

金蝶 Apusic 负载均衡器软件

用 户 手 册

Apusic 金蝶天燕

深圳市金蝶天燕云计算股份有限公司

版权声明

本用户手册内容版权属于金蝶天燕云计算股份有限公司。转载、摘编或利用其它方式使用本文文字或者观点的,应注明“来源:金蝶天燕云计算股份有限公司”。

目录

第 1 章 快速开始指南	5
1.1. 关于本快速开始指南	5
1.2. 基本介绍	5
1.3. 相关资源	5
1.4. 基本概念	5
第 2 章 安装部署	6
2.1. 安装部署前的准备	6
2.2. 安装部署 MICC	7
2.3. 停止 MICC	9
2.4. 常见问题解决	10
第 3 章 MICC 平台	11
3.1. 首页	11
3.1.1. 登录 MICC	11
3.1.2. 运行状态	11
3.1.3. 个人信息	12
3.2. 机器管理	13
3.2.1. 新增机器	13
3.2.2. 编辑机器、删除机器	14
3.3. 自动部署 ALB 负载均衡器	14
3.3.1. 自动部署 ALB	14
3.4. 负载均衡器管理	15
3.4.1. 启动、停止、重启、卸载负载均衡器	15
3.5. 用户管理	16
3.5.1. 新增用户	16
3.5.2. 编辑用户、删除用户	16
3.5.3. 授权用户	17
3.5.4. 激活用户、冻结用户	17
3.6. 操作审计	18
3.7. 密码与安全	18
3.13.1. 修改当前用户密码	18
3.13.2. 通过系统管理员修改密码	18
第 4 章 ALB 实例管控台	19
4.1. 路由管理	20
4.2.1. 创建路由	20
4.2.2. 查看路由详情	33
4.2.3. 下线、编辑、复制、删除路由	34

4.2. 上游管理	34
4.3.1. 上游列表:	34
4.3.2. 创建上游	35
4.3.3. 上游的下线、编辑、复制、删除	37
4.3. 服务管理	37
4.4.1. 服务列表	38
4.4.2. 创建服务	38
4.4.3. 服务的下线、编辑、复制、删除	39
4.4. Nginx 配置管理	40
4.5.1. 导入 nginx 配置	40
4.5.2. 发布、下线、编辑、删除、查看 nginx 配置	42
4.5. 插件管理	42
4.6.1. 插件列表	42
4.6.2. 如何开启插件	43
4.6.3. 编辑插件、删除插件	44
4.6.4. 插件示例	44
4.6. 证书管理	51
4.7.1. 证书列表	51
4.7.2. 创建证书	51
4.7. 系统信息	52
第5章 静态资源代理、PHP、TCP 代理	53
第6章 REST API 接口	53
7.1. REST API 请求的方法和参数	53
7.2. REST API 示例	56

第1章 快速开始指南

1.1. 关于本快速开始指南

本快速入门指南介绍了金蝶 Apusic 负载均衡器软件产品安装部署、负载均衡器监控、机器资源管理、负载均衡器管理、负载均衡器自动部署、用户管理等基本操作，为用户快速使用本产品提供指导意义。

1.2. 基本介绍

金蝶天燕负载均衡器软件（Apusic Load Balancer, ALB）具备高性能、高可用、灵活性和可扩展等特性，能够应对大规模集群场景下的访问流量管理和服务反向代理等场景，支持对服务的访问请求建立动态路由，进行验证、处理、转换和分发等操作，可作为应用集群的流量入口实现对应用服务的动态负载均衡。ALB 提供扩展插件功能，允许服务提供方在服务的请求和响应过程中，嵌入安全认证、访问监控和特定的业务逻辑，从而满足应用安全、运维管理以及业务上的个性化需求。

1.3. 相关资源

针对不同的操作系统，金蝶 Apusic 负载均衡器软件提供不同的安装包，也提供适合各个操作系统解压即用的 gz 格式的产品包。

更多信息，可以访问金蝶天燕官方网站 <http://www.apusic.com> 获取。

1.4. 基本概念

在正确使用 ALB 负载均衡器之前，需要先理解以下几个基本概念：

- ALB: Apusic Load Balance, 金蝶 Apusic 负载均衡器软件，实现对客户端请求的验证、处理、转换、分发
- MICC: ALB 多实例管控台 (multi instance control center)

- 路由：请求的入口点，通过定义一些规则来匹配客户端的请求，然后根据匹配结果加载并执行相应的插件，并把请求转发到指定后端服务
- 服务：由路由中公共的插件配置、上游目标信息组合而成的一组抽象，路由与服务之间，通常是 N:1 的关系
- 上游：指的是后端真实服务的访问入口，ALB 对给定的多个上游按照配置规则进行负载均衡，实现对上游目标服务的负载均衡和健康检查。

第2章 安装部署

ALB 是负载均衡器核心模块，ALB 的安装、部署与配置均由 ALB 管控台 MICC（多实例管控台，multi instance control center），其部署架构图如下：

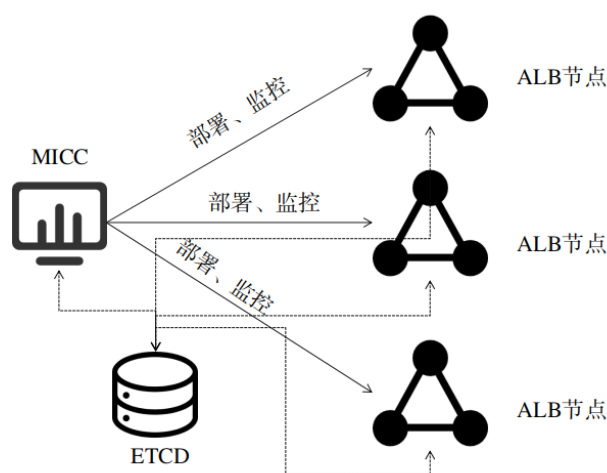


图 ALB 和 MICC 部署架构图

2.1. 安装部署前的准备

MICC 是 ALB 的 WEB 管理控制平台，部署 ALB 负载均衡器需要通过在选择 ALB 版本和上传 ALB license 进行部署，这部分内容由 MICC 的控制台的机器管理模块提供，本部分 MICC 的部署。

1. 获取 MICC 安装包

从 <http://www.apusic.com/> 下载金蝶 Apusic 负载均衡器软件安装包，或从金蝶 Apusic 负载均衡器软件产品光盘中获得相应的安装包文件。

安装包格式名： alb-micc-版本-芯片架构.tar.gz

下面的安装将以：alb-micc-2.1-x86.tar.gz 进行演示和操作。

2.支持的环境

平台类型	系统类型
芯片类型	x86、arm
操作系统	linux 系统

3.关于 License 文件

请使用官方授权的有效期内 license 文件。

2.2. 安装部署 MICC

MICC 是基于 web 的 ALB 管理和监控工具，支持 ALB 负载均衡器资源监控、ALB 负载均衡器自动部署、机器资源管理、ALB 负载均衡器管理、用户管理、操作审计、权限控制、路由管理、上游管理、服务管理、证书管理等功能。

部署 MICC 操作步骤：

为了避免安装依赖包权限问题请切换到 root 用户下进行操作, 其安装步骤主要有：

(1) 切换 root 用户, 上传 alb-micc 安装包(alb-micc-2.1-x86.tar.gz)至目标服务器的安装目录下(如：/opt 目录下)

(2) 解压安装包：tar -zxvf alb-micc-2.1-x86.tar.gz

```
[root@localhost opt]# tar -zxvf alb-micc-2.1-x86.tar.gz
alb-micc-2.1-x86/
alb-micc-2.1-x86/stop.sh
alb-micc-2.1-x86/micc-be/
alb-micc-2.1-x86/micc-be/micc.db
alb-micc-2.1-x86/micc-be/deploy/
alb-micc-2.1-x86/micc-be/deploy/configs/
```

(3) 进入解压后的文件夹：cd alb-micc-2.1-x86

```
[root@localhost opt]# cd alb-micc-2.1-x86
```

(4) 上传授权文件至当前目录。

(5) 修改 micc 配置文件和 dashboard 配置文件，分别增加 etcd 数据库 IP 地址

修改 micc 配置文件: /opt/alb-micc-2.1-x86/micc-be/config.yaml

如:将 micc 部署在 172.20.140.180 服务上则配置 etcdIP 为 172.20.140.180

```
[root@localhost micc-be]# vim /opt/alb-micc-2.1-x86/micc-be/config.yaml
```

```
use-multipoint: false
use-redis: false
iplimit-count: 15000
iplimit-time: 3600
etcd:
  - http://172.20.140.180:2379
dashboard-port: "9000"
nginx-conf-dir: /tmp/nginx_conf
login-failed-count: 5
login-failed-expire-minute: 10
tencent-cos:
  bucket: ""
```

配置 dashboard 配置文件: alb-dashboard/conf/conf.yaml

```
[root@localhost micc-be]# vim /opt/alb-micc-2.1-x86/alb-dashboard/conf/conf.yaml
```

```
allow_list:
# If we don't set any IP list, then any IP access is allowed by default.
#- 127.0.0.1 # The rules are checked in sequence until the first match is
#- ::1 # In this example, access is allowed only for IPv4 network 1
# It also support CIDR like 192.168.1.0/24 and 2001:0db8::/32
etcd:
  endpoints: # supports defining multiple etcd host addresses for an etcd cluster
    - 172.20.140.180:2379
    # yamllint disable rule:comments-indentation
    # etcd basic auth info
  # username: "root" # ignore etcd username if not enable etcd auth
  # password: "123456" # ignore etcd password if not enable etcd auth
  mtls:
    key_file: "" # Path of your self-signed client side key
    cert_file: "" # Path of your self-signed client side cert
    ca_file: "" # Path of your self-signed ca cert, the CA is used to sign callers'
  prefix: /alb # apisix config's prefix in etcd, /apisix by default
```

(5) 在安装目录执行运行启动脚本: ./start.sh。

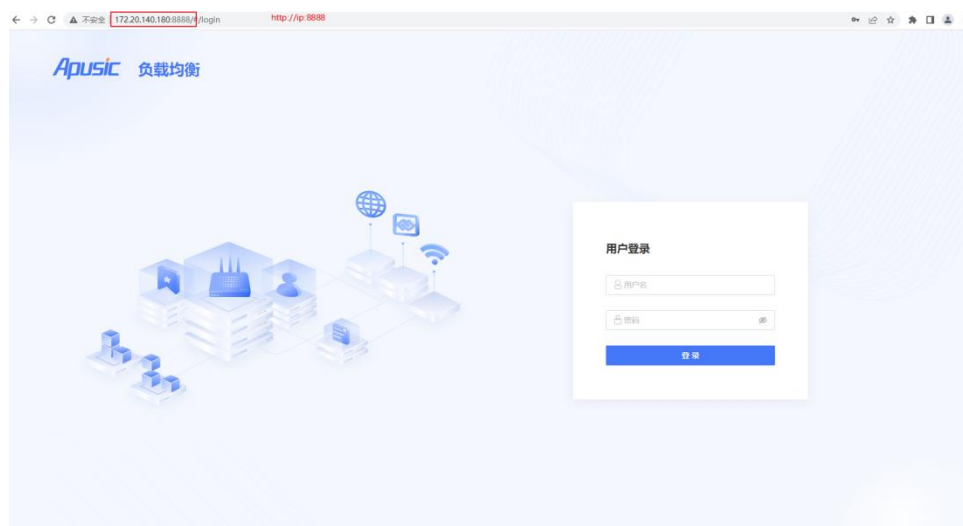
```
[root@localhost alb-micc-2.1-x86]# pwd
/opt/alb-micc-2.1-x86
[root@localhost alb-micc-2.1-x86]# ./start.sh
etcd服务未启动
micc-server服务未启动
manager-api服务未启动
/opt/alb-micc-2.1-x86/alb-dashboard
etcd服务正在运行
micc-server服务正在运行
manager-api服务正在运行
```


(6) 通过浏览器访问 MICC 控制台: `http://ip:port` (其中 ip 表示部署 MICC 控制台服务器的 IP, port 为控制台端口, 如: 172.20.140.180:8888) (建议用谷歌或火狐浏览器访问)

(7) 登录 MICC: MICC 根据登录的用户角色分配操作权限, 默认用户登录名密码如下:

- 系统管理员用户: micc-admin, 密码为: apusic\$micc
- 系统安全保密员: micc-ss, 密码为: apusic\$micc
- 系统安全审计员: micc-sa, 密码为: apusic\$micc
- 控制台操作员: micc-user, 密码为: apusic\$micc

启动完成后通过浏览器访问控制台



2.3. 停止 MICC

在 micc 安装根目录如 `/opt/alb-micc-2.1-x86` 目录, 执行停止脚本: `./stop.sh`

```
[root@localhost alb-micc-2.1-x86]# pwd
/opt/alb-micc-2.1-x86
[root@localhost alb-micc-2.1-x86]# ./stop.sh
etcd 服务正在运行
正在停止 etcd服务 ...
micc-server 服务正在运行
正在停止 micc服务 ...
manager-api 服务正在运行
正在停止 alb-dashboard服务 ...
etcd服务已停止
micc服务已停止
manager-api服务已停止
[root@localhost alb-micc-2.1-x86]#
```

2.4. 常见问题解决

(1) 端口占用与解决

MICC 占用端口表

端口	服务名称
2379	etcd 数据库
8888	MICC 管控台
9000	ALB 管控台

(2) 配置文件及其目录

配置内容	字段	默认值	配置文件地址
Micc	addr	8888	/opt/qiumeilun/alb2.1/alb-micc-2.1-x86/micc-be/config.yaml
控制台端口	dashboard-port	9000	/opt/qiumeilun/alb2.1/alb-micc-2.1-x86/micc-be/config.yaml
ALB 日志文件	access.log error.log	access.log error.log	/opt/alb-micc-2.1-x86/alb-dashboard/logs/access.log
MICC 日志文件			/opt/alb-micc-2.1-x86/micc-be/log/

第3章 MICC 平台

MICC 是基于 Web 的 ALB 的管理和监控工具，支持对 ALB 各个负载均衡器运行的监控、负载均衡器管理、机器资源管理、负载均衡器自动部署、操作审计、用户管理、权限控制等功能。

