应时间模式：需配置后端服务响应时间阈值，超出响应时间阈值为一次错误，到达熔断开启条件触发熔断。

b. 错误码模式：需配置异常状态 HTTP 状态码，状态码使用英文逗号隔开，请求返回异常状态码为一次错误，到达熔断开启条件触发熔断。

1.2 支持熔断后半开探测：第一次触发熔断，熔断2s，超过时间后，重新转发后端，进行错误条件和健康条件判断。

a. 满足错误条件时，熔断4s，以此类推，直到达到预设的最大探测等待时间，停止指数递增。

b. 满足健康条件（即正常状态达到连续正常请求次数）时，结束熔断。

1.3 支持自定义熔断返回：可自定义配置熔断状态 HTTP 状态码、熔断状态 HTTP 响应内容、熔断状态响应头。

## 插件配置

| 字段名称 | 是否必填 | 字段说明 |
| --- | --- | --- |
| 响应时间阈值 | 否 | >=1。上游服务后端最大响应时间，单位为秒。 |
| 异常状态 HTTP 状态码 | 否 | 默认503，支持[500,……599]。上游服务处于异常状态时返回的 HTTP 状态码范围。 |
| 错误率阈值 | 否 | 上游服务在一定时间内触发异常状态的请求次数比例阈值。 |
| 连续错误数阈值 | 否 | >=1，上游服务连续触发异常状态的请求次数阈值。 |
| 最大探测等待时间 | 否 | >=3，最大 300，单位秒，上游服务熔断的最大持续时间。 |
| 连续正常请求次数 | 是 | >=1，上游服务处于健康状态时的连续正常请求次数。 |
| 熔断状态 HTTP 状态码 | 否 | 当上游服务处于不健康状态时返回的 HTTP 错误码。 |
| 熔断状态 HTTP 响应内容 | 否 | 当上游服务处于不健康状态时返回的 HTTP 响应体信息。 |
| 熔断状态响应头 | 否 | 当上游服务处于不健康状态时返回的 HTTP 响应头信息。该字段仅在配置了 break_response_body 属性时生效。 |

版权所有：腾讯云计算（北京）有限责任公司

## 使用场景与操作步骤

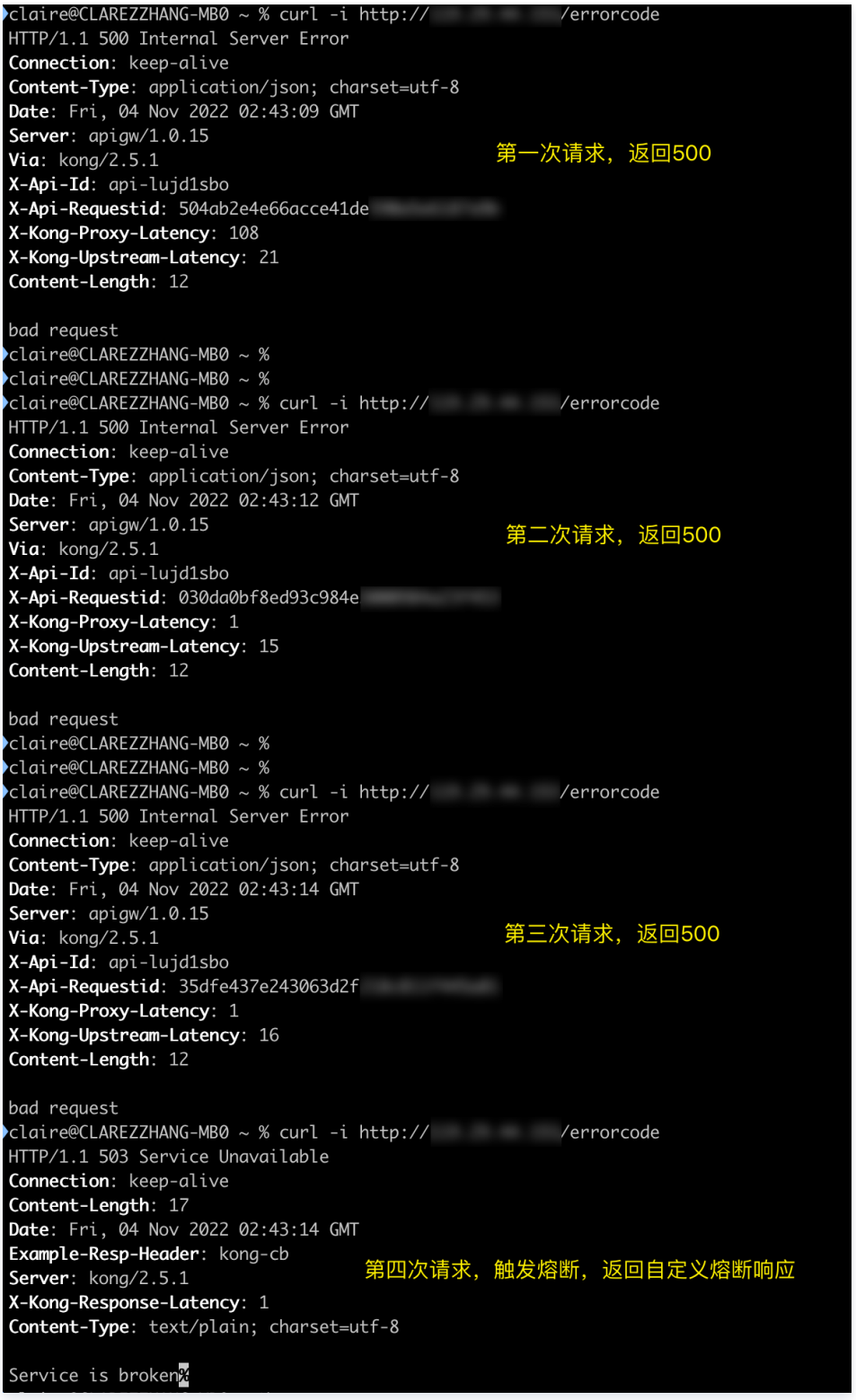
### 场景一：根据错误码进行熔断

1. 登录 Konga 管理控制台，进入需要配置熔断的 Service 详情页，单击 **Add Plugin** 添加插件，在 Traffic Control 分组下选择 TSE BREAKER 插件。
2. 本场景模拟当后端服务返回500，501，502时认为后端异常，连续3次请求错误时触发熔断，熔断返回自定义内容。
   在插件中填写以下配置：

- 错误判断条件：后端服务返回500,501,502。
- 熔断开启条件：连续错误数超过3次。
- 熔断响应配置（可选）。
  - 状态码：503
  - 响应头：example-resp-header:kong-cb
  - 响应内容：Service is broken

3. 发起API请求，查看在三次返回500后熔断成功，返回自定义熔断响应。

```
HTTP/1.1 503 Service Unavailable //自定义响应状态码
Example-Resp-Header: kong-cb  //自定义响应头

Service is broken    //自定义响应内容
```




## 场景二：根据响应时间进行熔断

1. 登录 Konga 管理控制台，进入需要配置熔断的 Service 详情页，单击 **Add Plugin** 添加插件，在 Traffic Control 分组下选择 TSE BREAKER 插件。
2. 本场景模拟当后端服务响应超过15秒时认为后端异常，连续4次请求错误时触发熔断，请求连续正确2次即关闭熔断。
   在插件中填写以下配置：
   - 错误判断条件：后端服务响应时间超过15秒
   - 熔断开启条件：连续错误数超过4次

- 熔断响应配置（可选）
  - 状态码：503
  - 响应头：example-resp-header:kong-cb
  - 响应内容：Service is broken
- 熔断关闭条件：连续正常请求达到2次

3. 发起 API 请求，查看在4次请求超时后熔断成功，2次请求正常后关闭熔断。

# 使用流量镜像插件

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何在 Kong 云原生网关上通过 Kong 流量镜像插件实现下述常见镜像场景：

- 将生产流量的一定比例镜像到测试后端。
- 将生产流量镜像到多个测试后端。

## 前置条件

- 已购买 Kong 网关实例，详情请参见 实例管理 。
- 配置了后端（Service）以及路由（Route）。

## 插件说明

- 插件名称：TSE TRAFFIC MIRROR。
- 插件分类：Traffic Control。
- 概述：实现对客户端的请求进行流量镜像，在线流量将被镜像到测试后端，镜像后端返回的响应数据会被自动丢弃，保证镜像后端的测试业务不会影响到线上业务，广泛用于在测试环境中模拟在线流量对新功能进行测试的场景。
- 具体能力：
  - 支持镜像到 Kong Upstream 或镜像服务地址。
  - 支持根据采样比例镜像：支持配置转发给镜像后端的流量比例。
  - 支持根据请求参数进行镜像：通过将插件配置在 Route 上，匹配到请求参数的请求会进行镜像。
  - 支持镜像到多后端。

## 插件配置

| 配置项 | 是否必填 | 配置说明 |
| --- | --- | --- |
| 镜像比例 | 是 | 转发到镜像服务的请求比例，支持1-100。 |
| 后端地址类型 | 是 | 镜像服务类型，支持填写 Host 或 Upstream。 |
| 后端地址-Host | 否 | 镜像服务地址，需包含 schema（http/https）和 Host 地址，例如 http://127.0.0.1。 |
| 后端地址-Port | 否 | 镜像服务端口，支持1-65535。 |
| 后端地址-Upstream | 否 | 镜像服务的 Kong Upstream，需包含 schema（http/https）和 Upstream 名称，例如 http://myupstream。 |
| 后端地址-Path | 否 | 镜像服务路径，不指定则默认使用当前请求路径。 |

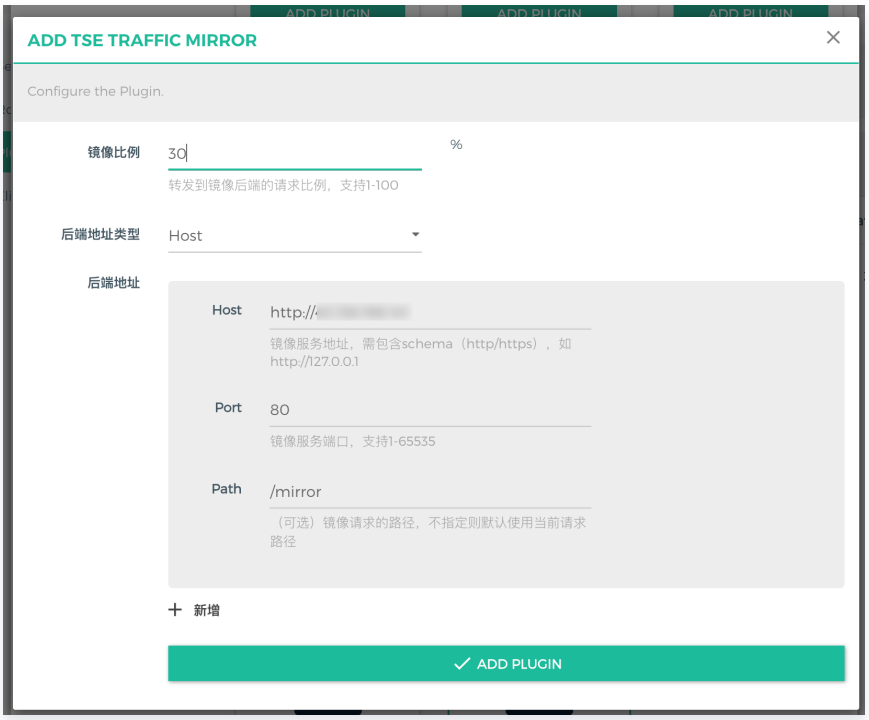
## 操作步骤

### 场景一：根据采样比例镜像

1. 登录 Konga 管理控制台，进入需要配置流量镜像的 Service 详情页，单击 **Add Plugin** 添加插件，在 Traffic Control 分组下选择 TSE TRAFFIC MIRROR 插件。
2. 在本场景中，模拟将线上30%的流量镜像到 Host 类型的测试后端。
3. 在插件中填写以下配置，单击 **ADD PLUGIN**：

- 镜像比例：30%

- 后端地址类型：Host
- 后端地址：
  - Host：填写测试服务访问地址，例如 http://example-test-service。
  - Port：填写测试服务端口。
  - Path：选填，不填使用真实请求路径。



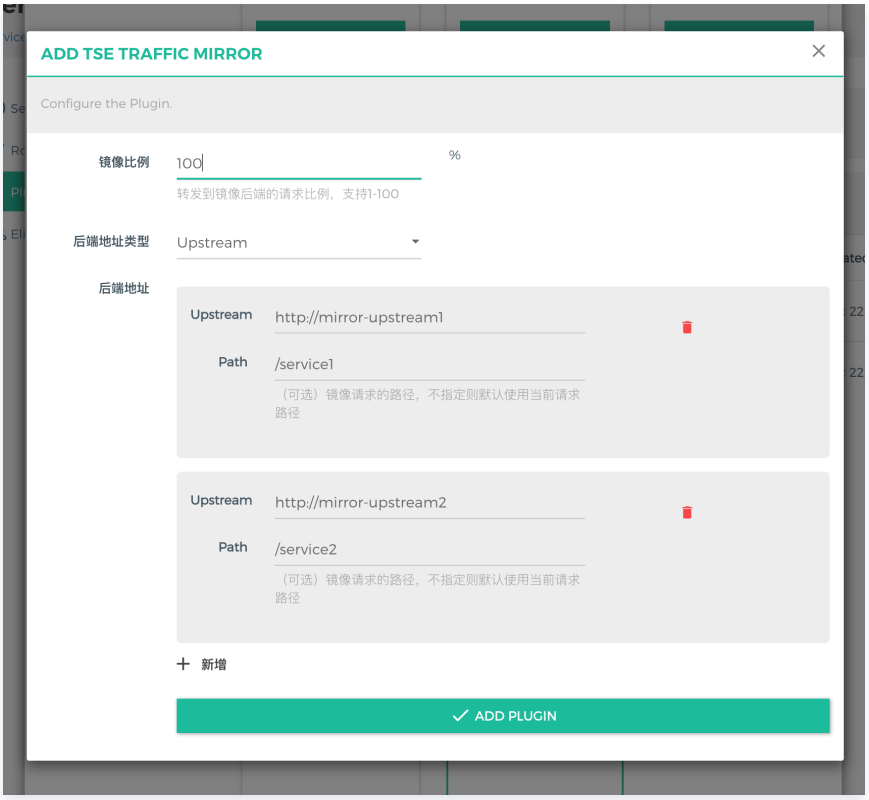

4. 发起 API 请求，查看镜像后端收到一定比例的镜像请求。

## 场景二：镜像到多个后端

1. 登录 Konga 管理控制台，进入需要配置镜像的 Service 详情页，单击 **Add Plugin** 添加插件，在 Traffic Control 分组下选择 TSE TRAFFIC MIRROR 插件。
2. 本场景模拟将流量镜像到多个 Upstream。
   在插件中填写以下配置：

- 镜像比例：100%。
- 后端地址类型：Upstream。
- 添加多个后端地址，每个后端地址填写：
  - Upstream：填写镜像服务所在的 Upstream。

- Path：选填，填写镜像请求路径。




3. 发起 API 请求，在两个 Upstream 查看发起的镜像请求。

# 请求路由
# 进行 URL 重写

最近更新时间：2025-06-09 16:07:02

## 操作场景

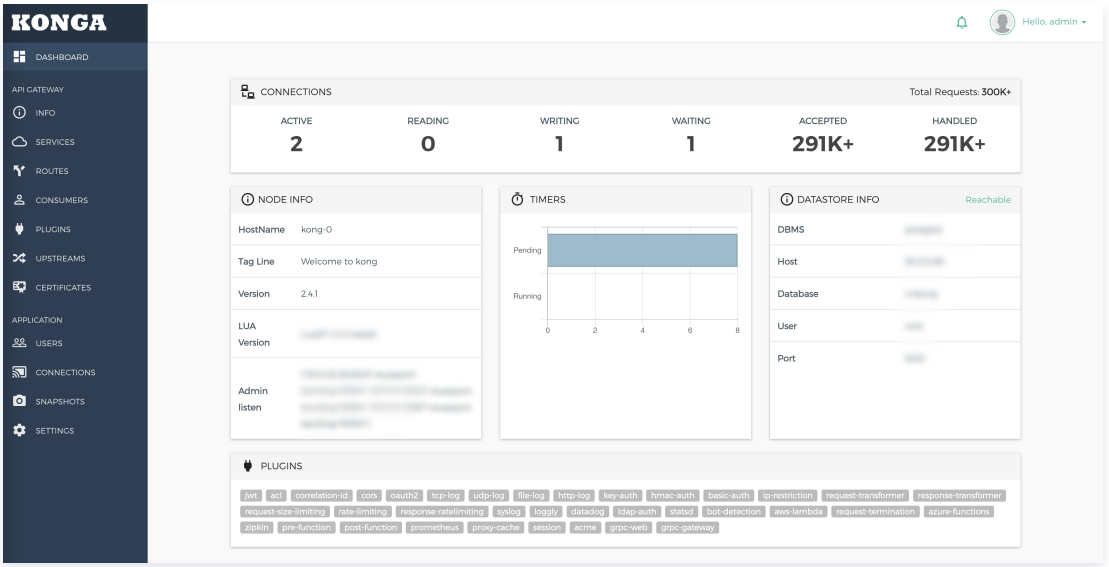
本文介绍如何将 API 发布到 Kong 网关上，并实现 URL 重写。

## 前置条件

1. 已购买 Kong 网关实例，详情请参见 网关实例管理 。
2. 进入 TSF 控制台，并找到需要的实例。
3. 进入实例详情页后，在 **Konga 控制台**页面查看管理控制台登录方式。




4. 访问 Konga 管理控制台。

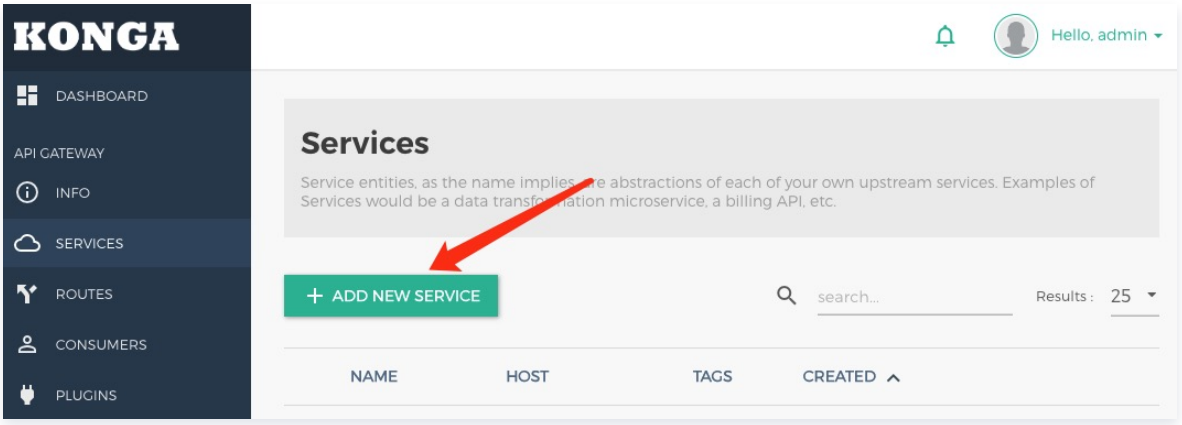



## 操作步骤

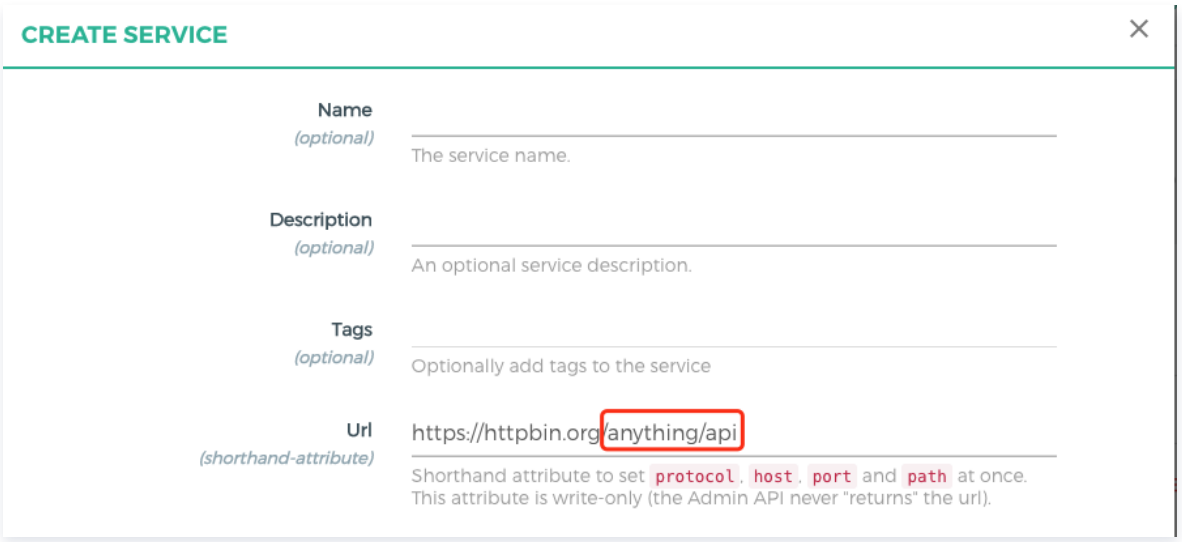
### 场景一：通过 Service 和 Route 配置实现 URL 前缀改写

**示例**
原始 API 调用 URL 为 `http://<backend>/anything/api/<sub_path>`
发布到 Kong 网关后，调用 URL 改写为 `http://<kong>/new/path/<sub_path>`

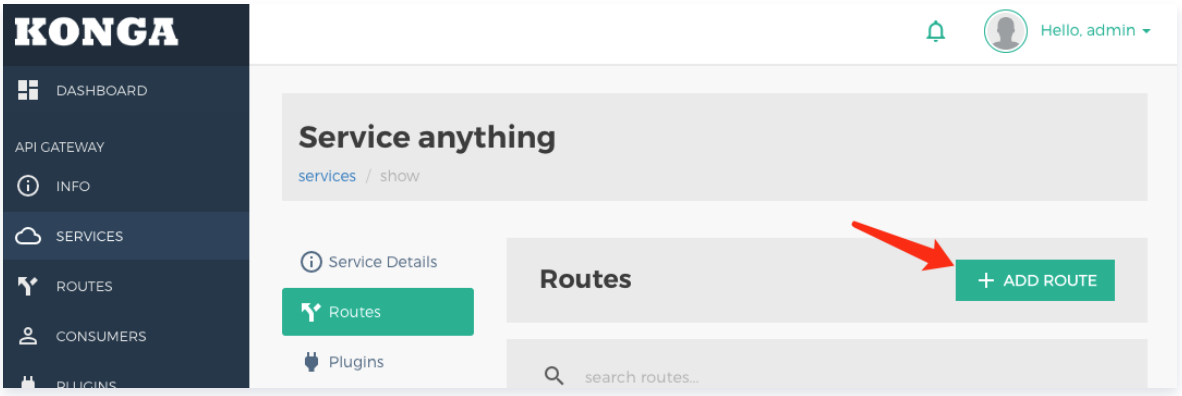
1. 单击 **ADD NEW SERVICE**创建 Service。



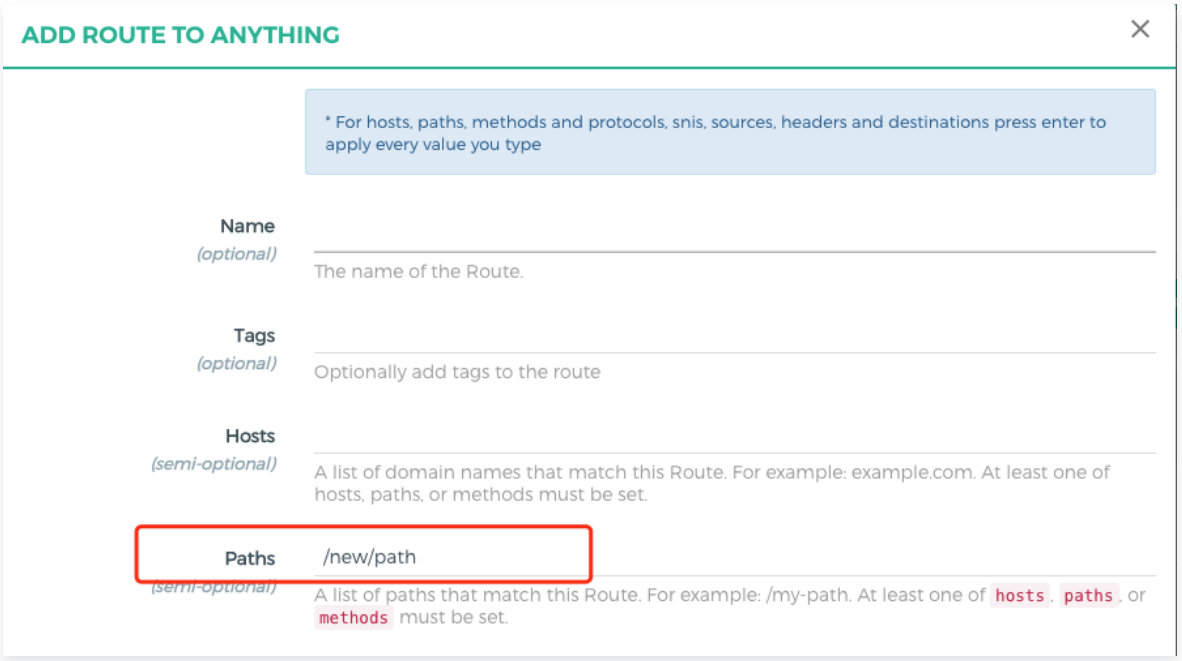

填写相应信息并设置 Path 为 `/anything/api` 。




2. 单击 **ADD ROUTE** 在该 Service 上创建 Route，



设置 Path 为 `/new/path`，且开启 Strip Path 选项（默认开启）。

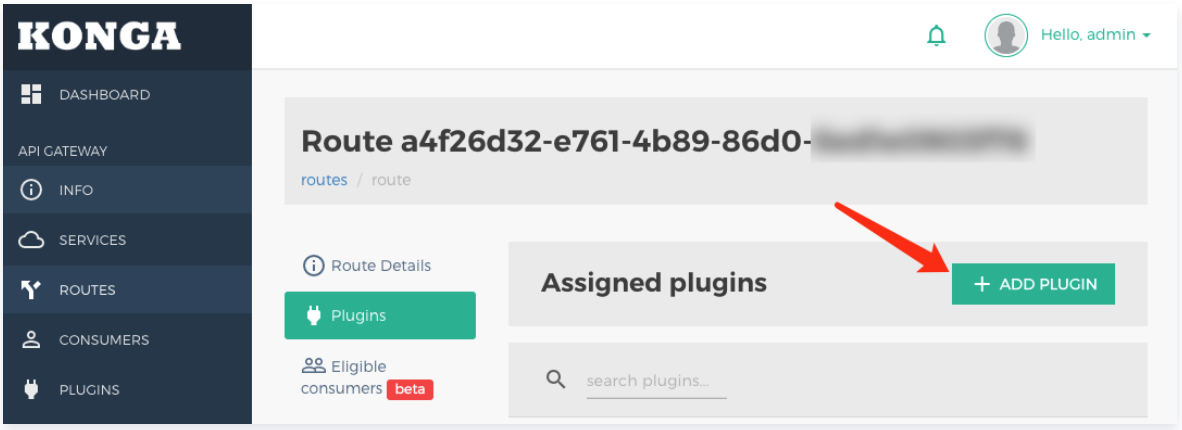



3. 请求 `http://<kong>/new/path/user` 时，后端接收到的请求路径为 `/anything/api/user`。

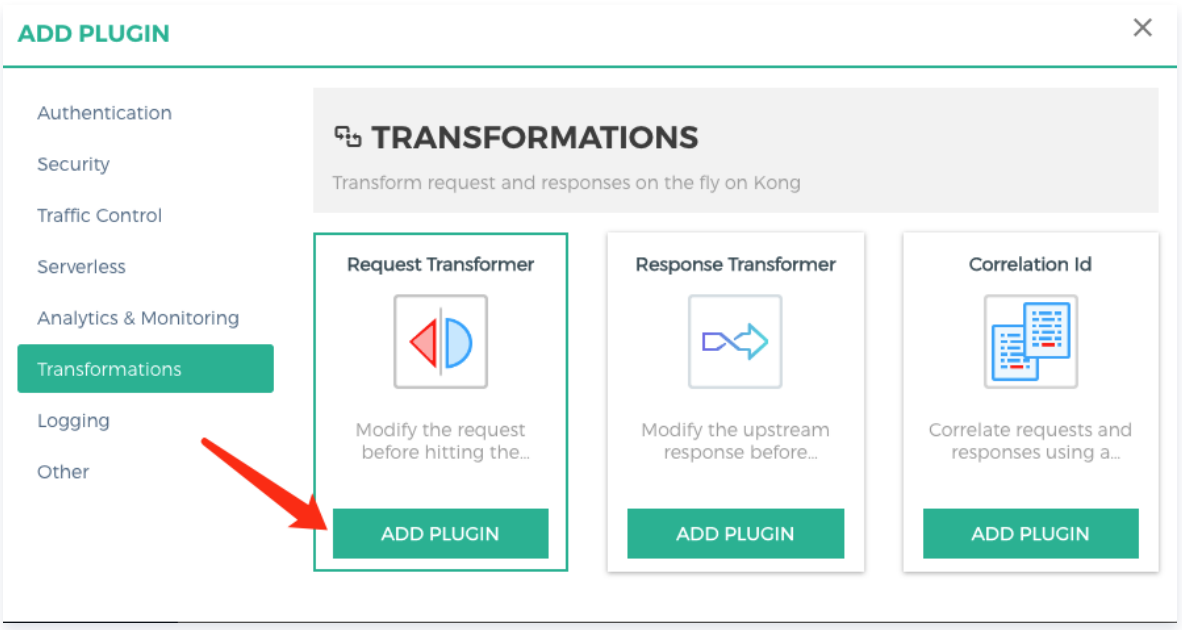
## 场景二：通过 Request Transformer 插件实现完整 URL 改写