3.1. 首页

3.1.1. 登录 MICC

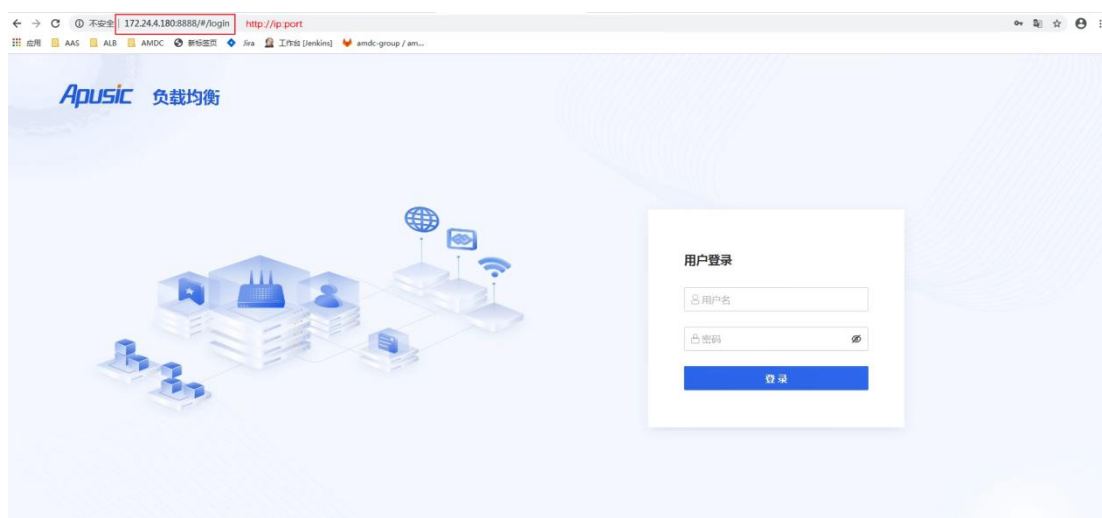
部署 MICC 完成后打开浏览器（推荐 Chrome、firefox），进入到 MICC 登录页面：<http://IP:serverPort/>。

例如：

将 MICC 部署在 172.24.4.180 服务器，端口默认为 8888，则 MICC 登录地址为：<http://172.24.4.180:8888>

输入用户名密码登录控制台，默认用户登录名密码如下：

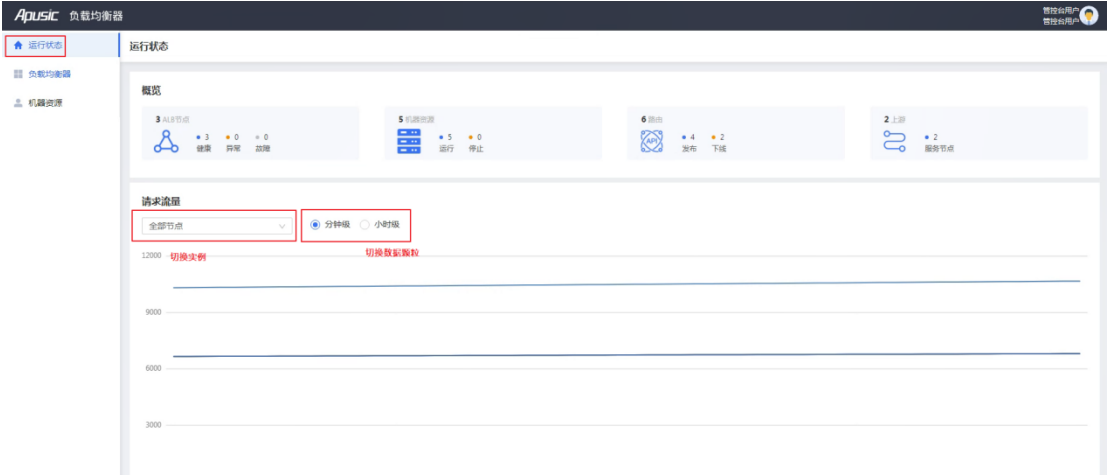
- 系统管理员用户：micc-admin，密码为：apusic\$micc
- 系统安全保密员：micc-ss，密码为：apusic\$micc
- 系统安全审计员：micc-sa，密码为：apusic\$micc
- 控制台操作员：micc-user，密码为：apusic\$micc



3.1.2. 运行状态

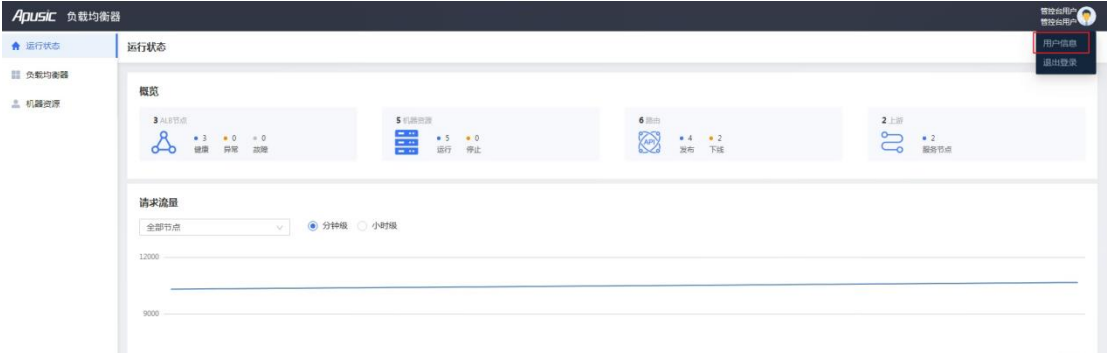
MICC 首页-运行状态，提供 ALB 运行监控指标数据可视化图表，汇总了 ALB 实例的运行状态数据、服务负载数据等指标，包括 ALB 节点、计算资源、路由数据、上游数据。

管控台用户登录, 进入首页【运行状态】



3.1.3. 个人信息

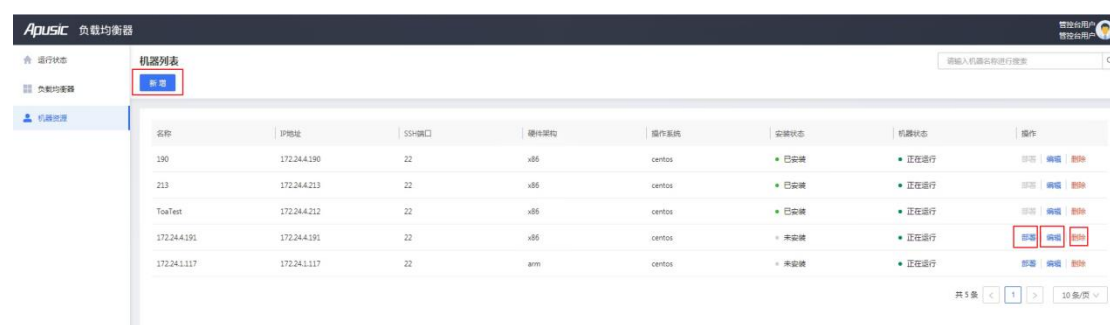
点击首页右上角头像，进入个人信息页面，可以查看用户信息或修改当前用户的密码、邮箱、电话



3.2. 机器管理

ALB 自动部署指的是在 MICC 管控台通过 SSH 链接服务器并执行自动部署脚本，因此自动部署 ALB 之前需要添加安装 ALB 负载均衡器的目标机器到 MICC 管理平台中。

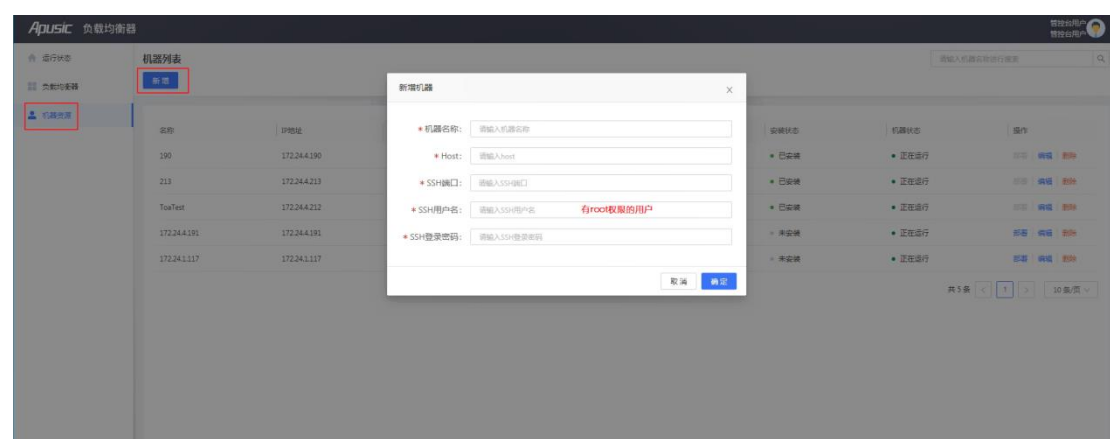
管控台用户登录（默认用户:micc-user 密码: apusic），进入机器资源页面, 机器资源页面提供新增机器、部署 ALB、编辑机器、删除机器的功能入口



3.2.1. 新增机器

管控台用户登录（默认用户:micc-user 密码: apusic），点击左侧【机器资源】进入机器管理页面，点击【新增】按钮，在输入框中输入相应的信息新增机器

备注：只能新增链接正常的服务器，因此请确认目标机器链接状态正常



- 机器名称：自定义机器名称
- Host：安装 ALB 负载均衡器的目标机器的 IP，如：172.24.4.180
- SSH 端口：登录远程机器的端口, 如：22
- SSH 用户名：登录远程机器用户名，仅支持 root 用户
- SSH 登录密码：登录远程机器的密码

3.2.2. 编辑机器、删除机器

管控台用户登录（默认用户:micc-user 密码: apusic）点击【机器资源】进入机器管理页面，机器列表右侧提供机器的【编辑】、【删除】入口



3.3. 自动部署 ALB 负载均衡器

3.3.1. 自动部署 ALB

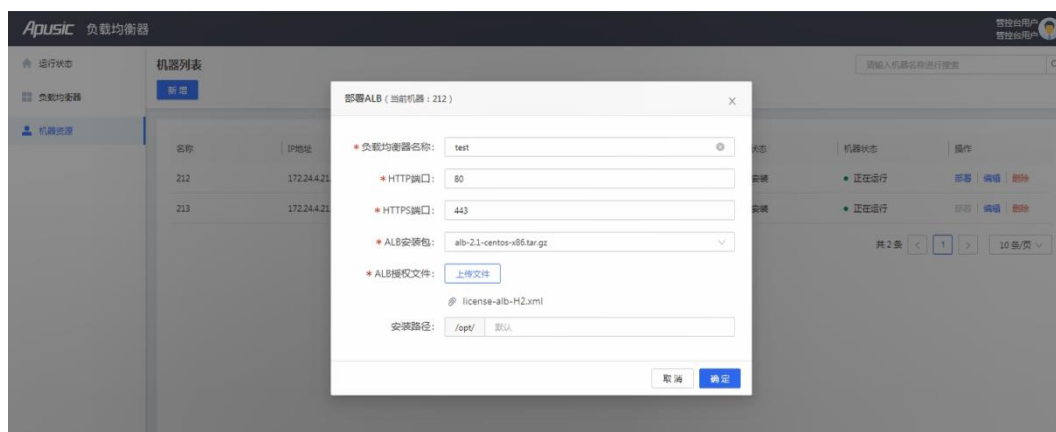
ALB 自动部署是指通过 MICC 图形界面化部署 ALB 负载均衡器，将 ALB 安装包、授权文件以及启动配置传送到目标机器的安装目录，并通过 SSH 命令启动 ALB 负载均衡器。部署主要流程包括：



注意：使用自动部署功能目标服务器用户具备root权限

步骤 1：进入【机器资源】，点击需要部署的目标机器右侧【部署】按钮，根据部署界面提示输入相应信息，点击【确定】按钮





- 负载均衡器名称：自定义负载均衡器名称
- HTTP 端口：开放的 http 路由端口, 默认是 80 端口
- HTTPS 端口：开放的 https 路由端口, 默认是 443 端口
- ALB 安装包：根据目标机器选择对应的安装包
- ALB 授权文件： 授权的 ALB license 文件

步骤 2. 验证自动部署结果

进入部署 ALB 负载均衡器的服务器，检查 80 端口和 443 端口，端口正常启用说明，ALB 部署成功

```
[root@linux-4-213 ~]# netstat -nlt|grep 80
tcp        0      0 127.0.0.1:6380      0.0.0.0:*           LISTEN      12982/redis-server
tcp        0      0 172.24.4.213:6380   0.0.0.0:*           LISTEN      12982/redis-server
tcp        0      0 0.0.0.0:80          0.0.0.0:*           LISTEN      14968/nginx: master
tcp6       0      0 :::80              :::*                LISTEN      14968/nginx: master
[root@linux-4-213 ~]# netstat -nlt|grep 443
tcp        0      0 0.0.0.0:443         0.0.0.0:*           LISTEN      14968/nginx: master
tcp6       0      0 :::443             :::*                LISTEN      14968/nginx: master
[root@linux-4-213 ~]#
```

3. 4. 负载均衡器管理

在 MICC 平台自动部署的负载均衡器，将展示在负载均衡器页面，并提供对负载均衡器操作入口。

3.4.1. 启动、停止、重启、卸载负载均衡器

控制台用户登录 MICC, 进入【负载均衡器】，负载均衡器列表右侧提供了启动、停止、重启、编辑、卸载、配置入口



3.5. 用户管理

MICC 中用户分为：系统管理员、系统安全保密员、系统安全审计员、管控台用户，对应角色及其操作选项如下：

- 系统管理员： 具备新增用户、修改用户信息、删除用户的功能
- 系统安全保密员： 具备用户激活、授权操作、修改用户密码。
- 系统安全审计员： 具备查看所有操作记录的权限。
- 管控台用户： 具备负载均衡运行状态监控、负载均衡器管理、机器资源管理、自动部署的权限