> **示例**
> 原始 API 调用 URL 为 `http://<backend>/anything/user_list`
> 发布到 Kong 网关后，调用 URL 改写为 `http://<kong>/users`

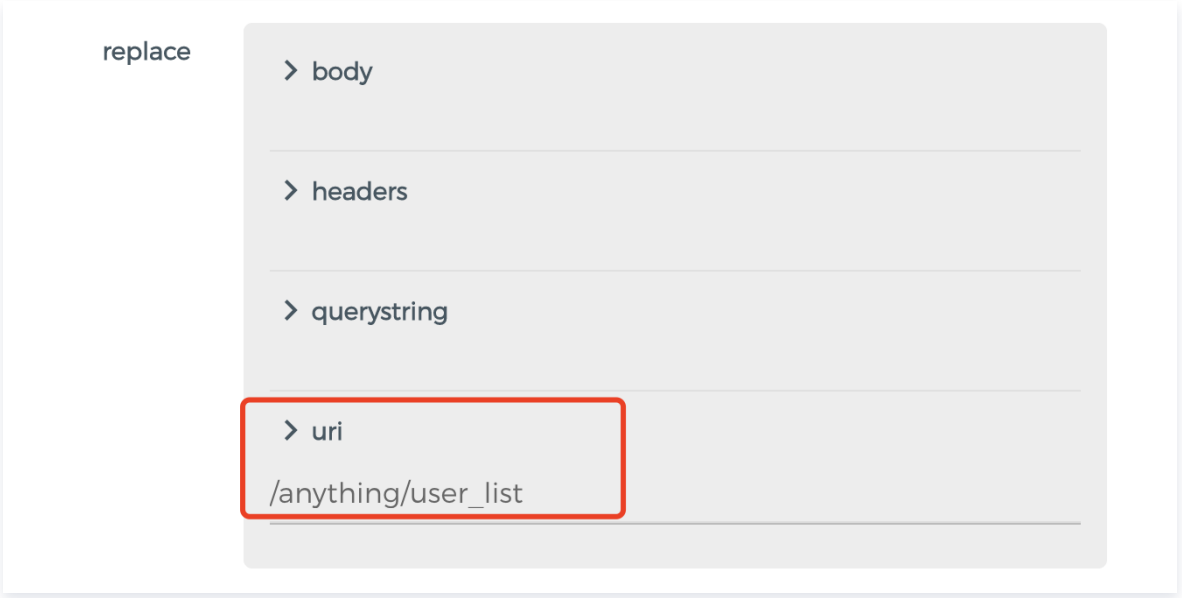
1. 创建 Service，Path 可以任意设置。
2. 在该 Service 上创建 Route，设置 Path 为 `/users`。
3. 单击 **ADD PLUGIN** 新建插件。




在该 Route 上创建 Request Transformer 插件。




并配置 `replace.uri` 为 `/anything/user_list`，该路径为后端接收到的实际请求路径。



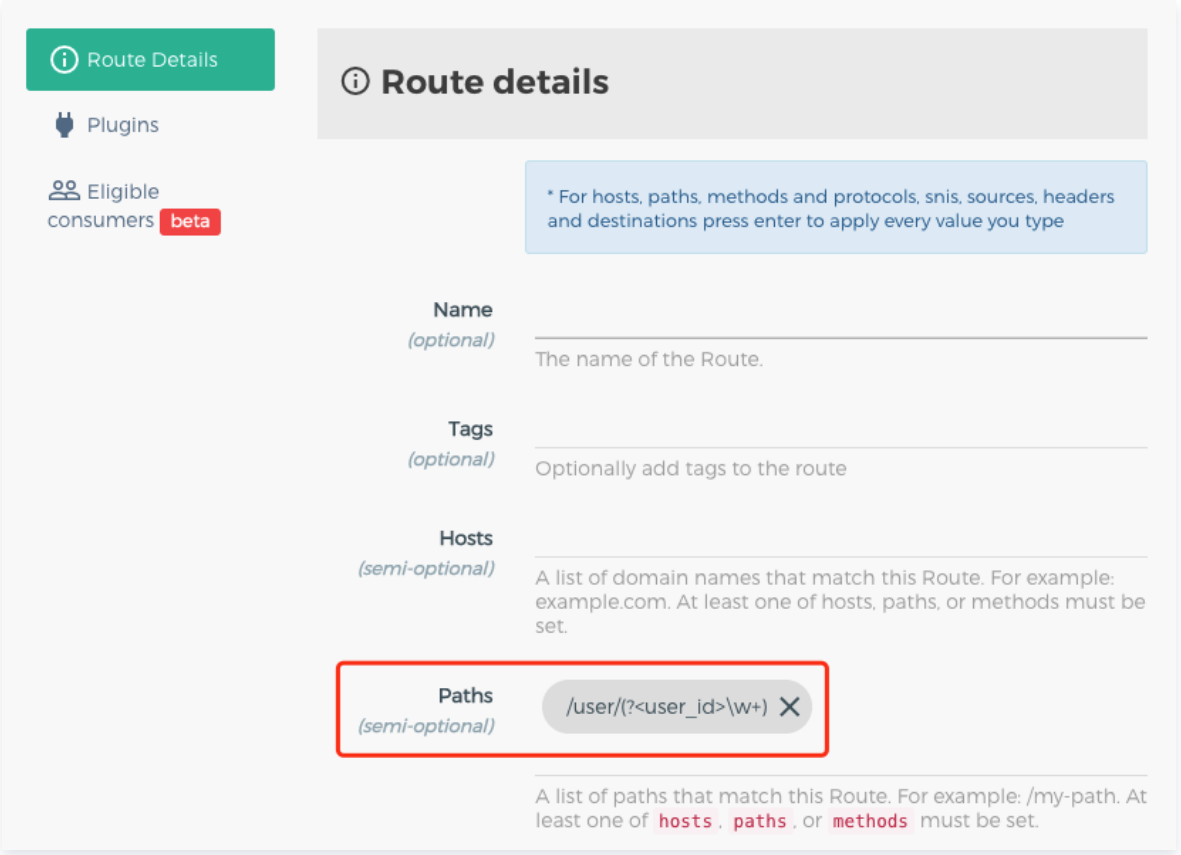

4. 请求 Route 后，后端接收到的请求路径为 `/anything/path`（原始 Service Path 配置被覆盖）。

## 场景三：通过 Request Transformer 插件实现部分 URL 改写

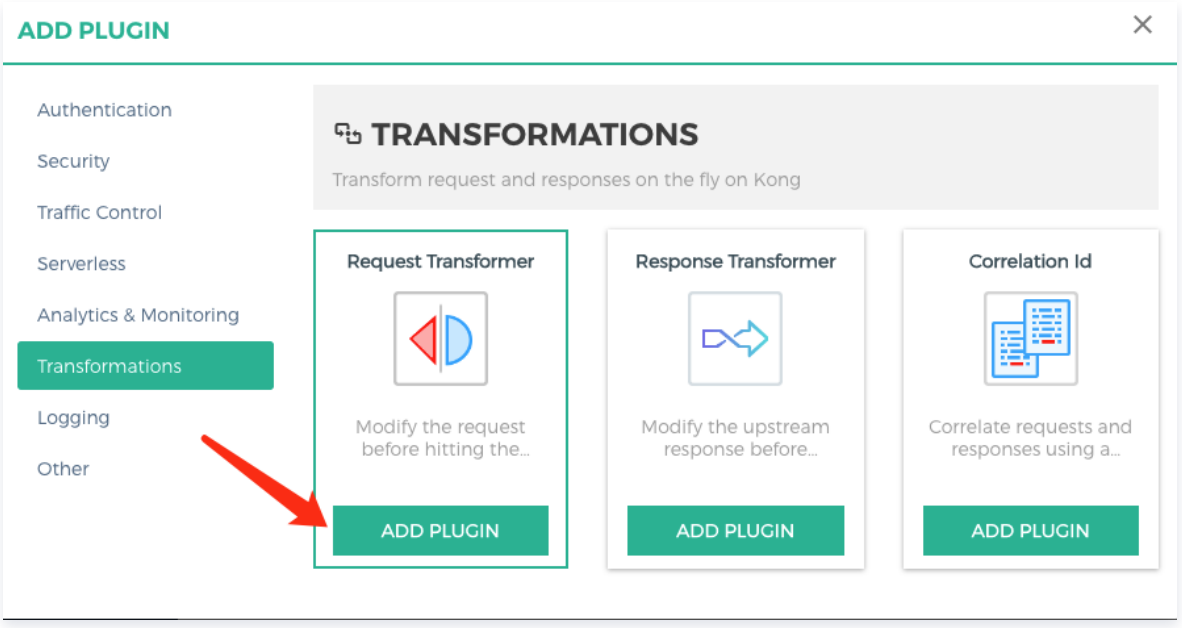
> **示例**
> 原始 API 调用 URL 为 `http://<backend>/anything/<user_id>/get`
> 发布到 Kong 网关后，调用 URL 改写为 `http://<kong>/user/<user_id>`

1. 创建 Service，Path 可以任意设置。

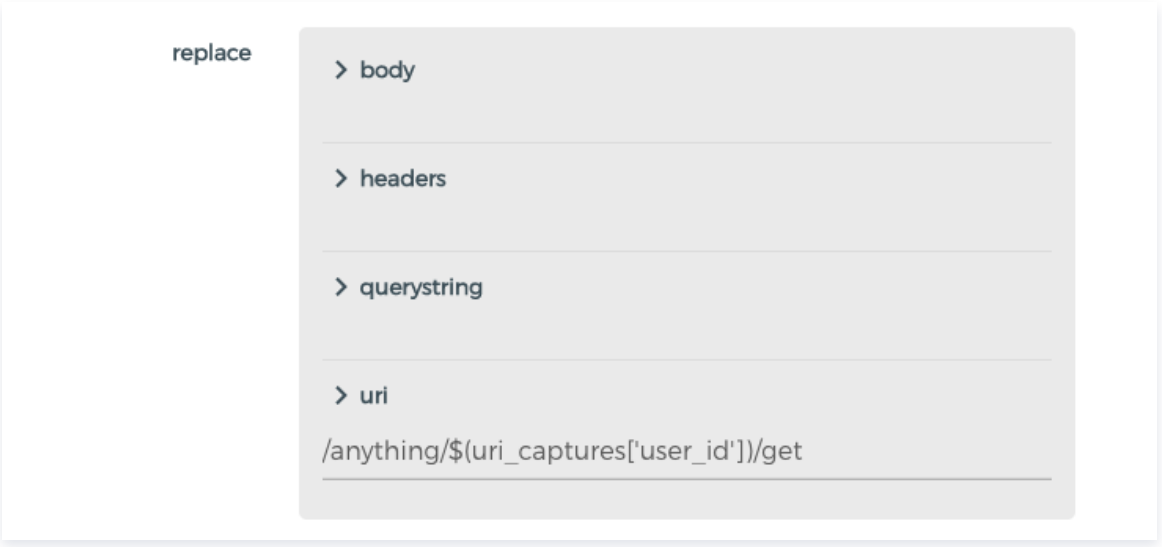
2. 在该 Service 上创建 Route，设置 Path 为 `/user/(?<user_id>\w+)` 。




3. 在该 Route 上创建 Request Transformer 插件。



并配置 `replace.uri` 为 `/anything/$(uri_captures['user_id'])/get`，该路径为后端接收到的实际请求路径。




4. 请求 `http://<kong>/user/user_a` 后，后端接收到的请求路径为 `/anything/user_a/get` 。

## 相关参考

- Kong 插件文档

# 使用重写插件改写请求或响应

最近更新时间：2025-06-09 16:07:02

## 操作场景

本文介绍如何在 Kong 云原生网关上通过 Kong 插件实现下述常见请求/响应改写场景：

- 支持在请求阶段对 header/query/body 进行改写
- 支持在响应阶段对 header/body 进行改写

## 前置条件

- 已购买 Kong 网关实例，详情请参见 实例管理 。
- 配置了后端（Service）以及路由（Route）。

## 插件说明

- 插件名称：TSE PROXY REWRITE
- 插件分类：Transformations
- 概述：实现对客户端的请求和响应进行改写，支持使用Nginx内置变量。
- 具体能力：
- 支持在请求或响应阶段对 query/header 参数进行新增、编辑、删除。
- 当 content-type 为 application/json, multipart/form-data, application/x-www-form-urlencoded 时，支持在请求或响应阶段对 body 字段进行新增、编辑、删除。
- 支持使用nginx变量重写请求/响应。

## 插件配置

| 配置项 | 是否必填 | 配置说明 |
| --- | --- | --- |
| Header 参数 | 否 | 支持配置多条，每条配置项包括生效阶段、操作类型、键、值。<br>• 生效阶段：支持请求、响应。<br>• 操作类型：支持新增、删除、编辑。<br>• 键值：输入请求/响应的 Header 参数名和值。<br>• 支持以 $var 的格式包含 NGINX 变量，例如 $remote_addr $balancer_ip。 |
| Query 参数 | 否 | 支持配置多条，每条配置项包括生效阶段、操作类型、键、值。<br>• 生效阶段：支持请求。<br>• 操作类型：支持新增、删除、编辑。<br>• 键值：输入请求的 Query 参数名和值。<br>• 支持以 $var 的格式包含 NGINX 变量，例如 $remote_addr $balancer_ip。 |
| Body | 否 | 支持配置多条，每条配置项填写生效阶段、操作类型、键、值。<br>• 生效阶段：支持请求、响应。<br>• 操作类型：支持新增、删除、编辑。<br>• 键值：当 content-type 为 application/json, multipart/form-data, application/x-www-form-urlencoded 时，输入 body 字段的参数名和值。 |

**注意**

当操作类型为编辑时，如果指定的参数不存在，则会按照新增操作新增键和值；如果指定的参数已存在，则会进行值替换。

## 操作步骤

### 场景一：实现响应阶段增加 Header 参数

1. 登录 Konga 管理控制台，进入需要配置响应改写的 Service 详情页，单击 **Add Plugin** 添加插件，在 Transformations 分组下选择 TSE PROXY REWRITE 插件。
2. 在插件中填写以下配置，实现响应阶段新增一个 Header 参数。




3. 发起 API 请求，查看后端服务的响应是否新增添加的 Header 参数。

```
HTTP/1.1 200 OK
Addheader: test
Connection: keep-alive
Content-Length: 20
Content-Type: application/octet-stream
Date: Thu, 22 Dec 2022 06:31:11 GMT
Server: nginx/1.20.1
Via: kong/2.5.1
X-Kong-Proxy-Latency: 30
X-Kong-Upstream-Latency: 66
```

```
this is test service
```

## 场景二：实现请求阶段改写 Query 参数

1. 登录 Konga 管理控制台，进入需要配置请求改写的 Service 详情页，单击 **Add Plugin** 添加插件，在 Transformations 分组下选择 TSE PROXY REWRITE 插件。
2. 在插件中填写以下配置，实现响应阶段改写 Query 参数。




3. 发起 API 请求，查看后端服务是否收到改写的 Query 参数。

## 场景三：实现响应阶段增加 Body

1. 登录 Konga 管理控制台，进入需要配置请求改写的 Service 详情页，单击 **Add Plugin** 添加插件，在 Transformations 分组下选择 TSE PROXY REWRITE 插件。

2. 在插件中填写以下配置，实现响应阶段增加 Body 参数。




> **注意**
> 对 body 参数的操作，需要请求传入 Content-Type 的 Header。仅当 Content-Type 为 application/json, multipart/form-data, application/x-www-form-urlencoded 时，才会进行 body 字段的改写。

3. 发起 API 请求，查看后端服务的响应是否收到新增的 Body 参数。

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: application/json
Date: Thu, 22 Dec 2022 06:53:43 GMT
Server: nginx/1.20.1
Via: kong/2.5.1
X-Kong-Proxy-Latency: 22
X-Kong-Upstream-Latency: 63
Content-Length: 54