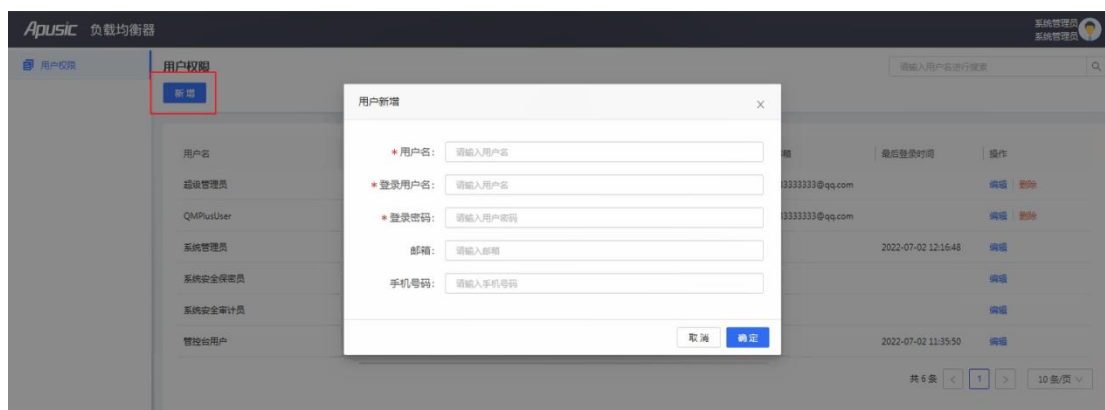
下表是系统默认用户，系统默认的用户不可删除、冻结、授权：

用户角色	登录用户名	密码
系统管理员	micc-admin	apusic\$micc
安全保密员	micc-ss	apusic\$micc
安全审计员	micc-sa	apusic\$micc
管控台用户	micc-user	apusic\$micc

注意：由系统管理员新增的用户，需要经过系统安全保密员授权并激活方可使用

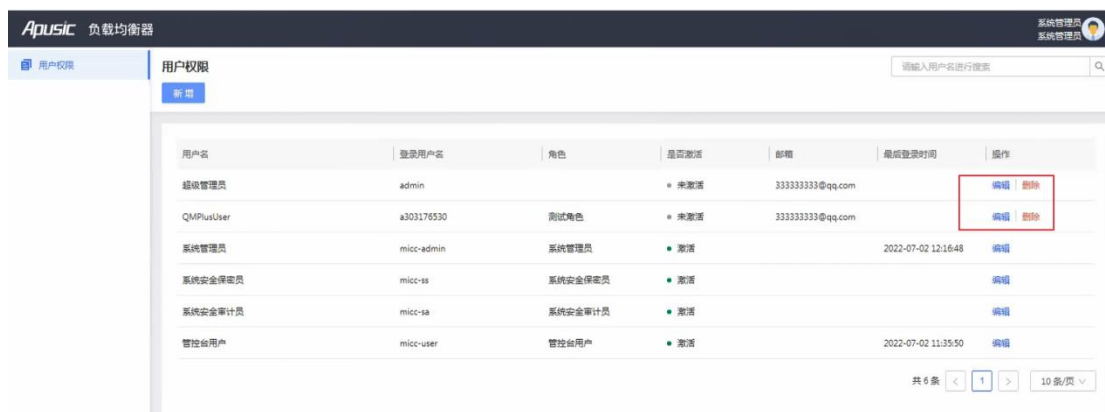
3.5.1. 新增用户

系统管理员用户登录 MICC,进入【用户权限】页面，点击【新增】按钮，输入相应的用户名、登录用户名、登录密码创建用户



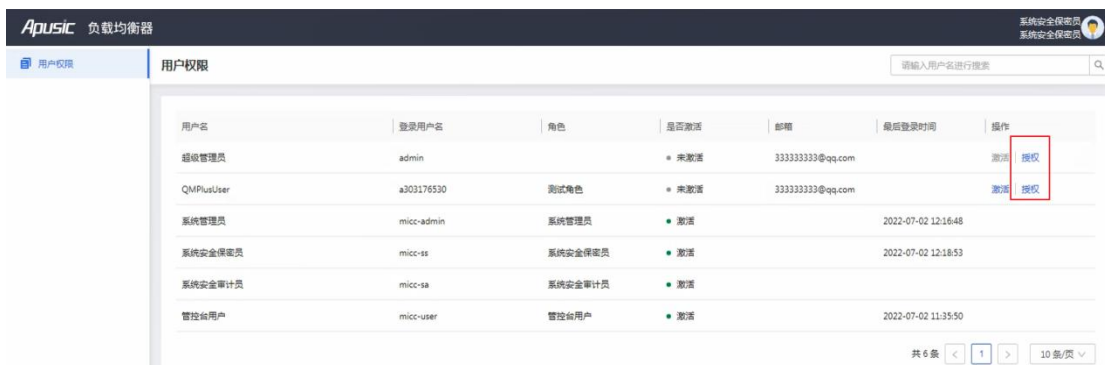
3.5.2. 编辑用户、删除用户

系统管理员用户登录 MICC（默认 micc-admin 密码 apusic）,进入【用户权限】页面，用户列表右侧提供了编辑、删除入口



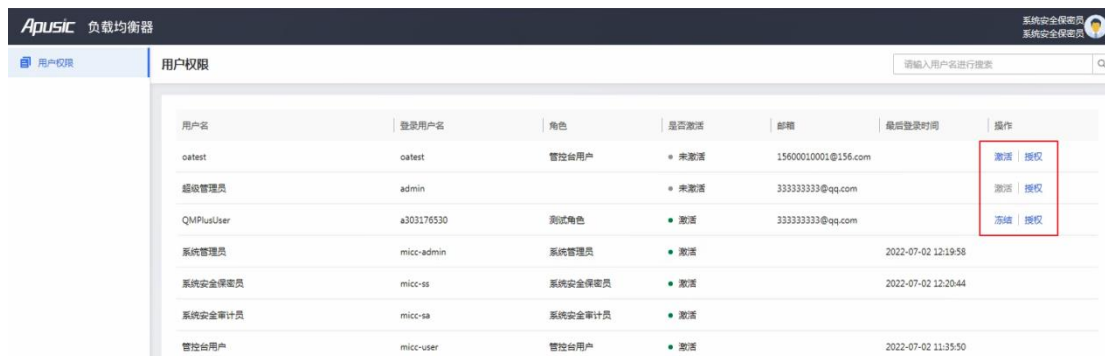
3.5.3. 授权用户

系统安全保密用户登录 MICC（默认 micc-ss 密码 apusic\$micc）,进入【用户权限】页面，点击用户列表右侧【授权】按钮，打开授权弹窗授予用户对应的权限



3.5.4. 激活用户、冻结用户

只有已经授权的用户才可激活，因此激活用户前需要先授权用户对应的角色（参考上面 3.5.3 部分）。系统安全保密用户登录 MICC（默认 micc-ss 密码 apusic），进入【用户权限】页面，用户列表右侧提供【激活】、【冻结】入口，只有激活成功的用户才能登录 MICC，被冻结的用户不能登录 MICC 控制台



3.6. 操作审计

安全审计员主要负责对系统管理员、安全管理员的操作行为进行审计跟踪分析和监督检查，以及及时发现违规行为，并定期向系统安全保密管理机构汇报相关情况。安全审计员用户登录 MICC 进入日志审计页面（默认用户名：micc-sa 密码：apusic\$micc）

Apusic 负载均衡器

登录 管理控制台

运行状态

操作审计

请输入用户名进行搜索

操作用户名	操作资源	操作方法	操作结果	登录IP	操作时间
运维人员	登录	登录	成功	172.24.4.56	2022-4-12 00:00:00
集团管理员	负载均衡器	停止	成功	172.24.4.56	2022-4-12 12:00:00
测试用户	部署机器	新增	成功	172.24.4.56	2022-4-12 13:00:00
admin	ALB路由	下线	成功	172.24.4.56	2022-4-12 13:00:00
测试用户	ALB路由	修改	成功	172.24.4.56	2022-4-12 14:00:00
测试用户	系统账户	新增	成功	172.24.4.56	2022-4-12 15:00:00

共5页 第 1 页 < > 10条/页

3.7. 密码与安全

密码修改说明：为了保证系统安全，要求密码长度需 6 位及以上，并且包含特殊字符。

3.13.1. 修改当前用户密码

登录控制台，点击首页右上角【用户信息】，在用户信息修改当前用户的登录密码。

Apusic 负载均衡器

系统安全审计

操作审计

请输入用户名进行搜索

用户信息 退出登录

操作用户名	操作资源	操作方法	操作结果	登录IP	操作时间
micc-user	获取机器资源以外的其他配置信息	GET	成功	127.0.0.1	2022-7-01 12:17:15
micc-user	获取请求流量相关信息	GET	成功	127.0.0.1	2022-7-01 12:17:15
micc-user	获取MachineManagementRunningStatus统计	GET	成功	127.0.0.1	2022-7-01 12:17:15
micc-user	获取机器资源以外的其他配置信息	GET	成功	127.0.0.1	2022-7-01 02:41:03

3.13.2. 通过系统管理员修改密码

系统管理员可以修改用户信息。系统管理员用户登录进入用户权限页面点击【编辑】，即可编辑用户密码、手机号、邮箱等信息。



第4章 ALB 实例管控台

管控台用户登录 MICC,进入【负载均衡器】页面>点击负载均衡器列表右侧【配置】按钮，即可进入实例管控台



ALB 实例管控台包含 ALB 实例节点运行监控、路由管理、上游管理、服务管理、插件管理、证书管理、系统信息查询。

- 路由：路由（Route）是请求的入口点，它通过定义一些规则来匹配客户端的请求，然后根据匹配结果加载并执行相应的插件，并把请求转发给到指定上游。

路由中主要包含三部分内容：匹配规则(比如匹配路径、优先级等)，插件配置(安全防护、流量控制等)和上游信息。

- 上游：上游是真实服务的访问入口即后端服务系统，ALB 对给定的多个上游

按照配置规则进行负载均衡，实现对上游目标节点的负载均衡和健康检查。

当我们的 Route 有比较多的上游重复配置（比如启用相同的插件配置或上游信息）时，可以单独配置上游，不同的路由课可以重复使用相同的上游。

- 服务：服务是由路由中公共的配置、插件、上游目标信息组合而成的一组抽象，当我们的 Route 有比较多的重复配置、插件和上游时可以通过同时绑定到一个服务上，减少冗余配置。

路由与服务之间，通常是 N:1 的关系，一个服务可对应一组上游节点、可被多条路由绑定。

- 插件：插件是获取 ALB 特殊体验的应用程序, 作用于 http 请求/响应期间，可直接绑定在路由上，也可绑定在服务或者消费者中，绑定在路由、服务或消费者中的同一个插件在路由中的优先级更高，其中在插件 tab 中启用的插件是全局插件，作用于所有的路由和服务。在路由或服务中启用的插件则是局部插件，作用于局部路由或服务。
- 证书：证书被网关用于处理加密请求，它与 SNI 关联，并与路由中主机名绑定，如 https 请求 SSL 证书。

4.1. 路由管理

4.2.1. 创建路由

管控台用户登录 MICC,进入【负载均衡器】页面>点击负载均衡器列表右侧【配置】按钮，进入管控页面，点击路由进入路由页面。

ALB 创建路由可通过两种方式：

- 界面创建
- 使用编辑器创建





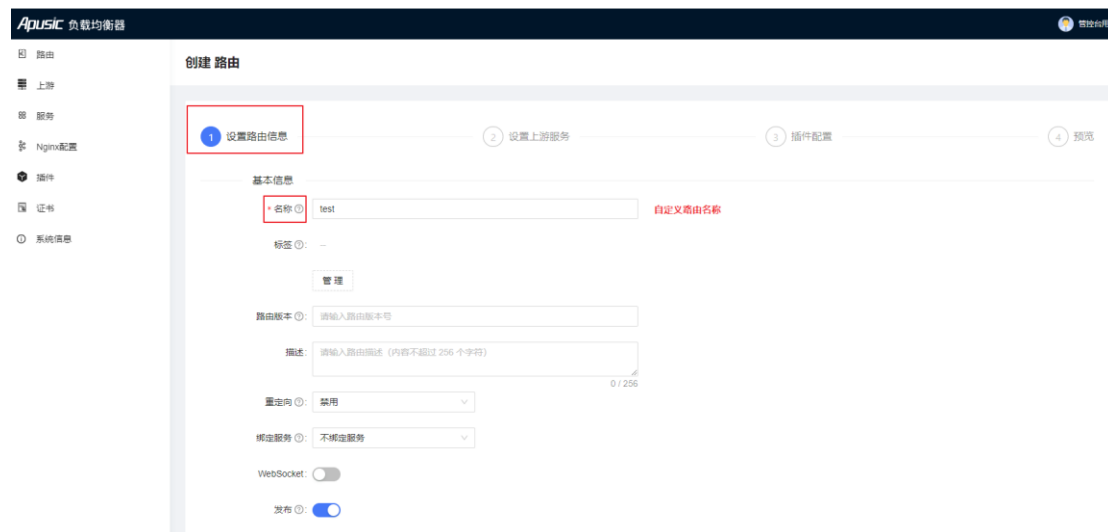
1. 图形界面创建路由

管控台用户登录 MICC,进入【负载均衡器】页面>点击负载均衡器列表右侧【配置】按钮，进入管控页面，点击【创建】按钮进入创建路由页面，创建路由的流程分为四个步骤：

设置路由信息>设置上游服务>插件配置>预览并提交

➤ 步骤 1. 设置路由信息

设置路由信息包括：设置路由基本信息和设置客户端请求匹配条件



- 名称：必填项，自定义路由名称，不允许配置重名路由
- 绑定服务：此项默认状态不绑定服务，有可绑定服务时下拉选择创建的服务
- 发布：默认开启，开启状态下路由才会生效
- 填写请求信息：这里需要定义客户端请求的匹配规则
- 域名：ALB 客户端服务器域名，无则不填
- 路径：必填项，默认所有路径/*，可以是路由的具体详细路径或者泛路径（以/*结尾）
- 客户端地址：允许访问的客户端 IP 或 IP 网段
- 优先级：必填项，默认为 0，路由配置相同时取优先级高的路由
- 请求改写：对客户端的请求的协议、路径、域名、请求头进行改写

➤ 步骤 2. 设置上游服务

设置上游服务包括设置后端业务系统匹配规则、链接设置、健康检查设置

- 选择上游服务：默认手动填写
- 负载均衡算法：必选项，包括带权轮询、一致性哈希、指数加权移动平均法、最小连接数，默认带权轮询。
- 上游类型：必选项，默认为节点
- 目标节点：必输项，输入上游目标服务器主机名或者 IP 地址、端口
- 权重：必输项，输入上游服务器的权重，权重高的节点承受的流量多，权重配置为 0 时熔断该节点
- 协议：必选项，选择后端服务器请求的类型选择 http、https、grpc、grpcs
- 超时时间：必输项，默认超时时间为 6S