{"myreturn":"this is test service","addbody":"return"}
```

# 可观测
# 观测中心

最近更新时间：2025-06-11 16:14:31

**腾讯云微服务平台（Tencent Service Framework，TSF）控制台升级后，观测中心统一入口，日志升级为腾讯云日志服务（CLS），监控升级为腾讯云应用性能监控（APM），功能更丰富。详细参见 升级公告 。**
**升级完成后才开始使用 TSF 的用户（下文简称为新用户），默认使用升级后的产品形态和功能，如果您是此类用户，请单击前往 用户指南 。**
**升级前已使用 TSF 的用户（下文简称为存量用户）产品功能与使用不受影响，仅控制台功能入口和名称进行了变更，如下图所示。如果您是此类用户，请单击前往 用户指南 。如果您想体验新功能，请联系我们 提交工单 处理。**



# 存量用户
# 升级指引（内测中）

最近更新时间：2025-05-27 15:24:22

**TSF 可观测功能迎来全面升级，应用监控、全局拓扑和依赖分析升级为 APM，日志服务升级为 CLS，注册配置中心升级为北极星，欢迎体验。当前功能处于处于测试阶段，请联系我们 提交工单 处理。**

# 资源概览

最近更新时间：2025-04-08 15:25:53

## 功能说明

微服务平台支持全局视角的用户资源统计，包含资源概览、集群使用概览、服务监控、配额概况、异常告警显示、命名空间等统计页面。
您只需登录 腾讯微服务平台控制台，在左侧菜单栏选择**观测中心 > 资源概览**，即可查看用户资源统计概览。

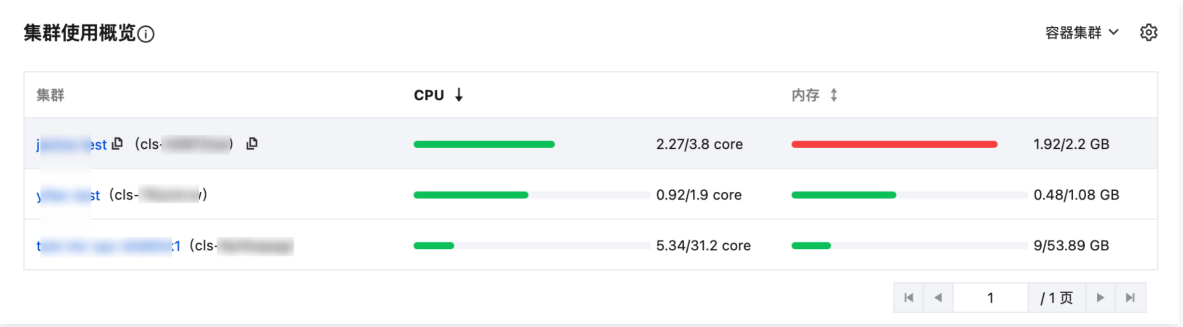
## 资源概览

支持查看租户下全局/子账号有读权限的资源概览情况，不受权限范围影响；当发生资源变更后，30分钟内刷新概览页数据。



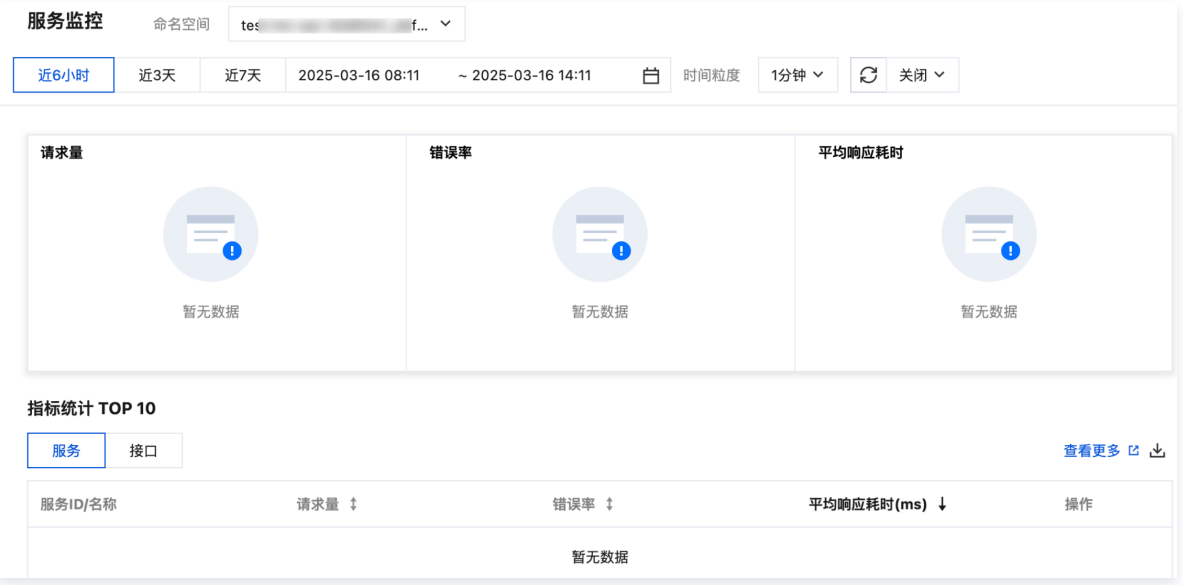

## 集群使用概览

支持查看 CPU / 内存使用率，并对分配率超过阈值的集群突出显示。




## 服务监控

支持查看命名空间维度请求量、错误率、平均响应耗时时间，并且统计服务、接口维度指标Top10。



## 配额情况

支持查看用户节点配额，应用仓库用量及日志存储用量，并支持进行节点规格调整和续费操作。



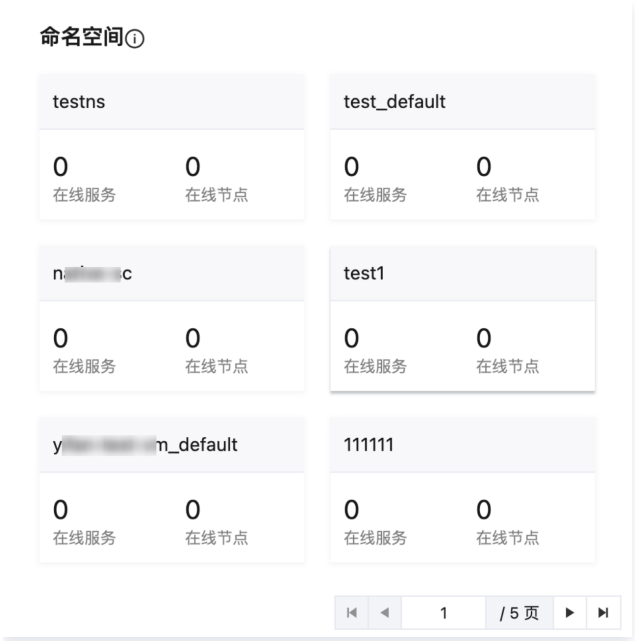

## 异常告警显示

支持查看近24小时内未恢复告警数 / 总告警数，也可单击**配置告警>**，前往腾讯云可观测平台控制台进行告警策略的配置。




## 命名空间

支持展示当前时间各个命名空间注册情况。




## 产品动态

支持查看产品最新发布动态。



# 业务应用监控

最近更新时间：2025-04-28 14:15:52

## 操作场景

TSF 支持监控业务应用的监控，支持应用、部署组、实例三种维度。

## 前提条件

- 已创建微服务业务应用（ 容器场景 、虚拟机场景 ）。

## 操作指南

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**观测中心**。
2. 在**观测中心**详情页选择**业务应用监控**，进入**业务应用监控**详情页面。
3. 选择目标**地域**、**应用/部署组/实例**信息后即可单击对应 Tab 查看监控信息，支持的功能如下。

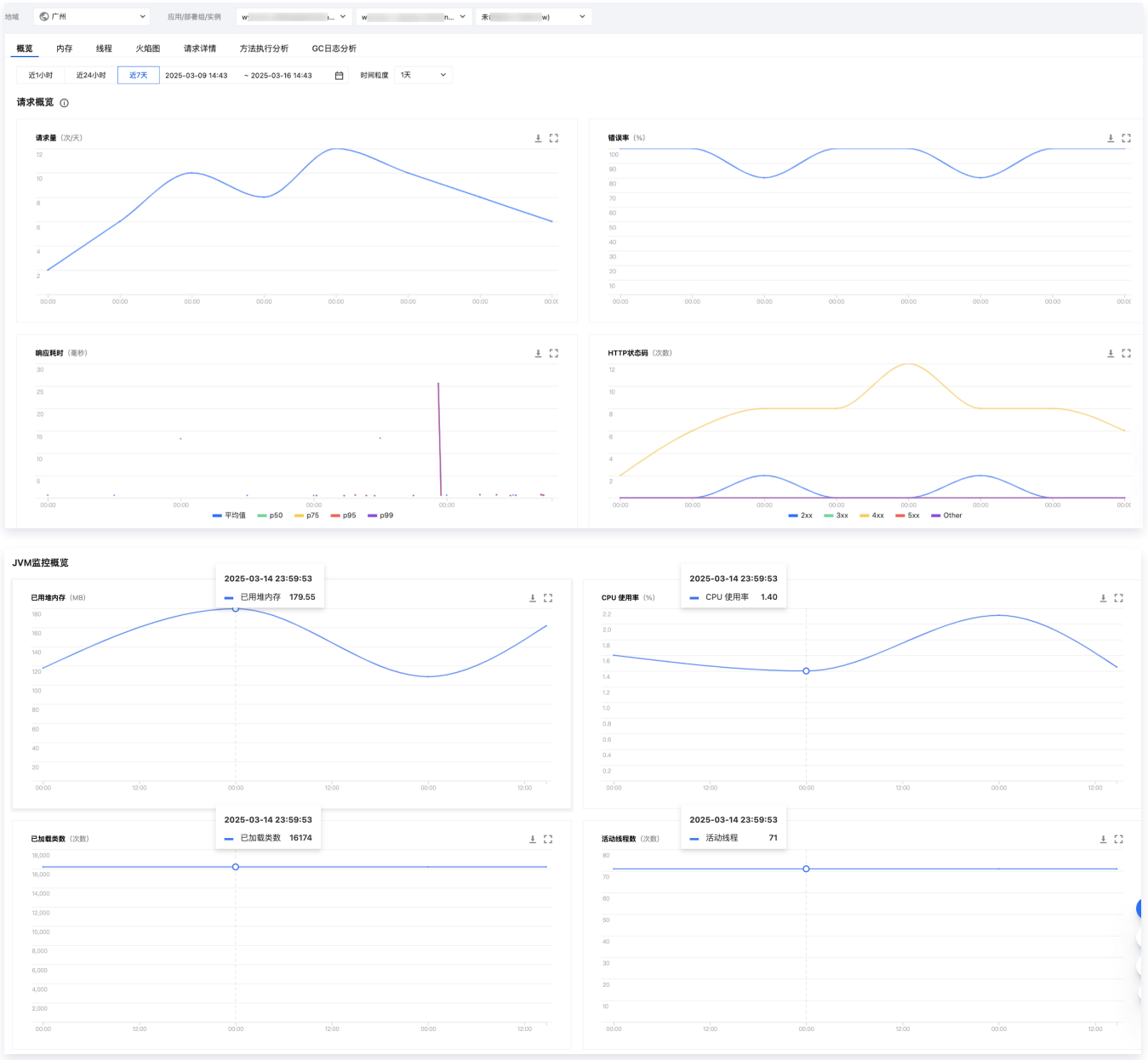
> **说明：**
> TSF 可观测功能迎来全面升级，应用监控、全局拓扑和依赖分析升级为 APM，日志服务升级为 CLS，注册配置中心升级为北极星，欢迎体验，点击查看 升级指引 。

### 概览

**功能说明：**统计实例被请求的详情。

| 指标类型 | 指标 | 单位 | 含义 |
| --- | --- | --- | --- |
| 请求概览 | 请求量 | • 次/分钟<br>• 次/小时<br>• 次/天 | 统计单位时间内服务的总请求次数，反映服务负载水平。 |
| | 错误率 | % | 统计请求中失败的比例（例如HTTP 5xx错误、服务内部异常等），用于衡量服务稳定性。 |
| | 响应耗时 | 毫秒 | 请求的平均处理时间，反映服务性能效率。 |
| | HTTP状态码 | 次数 | 统计不同 HTTP 响应码（例如2xx、3xx、4xx、5xx、其他）的分布情况，帮助定位接口级异常。 |
| JVM监控概览 | 已用堆内存 | MB | JVM 堆内存当前使用量，用于检测内存泄漏或 GC 效率问题。 |
| | CPU使用率 | % | JVM 进程占用的 CPU 资源比例。 |
| | 已加载类数 | 次数 | JVM 当前加载的类数量。 |
| | 活动线程数 | 次数 | 当前活跃线程数量。 |

版权所有：腾讯云计算（北京）有限责任公司



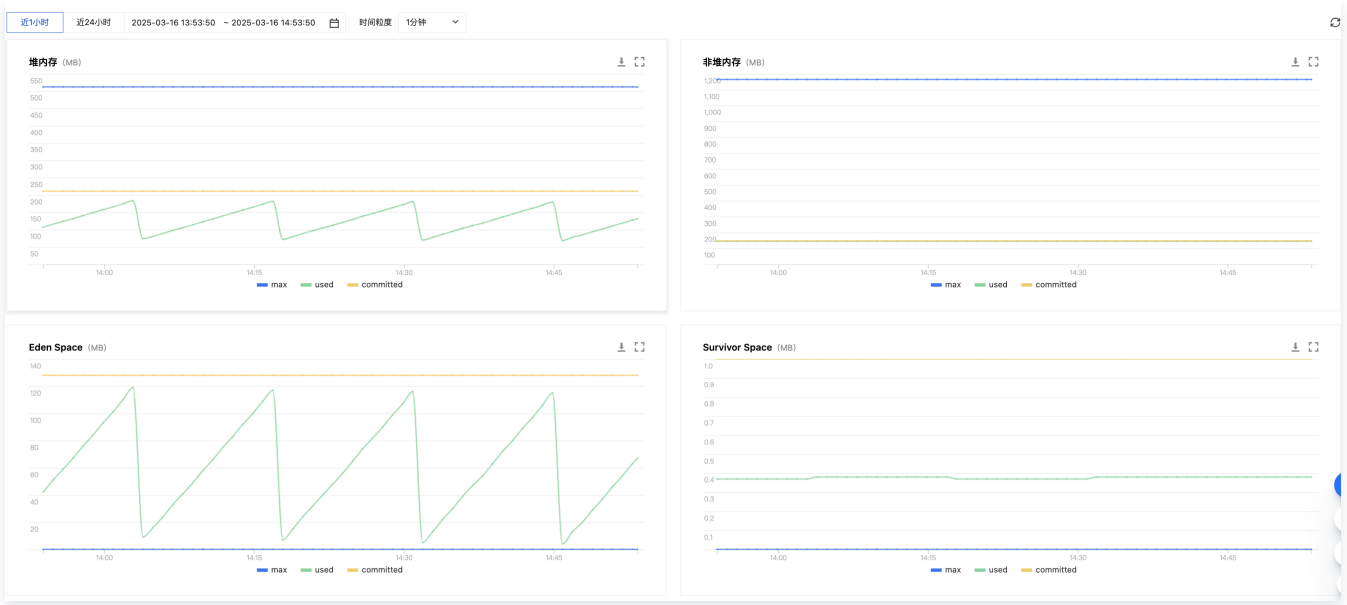

## 内存

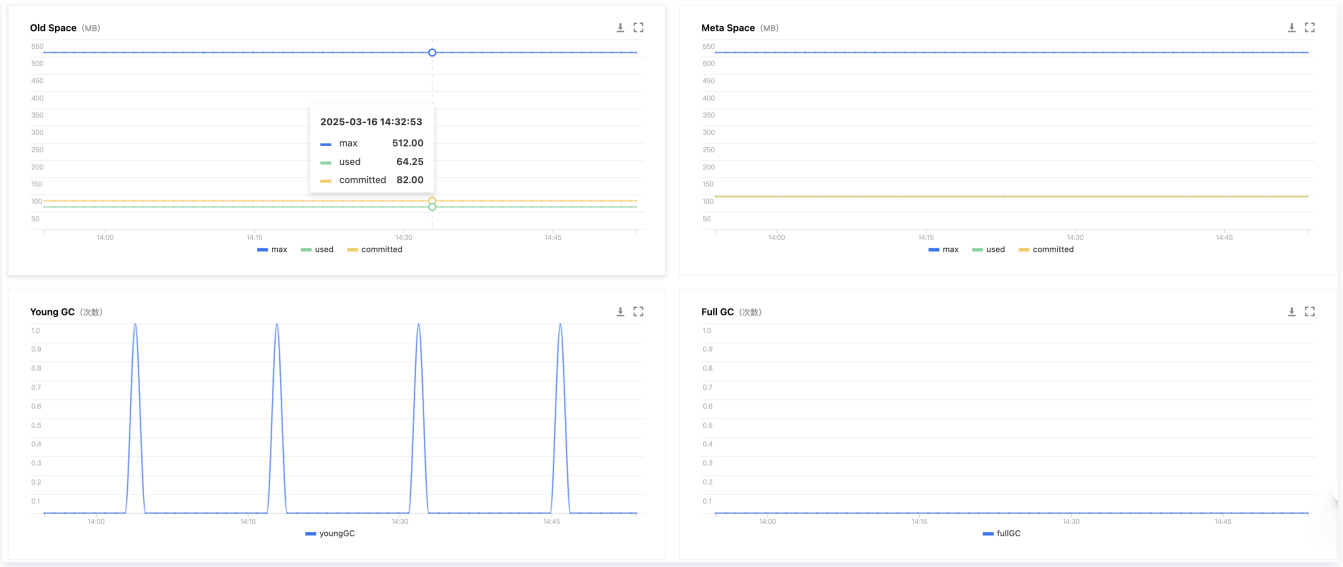
**功能说明：**您可在**内存**页中，查看到进程 JVM 内存主要指标变化曲线，了解选定时段内，内存指标随时间的变化情况。您可通过曲线图上方的时间范围选择器修改曲线显示的时间范围（时间粒度目前支持1分钟和1小时）。

| 内存信息 | 单位 | 含义 |
| --- | --- | --- |
| 当前堆大小 | MB | JVM 堆内存当前实际使用的容量。 |
| 最大堆大小 | MB | 堆内存真实可用的最大值（会扣除 to_space 内存），而不是配置的 xmx 参数。JDK1.8版本部分 GC 算法（例如 G1GC 的 Survivor space），其 max 值会设置为 max_int 值，此时值会显示为-1。 |
| commit堆内存 | MB | JVM 向操作系统实际申请的物理内存总量（包括已使用和未使用的堆内存）。 |
| Eden Space | MB | 新生代的核心区域，所有新对象首次分配内存的位置，支持展示堆内存最大 |

| | | |
|---|---|---|
| | | 值、当前值和 commit 值。 |
| Survivor Space | MB | 新生代的次要区域，分为 From Survivor 和 To Survivor，用于存放 Minor GC 后存活的对象。支持展示堆内存最大值、当前值和 commit 值。 |
| Old Space | MB | 存放长期存活的对象（例如缓存、会话数据），支持展示堆内存最大值、当前值和commit 值。 |
| Meta Space | MB | JDK8+替代永久代，存储类元数据（例如方法、字段信息）、运行时常量池等，使用本地内存（非堆内存）支持展示本地内存最大值、当前值和 commit 值。 |
| 待析构对象数 | 个 | 即将被垃圾回收器回收的对象数量。 |
| Young GC 次数 | 次 | 年轻代垃圾回收的次数。 |
| Young GC 总花费时间 | ms | 年轻代垃圾回收花费的总时间。 |
| FUll GC 次数 | 次数 | 全堆垃圾回收的次数，涉及新生代和老年代。 |
| FUll GC 总花费时间 | ms | 全堆垃圾回收花费的总时间。 |






您还可以通过单击图片卡片右上方的放大图标，放大当前图片；或通过单击下载图标，将当前图片下载到本地（.png格式）。

## 线程

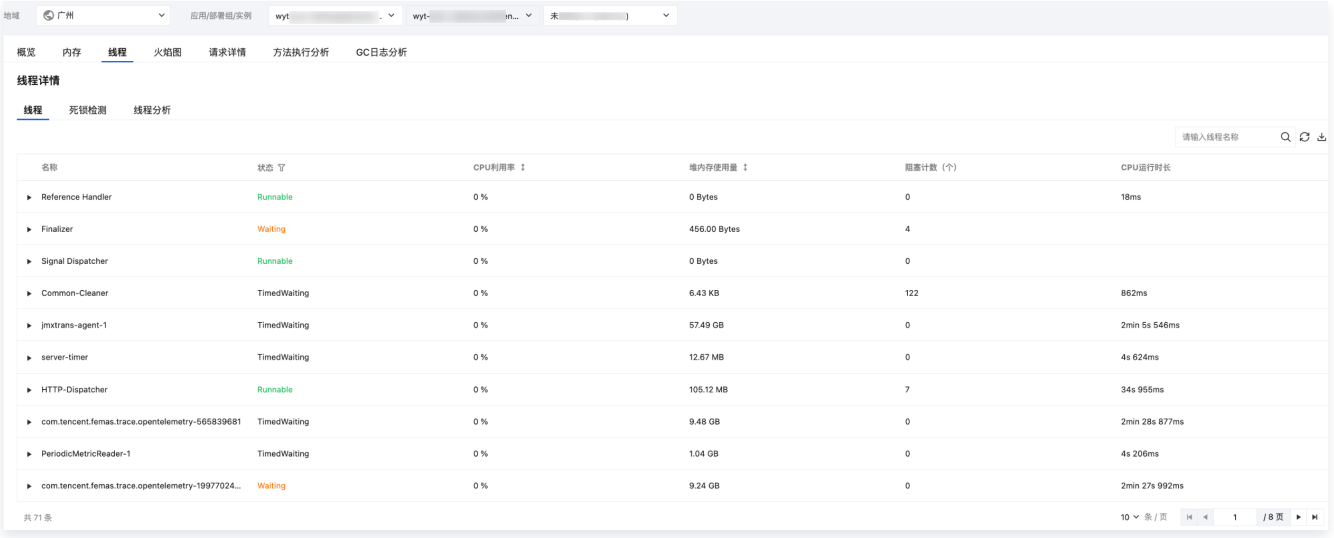
- 您可在实例详情的**线程**页中，查看到该实例的线程数变化情况、线程列表及线程详情，并进行死锁检测，了解实例的死锁情况。

### 线程详情

您可在线程详情卡片中查看到当前实例所有的 JVM 线程列表及其详细堆栈信息：

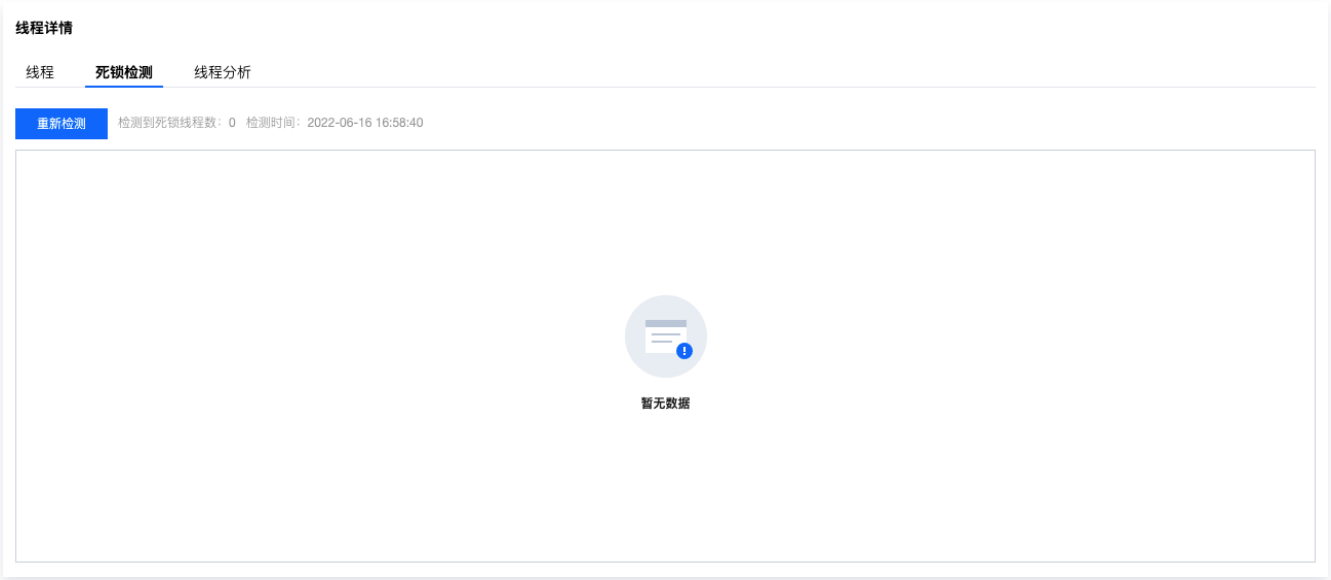
- 在**线程**页中，以列表形式列出了当前实例的所有线程。
  - 您可查看每个线程的名称、状态、CPU 利用率、堆内存使用量、阻塞计数、CPU 运行时长。
  - 您可根据“状态”对线程列表进行筛选，或根据“CPU利用率”或“堆内存使用量”对线程列表进行排序。
- 您可通过列表上方的搜索框，快速查找感兴趣的线程。
- 单击某线程行所在区域，可在下方的展开区域中，查看到所选线程的详细信息。

所展示的线程列表为进入页面时刻获取的数据，您可通过单击卡片右上方的刷新按钮，拉取当前的最新数据。



## 死锁检测

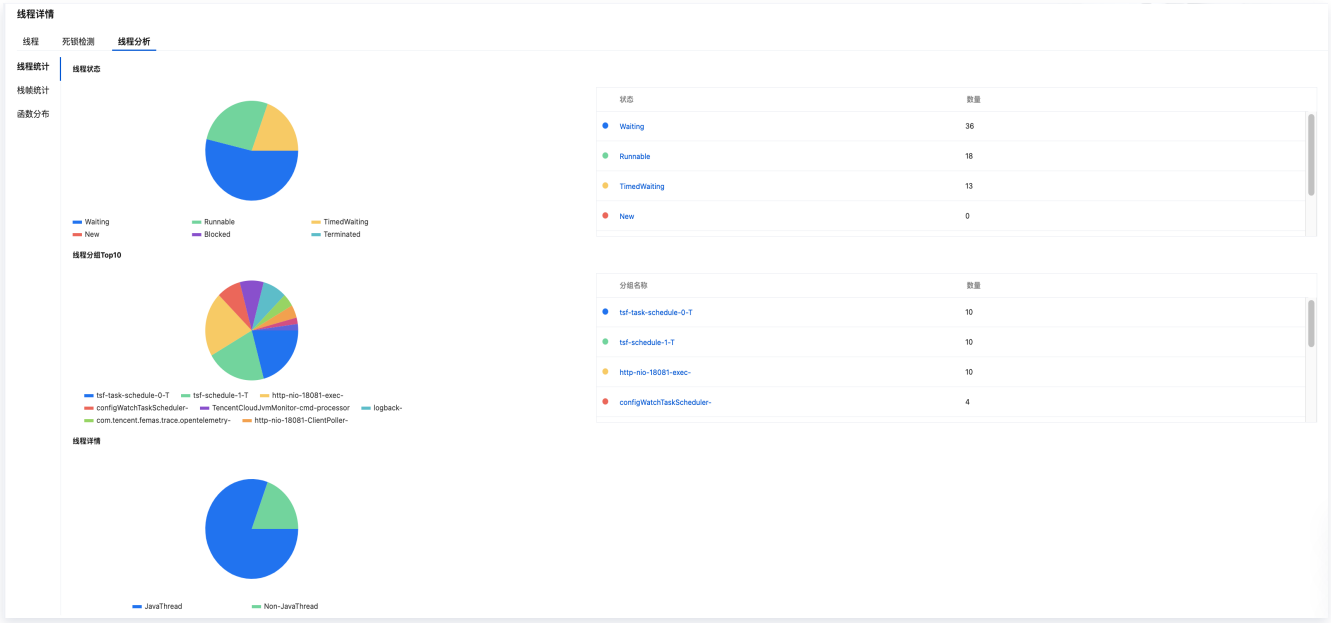
**在线程详情卡片中，单击死锁检测中的检测死锁或重新检测，实时检测当前进程中存在的死锁线程。**




> **说明：**
> 死锁检测目前的实现，是可以打印互相死锁的线程栈；多个线程等待同一个死锁的情况下，并不能检测出全部的死锁线程，只能找到死锁的根源；建议修复死锁后，重复检测以确认不出现嵌套死锁。

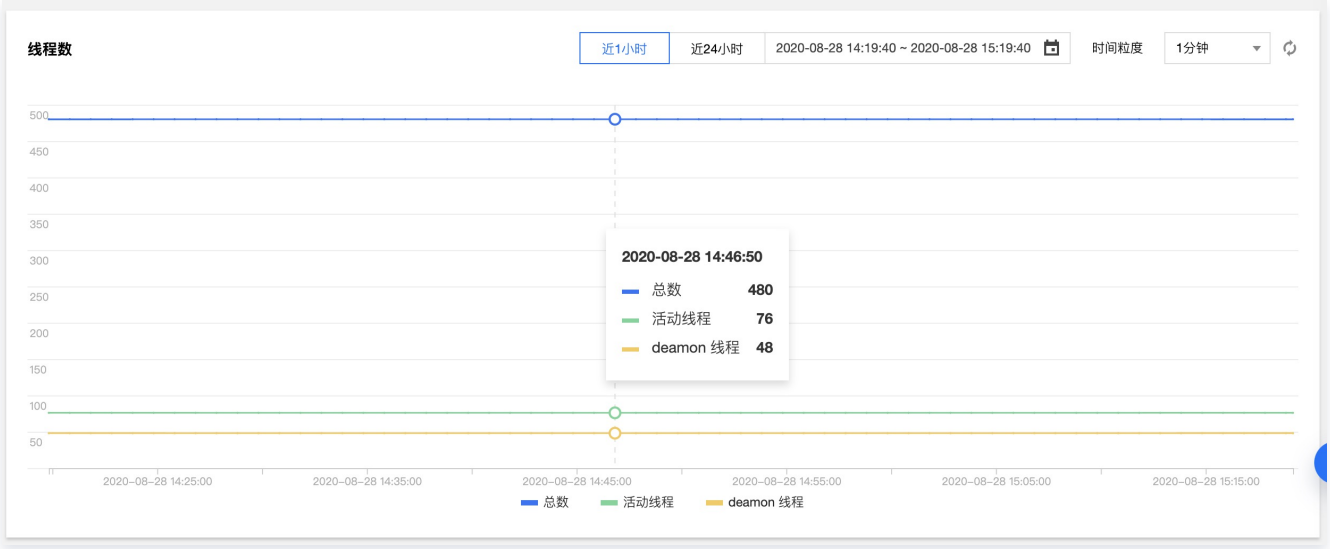
## 线程分析

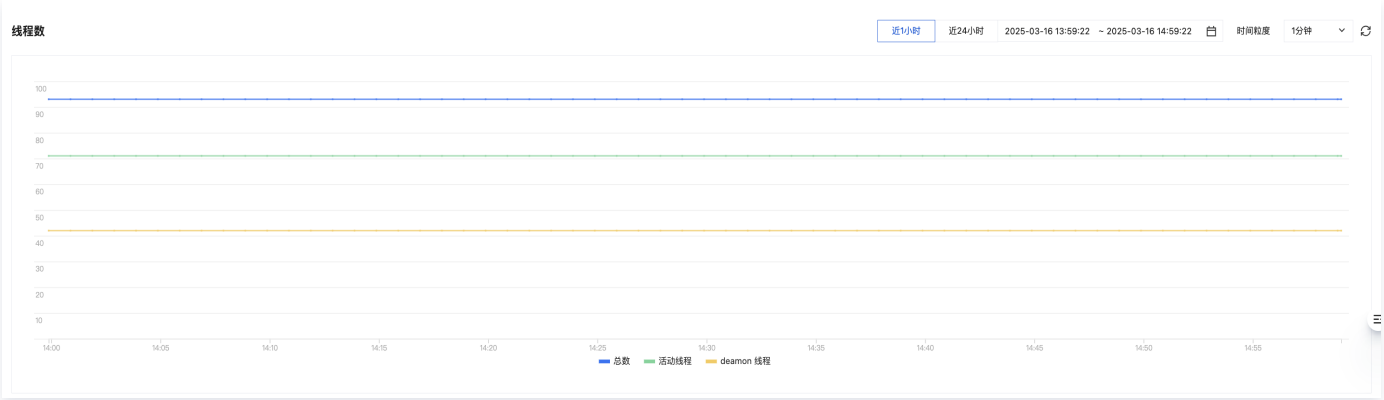
**线程分析**功能提供详细的每类线程统计、栈帧统计和函数分布，帮助您快速定位线程问题。
在线程详情卡片中，单击线程分析页签，您可以查看线程分析结果。




## 线程数

您可在线程数卡片中查看到当前实例的 JVM 线程总数、活动线程数、 daemon 线程数随时间的变化情况。您可通过曲线图上方的时间范围选择器修改曲线显示的时间范围（时间粒度目前支持1分钟和1小时）。




## 火焰图

**功能说明：**您可在**火焰图**页，采集并查看指定时段内程序执行过程所形成的“火焰图”，用以进行性能分析。操作步骤如下：

1.1 选择您所需的数据采集时长。目前支持的数据采集时长有：近5秒、近10秒、近30秒、近1分钟、近3分钟。

1.2 选择您需要采集的火焰图类型，目前支持 CPU、Latency、Allocation 三种类型。

1.3 单击开始采集，等待所选时长后，可在页面中查看到所采集到的数据形成的火焰图。

1.4 再次进入火焰图页面时，可查看到上一次采集的火焰图。您可通过步骤1.1和1.2重新采集当前实时的火焰图。

> **说明：**
> 数据量过大、无热点函数、或出现进程连接失败等异常情况时，火焰图可能采集失败。具体可查看 JVM 监控 常见问题 。




## 请求详情

**功能说明：**单击**请求详情**，可支持查看 Span 的详细信息。

> **说明：**
> 列表页中的“耗时”和“状态”，为服务处理请求的本地耗时和状态。



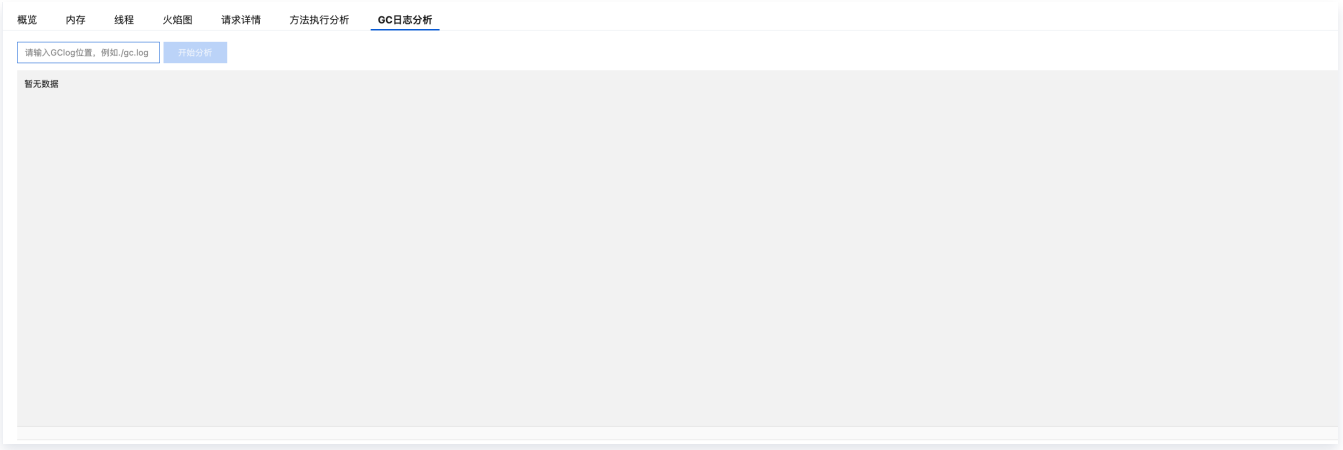

## 方法执行分析

**功能说明：**单击**方法执行分析**，查看实例级别本地方法调用情况，方便您进行性能调优和能力排障。

1. 在页面上方输入**采集类名**，**采集方法名**和**采集次数**。
2. 单击**开始采集**，可查看调用方法某次执行的信息，包含调用方法名称、行号、平均执行时间和该方法内部方法执行耗时时间。
3. 单击操作栏的**钻入**，即重新采样，停止原函数采样，采样次数为默认值1。
4. 单击**重置**，即停止当前采样，回到功能首页。

## GC日志分析

**功能说明：**支持对 GC 日志进行分析。




## 常见问题

- JVM 监控功能为何无法使用?
- JVM 如何升级 tsf-agent?
- 火焰图采集失败如何处理?
- 为何无法查看 JVM 日志?

# 服务依赖拓扑

最近更新时间：2025-04-08 15:25:52

## 操作场景

服务依赖拓扑包含了查询服务之间相互依赖调用的拓扑关系，查询特定集群特定命名空间下服务之间调用的统计结果等功能。您可以通过拓扑图的形式直观的了解应用的上下游组件与它们的调用关系和相关性能数据，从而更快速地找出应用的瓶颈。
本文介绍服务依赖拓扑功能的使用方法。

> **说明：**
> - 支持跨命名空间查看服务依赖拓扑，不仅全局命名空间。
> - 不支持 IE 浏览器。

## 操作步骤

### 创建视图

您可以通过固定视图，固定常用访问的命名空间。当视图中包含多个命名空间，可实现跨命名空间查看调用链。

> **说明：**
> 单个租户下最多可创建100个视图。

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**观测中心**。
2. 在**观测中心**详情页选择**服务依赖拓扑**，进入**服务依赖拓扑**详情页面。
3. 选择好地域后，在左侧视图栏单击+，勾选视图包含的命名空间，填写好试视图名称。
   - 视图类型：支持**仅包含一个命名空间**和**跨多个命名空间查看**，选择**跨多个命名空间查看**时可以同时最多勾选5个命名空间。
   - 视图名称：60个字符内。



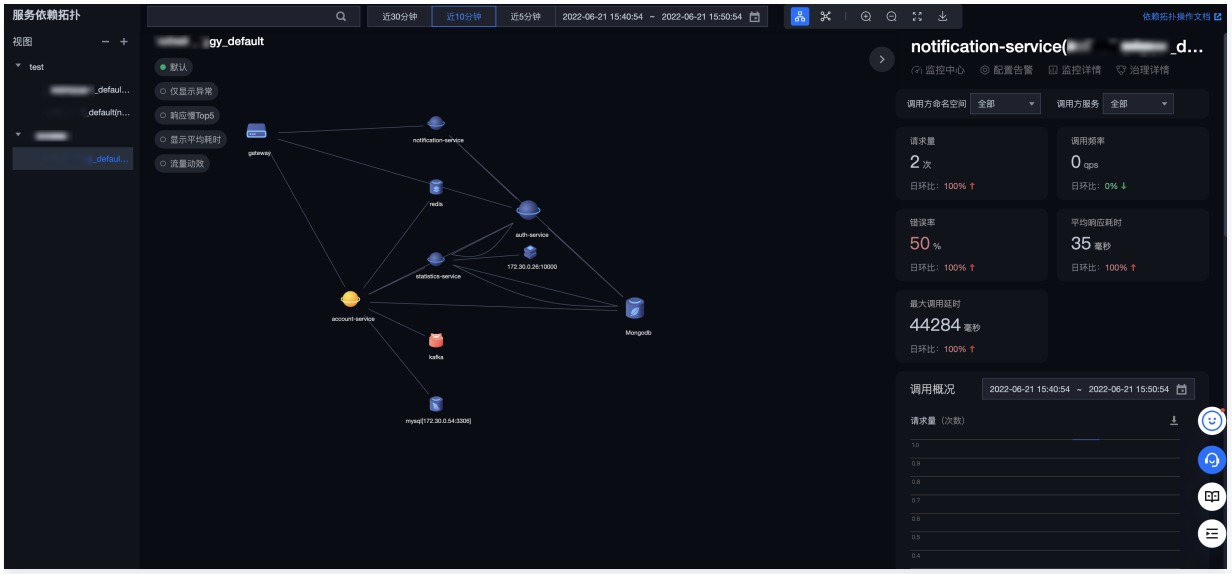

4. 单击**确定**，完成视图创建。 视图创建完成后，您可以在左侧视图栏进行以下操作：
   - 单击+继续创建视图，最多可创建100个视图。
   - 单击—删除您不再需要的视图。
   - 切换不同视图和命名空间查看不同命名空间下的服务拓扑依赖数据。
   - 服务拓扑依赖支持深色和浅色两个版本，您可以根据您的使用习惯单击右上角的 按钮进行切换选择。

### 查看拓扑关系

1. 在服务依赖拓扑页面，选择要查看的视图（展示视图下所有命名空间中的服务依赖关系）或者选择视图下的单个命名空间。

2. 设置好时间范围（支持近30分钟、近10分钟、近5分钟以及选择特定时间段，**特定时间段的时间跨度最长为7天**）后，查看对应的服务依赖调用关系。

> **说明：**
> - 当前服务依赖拓扑图中可以展示消息队列组件（Kafka、RocketMQ）、网关组件（微服务网关）、数据库（Redis、MySQL、PostgreSQL、MongoDB）和 httpclient (RestTemplate、Feign)。
> - 不支持展示服务名为大写字母。
> - 调用中出现消息队列组件时，暂时无法支持批量消费场景下的服务依赖拓扑图。




### 图例说明

- 曲线的箭头表示发出调用，起始图标表示主动调用的服务。
- 蓝色的图标表示成功调用，黄色表示调用失败。
- 曲线上的数字表示两个服务间调用的平均耗时。

在服务拓扑图页面支持以下5个展示筛选条件：

- 默认。
- 仅显示异常：仅展示调用异常的服务。
- 响应慢 Top5：展示调用平均耗时最长的5个调用关系。
- 显示平均耗时：选择后，调用曲线上显示调用平均耗时。
- 显示流量动效：点击查看动效，可以清晰地查看请求流动的方向。

在服务依赖拓扑关系图页面，您还可以进行以下操作：

- 您可以在通过右上角的  图标切换顺序排布和力导图排布。
- 您可以通过右上角的  切换到3D布局，长按鼠标左键进行旋转操作，鼠标右键进行平移操作。
- 您可以使用鼠标滚轮滑动，放大缩小视图。

## 查看监控数据

> **说明：**
> 要使用该功能，需要更新到1.12.0版本之后的 agent 和 SDK（Spring Cloud），否则无法看到调用概览。

在依赖拓扑依赖关系图中，单击某个服务，可锁定该服务整个上下游调用链路。页面显示服务名称、调用箭头、两个服务间调用的平均耗时。




- 单击服务旁的 图标可以进入到服务监控详情页面。详情请参见 服务监控 。
- 单击服务旁的 图标可以进入调用链详情页面。详情请参见 调用链跟踪 。

在页面右侧栏展示被选中服务的核心监控数据：

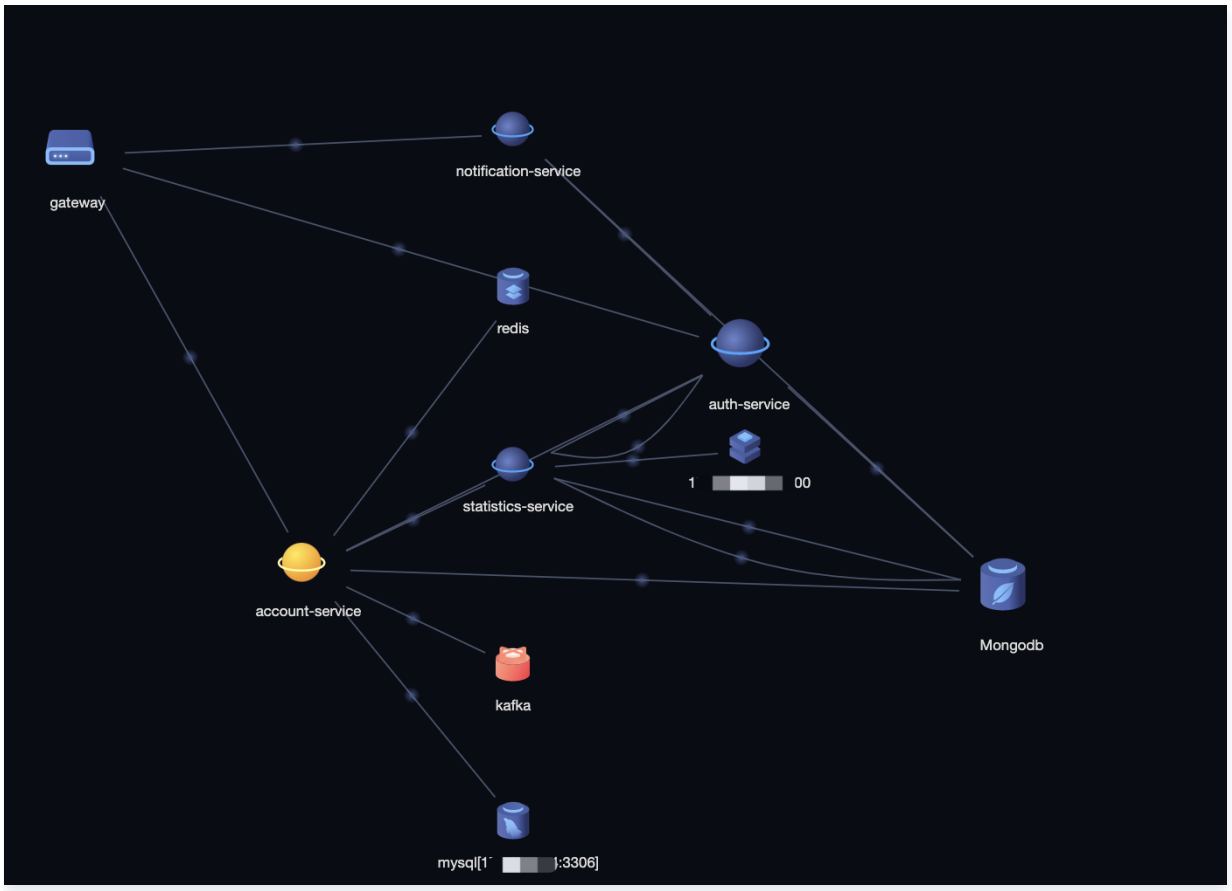
- 右侧栏上方展示选中服务的请求量、调用频率、错误率、平均响应耗时和最大调用时延，以及日环比数据。



- 右侧栏下方展示选中服务核心监控指标曲线图：请求量、错误率、响应耗时、响应耗时分布和 HTTP 状态码。




## 可视化参考




- **对依赖拓扑图的数据说明**

  图中，调用线上的时间，指从上游服务发出请求、到上游服务受到下游服务回包的时间。
  调用线上会经历 Client service send、Server service receive、Server service send、Client service receive 的过程。
  图中，服务圈内的数据，是站在服务 server 端采集到的。
  从平均耗时角度而言，会经历从 Server receive 到 Server send 的过程。
  Client-Server 过程可参考下图：




- **Apdex 释义**

  服务依赖拓扑使用应用性能指数（Apdex）对应用性能满意度进行量化，并使用不同颜色对不同区间 Apdex 的值进行标识，显示 span 健康度，方便您快速发现应用性能问题。
  应用性能指数（Apdex）的计算方式如下：




默认情况下，调用时延说明如下：

- 正常调用：指调用时延小于或等于200ms的调用。
- 慢调用：指调用时延大于200ms小于或等于800ms的调用（4 × 200ms）。
- 极慢调用：指调用时延大于800ms的调用。

Apdex 取值范围说明如下：

| Apdex 取值 | 说明 |
| --- | --- |
| Apdex > 0.75 | 表示调用延时正常，图标为蓝色。 |
| 0.75 ⩾ Apdex > 0.25 | 表示调用延时较大，图标为橙色。 |
| Apdex ⩽ 0.25 | 表示调用延时非常大，图标为红色。 |

**通用参考**

Apdex（Application Performance Index）是由 Apdex 联盟开发的用于评估应用性能的工业标准。Apdex 标准从用户的角度出发，将对应用响应时间的表现，转为用户对于应用性能的可量化范围为 0-1 的满意度评价。

**Apdex 的原理：**

Apdex 定义了应用响应时间的最优门槛为 T（即 Apdex 阈值，T 由性能评估人员根据预期性能要求确定），根据应用响应时间结合 T 定义了三种不同的性能表现：

| 性能表现 | 说明 | 示例 |
| --- | --- | --- |
| Satisfied（满意） | 应用响应时间低于或等于 T。 | 例如，T 为1.5s，则一个耗时1s的响应结果则可以认为是 Satisfied 的。 |
| Tolerating（可容忍） | 应用响应时间大于 T，但同时小于或等于 4T。 | 例如，应用设定的 T 值为1s，则4 × 1=4s为应用响应时间的容忍上限。 |
| Frustrated（烦躁期） | 应用响应时间大于 4T。 | - |

# 调用链查询

最近更新时间：2025-04-08 15:25:52

## 操作场景

调用分析功能展示服务的依赖拓扑调用信息和主被调统计信息。

## 功能入口

1. 登录 腾讯微服务平台控制台 。
2. 在左侧菜单栏中，单击**观测中心**，
3. 选择好时间范围和命名空间，维度选择**服务**，查看当前筛选条件下所有服务的监控图表信息。
4. 单击目标服务卡片，进入服务监控详情页面。
5. 选择**调用分析**页签，查看该服务的调用拓扑图和调用统计信息。

## 调用拓扑与调用链详情

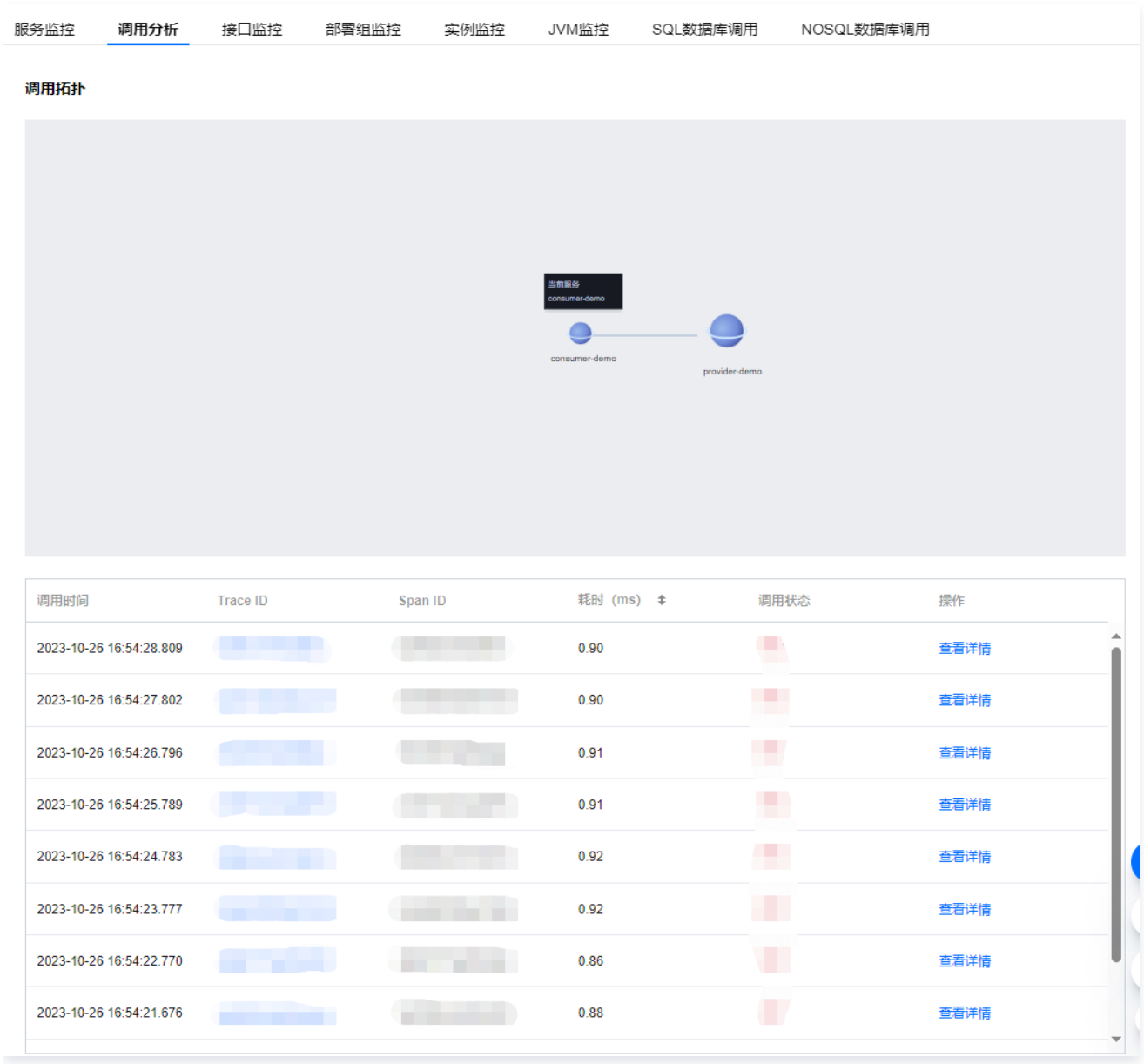
调用拓扑图展示了服务之间相互依赖调用的拓扑关系（包含服务及其上下游服务）和调用链信息。

- 关于服务依赖拓扑的详情，您可以参见 服务依赖拓扑 。
- 关于调用链详情，您可以参见 调用链查询 。

在调用拓扑区域，您可以进行以下操作：

- 单击服务间的依赖线条，弹出面板显示该调用的主被调用方信息、调用数、调用成功率等信息，单击弹出框上的“查看调用链”可以进入到调用链查询界面。
- 单击服务圈内（白色底），可以展示该服务的调用数、调用成功率和平均调用延时，单击弹出框上的“查看调用链”可以进入到调用链查询界面。
- 单击某一条调用链操作列的**查看详情**，可以查看具体的调用详情，包含基本信息、Tag、自定义 Metadata 和 Opentracing Format。

版权所有：腾讯云计算（北京）有限责任公司




## 主被调统计

### 功能说明

TSF 支持从主调和被调两个视角展示服务指标的统计信息。用户可以通过统计信息了解服务指标的变化情况。
服务指标以天为单位进行统计，在当天只能查看**昨天之前**的统计数据。

- 一个服务可能既是主调服务，又是被调服务。TSF 会统计服务作为主调服务调用其他服务及接口的情况，以及服务作为被调，其接口被其他服务调用的情况。
- TSF 服务统计支持查看指标的日环比和周同比。
- 统计指标：
  - 平均响应时间。
  - 按照状态码统计：2xx 响应、3xx 响应、4xx 响应、5xx 响应、其他状态码响应。
  - 按照异常请求统计：超时响应、不可用响应。其中超时响应表示服务端处理超时的请求响应，不可用响应表示服务端无可用实例时的异常请求响应。

> **说明**

> 成功请求为响应码为 "success"、"200"、"201"、"202"、"203"、"204"、"205"、"206"、"207"、"208"、"300"、"301"、"302"、"303"、"304"、"305"、"306"、"307"、"308"、"100"、"101"、"102" 的请求。其他响应码的请求被认为是错误请求。

服务统计仅能查询昨日起近1个月的数据。

## 使用说明

选择主调或被调视角、统计日期。

- **主调视角**下会展示被调方服务的指标列表。单击某个被调服务前的箭头会展示该服务不同接口被调用的统计信息。
- **被调视角**下会展示该服务的接口指标列表。单击某个接口前的箭头会展示该接口被不同主调服务调用的统计信息。

单击操作列的**查看日环比/周同比**，可以查看各指标日环比/周同比情况。

- 日环比表示查询日 T 和之前一天 T－1 的指标数据对比。
- 周同比表示查询日 T 和查询日前7天 T－7 的指标数据对比。

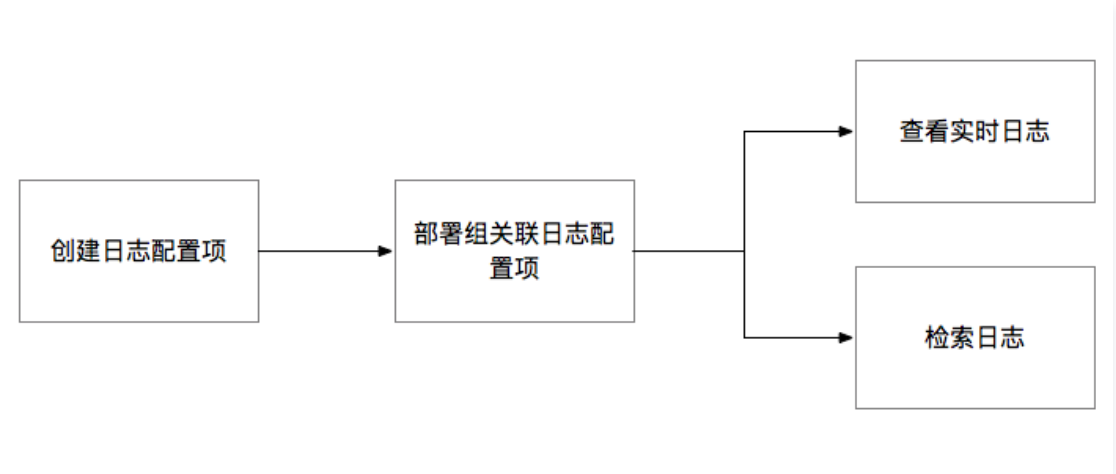


# 日志服务
# 概述

最近更新时间：2022-12-14 14:46:38

## 日志服务简介

日志服务为用户提供一站式日志服务，从日志采集、日志存储到日志内容搜索，帮助用户轻松定位业务问题。用户通过指定部署组的日志配置项来指定日志采集规则，TSF Agent 根据日志配置项采集指定路径下的文件日志，并上传日志到日志存储模块。用户可以通过 TSF 控制台查看部署组实时日志，并根据关键词来检索日志。
同时日志服务还支持日志告警功能，日志告警功能允许您通过配置业务日志中的关键词，设置关键词出现频率的告警，一旦触发告警条件，告警通知人将会收到告警信息。
下图是用户在 TSF 平台上使用日志服务的流程图：




以下视频将为您介绍 TSF 的日志服务功能：
观看视频

## 日志服务原理




**日志采集**
通过 Agent 来采集日志，无须手动安装。
**日志索引与查询**

- 实时索引：采集的日志数据建立索引。
- 查询灵活：支持全文检索、关键词检索（短语或者分词）。

# 日志存储额度与删除策略说明

- 存储额度：TSF 为用户提供了日志存储的免费额度。免费存储额度与客户使用的版本、节点数相关。

| 版本 | 基础版 | 专业版 | 铂金版 |
| --- | --- | --- | --- |
| 每个节点免费额度 | 3GB/节点/月 | 4GB/节点/月 | 20GB/节点/月 |

- 用户可在 TSF 控制台的概览页查看当前日志使用量与存储额度，该数据每小时刷新一次。TSF 在统计日志使用量时仅会统计标准输出日志与日志配置项日志，不包括 GC 日志与 Mesh Sidecar 日志。当日志达到免费额度上限时，平台将通过短信、邮件、微信等方式向您发送通知。超过上限时，我们将按照如下的删除策略删除日志直到日志总存储量低于免费额度。
- TSF 日志删除策略包括两部分：超量删除与超时删除
  - 超量删除针对用户日志使用量超过了日志额度的场景。超量删除仅针对标准输出日志与日志配置项日志。TSF 每小时会检测用户的日志使用量，如果超出用户日志存储额度，TSF 会依次删除最久的日志直到日志使用量低于存储额度，TSF 将优先删除标准输出日志，删除过程不会影响最新日志的写入和查询。TSF 删除日志的粒度由存储量（一般是 60 G）和创建天数（一般是 30 天）共同决定；
  - 超时删除是指 TSF 会每天检测用户的日志并删除写入超过 30 天的日志。超时删除针对标准输出日志、日志配置项日志、GC 日志和 Mesh Sidercar 日志。在用户日志使用量不大或日志存储额度较多的场景下，可能会出现能够查看 30 天之前日志的情况，但 TSF 不保证写入超时 30 天的日志数据的可见性。
- 如果您希望获取更长存储时间或希望对日志进行个性化分析统计，您可以使用腾讯云上的日志服务（CLS）。虚拟机部署业务接入 CLS 请参见 CLS 入门指南 。容器部署业务接入 CLS 请参见 TKE 日志采集 。
- 更多日志服务 CLS 产品能力请参见文档 日志服务产品概述 。

> **注意**
> 日志迁移到 CLS 中，您可以使用 CLS 中更丰富的日志分析能力，但 TSF 中提供的日志与调用链联动能力无法继续使用。

# 快速入门

最近更新时间：2025-04-08 15:25:52

## 操作场景

您可以通过本文介绍的四种方式，在控制台上查看应用程序实时打印的日志，快速了解 TSF 日志服务的使用方式。

## 操作步骤

### 查看标准输出（stdout）日志

TSF 默认提供 stdout 标准输出日志的查看，无需额外设置。假设用户已经完成了通过部署组部署应用的操作，以下为查看日志的步骤：

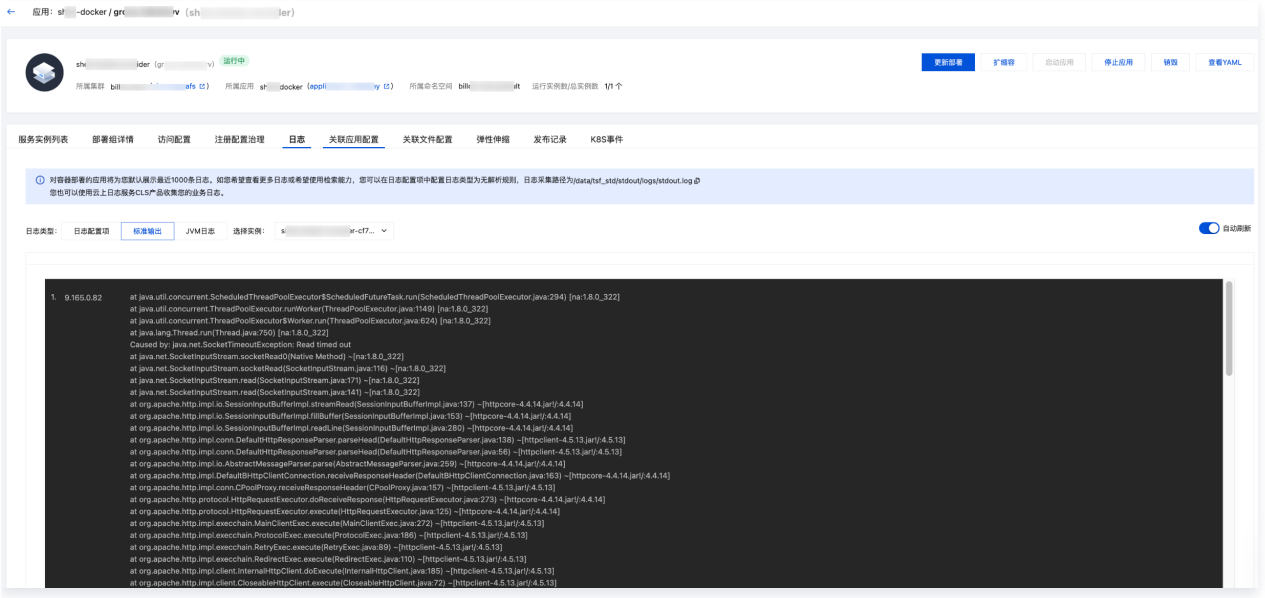
1. 登录 腾讯云微服务平台控制台 。
2. 在左侧导航栏选择**应用管理**，在**业务应用**详情页，单击目标应用，进入应用详情页。
3. 在应用详情页，选择**应用部署**菜单，进入部署组详情页。
4. 单击目标部署组操作栏的**更多 > 查看日志**。

### 使用 Spring Boot 默认日志格式

1. 在腾讯微服务平台控制台**日志配置**界面，创建日志配置项 `consumer-demo-log`，选择日志格式 **Spring Boot**，采集路径为 `/var/root.log` 。
2. 应用程序打印日志到指定目录，在配置文件（例如 application.yml）中设置打印文件日志的路径，和步骤1中日志配置项的日志采集路径保持一致。

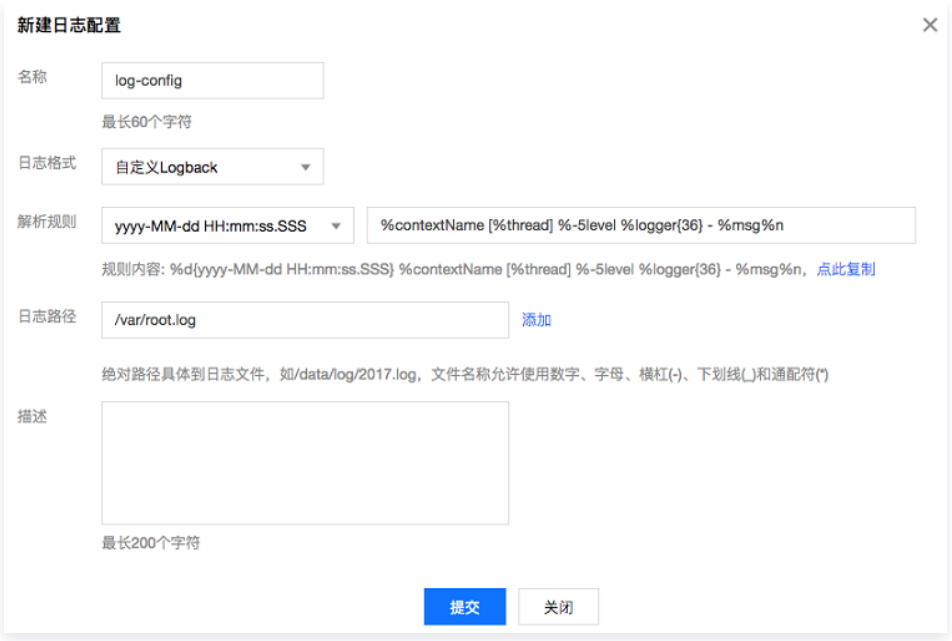
```
logging.file=/var/root.log
```

3. 将应用程序打包后，在控制台上新建部署组，选择创建好的日志配置项 `consumer-demo-log` 。
4. 部署应用。
5. 在部署组操作栏中，单击**更多 > 查看日志**。
6. 日志类型选择**标准输出**。



### 使用自定义 logback 日志格式

1. 在腾讯微服务平台控制台**日志配置**界面创建日志配置项 `log-config`，选择日志格式为**自定义logback**，设置解析规则和日志路径如下图：




2. 应用程序打印日志到指定目录，在配置文件 `logback.xml` 中配置日志的 Pattern。