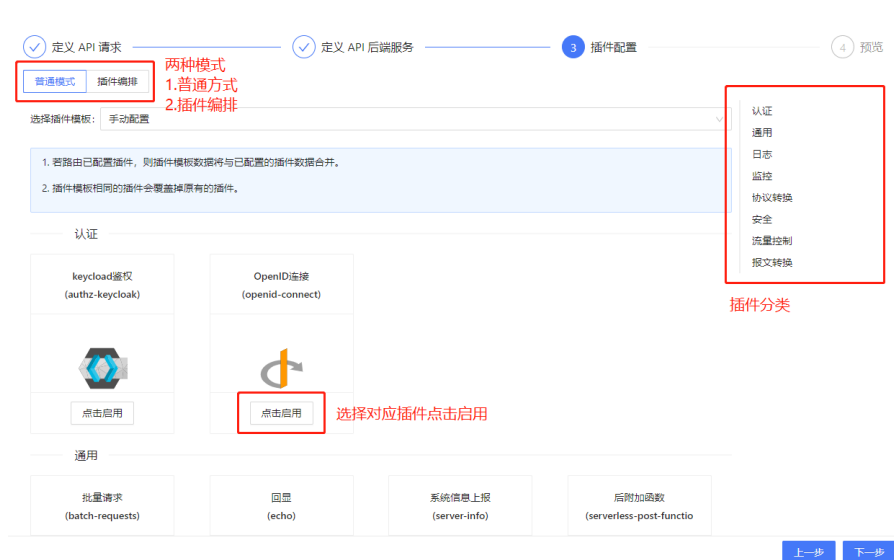
➤ 步骤 3. 插件配置

ALB 提供功能强大的插件系统，其中系统自带的插件列表及其功能有：

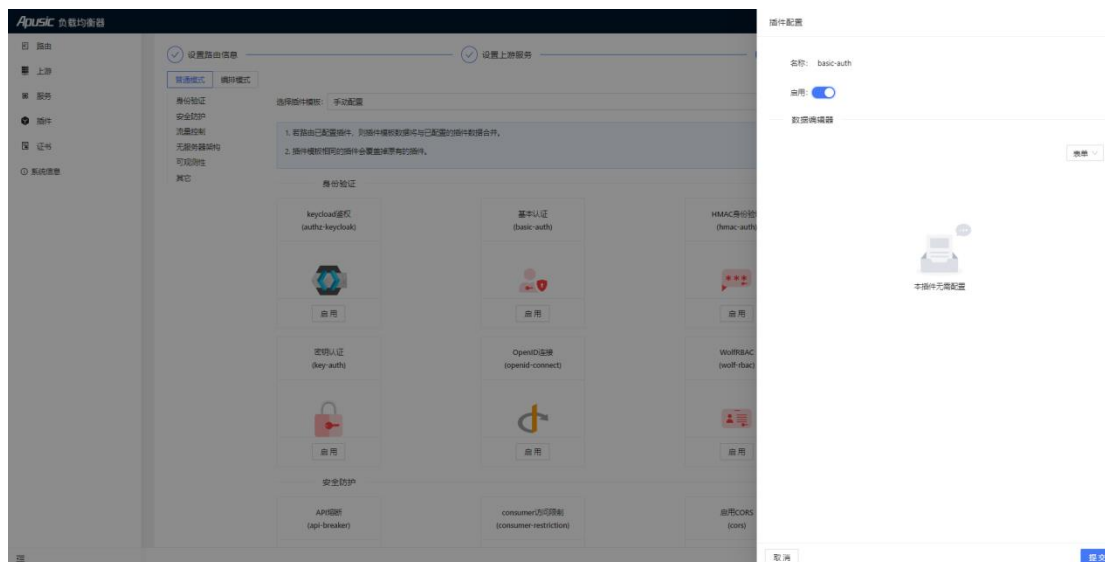
插件功能	插件名称
安全防护	API 熔断(api-breaker)、启用 CORS(cors)、故障注入(fault-injection)、ip 黑白名单(ip-restriction)、Referer 限制(referer-restriction)、请求验证(request-validation)、请求拦截(uri-blocker)
流量控制	限制并发连接(limit-conn)、限制请求次数(limit-count)、限制请求速度(limit-req)、
无服务架构	后附加函数(serverless-post-function)、前附加函数(serverless-pre-function)
可观测性	HTTP 日志推送(http-logger)、Kafka 日志推送(kafka-logger)、Prometheus (prometheus)、请求 ID(request-id)、系统日志推送(sls-logger)、系统日志推送(syslog)、TCP 日志推送(tcp-logger)、UDP 日志推送(udp-logger)、Zipkin(zipkin)
其他	批量请求(batch-requests)、ClientControl(client-control)、ext-plugin-post-req(ext-plugin-post-req)、ext-plugin-pre-req(ext-plugin-pre-req)、gRPC 转码(grpc-transcode)、gzip(gzip)、代理缓存(proxy-cache)、代理镜像(proxy-mirror)、响应信息重写(response-rewrite)、静态重写(static-rewrite)、响应信息正则替(string-replace)、User-Agent 黑白名单(ua-restriction)

启用插件配置步骤：

(1) 选择普通方式或插件编排方式选择插件：

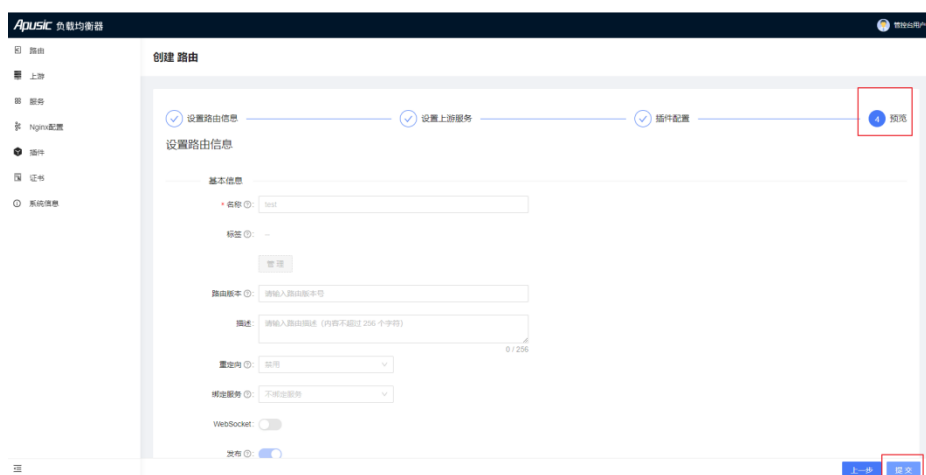


(2) 将启用开关开启，并在数据编辑器中编辑插件配置，启用插件：



➤ 步骤 4. 预览并提交

点击预览页面【提交】按钮，路由配置成功



2. 使用编辑器创建路由

点击路由列表右侧【高级特性】，在下拉框中选择【数据编辑器】，在大括号内编辑路由名称、路径、上游信息等路由详细信息，并提交，使用编辑器创建路由与界面创建路由一样需要包含三部分内容：匹配规则(比如匹配路径、优先级等)，插件配置(非必须)和上游信息。

路由列表

路由 (Route) 是请求的入口点, 它定义了客户端请求与服务之间的匹配规则。路由可以与服务 (Service)、上游 (Upstream) 关联, 一个服务可对应一个路由, 一个路由可以对应一个上游对象 (一组后端服务节点), 因此, 每个匹配到路由的请求将被网关代理到路由绑定的上游服务中。

名称:

路径:

标签:

重置

查询

展开

+ 创建

高级特性

+ 插件模板配置

+ 数据编辑器

下线

编辑

更多

名称	域名	路径	描述	标签	路由版本	状态	更新时间
沙袋		<input type="text" value="P"/>	-			已发布	2021-11-19 16:26:50

第 1-1 条/总共 1 条 < 1 > 10 条/页

Apusic 负载均衡器

路由列表

路由 (Route) 是请求的入口点, 它定义了客户端请求与服务之间的匹配规则。路由可以与服务 (Service)、上游 (Upstream) 关联, 一个服务可对应一个路由, 一个路由可以对应一个上游对象 (一组后端服务节点), 因此, 每个匹配到路由的请求将被网关代理到路由绑定的上游服务中。

名称:

路径:

描述:

标签:

路由版本:

状态:

更新时间:

45

test

已发布

已发布

数据编辑器

JSON

复制

文档

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

取消

提交

```
{
  "uri": "/+",
  "name": "45",
  "methods": [
    "GET",
    "POST",
    "PUT",
    "DELETE",
    "PATCH",
    "HEAD",
    "OPTIONS",
    "CONNECT",
    "TRACE"
  ],
  "upstream": {
    "nodes": [
      {
        "host": "172.20.140.45",
        "port": 6888,
        "weight": 1
      }
    ],
    "timeout": {
      "connect": 6,
      "send": 6,
      "read": 6
    },
    "type": "roundrobin",
    "scheme": "http",
    "pass_host": "pass",
    "keepalive_pool": {
      "idle_timeout": 60,
      "requests": 1000,
      "size": 320
    }
  }
}
```

定义API请求

定义后端服务

示例 2 创建 http 路由

以下示例介绍了 http 请求的创建和使用, 其中 micc 部署在 172.24.4.180 服务器上, ALB 实例部署在 172.24.4.213 服务器上, 后端真实的业务系统是 <http://172.24.4.212:6888/1M/1M.html>