```
<configuration scan="true" scanPeriod="60 seconds" debug="false">
 <property name="log.path" value="/var/root.log" />
 <appender name="stdout" class="ch.qos.logback.core.ConsoleAppender">
     <encoder>
         <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %contextName [%thread] %-5level %logger{36} -
%msg%n</pattern>
     </encoder>
 </appender>
</configuration>
```

3. 程序打包后，在控制台上新建部署组，选择日志配置项 `log-config` 。
4. 部署应用。
5. 单击部署组操作栏中**更多 > 查看日志**，选择日志配置项 `log-config` 查看日志信息。

## 应用程序不配置日志路径

如果用户不想配置日志的打印路径，但是仍然希望将日志采集后做检索，可以使用 TSF 提供的默认日志打印和采集功能。该功能要求工程满足以下条件：

- 该工程为 Spring Cloud 应用。
- 工程的配置文件中**不要设置** `logging.file` 参数（如果用户设置了该参数，默认日志功能不会生效）。
- 在 pom.xml 中依赖 TSF 的 logger 依赖，该依赖会打印日志到默认日志路径。

```
<dependency>
    <groupId>com.tencent.tsf</groupId>
    <artifactId>spring-cloud-tsf-logger</artifactId>
</dependency>
```

当工程满足上面三个条件后，在 TSF 上的操作步骤如下：

1. 在控制台上新建部署组时，选择日志配置项为 `default-log-config` ，该日志配置项会通知 TSF agent 去采集默认日志路径下的日志。
2. 部署应用。
3. 在部署组操作栏中，单击**更多 > 查看日志**，选择日志配置项 `default-log-config` 查看日志信息。

更多关于日志配置的详细操作请参见 日志配置 。

# 日志格式说明

最近更新时间：2025-04-08 15:25:52

TSF 目前支持 Spring boot、Nginx Access 、自定义 Logback、自定义 Log4j、自定义 Log4j2、单行/多行文本、无解析规则七种类型。

> **说明：**
> 请尽量保持**以年-月-日做日志开头**（例如：2019-09-21、11:09:48.395，因为分割日志时以这种格式作为分割）。否则，可能导致日志显示分行异常，甚至不能显示日志。

## Spring Boot

如果应用程序使用默认的 Spring Boot 日志，则选择 Spring Boot 日志类型。不需设置“日志格式”。

## Nginx Access

如果应用程序使用默认的 Nginx Access 日志，则选择 Nginx Access 日志类型。不需设置“日志格式”。

## 自定义 Logback

如果应用程序使用 Logback 日志配置，设置日志类型为 Logback，然后设置日志格式（即日志解析规则，对应 Logback 中的 pattern）。可参见 Logback 中关于 Pattern 的介绍，Pattern 支持说明和参考模板如下。

### Pattern 支持说明

TSF 支持以下 Pattern 的正常解析：

```
%d  %date
%p %le $level
%c %logger %lo
%C %class
%caller
%F %file
%L %line
%r %relative
%M %method
%trace
```

TSF 对以下 Pattern 贪心匹配：

```
%m %msg %message
%exception %ex %throwable %xEx %xException %xThrowable %rEx %rootException
```

其他 Pattern 不建议使用。

## Pom

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
            xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
```

```
        <groupId>com.tencent.tsf</groupId>
        <artifactId>tsf-demo</artifactId>
        <version>1.16.0-Edgware-RELEASE</version>
    </parent>

    <artifactId>provider-demo</artifactId>
    <packaging>jar</packaging>
    <name>provider-demo</name>

    <dependencies>
        <!-- TSF启动器 包含完整依赖 -->
        <dependency>
            <groupId>com.tencent.tsf</groupId>
            <artifactId>spring-cloud-tsf-starter</artifactId>
        </dependency>
    </dependencies>
</project>
```

## Logback.xml

使用时，应用应该依赖有 LogbackTraceConverter 类的 SDK。

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <springProperty scope="context" name="springAppName" source="spring.application.name"/>
    <conversionRule conversionWord="trace"
converterClass="com.tencent.tsf.logger.LogbackTraceConverter" />
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <!-- encoder 默认配置为PatternLayoutEncoder -->
        <encoder>
            <!-- <pattern>%d %L %c %t %msg%n</pattern>-->
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %trace %p %c [%t] %msg%n</pattern>
        </encoder>
    </appender>
    <appender name="TSF" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <File>${LOG_FILE:-Startup}
        </File>
        <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
            <FileNamePattern>${LOG_FILE:-Startup}.%d{yyyy-MM-dd}.%i.log
            </FileNamePattern>
            <MaxHistory>30</MaxHistory>
            <maxFileSize>100MB</maxFileSize>
            <totalSizeCap>1GB</totalSizeCap>
        </rollingPolicy>
        <encoder>
            <!-- <pattern>%d %L %c %t %msg%n</pattern>-->
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %trace %p %c [%t] %msg%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
    </appender>
    <root level="info">
        <appender-ref ref="STDOUT" />
        <appender-ref ref="TSF" />
    </root>
```