<http://172.24.4.213:6888/1M/1M.html>

第一步: 添加机器资源 172.24.4.213

Apusic 负载均衡器

机器列表

运行状态

负载均衡

机器资源

新增

请输入机器名称进行检索

名称

IP地址

SSH端口

硬件架构

操作系统

安装状态

机器状态

操作

213

172.24.4.213

22

x86

centos

未安装

正在运行

部署

编辑

删除

212

172.24.4.212

22

x86

centos

已安装

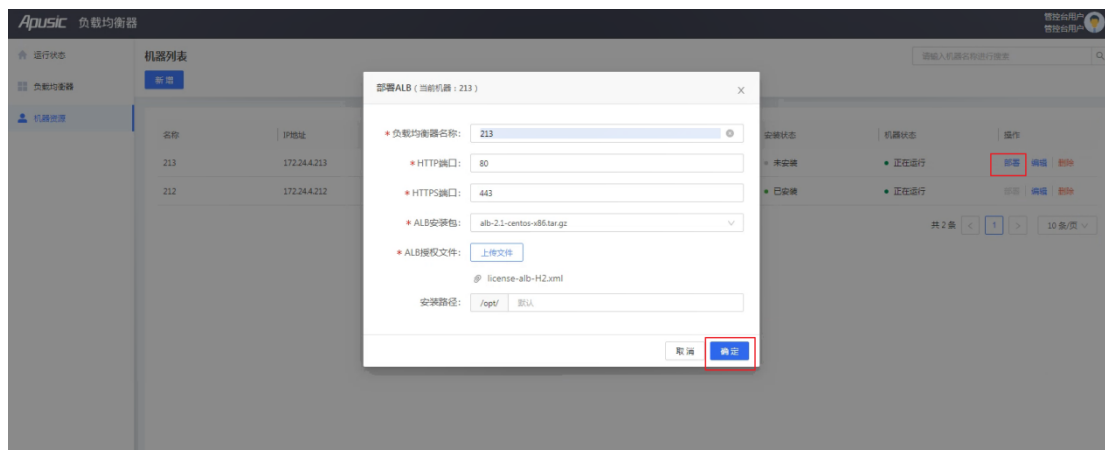
正在运行

部署

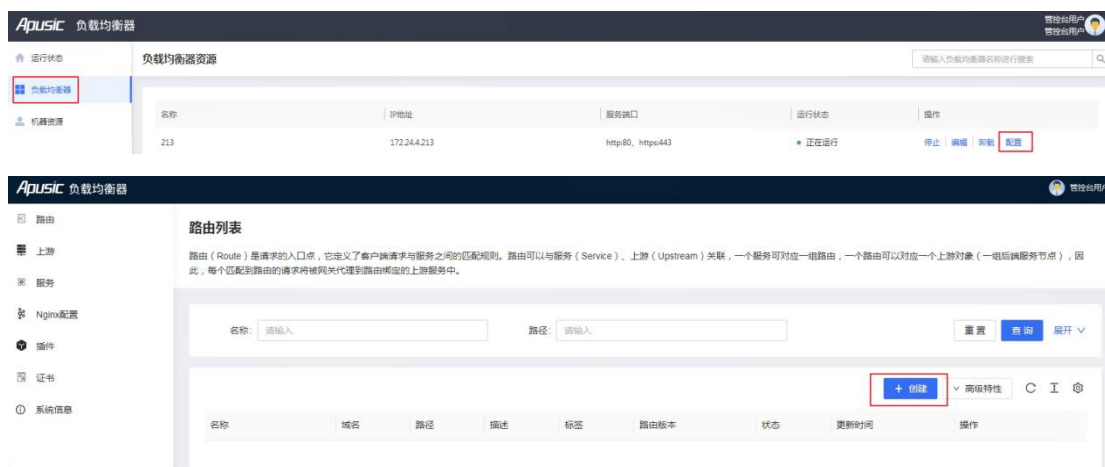
编辑

删除

第二部: 部署 ALB 到 213 机器上



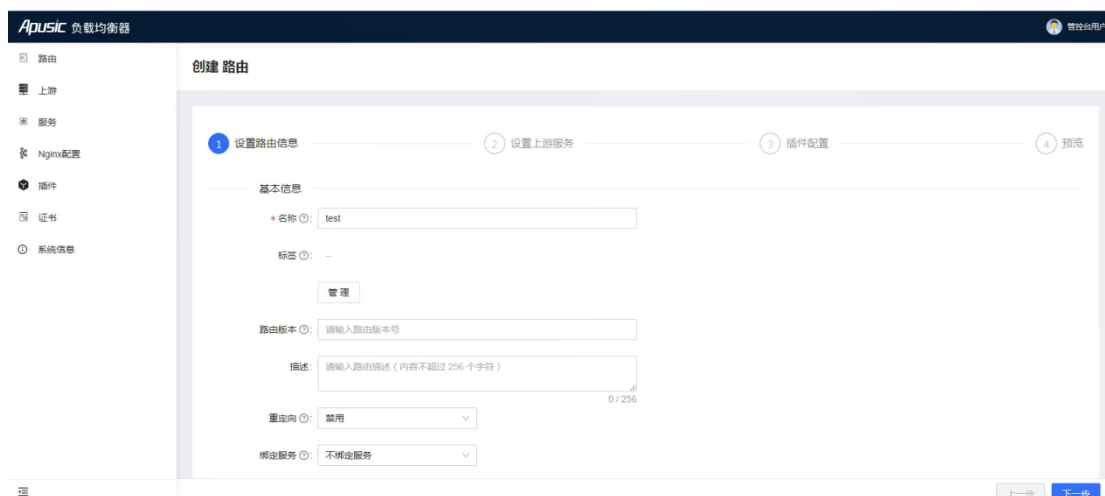
第三步：创建路由 test

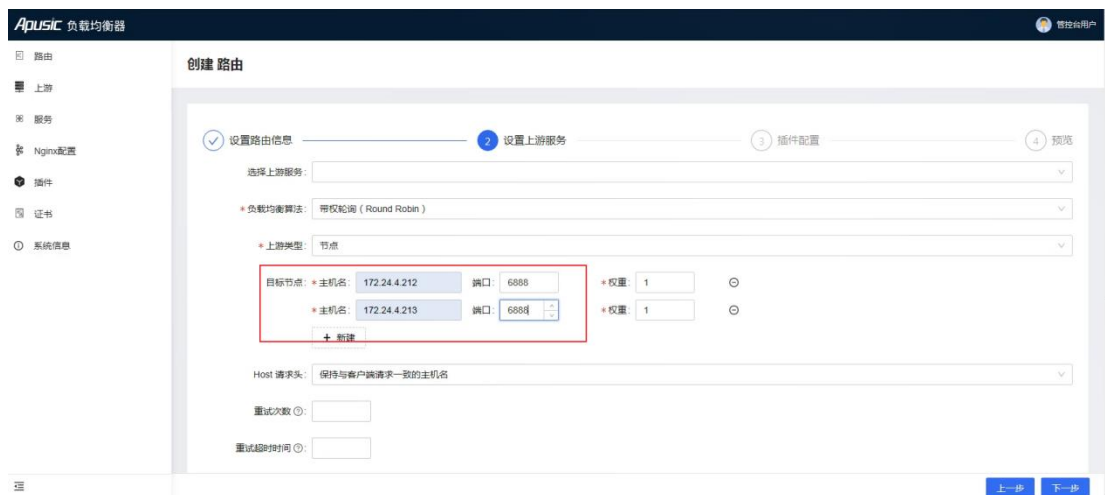


1. 请求的路径设置为：/1M/*(注意：*号为通配符，表示该目录下的所有路、文件径均生效)

2. 后端服务为： 172.24.4.212:6888

172.24.4.213:6888

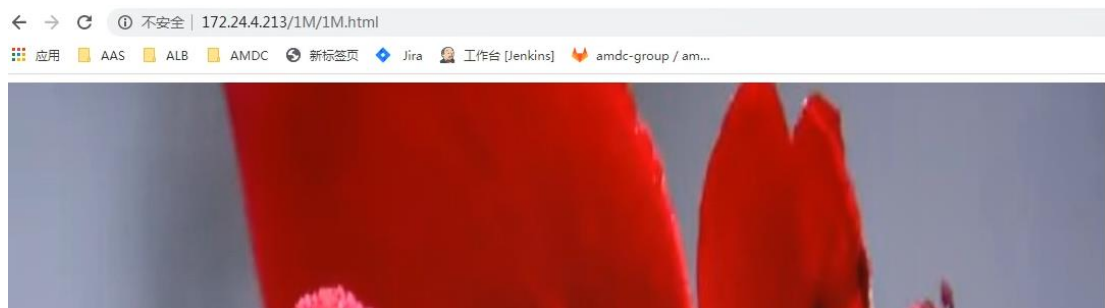




第四步 访问创建的路由

在浏览器中输入路由的地址和相应的路径：

<http://172.24.4.180:80/1M/1M.html>



第三步： 检查负载均衡是否生效

发送第二步中的请求多次，进入服务器日志执行 `tail -f /opt/安装目录/alb_install_dir/alb/logs/access.log` 查看日志文件内容，如下图，日志文件记录了发送到上游中的请求

```

^C
[root@linux-4-213 ~]# tail -f /opt/qiumeilun/alb/alb-2.1-centos-x86/alb_install_dir/alb/logs/access_log
172.24.6.65 - - [02/Jul/2022:14:21:09 +0800] 172.24.4.213 "GET /1M/1M.html HTTP/1.1" 304 0 0.007 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36" 172.24.4.213:6888 304 0.006 "http://172.24.4.213"
172.24.6.65 - - [02/Jul/2022:14:21:09 +0800] 172.24.4.213 "GET /1M/1M.html HTTP/1.1" 304 0 0.002 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36" 172.24.4.212:6888 304 0.002 "http://172.24.4.213"
172.24.6.65 - - [02/Jul/2022:14:21:09 +0800] 172.24.4.213 "GET /1M/1M.html HTTP/1.1" 304 0 0.007 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36" 172.24.4.213:6888 304 0.006 "http://172.24.4.213"
172.24.6.65 - - [02/Jul/2022:14:21:10 +0800] 172.24.4.213 "GET /1M/1M.html HTTP/1.1" 304 0 0.002 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36" 172.24.4.212:6888 304 0.001 "http://172.24.4.213"
172.24.4.190 - - [02/Jul/2022:14:22:04 +0800] 172.24.4.213:80 "GET /alb/status HTTP/1.1" 200 166 0.000 "-" "Go-http-client/1.1" - - "http://172.24.4.213:80"
172.24.4.190 - - [02/Jul/2022:14:23:04 +0800] 172.24.4.213:80 "GET /alb/status HTTP/1.1" 200 166 0.000 "-" "Go-http-client/1.1" - - "http://172.24.4.213:80"
172.24.4.190 - - [02/Jul/2022:14:24:04 +0800] 172.24.4.213:80 "GET /alb/status HTTP/1.1" 200 166 0.000 "-" "Go-http-client/1.1" - - "http://172.24.4.213:80"
172.24.4.190 - - [02/Jul/2022:14:25:04 +0800] 172.24.4.213:80 "GET /alb/status HTTP/1.1" 200 166 0.000 "-" "Go-http-client/1.1" - - "http://172.24.4.213:80"
172.24.4.190 - - [02/Jul/2022:14:26:04 +0800] 172.24.4.213:80 "GET /alb/status HTTP/1.1" 200 166 0.000 "-" "Go-http-client/1.1" - - "http://172.24.4.213:80"
172.24.4.190 - - [02/Jul/2022:14:27:04 +0800] 172.24.4.213:80 "GET /alb/status HTTP/1.1" 200 166 0.000 "-" "Go-http-client/1.1" - - "http://172.24.4.213:80"

```

示例 3 创建 https 路由

以下示例aas.alb.com的SSL证书使用方法，其中后端业务系统链接为：

http://172.24.4.212:6888/1M/1M.html.

http://172.24.4.213:6888/1M/1M.html.

第一步：创建 https 路由

将 ALB 部署在 172.24.4.213 服务器上，启动并创建如下路由：

路由名称：test

路由路径：/*

域名：aas.alb.com

目标节点：172.24.4.212:6888

172.24.4.213:6888

其他配置保持默认，点击【下一步】，插件页面点击【下一步】，预览页面点击【提交】，直到提示【提交成功】

基本信息

* 名称 ①: test

标签 ①: --

管理

路由版本 ①: 请输入路由版本号

描述: 请输入路由描述 (内容不超过 256 个字符)

0 / 256

重定向 ①: 禁用

绑定服务 ①: 不绑定服务

WebSocket: ☐

发布 ①: ☒

匹配条件

域名 ①: aas.alb.com

+ 新建

* 路径 ①: /*

+ 新建

客户端地址 ①: 请输入客户端地址

Apusic 负载均衡器

管理控制台

路由

上游

服务

Ngix配置

插件

证书

系统信息

编辑 路由

1 设置路由信息

2 设置上游服务

3 插件配置

4 预览

选择上游服务: 手动填写

* 负载均衡算法: 加权轮询 (Round Robin)

* 上游类型: 节点

目标节点: * 主机名: 172.24.4.212 端口: 6080 * 权重: 1

* 主机名: 172.24.4.213 端口: 6080 * 权重: 1

+ 新建

Host 请求头: 保持与客户端请求一致的主机名

重试次数 ①:

重试超时时间 ①:

第二步：在管理控制台上上传 aas.alb.com SSL 证书

进入管理控制台证书 TAB 上传 aas.alb.com 的 SSL 证书

Apusic 负载均衡器

管理控制台

路由

上游

服务

Ngix配置

插件

证书

系统信息

创建证书

1 完善证书信息

2 预览

* 方式: 上传

上传证书

@ aas_ssl.crt

上传私钥

@ aas_ssl.key

上传证书和key



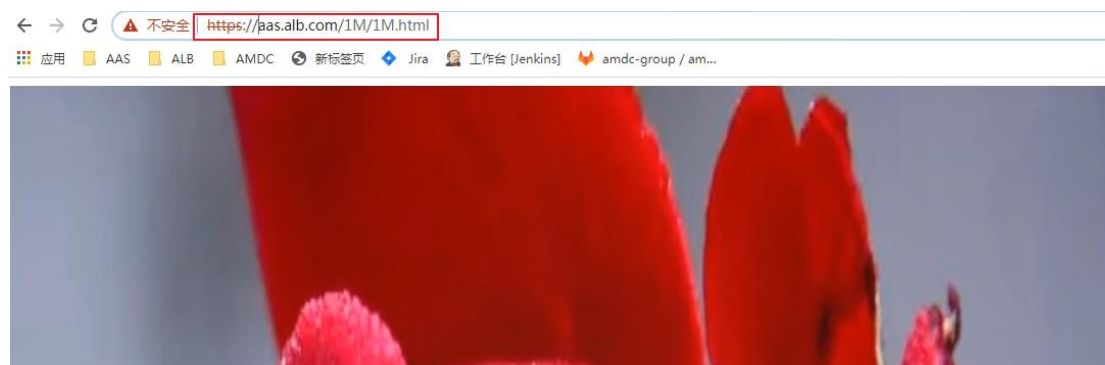
第三步：在客户端电脑添加 host 域名映射

客户端电脑(windows 电脑)，进入 C:\Windows\System32\drivers\etc\hosts,在文件末尾添加配置内容：172.24.4.213 aas.alb.com

```
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #       102.54.94.97       rhino.acme.com       # source server
17 #       38.25.63.10       x.acme.com           # x client host
18 #
19 # localhost name resolution is handled within DNS itself.
20 #       127.0.0.1         localhost
21 #       ::1               localhost
22 172.24.4.213 aas.alb.com
```

第四步：验证 https 请求是否生效

在浏览器输入地址：https://aas.alb.com/test/index.html，请求正确返回则为成功！



示例 4 请求 URI 改写

以下示例路由 URI 改写，其中上游（后端）业务系统是 http://172.24.4.212:6888/1M/1M.html,

<http://172.24.4.213:6888/1M/1M.html>,

改写到业务系统：172.24.212:6888/10K/10K.html，
172.24.212:6888/10K/10K.html 路径上

第一步：创建路由

将 ALB 部署在 172.24.4.213 服务器上，启动并创建如下路由：

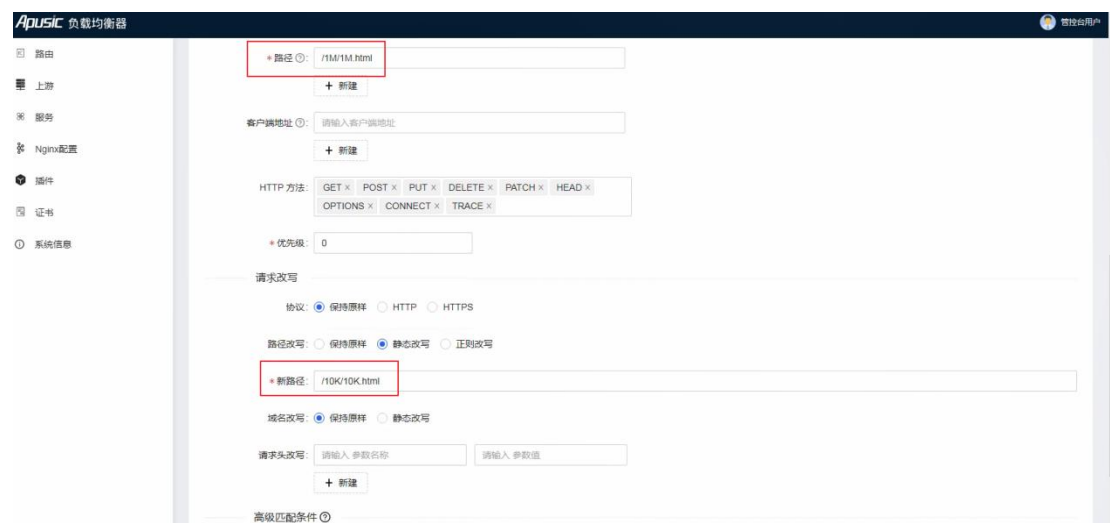
路由名称：test

路由路径：/1M/1M.html

请求改写，静态改写，新路径：//1M.html

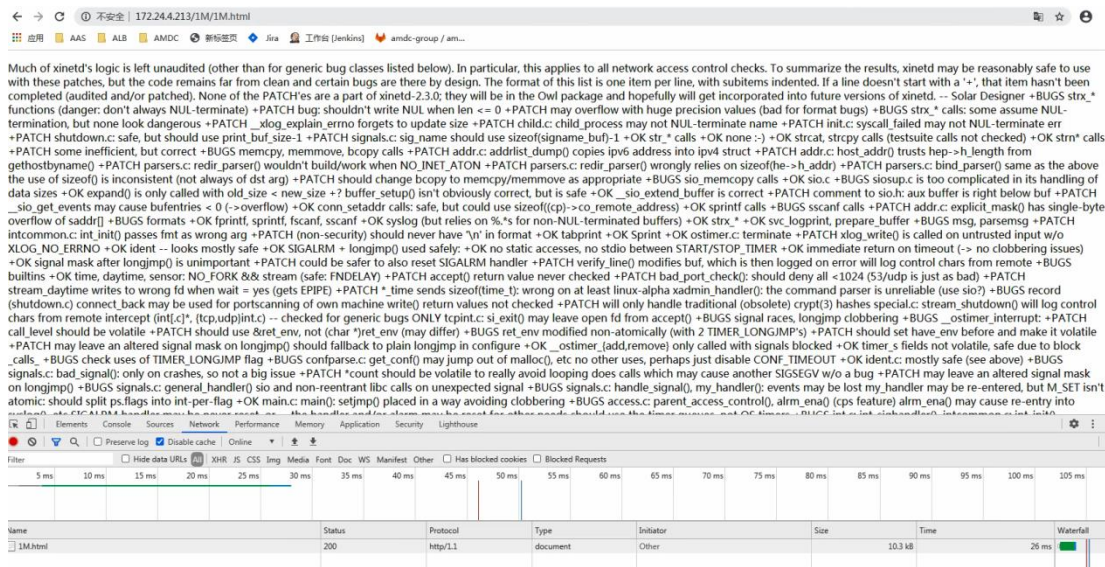
目标节点：172.24.4.212:6888 172.24.4.213: 6888

其他配置保持默认，点击【下一步】，插件页面点击【下一步】，预览页面点击【提交】，直到提示【提交成功】

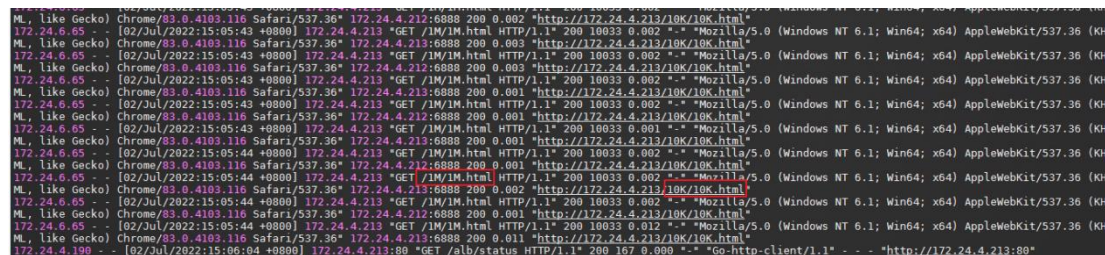




第二步：访问 <http://172.18.100.134:80/1M/1M.html>, 返回了新路径中，改写成功

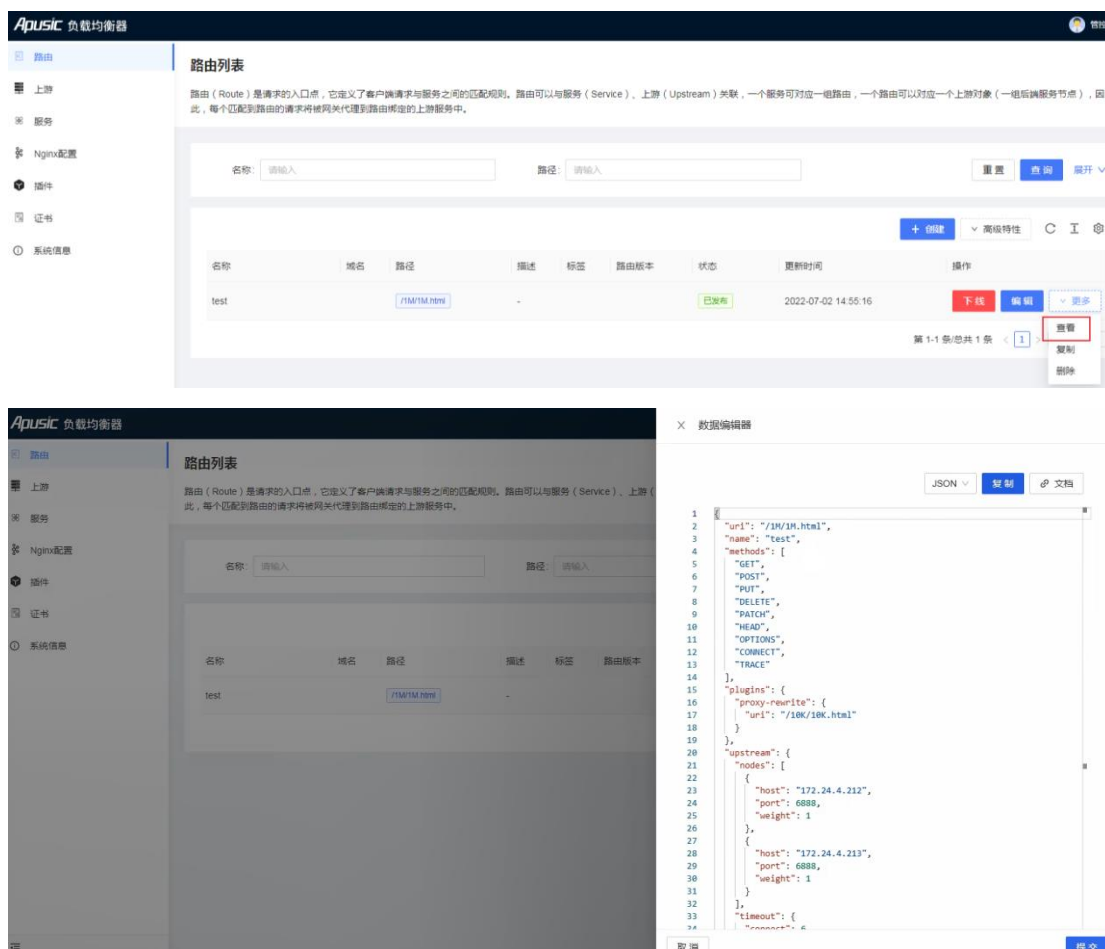


第三步：检查访问日志改写成功：



4.2.2. 查看路由详情

管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮，进入配置负载均衡器页面，点击路由列表页面右侧【查看】按钮，这里提供 JSON 格式或 YAML 格式的路由信息。



4.2.3. 下线、编辑、复制、删除路由

管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮进入配置负载均衡器页面，选择【路由】tab,路由列表右侧提供路由下线、编辑、复制和删除的操作入口

4.2. 上游管理

上游是真实服务的访问入口即后端服务系统, ALB 对给定的多个上游按照配置规则进行负载均衡, 实现对上游目标节点的负载均衡和健康检查。

当我们的 Route 有比较多的上游重复配置 (比如启用相同的插件配置或上游信息) 时, 可以单独配置上游, 不同的路由课可以重复使用相同的上游。

4.3.1. 上游列表:

上游列表包含了已创建的上游服务, 创建路由时可以在上游列表中选择上游,

需要注意的是在路由中添加的上游不可编辑。

管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮，进入配置负载均衡器页面，点击右侧上游 tab 进入上游列表页面。



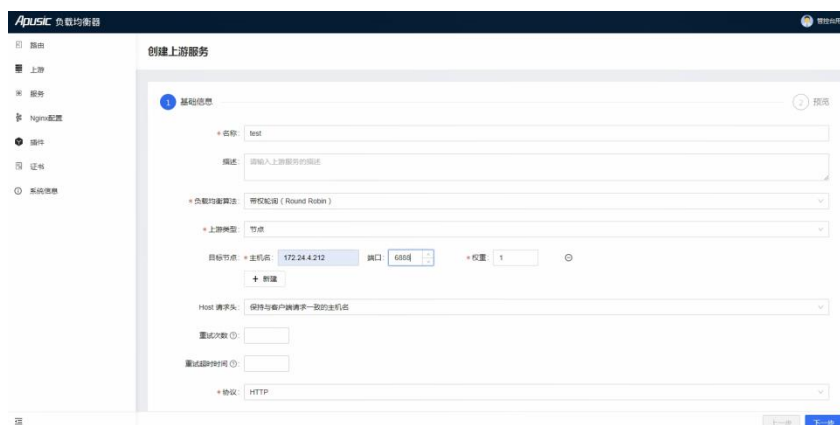
4.3.2. 创建上游

可以通过两种方式创建上游：

- 界面创建上游
- 使用编辑器创建上游

1.使用界面创建上游

管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮进入配置负载均衡器页面，进入【上游】页面，进入上游 Tab 页，点击上游列表中【创建】按钮，即可根据页面提示创建上游



- 上游名称：自定义上游名称，不允许创建相同名称上游
- 负载均衡算法：必选项，包括带权轮询、一致性哈希、指数加权移动平均法、最小连接数，默认带权轮询。
- 上游类型：必选项，默认为节点
- 目标节点：必输项，输入上游目标服务器主机名或者 IP 地址、端口
- 权重：必输项，输入上游服务器的权重，权重高的节点承受的流量多，权重配置为 0 时熔断该节点
- 协议：必选项，选择后端服务器请求的类型选择 http、https、grpc、grpcs
- 超时时间：必输项，默认超时时间为 6S

2.使用编辑器创建上游

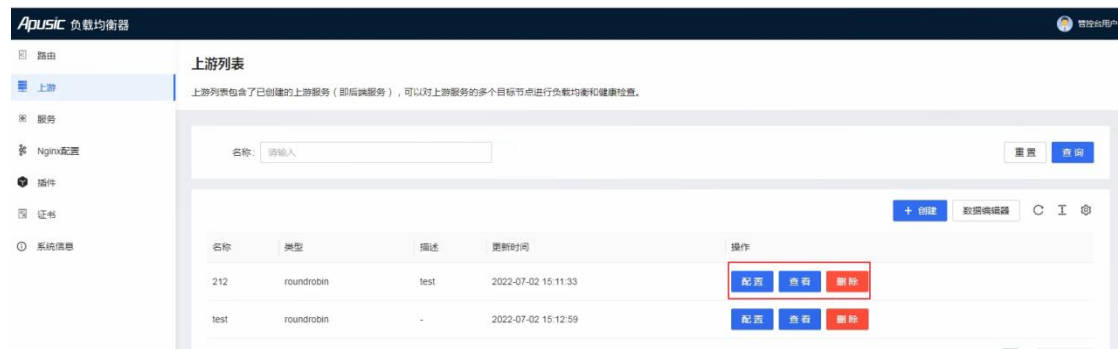
点击上游列表右侧【数据编辑器】入口，在大括号内输入上游信息如上游名称、负载均衡算法、链接超时时间等内容，并提交





4.3.3. 上游的下线、编辑、复制、删除

管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮进入配置负载均衡器页面，进入【上游 tab】上游列表右侧提供上游配置、查看、删除的操作入口



4.3. 服务管理

服务是由路由中公共的配置、插件、上游目标信息组合而成的一组抽象，当我们的 Route 有比较多的重复配置、插件和上游时可以通过同时绑定到一个服务上，减少冗余配置。

路由与服务之间，通常是 N:1 的关系，一个服务可对应一组上游节点、可

被多条路由绑定。

4.4.1. 服务列表

服务列表包含了已创建的服务，管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮，进入配置负载均衡器页面，点击左侧【服务】tab 进入服务页面