```
</configuration>
```

## 自定义 Log4j

如果应用程序使用 Log4j 日志配置，设置日志类型为 Log4j，然后设置日志格式（即日志解析规则，对应 Log4j 中的 pattern）。可参见 Log4j 中关于 Pattern 的介绍。Pattern 支持说明和参考模板如下。

### Pattern 支持说明

TSF 支持以下 Pattern 的正常解析：

```
%d
%p
%C
%c
%F
%L
%r
%M
%t
%m
%T（trace）
```

## Pom

```
#排除日志中默认的logback依赖，引入log4j的依赖
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
            xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>com.tencent.tsf</groupId>
        <artifactId>tsf-demo</artifactId>
        <version>1.16.0-Edgware-RELEASE</version>
    </parent>

    <artifactId>consumer-demo</artifactId>
    <packaging>jar</packaging>
    <name>consumer-demo</name>

    <dependencies>
        <!-- TSF启动器 包含完整依赖 -->
        <dependency>
            <groupId>com.tencent.tsf</groupId>
            <artifactId>spring-cloud-tsf-starter</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <!--log4j-->
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-log4j</artifactId>
            <version>1.3.8.RELEASE</version>
        </dependency>
    </dependencies>
</project>
```

### Log4j.xml

使用时，应用应该依赖有 Log4JPatternLayout 类的 SDK。请注意日志输出路径。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

 <log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
     <appender name="console" class="org.apache.log4j.ConsoleAppender">
         <param name="Target" value="System.out"/>
         <layout class="com.tencent.tsf.logger.Log4JPatternLayout">
             <param name="ConversionPattern" value="%d %p %l %c %m%n"/>
         <!-- <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss,SSS} %T %p %F %r
%m%n"/>-->
         </layout>
     </appender>
     <appender name="TSF" class="org.apache.log4j.RollingFileAppender">
        <param name="File" value="/tsf-demo-logs/provider-demo/root.log" /><!-- 设置日志输出文件名
-->
        <!-- 设置是否在重新启动服务时，在原有日志的基础添加新日志 -->
        <param name="Append" value="true" />
        <param name="MaxBackupIndex" value="10" />
        <layout class="com.tencent.tsf.logger.Log4JPatternLayout">
            <param name="ConversionPattern" value="%d %p %l %c %m%n"/>
        <!-- <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss,SSS} %T %p %F %r
%m%n" />-->
        </layout>
     </appender>
     <root>
         <priority value ="info" />
         <appender-ref ref="console" />
         <appender-ref ref="TSF"/>
     </root>
</log4j:configuration>
```

## 自定义 Log4j2

如果应用程序使用 Log4j2 日志配置，设置日志类型为 Log4j2，然后设置日志格式，可参见 Log4j2 的介绍。Pattern 支持说明和参考模板如下。

### Pattern 支持说明

TSF 支持以下 Pattern 的正常解析：

```
%d %date
%p $level
%c %logger
```

```
%C %class
%F %file
%L %line
%r %relative
%M %method
%t %tn %thread %threadName
%pid %processId
%N %nano
%tp %threadPriority
%T %tid %threadId
%sn %sequenceNumber
%u %uuid
%trace
```

**TSF 对以下 Pattern 贪心匹配：**

```
%m %msg %message
%exception %ex %throwable %xEx %xException %xThrowable %rEx %rootException
```

**其他 Pattern 不建议使用。**

## Pom

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
            xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>com.tencent.tsf</groupId>
        <artifactId>tsf-demo</artifactId>
        <version>1.16.0-Edgware-RELEASE</version>
    </parent>

    <artifactId>provider-demo</artifactId>
    <packaging>jar</packaging>
    <name>provider-demo</name>

    <dependencies>
        <!-- TSF启动器 包含完整依赖 -->
        <dependency>
            <groupId>com.tencent.tsf</groupId>
            <artifactId>spring-cloud-tsf-starter</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <!--log4j2-->
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-log4j2</artifactId>
            <version>2.1.6.RELEASE</version>
        </dependency>
        <!--log4j2-->
        <dependency>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
            <version>1.2</version>
        </dependency>

    </dependencies>
</project>
```

## Log4j2.xml

**日志文件名应为 Log4j2.xml。请注意日志输出路径。**

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration status="WARN">
    <Properties>
        <Property name="basePath">/tsf-demo-logs/provider-demo</Property>
    </Properties>
    <appenders>
        <!--  输出控制台的配置 -->
        <Console name="console" target="SYSTEM_OUT">
            <!--  控制台只输出level及以上级别的信息（onMatch），其他的直接拒绝（onMismatch） -->
            <ThresholdFilter level="trace" onMatch="ACCEPT" onMismatch="DENY"/>

                <PatternLayout pattern="%d %nano %p %M %uuid %m%n" />
        <!-- <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss,SSS} %trace %level %F [%l]
%message%n" />-->
        </Console>
        <RollingFile name="TSF" fileName="${basePath}/root.log"
filePattern="${basePath}/root.%d{yyyy-MM-dd}.%i.log">
            <PatternLayout>
                <PatternLayout pattern="%d %nano %p %M %uuid %m%n" />
            <!-- <pattern>%d{yyyy-MM-dd HH:mm:ss,SSS} %trace %level %F [%l]
%message%n</pattern>-->
            </PatternLayout>
            <Policies>
                <TimeBasedTriggeringPolicy interval="1" modulate="true" />
            </Policies>
        </RollingFile>
    </appenders>

    <!-- 然后定义logger，只有定义了logger并引入的appender，appender才会生效 -->
    <loggers>
        <root level="info">
            <appender-ref ref="console"/>
            <appender-ref ref="TSF" />
        </root>

    </loggers>
```

```
</configuration>
```

## 单行/多行文本

当日志为自定义格式时，可选择日志类型“单行/多行文本”，日志日期时间部分提供四种可选的格式，然后设置剩余部分内容的格式。日期格式支持以下四种：

- yyyy-MM-dd HH:mm:ss.SSS
- yyyy-MM-dd HH:mm:ss,SSS
- dd/MMM/yyyy:HH:mm:ss.SSS
- dd/MMM/yyyy:HH:mm:ss,SSS

## 无解析规则

指一行日志内容为一条完整的日志。日志服务在采集的时候，将使用换行符 `\n` 来作为一条日志的结束符。为了统一结构化管理，每条日志都会存在一个默认的键值 `__CONTENT__`，但日志数据本身不再进行日志结构化处理，也不会提取日志字段，日志属性的时间项由日志采集的时间决定。

版权所有：腾讯云计算（北京）有限责任公司

# 日志配置项

最近更新时间：2025-04-08 15:25:53

## 操作场景

日志配置项用于指定采集日志的规则，包括日志的采集路径和日志解析格式。用户可以在 TSF 控制台上创建日志配置项，然后将配置项发布到部署组上。同一个部署组可以关联多个日志配置项。

## 操作步骤

### 创建日志配置项

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**观测中心**。
2. 在**观测中心**详情页选择**日志服务**，进入日志服务详情页面后选择日志配置标签页。
3. 单击**新建日志配置项**，在“创建日志配置”页面中设置您的日志配置项信息。



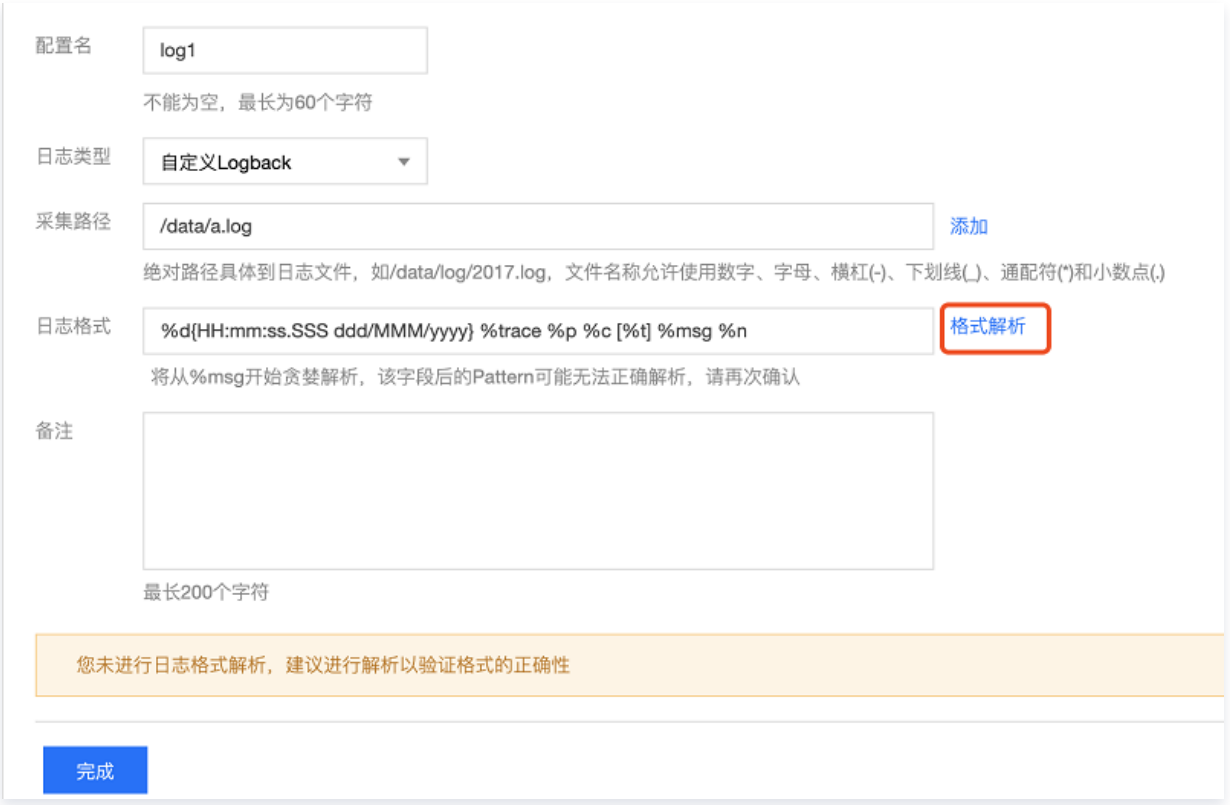

- **配置名**：填写日志配置项名称，不能为空，不超过 60 个字符。
- **日志类型**：根据应用程序的日志类型选择一种日志类型，目前支持 Spring boot、 Nginx Access 、自定义logback、自定义log4j、自定义log4j2 、单行/多行文本、无解析规则七种类型。
- **采集路径**：设置您的日志采集路径，可配置一个或者多个日志采集路径。目前支持**绝对路径具体到日志文件**，如/data/log/2017.log，文件名称允许使用数字、字母、横杠 `-` 、下划线 `_` 、通配符 `\*` 和小数点 `.` 。日志文件必须具有文件后缀，文件名和目录支持通配符，如 `/data/log/*.log` 或 `/var/log/*/*.log` 。

  > **注意：**
  > 日志配置项目的根目录不可以配置成根目录。

- **日志格式**：当日志类型为“自定义 logback”、“自定义 log4j”、“自定义log4j2” 及 “单行/多行文本”时，需设置日志格式（参见 日志格式说明 ）。

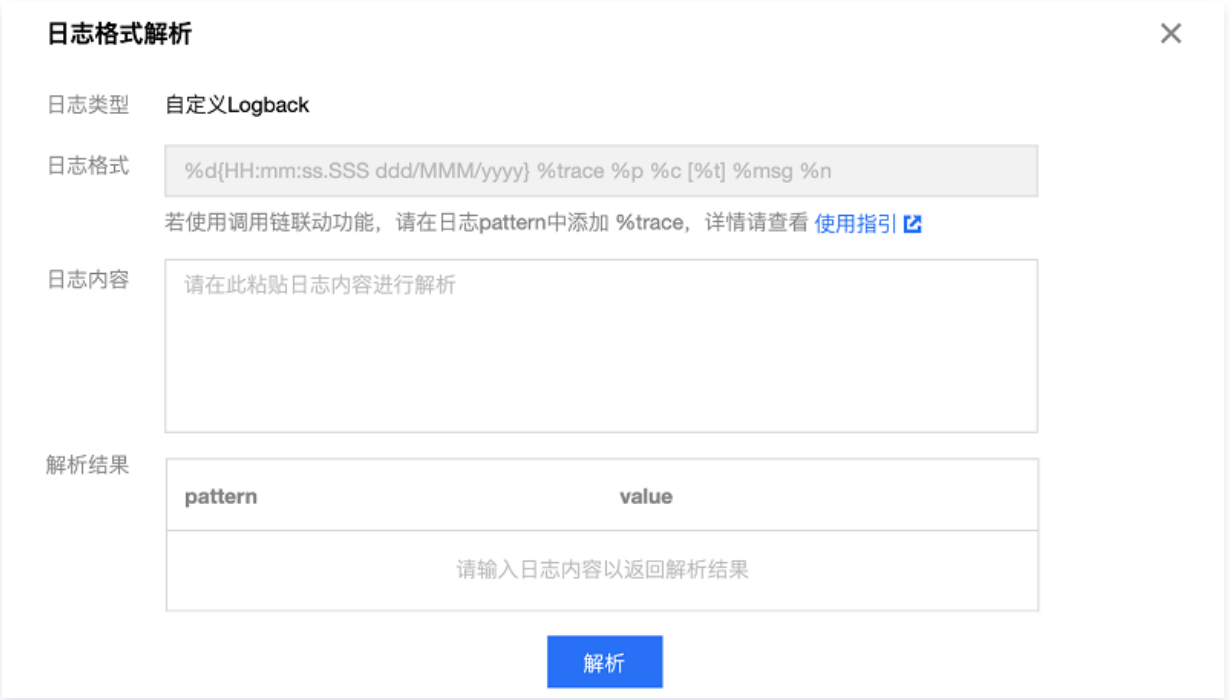
  > **说明**
  > 若要实现调用链与业务日志联动，部署组关联的日志配置项必须遵守日志格式规范。
  > - Spring Boot 格式默认可以支持日志调用链联动。
  > - 自定义 Logback、自定义 Log4j、自定义 Log4j2 和单行/多行文本格式日志需要日志 pattern 中添加 %trace 才可以支持日志和调用链联动。

4. Nginx Access 和无解析规则格式日志无法支持日志和调用链联动。
   设置日志格式后，您可以通过“格式解析”功能，检验当前的设置是否正确：
   4.1 单击**格式解析**，打开日志格式解析弹框。




   - **备注：**选填，日志配置项的描述信息。

   4.2 复制部分日志内容粘贴在“日志内容”区域，单击**解析**，查看“解析结果”区域的输出是否符合预期。




5. 单击**提交**完成创建。

## 默认日志配置项

TSF 提供默认日志配置项，格式是 Spring Boot。日志路径包括三部分，相对于应用启动路径：

- `./*.log`
- `./log/*.log`
- `./logs/*.log`

用户可以在微服务应用的配置文件中配置 `logging.file` 为上述任意一个路径，然后在创建部署组时，选择默认日志配置项 `default-log-config` 。

> **注意：**
> 默认日志配置项仅适用于虚拟机部署的应用。

## 部署组关联日志配置项

部署组关联日志配置项有两个入口：

- 用户可以在创建应用时将日志配置项关联到应用。
- 在日志配置项列表，右侧操作栏中单击**发布规则**。

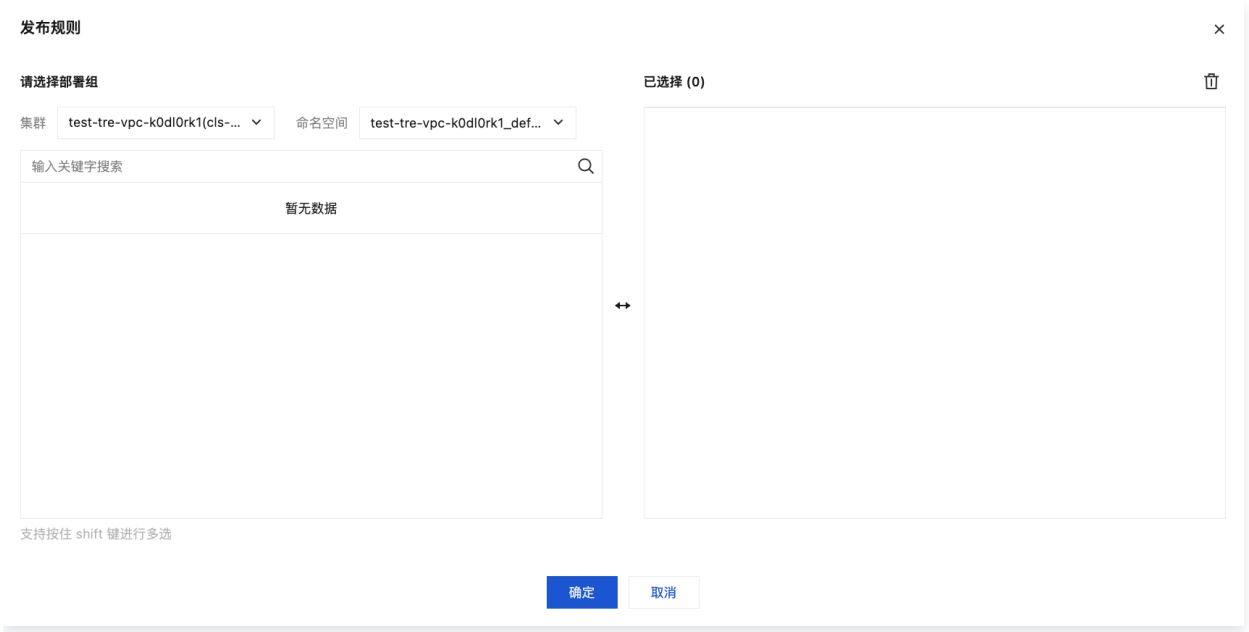
**入口1：创建部署组时选择日志配置项**

1. 登录 腾讯云微服务平台控制台 。
2. 在左侧导航栏选择**应用管理**，在与业务应用详情页，单击目标应用，进入应用详情页。
3. 单击**应用部署**，在第4步：**服务注册配置&可观测**的可观测部分选择关联的**日志配置项**。




**入口2：在日志配置项列表发布配置**

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**观测中心**。
2. 在**观测中心**详情页选择**日志服务**，进入日志服务详情页面后选择日志配置Tab页。
3. 在日志配置页，单击操作列的**发布规则**。
4. 选择目标部署组后，单击**提交**完成发布。




## 删除日志配置项

当日志配置项没有被其他部署组关联时，才可以删除日志配置项。

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**观测中心**。
2. 在**观测中心**详情页选择**日志服务**，进入日志服务详情页面后选择日志配置Tab页。
3. 在日志配置页，单击操作列的**更多 > 删除**。
4. 在删除确认框中单击**确认**，即可删除日志配置项。

## 修改日志配置项

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**观测中心**。
2. 在**观测中心**详情页选择**日志服务**，进入日志服务详情页面后选择日志配置Tab页。
3. 单击目标日志配置项的“ID”，进入日志配置基本信息页面。
4. 单击右上角的**编辑**，完成修改。

> **说明：**
> - 不支持修改默认日志配置项。
> - 日志配置项修改后，虚拟机部署的应用会自动使用修改后的日志配置项来采集日志，容器部署的应用需要重新部署后才会使用修改后的日志配置项来采集日志。

# 日志投递

最近更新时间：2025-04-08 15:25:53

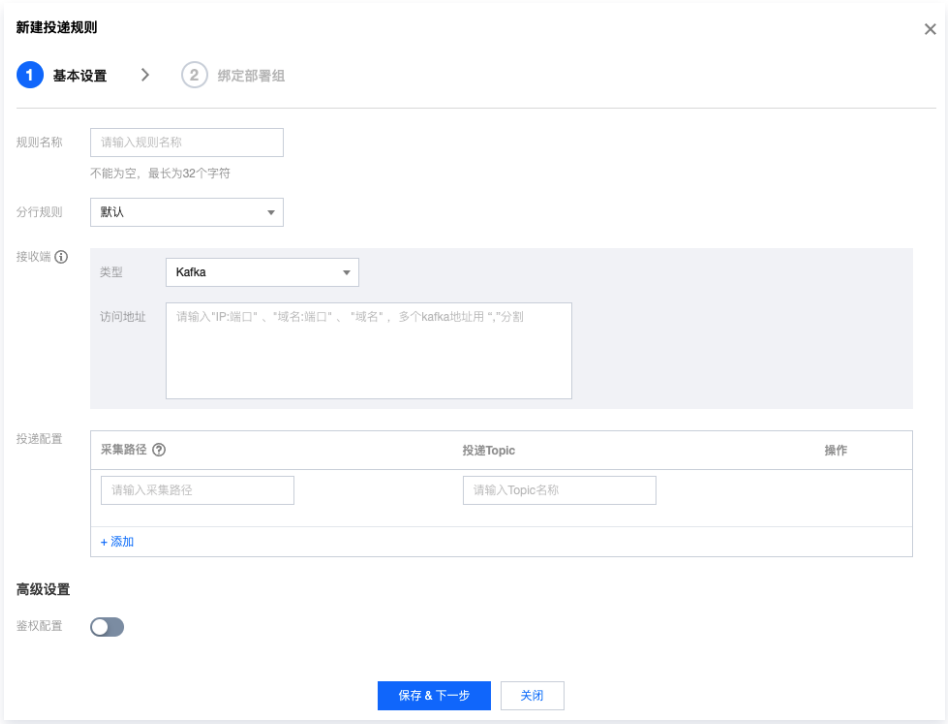
## 操作场景

TSF 的日志服务，提供日志投递功能。您可通过配置日志投递规则，并将其发布至目标部署组，从而实现部署组日志的转存。

## 操作步骤

### 新建日志投递规则

1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**观测中心**。
2. 在**观测中心**详情页选择**日志服务**，进入日志服务详情页面后选择日志投递标签页。
3. 单击**新建投递规则**，在弹出的**新建投递规则**弹框中输入您的规则设置。

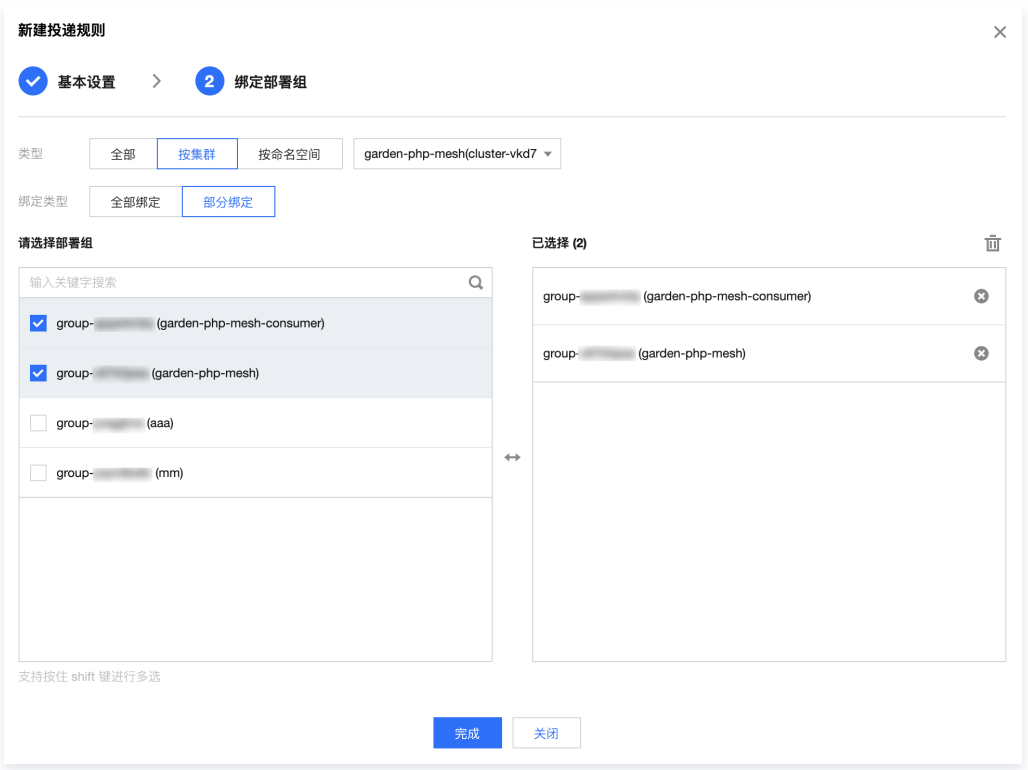



- 规则名称：必填，文本，长度限制32个字符。
- 接收端：日志转存至的目标位置。
  - 仅支持设置一个接收端。
  - 目前仅支持 Kafka 类型的接收端，您需设置 “IP:端口”、“域名:端口” 或“域名”，多个 Kafka 地址用","分隔。输入集群后可以检测集群状态，如果通过可以进行下一步操作；失败返回失败信息，您可以查看异常信息排查问题。
  - 所填写的 Kafka 集群，可为自建 Kafka 集群或腾讯云 CKafka。
- 投递配置：支持从不同路径采集到的日志投递到同一个 kafka 实例的不同 Topic 中，可通过**添加**按钮添加多个路径输入框，以设置多个路径。
  - 采集路径：需要具体到某一个日志文件
  - 投递 Topic：填写该目录的文件投递到具体哪一个 Topic 中
- 高级设置：配置投递到 Kafka 的鉴权规则。当打开鉴权配置后，需要填写 Kafka 访问的用户名与密码。其中当投递目的地为腾讯云 CKafka 时，用户名和密码的填写方式可以参见文档 配置 ACL 策略 。

> ⚠ **注意：**
> - 所配置的 Kafka 必须和应用部署的 VPC 相同。

- 请在 CKafka 上开启白名单，以保证日志采集端和 Kafka 的网络联通性。
- 日志采集路径请勿以 /data/tsf_apm 开头。
- 同一个部署组上只支持添加一条投递规则。如果配置多条规则，最后配置的规则将覆盖原有规则。

4. 单击**保存&下一步**，进入绑定部署组页面。
5. 绑定部署组，并单击**完成**。
6. 规则创建完成后，您可将规则绑定至多个部署组。绑定后，此投递规则即会在所选的部署组中生效，部署组中规则指定路径下的日志，会被自动转存至目标接收端。




## 查看日志投递规则详情

日志投递规则创建后，会以列表项的形式呈现在**日志投递**页面的规则列表中。

1. 单击某列表项中的**规则 ID**，进入该规则的详情页。
2. 在规则详情页中，查看其基本信息和发布情况（规则已绑定的部署组列表）。

## 发布日志投递规则

投递规则创建完成后，您可通过**发布规则**将投递规则绑定至更多部署组。

1. 选择目标规则，进入发布规则页面。您可以通过以下任一方式操作：
   - 方式一：在投递规则列表页中找到目标规则，单击操作列的**发布规则**。
   - 方式二：进入目标规则的**规则详情** > **发布情况**页中，单击左上方的**发布规则**。
2. 选择所需绑定的部署组，单击**提交**即可。

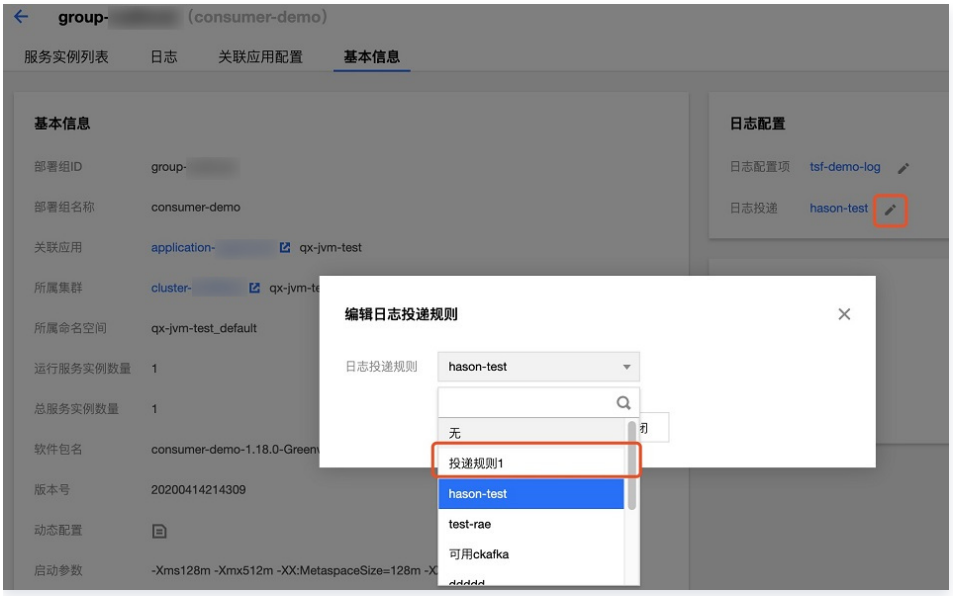
> **说明**
> 已绑定当前规则的部署组不可被选中；已绑定其它规则的部署组，被绑定新规则时，部署组的投递规则被替换为新的规则。