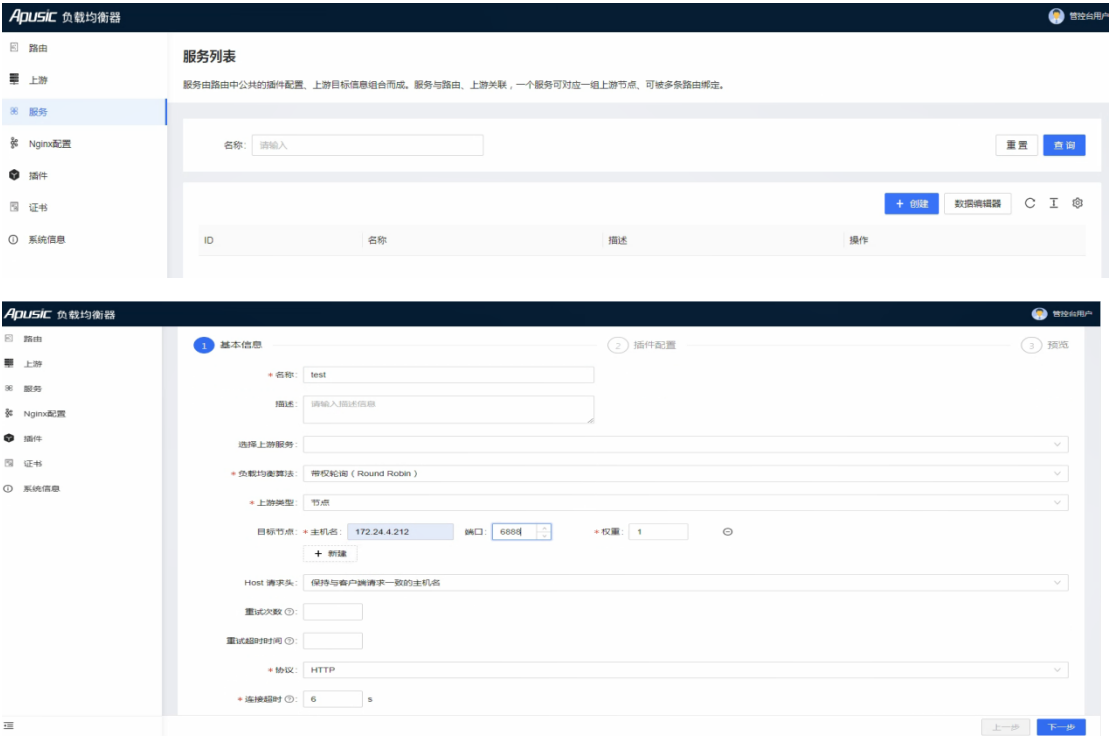
4.4.2. 创建服务

可通过两种方式创建服务。

- 界面创建
- 使用编辑器创建

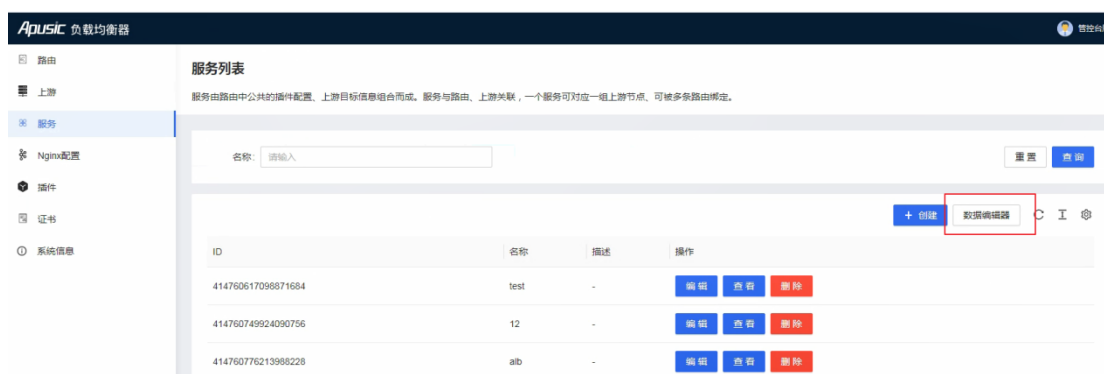
1. 界面创建服务

进入服务 Tab 页，点击服务列表中【创建】，进入图形界面创建服务，根据页面提示进行创建



2. 使用编辑器创建服务

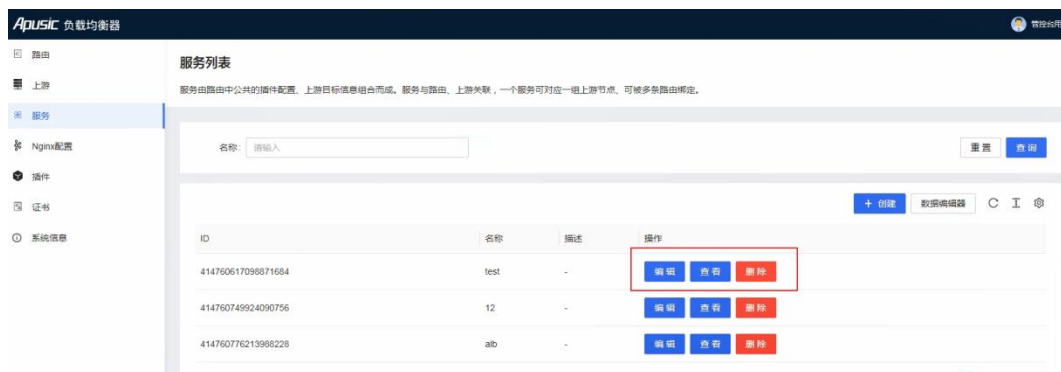
点击服务列表右侧【数据编辑器】，调取数据编辑器进行创建



注意：当 Route 和 Service 都开启同一个插件时，Route 参数的优先级是高于 Service 的。

4.4.3. 服务的下线、编辑、复制、删除

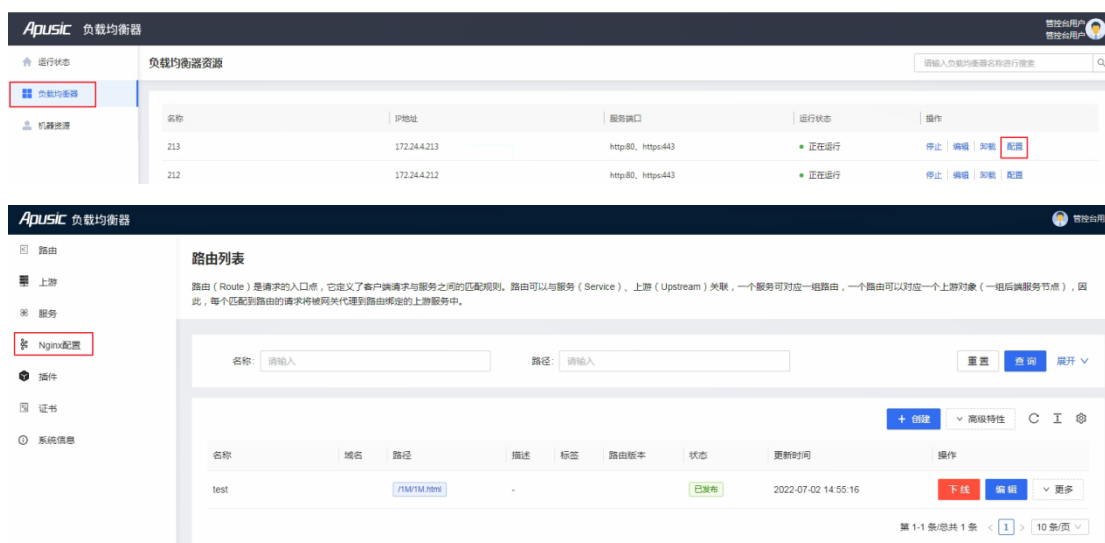
管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮进入配置负载均衡器页面，进入【服务 tab】服务列表右侧提供服务编辑、查看、删除操作入口



4.4. Nginx 配置管理

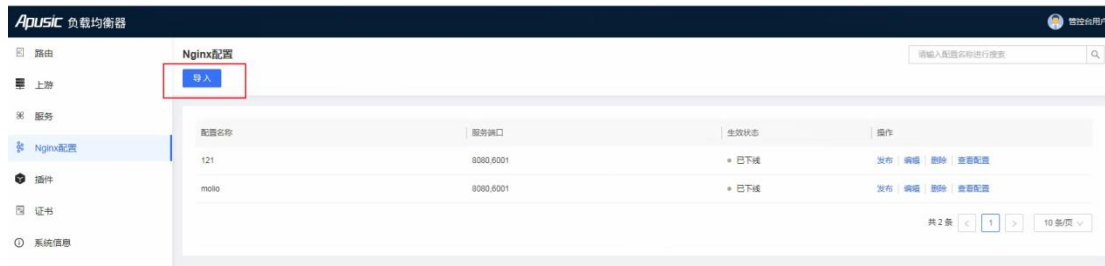
ALB 支持将原生 nginx 的配置导入系统，并复现原有的 Nginx 的能力，同时对导入的 nginx 配置进行监控，提供对 nginx 配置的导入、下线、编辑、删除、查看配置详情操作入口。

管控台用户登录（默认 micc-user/apusic）进入负载均衡器，点击列表右侧【配置】按钮，进入示例管控台，选择 Nginx 配置进入 nginx 配置页面。



4.5.1. 导入 nginx 配置

管控台用户登录进入负载均衡器，点击列表右侧【配置】按钮，进入管控台，选择 Nginx 配置进入 nginx 配置页面，点击【导入】弹出导入 nginx 配置弹窗。



- 配置名称：自定义配置名称
- 配置文件：conf 格式的 nginx 配置文件，文件中的内容是以 server 开头的 nginx 的服务器配置
- 是否生效：只有选择生效才会启用该配置

示例 5 导入 nginx 配置文件并访问

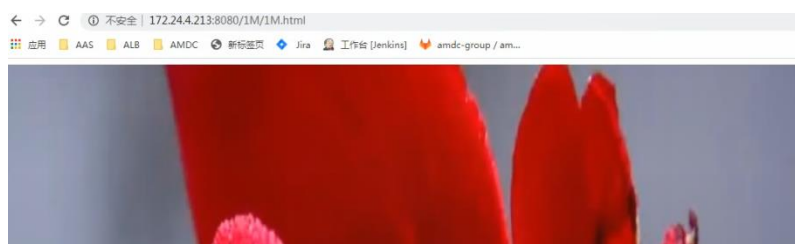
1. 准备 nginx 配置文件 nginx_conf.conf, 文件内容如：

```
upstream tomcat_server{
    server 172.24.4.212:6888;
    server 172.24.4.213:6888;
    keepalive 320;
}

server {
    keepalive_requests 6000;
    listen 8080;
    server_name 172.0.0.1;
    location / {
        proxy_pass http://tomcat_server;
    }
}
```

2. 管控用户登录 MICC>进入负载均衡器>点击【配置】按钮进入示例管控页面>进入 nginx 配置，上传 1 中的 nginx_conf.conf 文件并启用

3. 用浏览器访问 <http://172.24.4.180:8080/1M/1M.html>，检查访问结果



4.5.2. 发布、下线、编辑、删除、查看 nginx 配置

管控台用户登录（默认用户：micc-user/apusic）进入负载均衡器，点击列表右侧【配置】按钮，进入管控台，选择 Nginx 配置进入 nginx 配置页面，nginx 配置列表右侧提供了 nginx 配置发布、下线、编辑、删除和查看配置的入口。



4.5. 插件管理

插件是获取 ALB 特殊体验的应用程序, 作用于 http 请求/响应期间, 可直接绑定在路由上, 也可绑定在服务或者消费者中, 绑定在路由、服务或消费者中的同一个插件在路由中的优先级更高, 其中在插件 tab 中启用的插件是全局插件, 作用于所有的路由和服务。在路由或服务中启用的插件则是局部插件, 作用于局部路由或服务。

ALB 中提供的插件包含内置插件和自定义插件两种

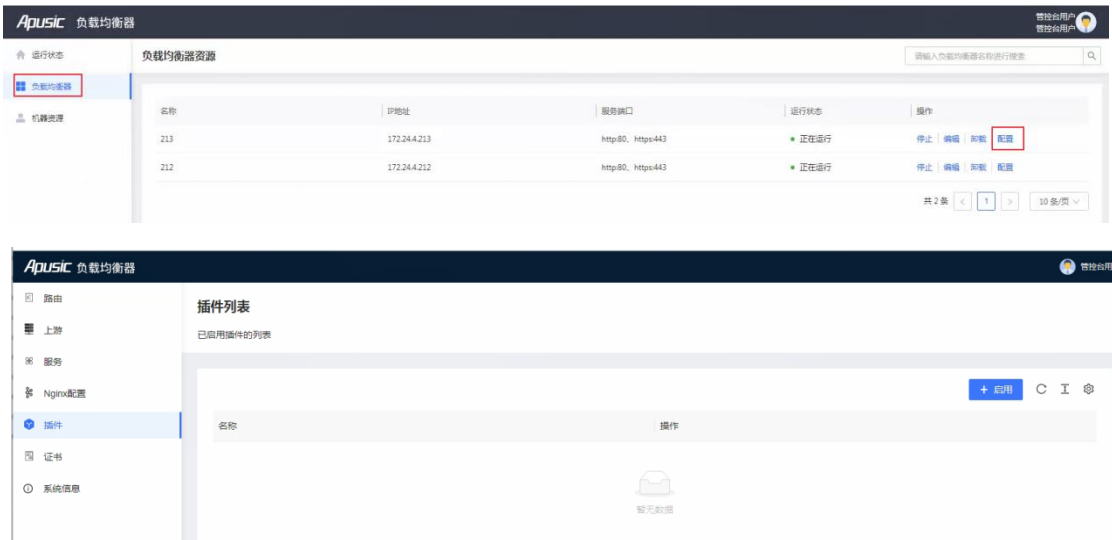
- 内置插件：
 - 安全防护：API 熔断、consumer 访问限制、启用 CORS、故障注入、Referer 限制、ip 黑白名单、Referer 限制、请求验证、请求拦截等
 - 流量控制：限制并发连接、限制请求次数、限制连接数、流量划分等
 - 通用：批量请求、回显、重定向、前附加函数、后附加函数等
 - 可观测性：错误日志、HTTP 日志推送、kafka 日志推送、阿里云日志推送、TCP 日志推送、请求 ID、UDP 日志推送、Zipkin 等
 - 监控：节点状态、prometheus、skywalking 等
 - 协议转换：gRPC、dubbo 代理
 - 报文转换：故障注入、gRPC 转码、响应信息重写、静态重写、相应信息正则替换、User-Agent 黑白名单等

可以根据实际业务编写插件，插件允许在常见阶段进行挂载生效，例如 rewrite, access, header filter, body filter 阶段。

4.6.1. 插件列表

插件列表中包含已启用插件的列表，管控台用户登录 MICC, 进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮，进入配置负载均衡器页面，点击

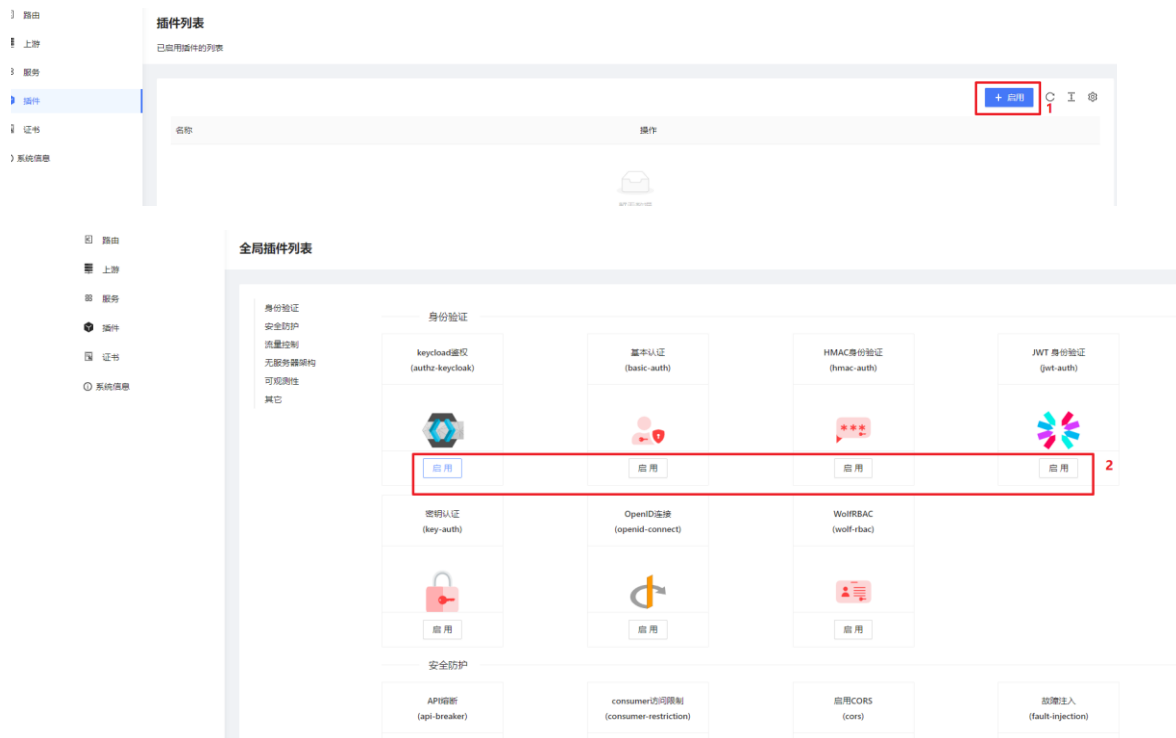
左侧【插件】tab 进入插件页面



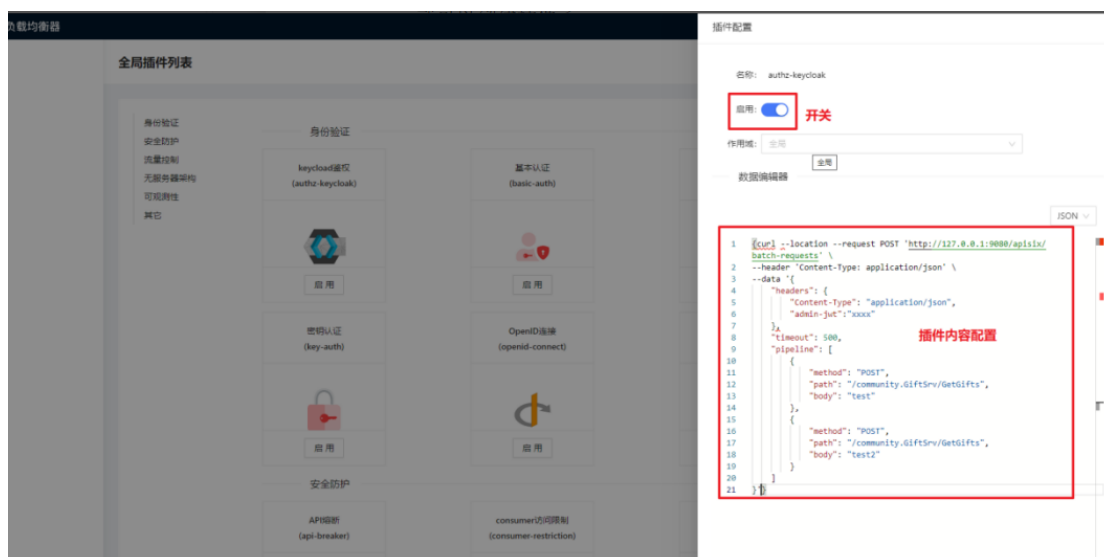
4.6.2. 如何开启插件

在全局插件列表中可以根据需要选择相应插件进行启用

1. 点击插件列表【启用】按钮，进入全局插件列表页面点击【启用】，调起右侧编辑器



2.在编辑器右侧开启【启用】开关，在数据编辑器内编辑插件内容并提交，就可以创建插件了



4.6.3. 编辑插件、删除插件

管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮进入配置负载均衡器页面，进入【插件 tab】插件列表右侧提供插件编辑、删除操作入口