## 修改部署组的投递规则

您可通过以下方式，修改部署组所绑定的投递规则。

- 方式一：您可通过将新的规则绑定至部署组，替换其原有投递规则。

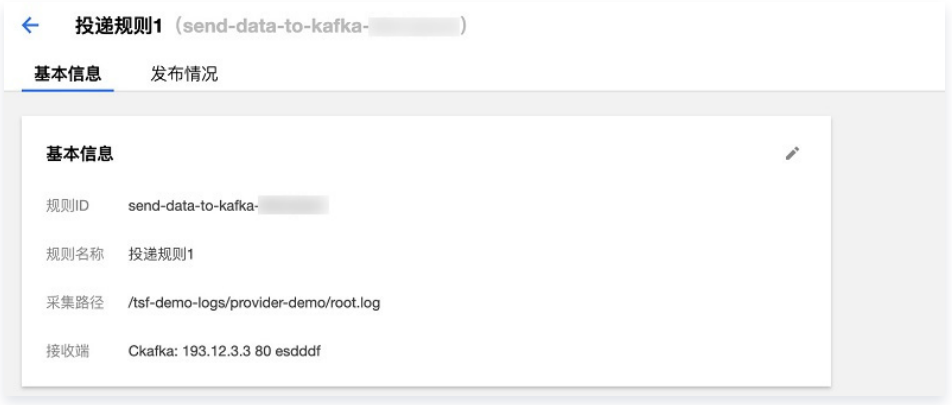
- 方式二：在部署组的详情页的**基本信息**中，编辑其投递规则；选择新的规则并保存后，部署组的投递规则即更新为新选的规则。




## 编辑日志投递规则

规则创建成功后，您可随时修改规则的配置信息。

1. 在规则列表页中找到所需修改的规则项，单击规则 ID 进入其详情页。




2. 单击详情页**基本信息**卡片右上方的“编辑”图标，在打开的**编辑投递规则**弹框中，编辑规则配置项后，单击**保存**即可。



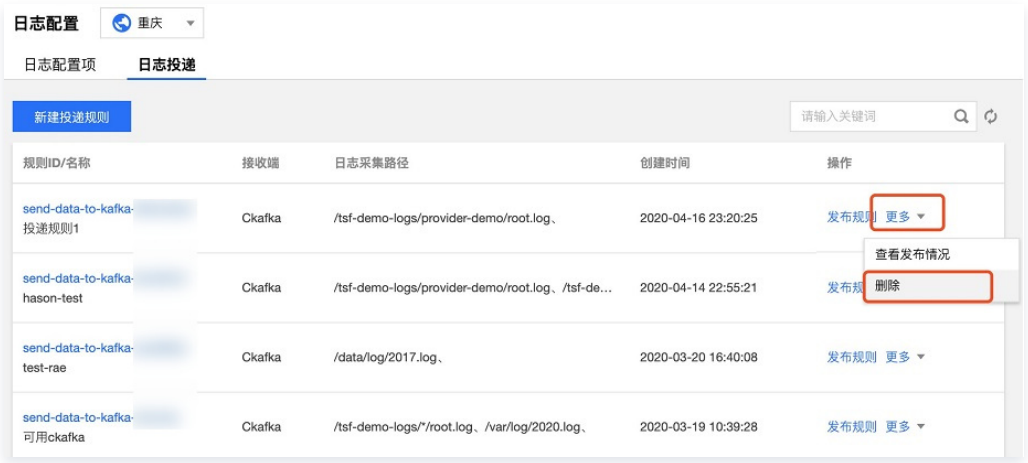
3. 规则修改完成后，自动在规则已绑定的部署组中生效。

## 删除日志投递规则

投递规则创建后，您也可以随时将其删除。

> **注意：**
> 规则删除后，自动把该规则从所有已绑定该规则的部署组中清除，并停止此规则的日志投递。

1. 在投递规则列表页中找到目标规则，单击操作列中的**更多 > 删除**。
2. 二次确认后即可将规则删除。



# 日志告警

最近更新时间：2025-04-08 15:25:53

## 操作场景

日志告警功能允许您通过配置业务日志中的关键词，设置关键词出现频率的告警。您需要在 TSF 控制台上配置需要告警的关键词和监控对象，并在腾讯云可观测平台界面上配置告警通知人。

## 前提条件

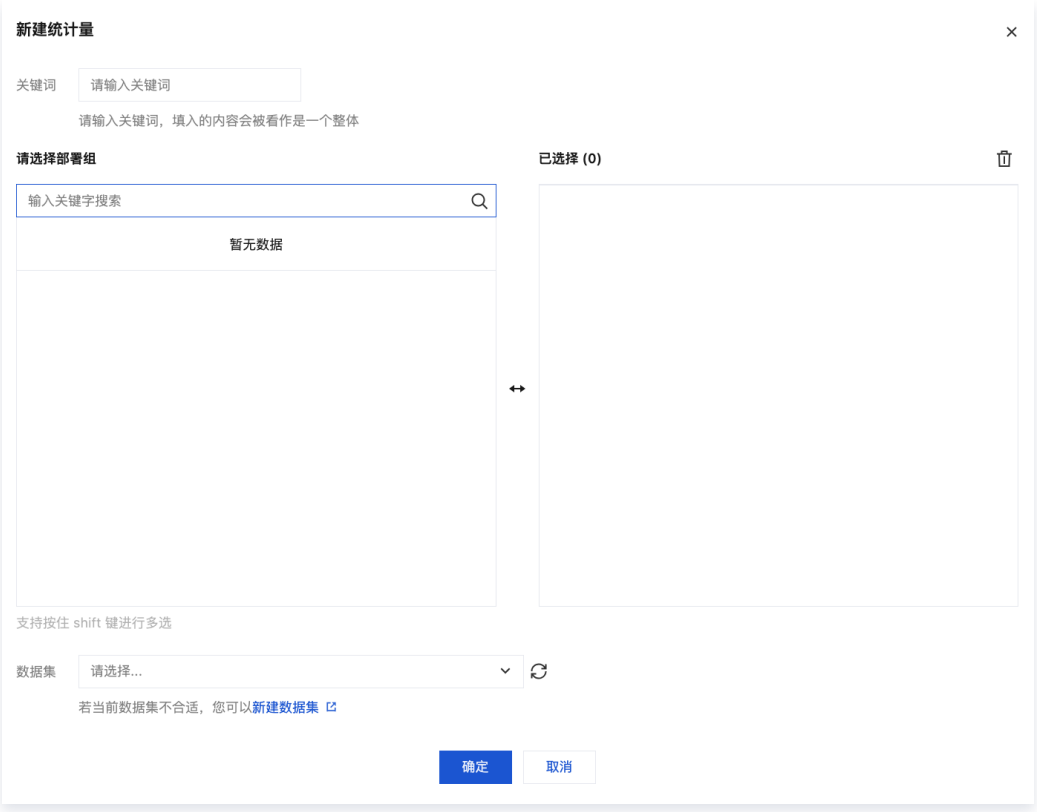
要触发日志告警，您需要先完成以下操作：

1. 已采集日志数据（参见日志服务 快速入门 ）
   - 部署组已经关联了日志配置项
   - 程序打印的日志的路径和解析规则和日志配置项保持一致
2. 日志中有告警统计规则中的关键词

## 操作步骤

### 新建告警统计

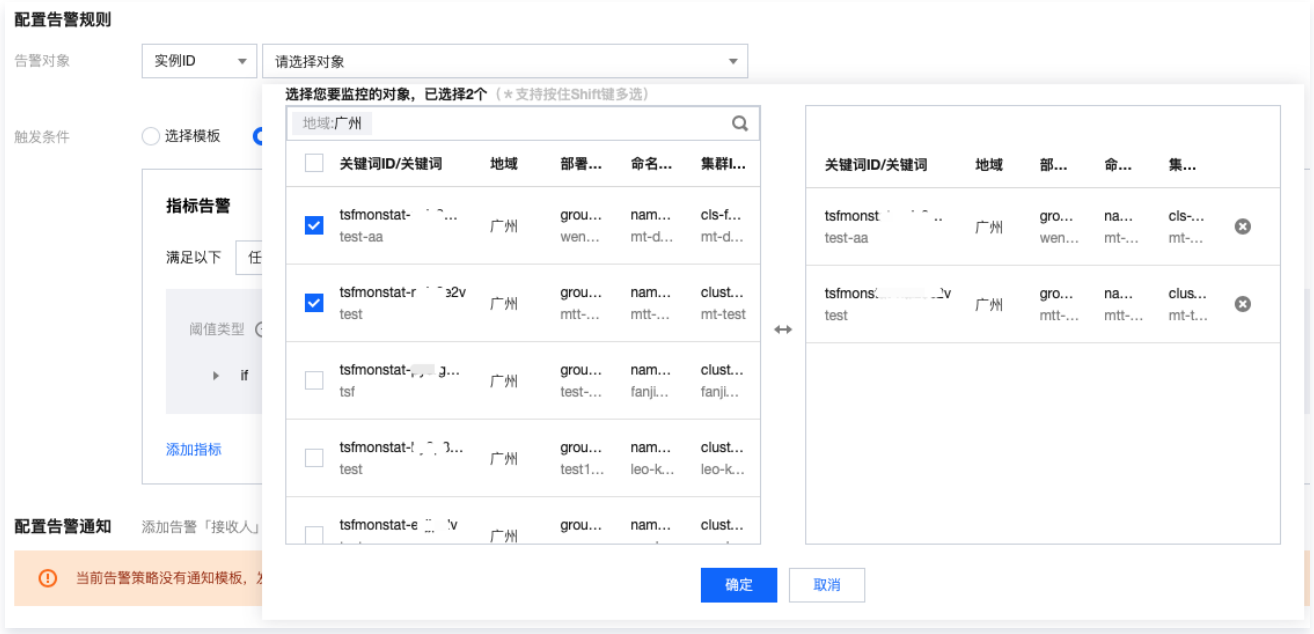
1. 登录 腾讯微服务平台控制台 ，在左侧导航栏，选择**观测中心**。
2. 在**观测中心**详情页选择**日志服务**，进入日志服务详情页面后选择日志告警列表标签页。
3. 在日志告警列表页面，单击左上角**新建告警统计**，在弹框中填写相关信息。
   - 关键词：填写需要设置告警的关键词，关键词中可以包含空格、引号、逗号、句号、冒号等常见特殊符号。填入关键词输入框的内容都将被视为一个整体进行监控告警。整体输入内容不能超过20字符。当前暂不支持正则表达式。
   - 部署组：选择需要监控部署组。此时可以选择某集群和某命名空间下的多个部署组。统计时多个部署组分别统计。
   - 数据集：选择需要关联的数据集。




4. 单击**提交**，完成告警统计。

## 配置告警策略

1. 配置好告警统计后，在 日志告警 列表页面，单击**前往配置告警**，跳转至 腾讯云可观测平台控制台 。
2. 在**腾讯云可观测平台** > **告警配置** > 告警策略 页面，单击左上角的**新建**。
3. 在新增策略界面，填写新建策略内容。
   - 策略名称：20字以内。
   - 策略类型：选择 **TSF/日志告警**。
   - 所属项目：通常为默认项目。
   - 告警对象：选择在已经在 TSF 控制台上配置的告警统计对象（关键词和需要对这个关键词进行统计的部署组）。此处支持多选。




   - 触发条件：针对日志告警，需要填写上面选择的关键词在一定长度的统计周期下出现次数超过一定范围触发告警。
   - 告警渠道：支持选择**已有模板**或者**新建模板**。
   - 接收对象：支持用户或者用户组，单击输入框后面的**新增用户组**，跳转至 访问管理控制台 配置。
   - 接收渠道：支持邮件、短信、微信、企业微信和电话等。
   - 接口回调：填写回调地址方便腾讯云可观测平台将告警信息推送到该地址。




4. 单击**完成**。当被监控的对象发生告警时，告警接收组的用户即可在配置的接收渠道上收到监控信息。

## 查看告警策略详情

1. 登录 腾讯云可观测平台控制台 。
2. 在**告警配置** > 告警策略 中，查看当前配置的告警策略列表。
3. 在策略列表中，单击目标策略的策略名称，即可查看策略详情。
   在这个页面上，用户可以修改当前的告警策略。**当某一条策略被修改时，该策略导致的告警条目在告警列表中告警状态将展示为“数据不足”**。
4. 在管理告警策略页面底部，您可以填写告警回调信息，填写公网可访问到的 URL 作为回调接口地址（域名或 IP[:端口][/path]），腾讯云可观测平台将及时把告警信息推送到该地址。

## 查看告警列表

1. 登录 腾讯云可观测平台控制台 。
2. 在左侧导航栏中，选择 告警历史，进入告警列表页。
   在此页面上，可以看到近期的告警信息。其中告警状态表明的是曾经发生的告警在当前的状态是否依然能够触发告警。当显示为 “已恢复” 时，表明此时告警情况已经被修复。

## 编辑告警对象

告警对象，即需要配置告警的关键词和部署组，需要在 TSF 控制台 上进行编辑。
在**日志服务** > 日志告警 > **日志告警列表**的操作栏，可以对告警对象进行编辑和删除。

版权所有：腾讯云计算（北京）有限责任公司

# 查看日志

最近更新时间：2025-04-08 15:25:53

## 操作场景

TSF 会根据部署组的日志配置项来采集业务日志。例如日志配置项的采集路径是 `/tsf-demo-logs/tsf-inventory/*.log`，TSF 会采集该路径下的文件日志。
TSF 默认提供 stdout 标准输出日志的查看，无须额外设置。

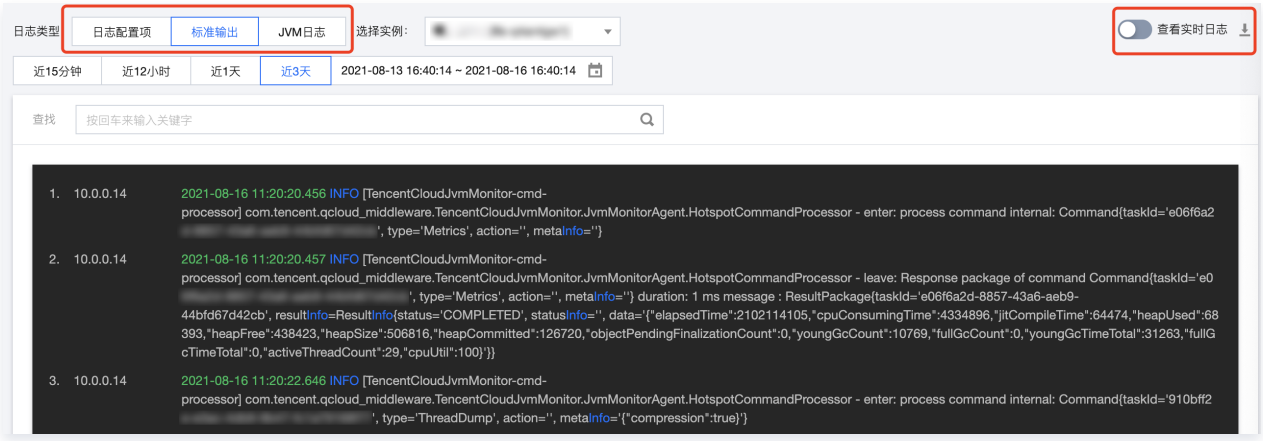
## 操作步骤

1. 登录 腾讯微服务平台控制台，在左侧导航栏单击**应用管理**，**应用详情** > **应用部署**进入部署组列表页。
2. 在目标部署组列表页，您可以通过两种方式查看日志：
   - 方式一：单击操作列的**更多** > **查看日志**，即可查看实时日志。
   - 方式二：单击部署组 ID 进入详情页中，单击顶部的**日志**标签页查看实时日志。




3. 日志信息如下图所示，切换日志类型查看日志配置项的业务日志或者 stdout 标准输出日志，切换右上角按钮查看实时日志和历史日志。
   - 日志配置项：查看日志配置项的采集路径下的业务日志。
   - stdout（标准输出）日志：查看实例标准输出日志。

> **说明**
> 查看标准输出日志时，对容器部署的应用将为您默认展示最近1000条日志。如您希望查看更多日志或希望使用检索能力，您可以在日志配置项中配置日志类型为无解析规则，日志采集路径为 `/data/tsf_std/stdout/logs/stdout.log`。您也可以使用云上日志服务 CLS 产品收集您的业务日志。




## 常见问题

- 为什么无法显示 stdout 日志?
- 为什么无法显示自定义的文件日志?
- 为什么实时日志一直加载不出来?
- 能否关掉 tsf-task-schedule-xxx 线程或者日志?

# 日志检索

最近更新时间：2025-04-08 15:25:53

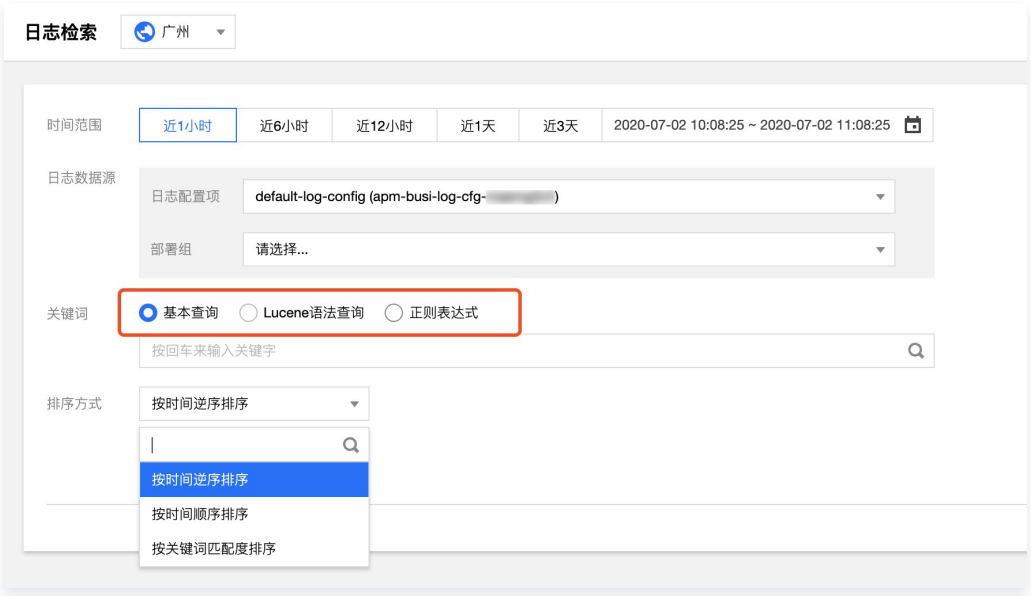
## 操作场景

当部署组关联了日志配置项后，TSF 会对采集的日志数据实时建立索引。用户可以在 TSF 控制台中，通过日志检索功能使用关键词检索出关键日志信息。

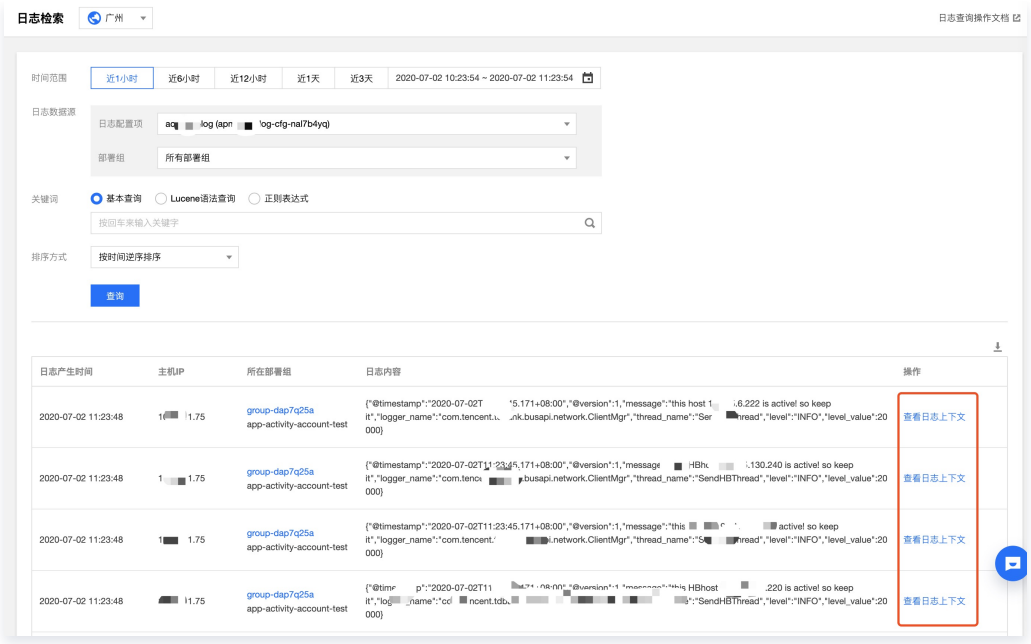
## 操作步骤

1. 登录 腾讯微服务平台控制台，单击左侧导航栏**观测中心**。
2. 在**观测中心**详情页，选择**日志服务 > 日志检索**。
3. 选择搜索的时间范围。
4. 选择日志数据源
   - 日志配置项：日志配置项决定了日志格式、日志采集路径等规则；
   - 部署组：从关联了日志配置项的部署组中进行选择；
   - 实例：当选择单个部署组时，可以将数据源范围精细化到部署组内某个实例。
5. 选择排序方式，默认是**按关键词匹配度排序**。
6. 输入日志关键词，单击**查询**。
   关键词搜索包含三种查询方式：基本查询 、Lucene 语法查询 及 正则表达式 。用户可根据需求自行进行选择。在选定其一查询方式后，输入关键字。

   > **说明：**
   > 中文日志场景下，请使用正则表达式方式进行检索。




7. 单击**查看日志上下文**，查看该条日志的上下文信息。




**说明：**
时间范围选择组件的结束时间为进入日志检索页面的时间。

# 关键词查询

## 基本查询

输入待查询的关键词或短语，按下回车键，输入下一个待查询的关键词或短语，按下回车，直至全部输入完成。单击**查询**，日志列表将展示出所有符合条件的日志。
本查询方式默认将 `任意空白字符, . [ ] { } ( ) ? : =` 认定为分词符，含有上述符号的日志被按照分隔符所在位置拆分。例如 `org.apache.kafka.clients.NetworkClient` 将被分词为 `org apache kafka clients NetworkClient` 。

- 支持单词的搜索
  例如：输入 `available` ，输出日志为所有含有单词 `available` 的日志。
- 支持短语的搜索
  例如：输入 `Broker may not be available` ，输出日志为所有含有短语 `Broker may not be available` 的日志。

## Lucene 语法查询

| 支持检索类型 | 规则 | 示例 |
| --- | --- | --- |
| 全文检索-单词检索 | 输入单个词语进行搜索，精准输出所有包含该词语的日志 | startup |
| 全文检索-短语检索 | 输入双引号“”囊括短语进行搜索，精准输出所有包含该短语的日志 | startup |
| 布尔表达式搜索 | 支持 A AND B 搜索，输出所有同时包含 A 和 B 的日志；支持 A OR B 搜索，输出所有含有 A 或所有含有 B 或同时含有 A 和 B 的日志；支持 A NOT B 搜索，输出所有含有 A 同时不含有 B 的日志；支持 +A，输出所有包含 Term 的日志；支持 -A，输出的日志不包含 A | - |
| 通配符搜索 | 支持？搜索，？可匹配单个字符； 支持 * 搜索，* 可匹配0到多个字符 | - |
| 模糊搜索 | 支持 A~Beta（数值）启动模糊搜索，输出包含所有距离 A 相似度为 Beta 的单词的日志。该模糊检索查询出拼写错误的单词 | - |
| 优先级搜索 | 支持 A^beta（数值）B 搜索，使 A 的搜索优先级相较于 B 提高 Beta | - |
| 转义特殊字符 | 以下字符当作值搜索时需要用\转义： | (1+1)\=2用来查询(1+1)=2 |

| | | |
|---|---|---|
| | `+- = && \| > < ! ( ) { } [] ^ " ~ * ? : \ /` | |

### 正则表达式查询

支持使用正则表达式对日志进行搜索。
例如：搜索形如以 p 开头 r 结尾的关键词，我们可以直接在搜索框内输入 `/p.*r/` 进行匹配。

# 事件中心

最近更新时间：2025-04-08 15:25:53

事件中心将 TSF 所生成的事件数据进行统一管理、存储、分析和展示，方便您查看和分析，您可以在事件中心查看事件数据详情，同时您也可以为事件配置告警通知规则，及时发现问题并进行处理。

TSF 当前支持的事件类型有：

- 资源变更
  - 部署组状态异常
  - 应用仓库容量告警
- 发布变更
  - 部署组部署
  - 部署组启动
  - 部署组停止
  - 部署组删除
  - 部署组扩容
  - 部署组缩容
- 服务事件
  - 服务熔断
  - 服务离线
  - 服务限流
- 弹性伸缩
  - 部署组自动扩容
  - 部署组自动缩容

本文介绍如何在 TSF 控制台上查看事件详情和配置事件告警规则。

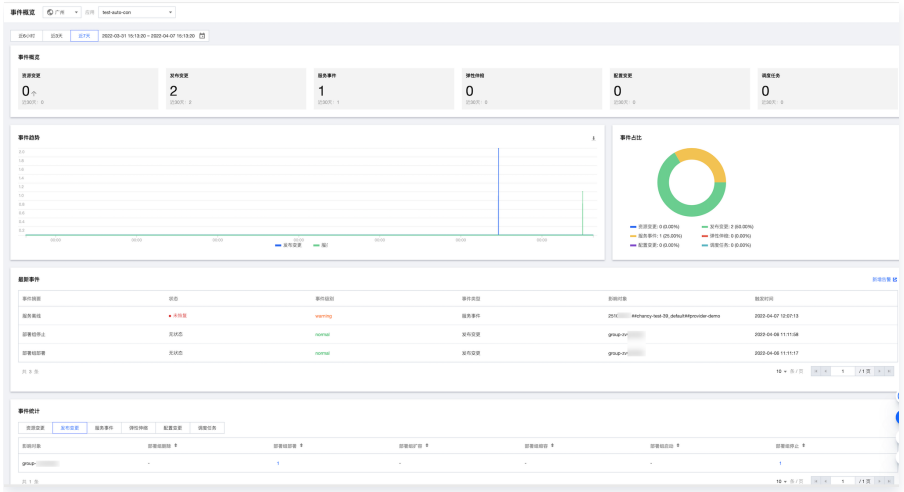
## 进入事件中心

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择**观测中心** > **事件中心** > **事件概览**，进入事件概览页面。
3. 在顶部菜单栏选择地域并在页面上方选择要查看的应用，设置好时间范围（支持近6小时、近3天、近7天和自定义时间范围）。

## 事件概览模块介绍

- **事件概览**：展示 TSF 支持的6类事件类型在设定时间范围内（可在页面上方修改时间范围）发生变更的次数和近30天内发生变更的次数。
- **事件趋势**：展示不同类型来源事件在选定时段内的发生次数变化的走势，您可以在右上角下拉框中选择需要展示的事件类型。
- **事件占比**：展示在选定时段内不同类型来源事件的数量占比。
- **最新事件**：展示最新发生的事件列表，包含事件摘要、状态、所属事件类型、影响对象、开始时间和更新时间。您可以单击列表右上角**新增告警**跳转至事件总线页面为事件配置告警通知规则，详情参见 配置事件告警规则 。
- **事件统计**：展示所选应用下各部署组中在所选时间段内不同事件类型不同事件摘要的统计信息。

版权所有：腾讯云计算（北京）有限责任公司



## 查看事件详情

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择**观测中心 > 事件中心 > 事件详情**，进入事件详情页面。
3. 在顶部菜单栏选择好地域，设置好时间范围（支持近6小时、近3天、近7天和自定义时间范围）。

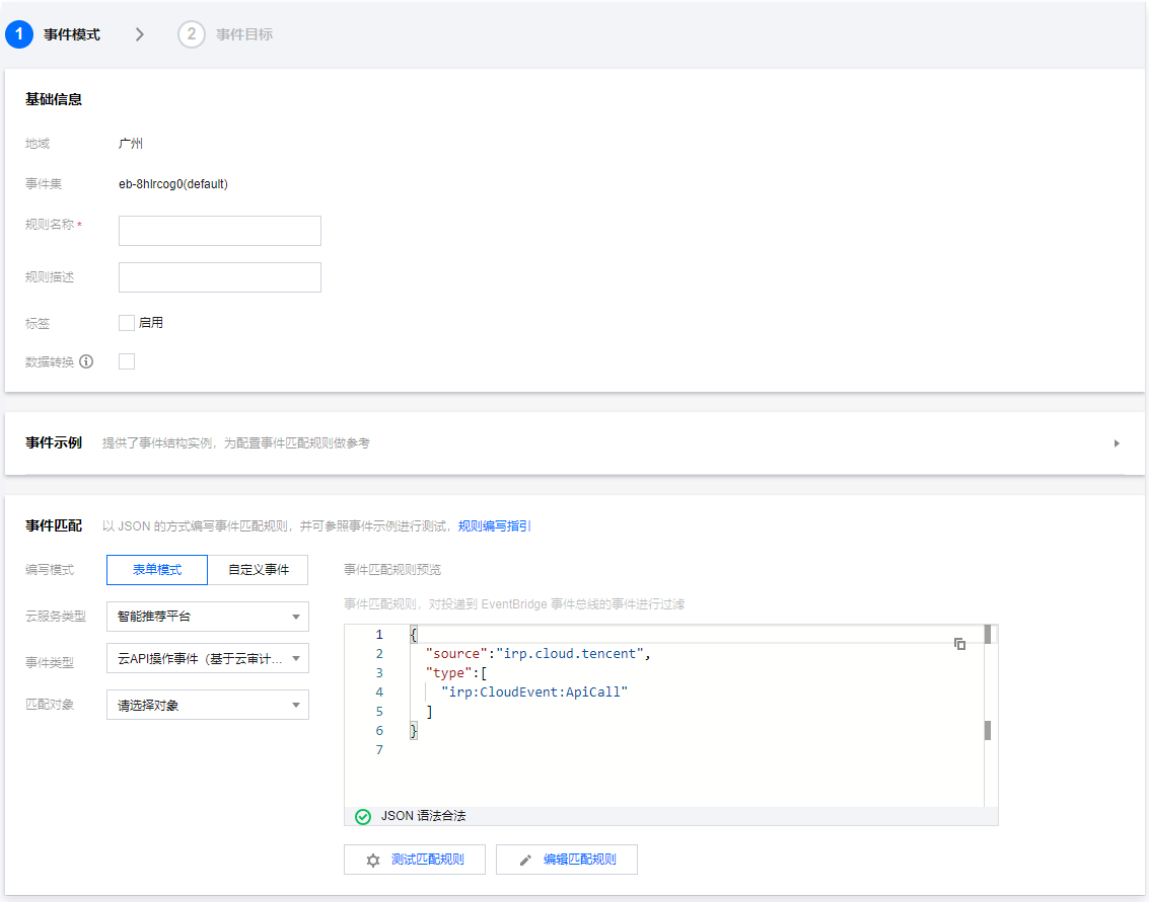



| 字段 | 描述 |
| --- | --- |
| 事件摘要 | 事件信息概要描述，例如自动扩容触发。 |
| 状态 | 在下拉列表中筛选状态，包含已恢复、未恢复和无状态三种。 |
| 事件级别 | 在下拉列表中筛选事件级别，包含 warning、normal 和 error 三种。 |
| 事件类型 | 包含资源变更，发布变更，服务事件，弹性伸缩四种。 |
| 影响对象 | 应用名/ID、集群名ID、部署组名/ID、服务名/ID、配置名/ID、命名空间名称/ID。 |
| 对象详情 | 事件影响对象的维度信息，例如应用名/ID、集群名ID、部署组名/ID等基本维度信息，可用于聚合数据，由上报方确定。 |
| 事件详情 | 事件明细信息，例如事件摘要、对象及事件内容等，由上报方确定。 |
| 触发时间 | 事件触发时间，精确到微秒。 |

## 配置事件告警规则

1. 在**事件详情**页面，单击左上角的**前往事件总线配置告警**跳转至事件规则配置页面。

2. 单击**新建事件规则**，配置事件模式信息。




- 规则名称：填写事件规则名称，规则名称一旦创建无法修改。
- 事件示例：选择**默认模板**。
- 云服务类型：选择**微服务平台 TSF**。
- 事件类型：勾选您要配置告警规则的事件类型。

3. 单击**下一步**，填写事件目标信息。




- 触发方式：选择**消息推送**。
- 消息模板：根据事件类型，提供两种不同类型推送模板，您可以根据实际情况选择，对于告警事件，建议统一选择告警通知模板。
- 用户通知：根据您的需要自定义选择通知方式和告警接收对象。

4. 单击**完成**，完成事件告警规则配置。

# 新用户
# 资源概览

最近更新时间：2025-04-08 15:25:53

## 功能说明

微服务平台支持全局视角的用户资源统计，包括资源概览、配额概况、集群使用概览、异常告警显示页面。
您只需登录 腾讯微服务平台控制台，在左侧菜单栏选择**观测中心** > **资源概览**，即可查看用户资源统计概览。

## 资源概览

支持查看租户下全局/子账号有读权限的资源概览情况，不受权限范围影响；当发生资源变更后，30分钟内刷新概览页数据。



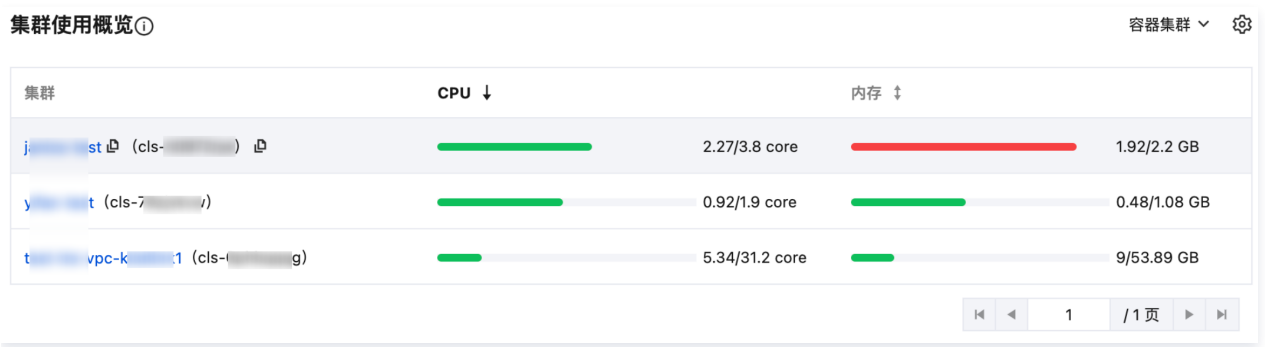

## 配额情况

支持查看用户节点配额，应用仓库用量及日志存储用量，并支持进行节点规格调整和续费操作。




## 集群使用概览

支持查看 CPU / 内存使用率，并对分配率超过阈值的集群突出显示。




## 异常告警显示

支持查看近24小时内未恢复告警数/总告警数，也可单击**配置告警>**，前往“腾讯云可观测平台”控制台进行告警策略的配置。



# 业务应用监控

最近更新时间：2025-04-08 15:25:53

TSF控制台升级后，业务应用监控监控升级为 腾讯云应用性能监控（APM），为您提供应用性能监控一站式解决方案。

> **说明：**
> APM 默认为每位 TSF 用户提供每天 2000 万 Span 的上报量与最近 3 天的数据存储。

本文为您介绍如果使用 APM 实现 TSF 业务应用的性能监控。

## 操作场景

业务应用监控功能用于监控应用、部署组、实例级别的监控。

## 前提

TSF 应用已绑定 APM 业务系统。在TSF控制台新建应用时会自动关联免费APM业务系统。关于业务系统的详细介绍，请参考「观测中心-业务系统」相关内容介绍。

## 操作步骤

1. 登录 腾讯微服务平台控制台 。
2. 在左侧菜单栏中，选择**观测中心**，选择好地域、时间范围、应用/部署组/实例信息后，即可查看当前筛选条件下应用/部署组/实例的监控图表信息。
3. 支持的监控能力如下表。

| 性能概览 |
| --- |
| 性能概览模块为您展示了所服务端或客户端、某时间段内应用平均响应时间、平均吞吐量、平均错误率以及 Apdex 的变化趋势。通过单击每个图表右上角的时钟图标，您可以自定义30天内任意日期的数据与当前时间段进行对比分析。 |

| 应用拓扑 |
| --- |
| 应用拓扑当前以筛选服务为中心，展示上下游局部依赖拓扑。将鼠标悬浮在代表应用的节点上，您可以看到对应应用平均吞吐量，响应时间，错误率。<br>应用性能监控使用不同的拓扑图标颜色进行标识，绿色图表示应用健康、橙色图标表示应用有延时情况、红色表示该业务出现异常情况。 |

| 接口监控 |
| --- |
| 该模块展示应用接口性能指标。您还可以单击接口名查看该接口的更多详情。 |

| 数据库调用分析 |
| --- |
| 该模块展示应用访问数据库的情况，由类型、数据库名称、数据库地址组成的三元组，可以确定唯一的数据库。您还可以单击查看概览和查看 SQL 查看更多数据库调用情况。 |

| SQL分析 |
| --- |

该模块展示应用访问数据库时每一种调用语句的执行情况。您还可以单击查看调用查看相关调用链的执行情况。

### 实例监控

该模块展示每一个应用实例的性能指标。

### 链路监控

该模块展示该应用相关调用链路，包括 Trace 和 Span 信息。您还可以单击 Trace ID 查看链路详情。

### 错误监控

该模块展示该应用相关异常信息，包括对每种异常类型的统计。

### 性能剖析

详细参见 腾讯云可观测平台性能剖析 。

### GC日志分析

详细参见 腾讯云可观测平台GC日志分析 。

### 线程池分析

详见 腾讯云可观测平台线程池分析 。

### 连接池分析

详见 腾讯云可观测平台连接池分析 。

版权所有：腾讯云计算（北京）有限责任公司


性能概览
应用拓扑
接口监控
数据库调用分析
SQL分析
实例监控
链路追踪
错误监控
性能剖析 高级功能
GC日志分析 高级功能
线程池分析 高级功能
连接池分析 高级功能
平均请求量(个/秒)
100(个/秒)
80(个/秒)
60(个/秒)
40(个/秒)
20(个/秒)
0(个/秒)
暂无数据
01:00 01:30 02:00 02:30 03:00 03:30 04:00 04:30 05:00 05:30 06:00 06:30
平均响应时间(毫秒)
100毫秒
80毫秒
60毫秒
40毫秒
20毫秒
0毫秒
暂无数据
01:00 01:30 02:00 02:30 03:00 03:30 04:00 04:30 05:00 05:30 06:00 06:30
P99响应时间(毫秒)
100毫秒
80毫秒
60毫秒
平均错误率(%)
100%
80%
60%

# 调用查询

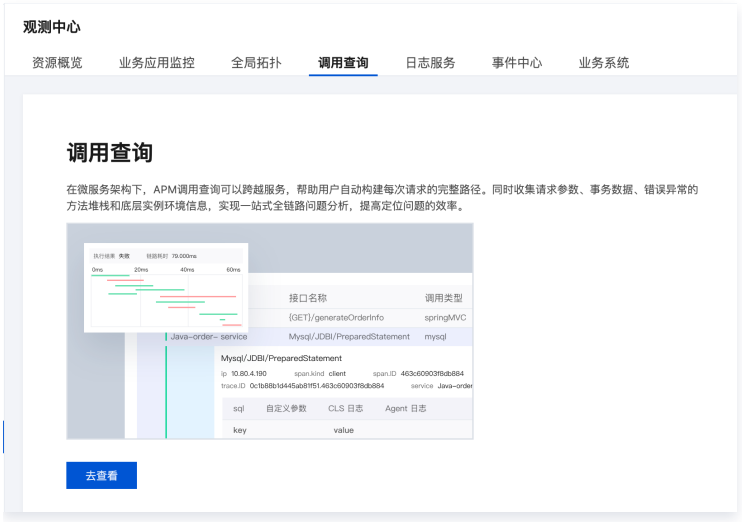
最近更新时间：2025-04-08 15:25:53

## 功能说明

链路追踪是应用性能监控（APM）的重要能力，可以实现跨应用的多维度调用链检索与分析。在链路追踪主页面，您可以根据多种过滤条件组合进行调用查询，查询结果中的每一条记录代表对一次调用的记录，等同于一个 Span。您可以单击对应的 TraceID 进入链路详情视图，进一步分析链路中的每个环节。

## 操作步骤

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择 **观测中心** > **调用查询** ，单击**去查看**进入腾讯云可观测平台控制台。
3. 腾讯云可观测平台控制台上关于链路追踪的操作与说明具体可参见 链路追踪 。



# 日志服务

最近更新时间：2025-04-08 15:25:53

TSF 使用腾讯云产品 日志服务（Cloud Log Service，CLS）为用户提供一个一站式日志服务平台。CLS提供了从日志采集、日志存储到日志检索，图表分析、监控告警、日志投递等多项服务，协助用户通过日志来解决业务运维、服务监控、日志审计等场景问题。

## 操作场景

日志配置项用于指定采集日志的规则，包括日志的采集路径和日志解析格式。用户可以在腾讯微服务平台控制台上创建日志配置项，然后将配置项发布到部署组上。同一个部署组可以关联多个日志配置项。

## 前提条件

已开通 CLS 日志服务。

## 操作步骤

### 新建日志配置项目

- **采集配置：**
  - 配置名：输入配置名称，不能为空，最长为60个字符。
  - 日志消费端：
    - 日志集：选择合适的日志集。
    - 日志主题：选择日志集下合适日志主题，也支持选择自动创建一个新的日志主题。

> **说明：**
> 1. 日志集：日志集（Logset）是对日志主题及指标主题的分类，一个日志集可包含多个日志主题和指标主题。一个日志集通常对应公司内的某一个项目/业务，建议将某个项目/业务下的多个应用/服务的日志主题归属到同一个日志集下。
> 2. 日志主题：日志主题（Log Topic）是日志数据在日志服务（Cloud Log Service，CLS）平台进行采集、存储、检索和分析的基本单元，采集到的海量日志以日志主题为单元进行管理，包括采集规则配置、保存时间配置、日志检索分析以及日志下载/消费/投递等。

  - 采集路径：设置您的日志采集路径，可配置一个或者多个日志采集路径。日志目录前缀以/开头，文件名以非/开头。/**/代表LogListener将监听所填前缀目录下的所有层级匹配上的日志文件。多文件路径之间为或关系。目录前缀和文件名支持?和*通配符，不支持逗号。

> **注意：**
> Linux 日志路径必须以/开头。 Windows文件路径必须以盘符开头，例如C:\。

    - Linux 系统中的日志路径：/[目录前缀表达式]/**/[文件名表达式]，例如：/data/log/**/*.log。
    - Windows 系统中的日志路径：[盘符]:\[目录前缀表达式]**\[文件名表达式]，例如：C:\Program Files\Tencent\...*.log。
    - 填写日志采集路径后，LogListener 会按照[目录前缀表达式]匹配所有符合规则的公共前缀路径，并监听这些目录（包含子层目录）下所有符合[文件名表达式]规则的日志文件。其参数详细说明如下：

| 字段 | 说明 |
| --- | --- |
| 目录前缀 | 日志文件前缀目录结构，仅支持通配符 * 和 ？<br>* 表示匹配多个任意字符<br>？表示匹配单个任意字符<br>不支持填写逗号 |
| ** | 表示当前目录以及所有子目录 |

| | |
|---|---|
| 文件名 | 日志文件名，仅支持通配符 * 和 ？<br>* 表示匹配多个任意字符<br>？表示匹配单个任意字符<br>不支持填写逗号 |

- 备注：填写日志配置项的备注下信息，选填，200字符内。
- 数据集：选择“无”。用户可以通过数据集管理配置不同的子账号和协作者使用不同资源的权限，管理方法详情请参见 数据集管理 。




- **解析规则配置：**
  - 采集策略：选择您的采集策略；支持全量/增量采集。
    - 回溯采集：对于增量采集支持策略，支持配置回溯采集的位置，即从从最新的位置，往前采集一定数量字节（Byte）的日志。
  - 编码模式：选择您的 UTF-8，
  - 提取模式：选择您的日志提取模式，支持结构化提取和非结构化提取，详细参见 CLS 关于日志提取规则 。
  - 过滤器：选择是否开启过滤器；开启后可以仅采集符合过滤规则的日志。

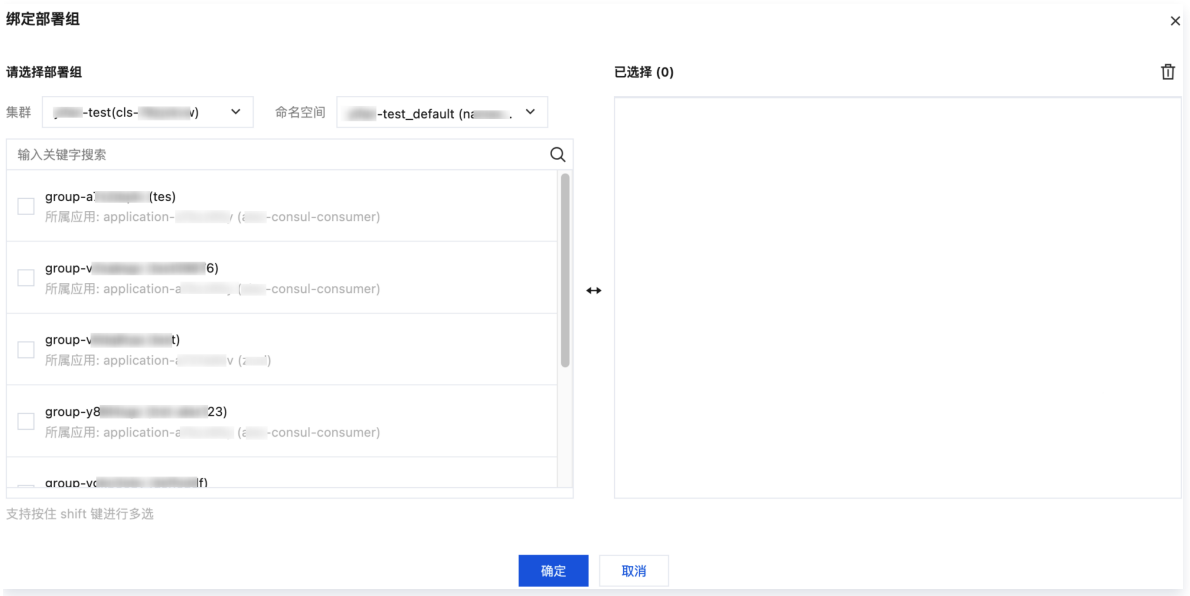
> **说明：**
> 例如，您希望原始日志内容中 response_code 为400或500的所有日志数据被采集，那么 key 处配置 response_code，过滤规则选择等于，Value 处配置正则表达式 400|500。

  - 例如，您希望原始日志内容中 response_code 为400或500的所有日志数据被采集，那么 key 处配置 response_code，过滤规则选择等于，Value 处配置正则表达式 400|500。
  - 上传解析失败的日志：开启后，LogListener 会上传解析失败的日志；关闭会丢弃失败的日志。
  - 解析失败的日志的键名称：输入您希望的名称；默认值为 LogParseFailure，所有解析失败的日志，均以 LogParseFailure 作为键名称（Key），原始日志内容作为值（Value）进行上传。
  - 高级配置：
    - 超时属性：该配置控制日志文件的超时事件。支持“自定义”、“不超时”。如果一个日志文件在指定时间内没有任何更新，则为超时。超时的日志文件 LogListener 将不再监控。当您的日志文件数量较大时，建议降低超时时间，避免 LogListener 性能浪费。
    - 最大目录深度：该配置控制日志采集的最大目录深度。LogListener 不会采集所在目录层级超过指定最大目录深度的日志文件。当您目标采集路径包含模糊匹配时，建议配置合适的最大目录深度，避免 LogListener 性能浪费。




## 绑定部署组

保存上述 新建日志配置项 的相关配置后，可以将配置绑定至指定部署组，一个配置支持绑定到多个部署组。
依次选择集群、命名空间后，勾选目标部署组即可。选择完成后点击确定即可完成日志配置的发布。发布完成后可以前往日志配置项列表页，点击日志配置项名称，进入详情页，选择发布情况，查看日志发布情况。




## 日志检索

完成日志配置项的创建后发布后，回到志配置项列表页，在操作栏点击“日志检索”后跳转至 CLS 控制台检索分析页面，进行日志分析相关操作，关于检索分析相关介绍，请参见 检索分析 。

## 仪表盘

日志服务提供丰富的图表类型，支持用户使用 SQL 灵活统计日志内的系统及业务指标，并通过图表进行展示。用户还可以将图表分析结果保存到仪表盘，进行长期监控。关于仪表盘相关介绍，请参见 仪表盘 。

## 数据处理

CLS 支持对日志 数据加工 ，提供对日志数据的过滤、清洗、脱敏、富化、分发等能力。以及 定时 SQL 分析，可根据您定义的时间窗口、调度周期， 周期性地对日志数据进行查询，并将查询的结果保存到您指定的日志主题中。

## 投递与消费

CLS 可以将日志主题中的数据投递到腾讯云 COS（对象存储）、CKafka（消息队列），以满足归档或者计算的需求。关于日志投递与消费的相关介绍，请参见 投递与消费 。

## 日志告警

日志服务支持对日志及指标主题设置告警策略，告警策略会周期性执行监控任务，对监控对象中的主题执行检索分析，当查询结果满足触发条件时发送告警通知，方便用户及时发现异常问题。关于日志告警相关介绍，请参见 监控告警 。

# 事件中心

最近更新时间：2025-04-08 15:25:53

事件中心将 TSF 所生成的事件数据进行统一管理、存储、分析和展示，方便您查看和分析，您可以在事件中心查看事件数据详情，同时您也可以为事件配置告警通知规则，及时发现问题并进行处理。

TSF 当前支持的事件类型有：

- 资源变更
  - 部署组状态异常
  - 应用仓库容量告警
- 发布变更
  - 部署组部署
  - 部署组启动
  - 部署组停止
  - 部署组删除
  - 部署组扩容
  - 部署组缩容
- 服务事件
  - 服务熔断
  - 服务离线
  - 服务限流
- 弹性伸缩
  - 部署组自动扩容
  - 部署组自动缩容

本文介绍如何在 TSF 控制台上查看事件详情和配置事件告警规则。

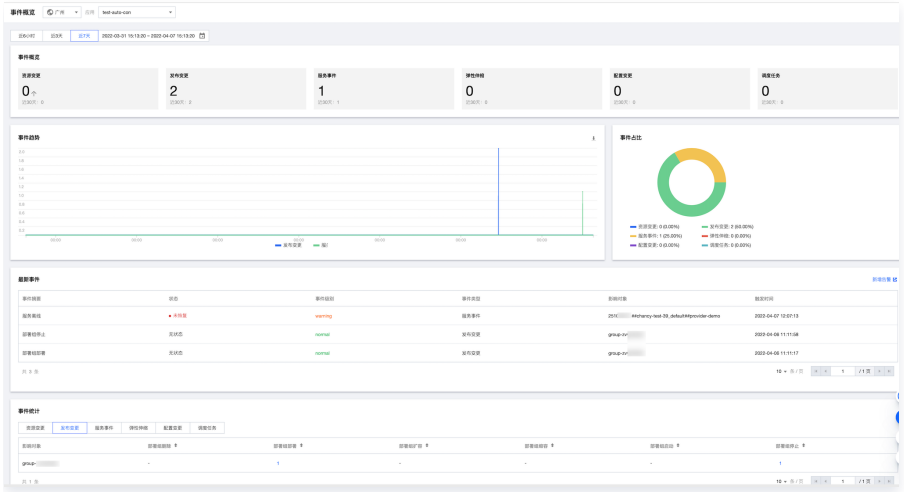
## 进入事件中心

1. 登录 腾讯微服务平台控制台 。
2. 在左侧导航栏选择 **运维中心** > **事件中心** > **事件概览**，进入事件概览页面。
3. 在顶部菜单栏选择地域并在页面上方选择要查看的应用，设置好时间范围（支持近6小时、近3天、近7天和自定义时间范围）。

## 事件概览模块介绍

- **事件概览**：展示 TSF 支持的6类事件类型在设定时间范围内（可在页面上方修改时间范围）发生变更的次数和近30天内发生变更的次数。
- **事件趋势**：展示不同类型来源事件在选定时段内的发生次数变化的走势，您可以在右上角下拉框中选择需要展示的事件类型。
- **事件占比**：展示在选定时段内不同类型来源事件的数量占比。
- **最新事件**：展示最新发生的事件列表，包含事件摘要、状态、所属事件类型、影响对象、开始时间和更新时间。您可以单击列表右上角**新增告警**跳转至事件总线页面为事件配置告警通知规则，详情参见 配置事件告警规则 。
- **事件统计**：展示所选应用下各部署组中在所选时间段内不同事件类型不同事件摘要的统计信息。

版权所有：腾讯云计算（北京）有限责任公司



## 查看事件详情

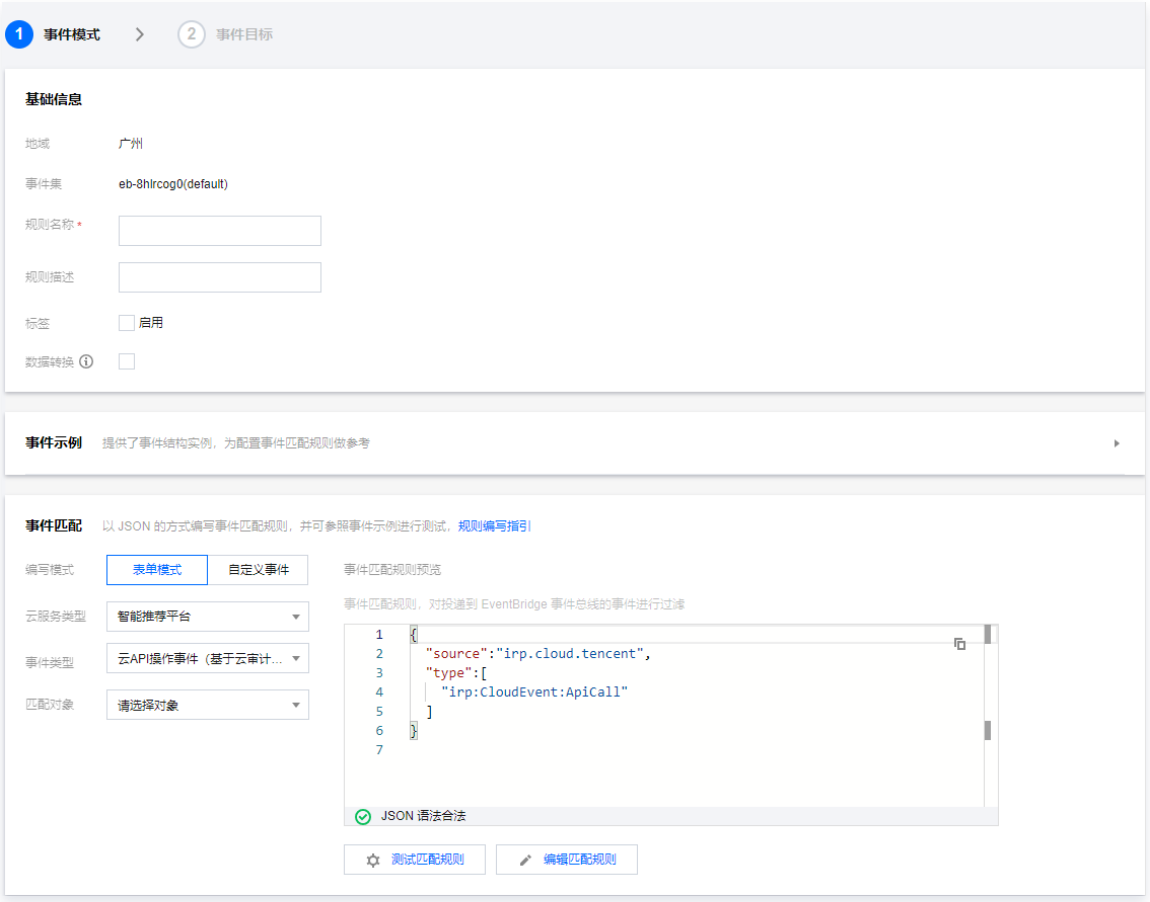
1. 登录 TSF 控制台。
2. 在左侧导航栏选择 **运维中心 > 事件中心 > 事件详情**，进入事件详情页面。
3. 在顶部菜单栏选择好地域，设置好时间范围（支持近6小时、近3天、近7天和自定义时间范围）。
4. 

| 字段 | 描述 |
| --- | --- |
| 事件摘要 | 事件信息概要描述，如自动扩容触发 |
| 状态 | 在下拉列表中筛选状态，包含已恢复、未恢复和无状态三种。 |
| 事件级别 | 在下拉列表中筛选事件级别，包含 warning、normal 和 error 三种。 |
| 事件类型 | 包含资源变更，发布变更，服务事件，弹性伸缩四种。 |
| 影响对象 | 应用名/ID、集群名ID、部署组名/ID、服务名/ID、配置名/ID、命名空间名称/ID |
| 对象详情 | 事件影响对象的维度信息，例如应用名/ID、集群名ID、部署组名/ID等基本维度信息，可用于聚合数据，由上报方确定。 |
| 事件详情 | 事件明细信息，例如事件摘要、对象及事件内容等，由上报方确定。 |
| 触发时间 | 事件触发时间，精确到微秒 |

## 配置事件告警规则

1. 在**事件详情**页面，单击左上角的**前往事件总线配置告警**跳转至事件规则配置页面。
2. 单击**新建事件规则**，配置事件模式信息。

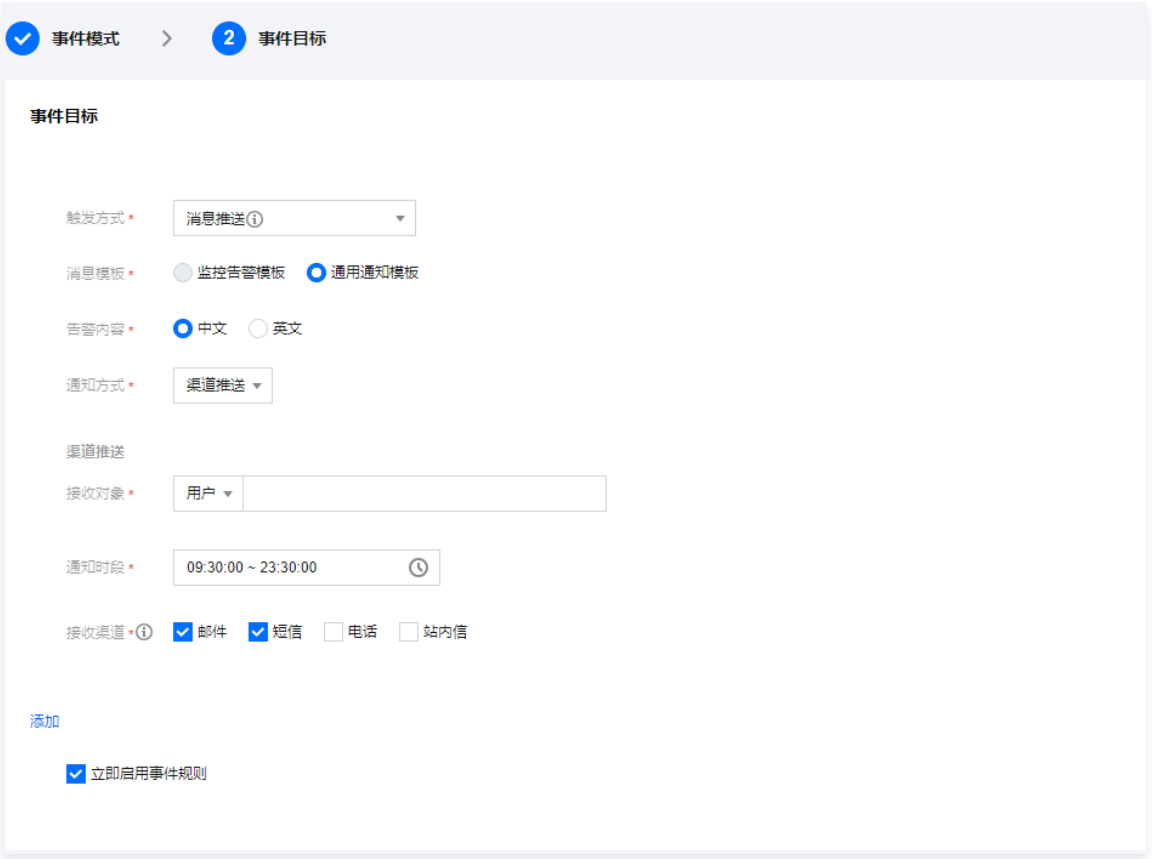
3.




- 规则名称：填写事件规则名称，规则名称一旦创建无法修改。
- 事件示例：选择**默认模板**。
- 云服务类型：选择**微服务平台 TSF**。
- 事件类型：勾选您要配置告警规则的事件类型。

4. 单击**下一步**，填写事件目标信息。

5.




- 触发方式：选择**消息推送**。
- 消息模板：根据事件类型，提供两种不同类型推送模板，您可以根据实际情况选择，对于告警事件，建议统一选择告警通知模板。
- 用户通知：根据您的需要自定义选择通知方式和告警接收对象。

6. 单击**完成**，完成事件告警规则配置。

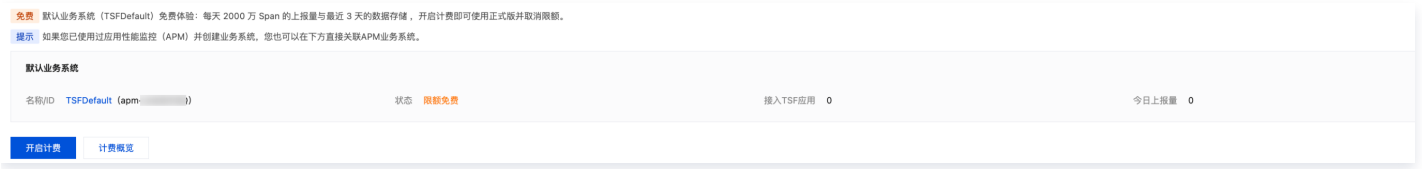
# 业务系统

最近更新时间：2025-04-08 15:25:53

## 功能说明

业务系统是腾讯云应用性能监控（APM）中的概念，业务系统用于分类管理您的应用，您可以根据不同业务系统设置不同的存储时长、上报限额等。
对于每一位 TSF 用户，APM 免费一个默认业务系统（TSFDefault），免费提供每天 2000 万 Span 的上报量与最近 3 天的数据存储 。
如果您对于监控有更高的存储要求，可以将此系统升级为付费业务系统，在页面进行开启计费操作即可，开启计费后，默认业务系统（TSFDefault）会升级为APM正式版，享受无限额的上报量与存储时长。详细可参见 APM 计费概览 。

> **说明：**
> 在 腾讯微服务平台控制台 新建应用时会自动关联免费版 APM 业务系统。




如果您已使用过应用性能监控（APM）并创建业务系统，创建应用时也可以直接关联目标 APM 业务系统。业务系统标签页也会展示您关联的其他业务系统。



# TCT 任务迁移 XXL-JOB 通用指引

最近更新时间：2025-06-17 17:34:42

## 背景

分布式任务调度（TCT）在公有云下架，协助客户完成下架迁移至替代开源产品 XXL-JOB 。

## TCT 和 XXL-JOB 功能清单

| TCT模块 | 功能点 | XXL-JOB | 说明 |
| --- | --- | --- | --- |
| 任务管理 | 新建/编辑/删除/启用/停用 | ✓ | - |
| 任务触发 | 手动触发任务执行一次 | ✓ | - |
| | 周期触发 | ✓ | - |
| | 定时触发 | ✓ | - |
| | 工作流触发 | × | - |
| 任务调度/执行 | Java类型任务 | ✓ | - |
| | 随机单节点执行 | ✓ | - |
| | 广播执行 | ✓ | - |
| | 分片执行 | ✓ | XXL-JOB 不支持配置分片数 |
| | 任务超时设定 | ✓ | - |
| | 失败重试 | ✓ | - |
| | 任务参数传递 | ✓ | - |
| | 分片任务参数传递 | ✓ | - |
| | 分片任务单机并发控制 | × | - |
| | 手动重新执行失败任务 | × | - |
| | 停止任务 | ✓ | - |
| | 分片子任务失败断点续跑 | × | - |
| 工作流管理 | 新建/编辑/删除/启用/停用 | × | XXL-JOB 可通过父子任务实现简单任务编排 |
| 执行记录 | 搜索/查看任务执行记录 | ✓ | - |
| | 搜索/查看工作流执行记录 | × | - |
| | 执行记录过期自动清理 | × | XXL-JOB 支持手动批量清理 |
| 执行日志 | 查看任务执行日志 | ✓ | - |
| 监控告警 | 任务失败告警 | ✓ | - |
| | 工作流失败告警 | × | - |
| 认证鉴权 | 执行器注册 Token 认证 | ✓ | - |

| | | | |
|---|---|---|---|
| | 用户登录认证 | ✓ | - |
| | 用户资源鉴权 | ✓ | XXL-JOB 仅支持执行器资源维度鉴权 |

> **注意：**
> XXL-JOB 支持父子任务，对于不涉及原 TCT 工作流逻辑节点的简单工作流可以在 XXL-JOB 中通过父子任务实现简单任务编排

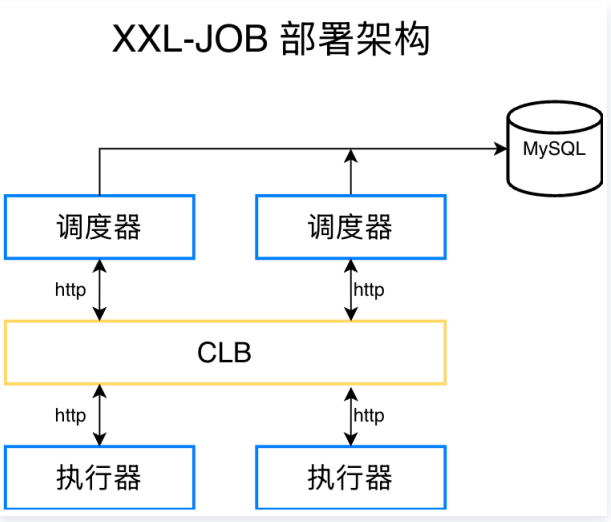
## 迁移步骤

### 部署 XXL-JOB 调度中心

用户可根据自己的网络和机器资源自行规划配置，参考 XXL-JOB 官网 分布式任务调度平台 XXL-JOB 部署即可。以下规划选项供参考：
资源清单（推荐）：

| 模块 | 副本数 | 节点规格 | 部署模式 |
|---|---|---|---|
| 调度中心 | 2 | 4c8g | 集群部署 |
| 数据库MySQL | 2 | 4c8g | 主从部署 |
| CLB | - | - | - |

部署架构（推荐）：




CLB 可选，执行器可直连多个调度器节点。部署完成后，访问 XXL-JOB 控制台页面验证部署成功。

### TCT 任务改造

TCT 以 Java 类为单位开发任务，在新建/编辑任务时输入完整类名识别任务。
XXL-JOB 同时支持类任务和方法任务，因此在我们既可以 TCT 任务类改造为 XXL-JOB 任务，也可以将其改造为任务方法。

我们保持 TCT 任务使用习惯以一个例子来说明 TCT 任务类改造为 XXL-JOB 任务类流程（用户也可参考 XXL-JOB 官方文档将其改造为方法），一个典型的 TCT 任务实现如下：

```
/**
 * 模拟简单的任务， 执行时间在10s ~ 15s之间， 可以终止。
 */
@Component
public class SimpleLogExecutableTask implements ExecutableTask, TerminableTask {
```

```
    @Override
    public ProcessResult execute(ExecutableTaskData taskData) {
        ProcessResult result = new ProcessResult();
        try {
            LogReporter.log(taskData, "start to execute SimpleLogExecutableTask...");
            LogReporter.log(taskData, "hello, this is a demo for SimpleLogExecutableTask");
            long sleepTime = RandomUtils.nextLong(10000, 15000);
            Thread.sleep(sleepTime);
            result.setResultCode(ProcessResultCode.SUCCESS);
        } catch (InterruptedException e) {
            result.setResultCode(ProcessResultCode.TERMINATED);
            LogReporter.log(taskData, "task is terminated... ");
        } catch (Throwable t) {
            result.setResultCode(ProcessResultCode.FAIL);
        }
        return result;
    }

    @Override
    public TerminateResult cancel(TaskExecuteFuture future, ExecutableTaskData taskData) {
        LogReporter.log(taskData, "task start to cancel");
        future.cancel(true);
        LogReporter.log(taskData, "task cancel success");
        return TerminateResult.newTerminateSuccessResult();
    }
}
```

**以 Maven 项目为例，在 POM 中引入 xxl-job-core 依赖：**

```
        <dependency>
            <groupId>com.xuxueli</groupId>
            <artifactId>xxl-job-core</artifactId>
            <version>3.1.0</version>
        </dependency>
```

**修改 TCT 任务实现为 XXL-JOB 的实现， 将其中实现 ExecutableTask 的方式修改为继承自"com.xxl.job.core.handler.IJobHandler"的JobHandler类，实现 execute 方法。**

```
public class SimpleLogExecutableTask extends IJobHandler {
    public void execute() {
        try {
            String param = XxlJobHelper.getJobParam();
            XxlJobHelper.log("xxl job param: {}", param);
            XxlJobHelper.log("start to execute SimpleLogExecutableTask...");
            XxlJobHelper.log("hello, this is a demo for SimpleLogExecutableTask");
            long sleepTime = RandomUtils.nextLong(10000, 15000);
            Thread.sleep(sleepTime);
        } catch (InterruptedException e) {
            XxlJobHelper.log("task is terminated... ");
        } catch (Throwable t) {
            XxlJobHelper.log("task is failed... ");
        }
    }
```

版权所有：腾讯云计算（北京）有限责任公司

```
}
```

手动通过如下方式注入到执行器容器：

```
    @PostConstruct
    public void registerJobHandlers() {
        XxlJobExecutor.registJobHandler("SimpleLogExecutableTask", new
SimpleLogExecutableTask());
    }
```

完整代码请单击跳转参考示例。

> ⚠ **注意：**
> 在实现任务终止时，TCT 任务提供了 cancel 接口接收任务终止指令，但是 XXL-JOB 中没有类似接口，XXL-JOB 任务终止指令是向任务执行线程发送 interrupt 信号，如果 XXL-JOB 任务内部未捕获该异常或是 catch 到了该异常并消化掉的话，任务终止功能将不可用。

## 测试

### 执行器部署

执行器配置如下：

```
### 调度中心部署根地址 [选填]：如调度中心集群部署存在多个地址则用逗号分隔。执行器将会使用该地址进行"执行器心跳注册"和"任务结果回调"；为空则关闭自动注册；
xxl.job.admin.addresses=http://127.0.0.1:8080/xxl-job-admin
### 调度中心通讯TOKEN [选填]：非空时启用；
xxl.job.admin.accessToken=default_token
### 调度中心通讯超时时间[选填]，单位秒；默认3s；
xxl.job.admin.timeout=3
### 执行器AppName [选填]：执行器心跳注册分组依据；为空则关闭自动注册
xxl.job.executor.appname=xxl-job-executor-sample
### 执行器注册 [选填]：优先使用该配置作为注册地址，为空时使用内嵌服务 ”IP:PORT“ 作为注册地址。从而更灵活的支持容器类型执行器动态IP和动态映射端口问题。
xxl.job.executor.address=
### 执行器IP [选填]：默认为空表示自动获取IP，多网卡时可手动设置指定IP，该IP不会绑定Host仅作为通讯使用；地址信息用于 "执行器注册" 和 "调度中心请求并触发任务"；
xxl.job.executor.ip=
### 执行器端口号 [选填]：小于等于0则自动获取；默认端口为9999，单机部署多个执行器时，注意要配置不同执行器端口；
xxl.job.executor.port=9999
### 执行器运行日志文件存储磁盘路径 [选填] ：需要对该路径拥有读写权限；为空则使用默认路径；
xxl.job.executor.logpath=/data/applogs/xxl-job/jobhandler
### 执行器日志文件保存天数 [选填] ： 过期日志自动清理，限制值大于等于3时生效；否则，如-1，关闭自动清理功能；
xxl.job.executor.logretentiondays=30
```

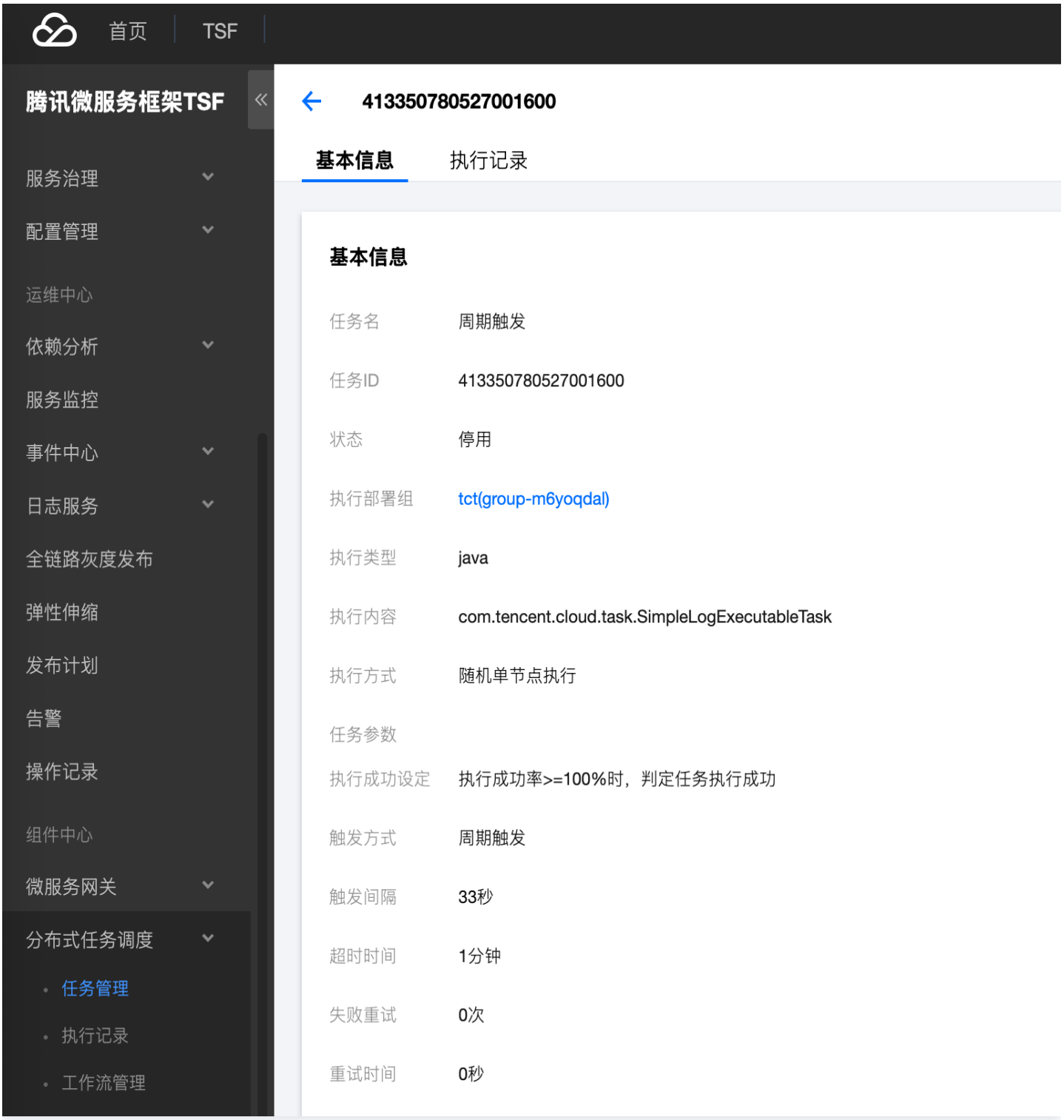
关键配置说明：

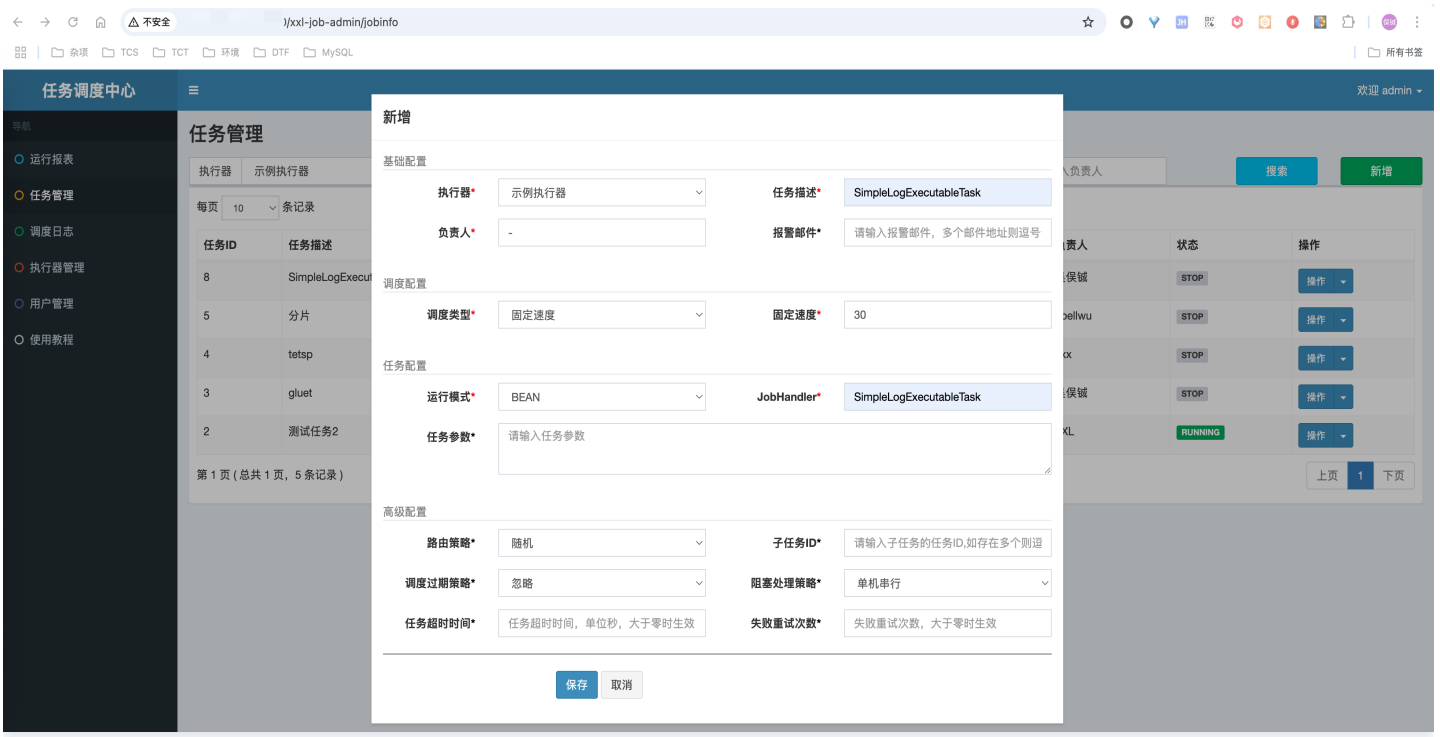
| 配置项 | 说明 |
| --- | --- |
| xxl.job.admin.addresses | 调度中心地址，如果使用了 LB 则填入 LB 地址。 |

| | |
|---|---|
| xxl.job.admin.accessToken | 需要和服务端保持一致。调度中心和执行器，如果需要正常通讯，只有两种设置：<br>• 设置一：调度中心和执行器，均不设置 AccessToken，关闭安全性校验；<br>• 设置二：调度中心和执行器，设置了相同的 AccessToken。 |
| xxl.job.executor.appname | 分组依据。相同 appname 的执行器视为同一个分组，限定了分片/广播任务调度时作用域。类似 TCT 部署组概念。 |
| xxl.job.executor.address | 执行器 HTTP 服务地址，用于接收调度中心请求。 |
| xxl.job.executor.port | 执行器 HTTP 服务端口，用于接收调度中心请求。 |

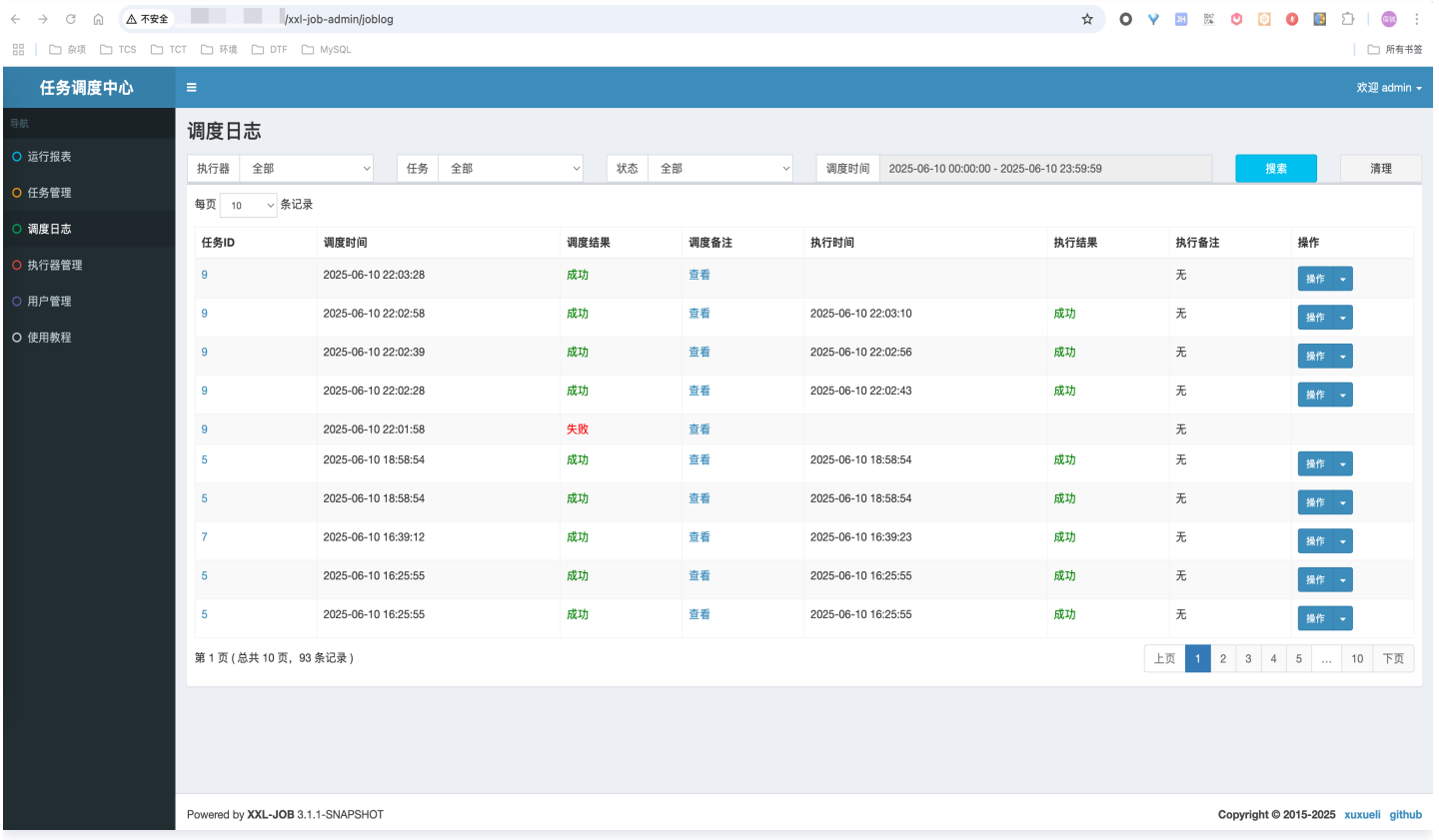
### 创建 XXL-JOB 任务

登录 XXL-JOB 控制台，默认账号 admin/123456，新建 XXL-JOB 任务与如下 TCT 任务有相同配置：


首页
TSF
腾讯微服务框架TSF
服务治理
配置管理
运维中心
依赖分析
服务监控
事件中心
日志服务
全链路灰度发布
弹性伸缩
发布计划
告警
操作记录
组件中心
微服务网关
分布式任务调度
任务管理
执行记录
工作流管理
413350780527001600
基本信息
执行记录
基本信息
任务名 周期触发
任务ID 413350780527001600
状态 停用
执行部署组 tct(group-m6yoqdal)
执行类型 java
执行内容 com.tencent.cloud.task.SimpleLogExecutableTask
执行方式 随机单节点执行
任务参数
执行成功设定 执行成功率>=100%时，判定任务执行成功
触发方式 周期触发
触发间隔 33秒
超时时间 1分钟
失败重试 0次
重试时间 0秒


任务调度中心
任务管理
新增
基础配置
执行器*
示例执行器
任务描述*
SimpleLogExecutableTask
负责人*
-
报警邮件*
请输入报警邮件，多个邮件地址则逗号
调度配置
调度类型*
固定速度
固定速度*
30
任务配置
运行模式*
BEAN
JobHandler*
SimpleLogExecutableTask
任务参数*
请输入任务参数
高级配置
路由策略*
随机
子任务ID*
请输入子任务的任务ID,如存在多个则逗
调度过期策略*
忽略
阻塞处理策略*
单机串行
任务超时时间*
任务超时时间，单位秒，大于零时生效
失败重试次数*
失败重试次数，大于零时生效
保存
取消

**验证任务可以成功执行：**


任务调度中心
调度日志
执行器 全部
任务 全部
状态 全部
调度时间 2025-06-10 00:00:00 - 2025-06-10 23:59:59
搜索
清理
每页 10 条记录
任务ID 调度时间 调度结果 调度备注 执行时间 执行结果 执行备注 操作
9 2025-06-10 22:03:28 成功 查看 无 操作
9 2025-06-10 22:02:58 成功 查看 2025-06-10 22:03:10 成功 无 操作
9 2025-06-10 22:02:39 成功 查看 2025-06-10 22:02:56 成功 无 操作
9 2025-06-10 22:02:28 成功 查看 2025-06-10 22:02:43 成功 无 操作
9 2025-06-10 22:01:58 失败 查看 无
5 2025-06-10 18:58:54 成功 查看 2025-06-10 18:58:54 成功 无 操作
5 2025-06-10 18:58:54 成功 查看 2025-06-10 18:58:54 成功 无 操作
7 2025-06-10 16:39:12 成功 查看 2025-06-10 16:39:23 成功 无 操作
5 2025-06-10 16:25:55 成功 查看 2025-06-10 16:25:55 成功 无 操作
5 2025-06-10 16:25:55 成功 查看 2025-06-10 16:25:55 成功 无 操作
第 1 页 ( 总共 10 页，93 条记录 )
Powered by XXL-JOB 3.1.1-SNAPSHOT
Copyright © 2015-2025 xuxueli github




## TCT 任务迁移

按照以上步骤改造完成后，可灰度迁移。以迁移 TSF 部署组 A 下所有任务和执行器为例，步骤如下：

1. 按照以上步骤改造 TSF 部署组A下 TCT 执行器为 XXL-JOB 类型执行器并部署注册；
2. 新建 XXL-JOB 任务（注意与原 TCT 任务同名同配置）并停用；
3. 手动测试验证 XXL-JOB 任务正常执行；
4. 启用 XXL-JOB任务，停用 TCT 任务；
5. 业务观察一段时间（比如一周），如无问题，可删除所有 TCT 任务，并下线 TSF 部署组A下所有执行器。

版权所有：腾讯云计算（北京）有限责任公司