4.6.4. 插件示例

在 ALB 中插件是动态生效的，系统中新增、删除或者修改的插件，不需要重启服务，通过 admin API 发送一个 HTTP 请求即可

示例 6 安全防护插件

ALB 中内置的安全防护插件包括：API 熔断、consumer 访问限制、启用 CORS、故障注入、Referer 限制、ip 黑白名单、Referer 限制、请求验证、请求拦截等，下面以 ip 黑白名单插件的使用进行介绍。

1) ip 黑白名单插件

ip 黑白名单插件，主要作用于将 IP 地址列入白名单或黑名单用于限制对服

务或路线的访问。 只能单独启用白名单或黑名单，用户可自定义 message 内容

单个 IP 地址，多个 IP 地址 或 CIDR 范围，可以使用类似 10.10.10.0/24 的 CIDR 表示法。

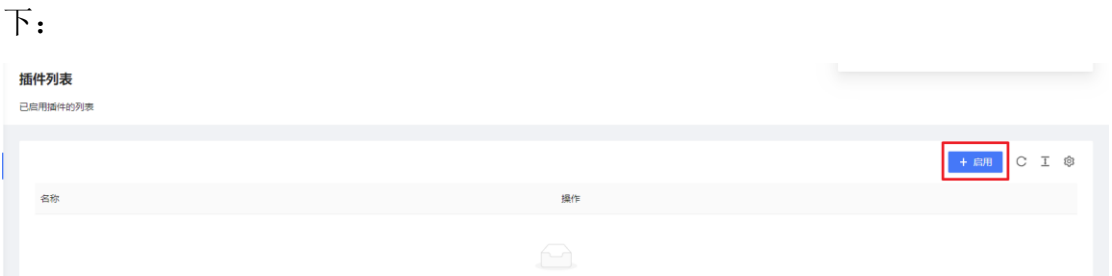
参数表:

参数名	类型	可选项	默认值	有效值	描述
whitelist	array[string]	可选			加入白名单的 IP 地址或 CIDR 范围
blacklist	array[string]	可选			加入黑名单的 IP 地址或 CIDR 范围
message	string	可选	Your IP address is not allowed.	[1, 1024]	在未允许的 IP 访问的情况下返回的信息

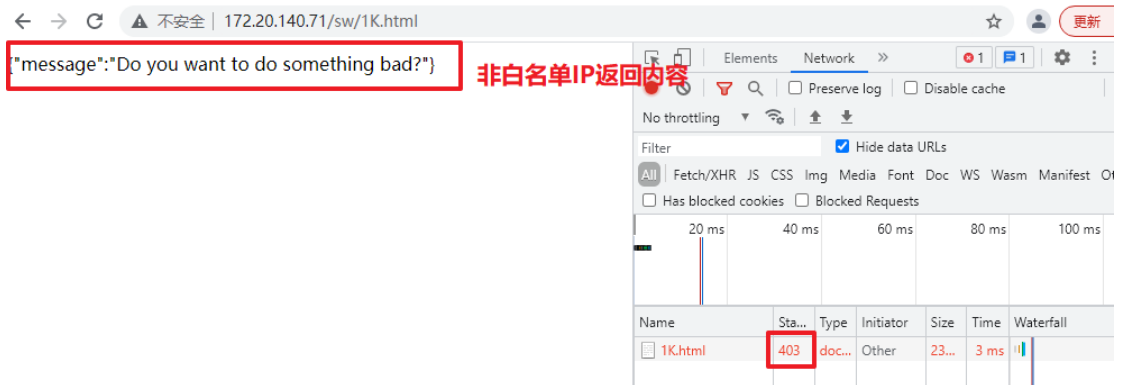
2) 如何启用 ip 黑白名单插件

点击插件列表右上角【启用】按钮进入全局插件列表,在 ip 黑白名单下点击【启用】，在弹出的右侧编辑器页面打开【启用】开关，并输入配置内容

"whitelist": ["127.0.0.1", "113.74.26.106/24"],
"message": "Do you want to do something bad?"并提交，相关操作截图如下：



非白名单用户访问，则返回插件配置中的 message 信息，如下图：



示例 7 流量控制插件

ALB 中内置的流量控制插件有：限制并发连接、限制请求次数、限制请求速度、流量划分等，下面以限制请求次数插件的使用进行介绍。

1) 制请求次数插件

限制请求次数指在指定的时间内，限制总的请求个数。并且在 HTTP 响应头中返回剩余可以请求的个数。

参数表：

名称	类型	必选项	默认值	有效值	描述
count	integer	必须		count > 0	指定时间窗口内的请求数量阈值
time_window	integer	必须		time_window > 0	时间窗口的大小（以秒为单位），超过这个时间就会重置
key	string	可选	"remote_addr"	["remote_addr", "server_addr", "http_x_real_ip", "http_x_forwarded_for", "consumer_name", "service_id"]	用来做请求计数的有效值。 例如，可以使用主机名（或服务器区域）作为关键字，以便限制每个主机名规定时间内的请求次数。我们也可以使用客户端地址作为关键字，这样我们就可以避免单个客户端规定时间内多次的连接我们的服务。 当前接受的 key 有："remote_addr"（客户端 IP

名称	类型	必选项	默认值	有效值	描述
					地址), "server_addr" (服务端 IP 地址), 请求头中的 "X-Forwarded-For" 或 "X-Real-IP", "consumer_name" (consumer 的 username), "service_id" 。
rejected_code	integer	可选	503	[200,...,599]	当请求超过阈值被拒绝时, 返回的 HTTP 状态码
policy	string	可选	"local"	["local", "redis", "redis-cluster"]	用于检索和增加限制的速率限制策略。可选的值有: local(计数器被以内存方式保存在节点本地, 默认选项) 和 redis(计数器保存在 Redis 服务节点上, 从而可以跨节点共享结果, 通常用它来完成全局限速); 以及 redis-cluster, 跟 redis 功能一样, 只是使用 redis 集群方式。
redis_host	string	redis 必须			当使用 redis 限速策略时, 该属性是 Redis 服务节点的地址。
redis_port	integer	可选	6379	[1,...]	当使用 redis 限速策略时, 该属性是 Redis 服务节点的端口
redis_password	string	可选			当使用 redis 限速策略时, 该属性是 Redis 服务节点的密码。
redis_database	integer	可选	0	redis_database >= 0	当使用 redis 限速策略时, 该属性是 Redis 服务节点中使用的 database, 并且只针对非 Redis 集群模式 (单实例模式或者提供单入口的 Redis 公有云服务) 生效。
redis_timeout	integer	可选	1000	[1,...]	当使用 redis 限速策略时, 该属性是 Redis 服务节点以毫秒为

名称	类型	必选项	默认值	有效值	描述
					单位的超时时间
redis_cluster_nodes	array	当 policy 为 redis-cluster 时必填			当使用 redis-cluster 限速策略时，该属性是 Redis 集群服务节点的地址列表（至少需要两个地址）。
redis_cluster_name	string	当 policy 为 redis-cluster 时必填			当使用 redis-cluster 限速策略时，该属性是 Redis 集群服务节点的名称。

2) 启用限制请求次数插件

启用限制请求次数插件，第一步，需要在插件列表中选择流量控制 tab,在限制请求次数插件下点击【启用】。第二步，需要在调出的编辑页面开启【启用】开关，并输入相应的参数，然后提交

插件配置

名称: limit-count

启用: ☒

作用域: 全局

数据编辑器

表单

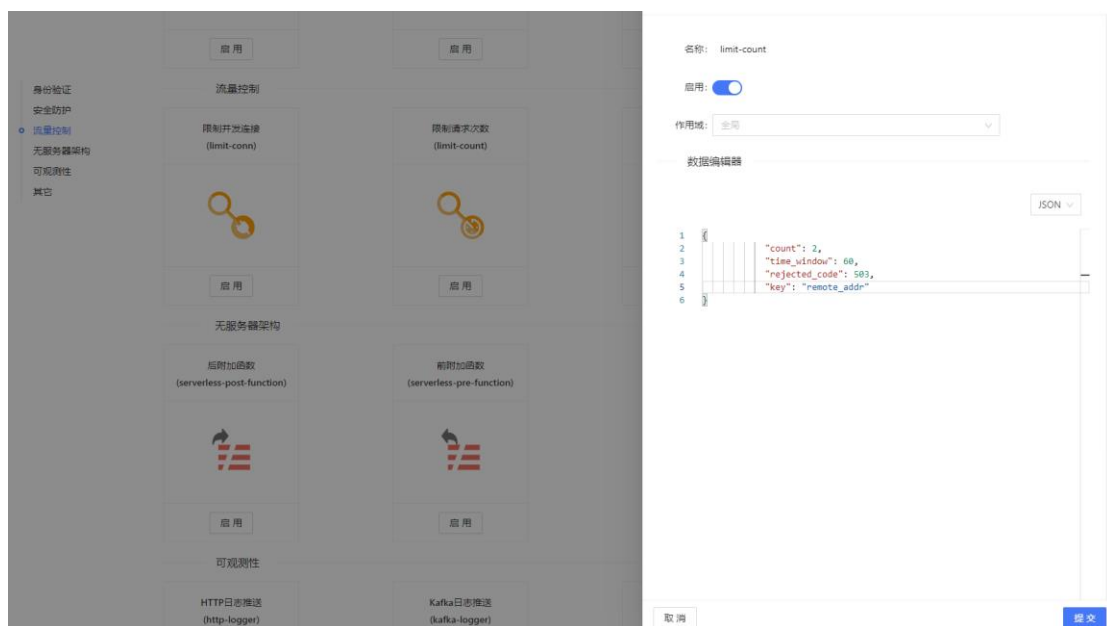
* count: 2

* time_window: 60

key: remote_addr

rejected_code: 503

policy: local



3) 限制请求次数示例

启用限制请求次数插件后，根据上面定义的限制请求次数 2，超出 2 次后则返回 503 如下图所示：

GET

http://172.20.140.71/sw/1K.html

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Headers

6 hidden

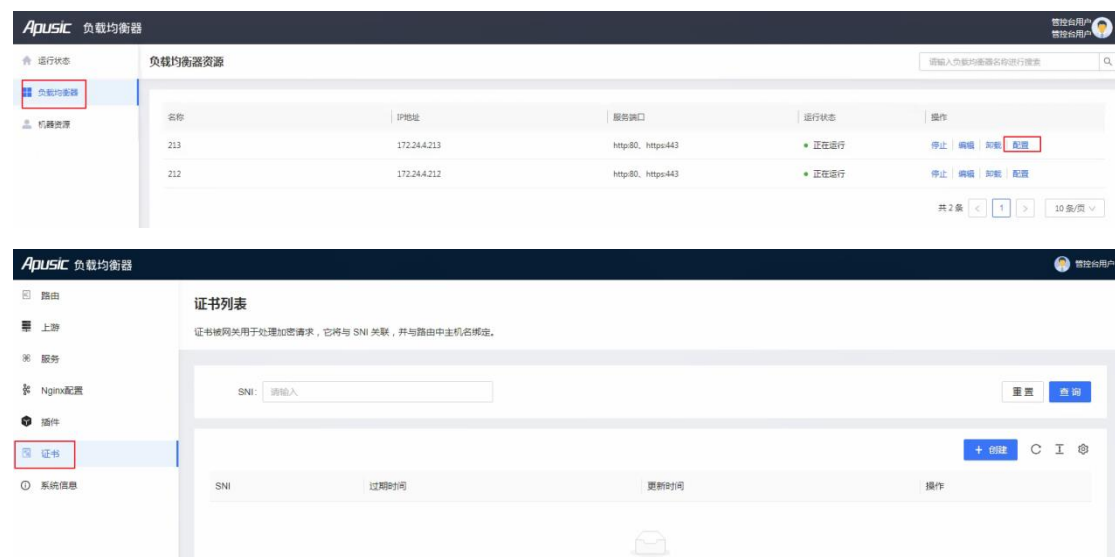
KEY	VALUE
Content-Type	text/html; charset=utf-8
Content-Length	1024
Connection	keep-alive
X-RateLimit-Limit	60
X-RateLimit-Remaining	58
Accept-Ranges	bytes
ETag	W/"1024-1637661872978"
Last-Modified	Tue, 23 Nov 2021 10:04:32 GMT
Server	APISIX/2.8

4.6. 证书管理

证书被网关用于处理加密请求，它与 SNI 关联，并与路由中主机名绑定，如 https 请求 SSL 证书。

4.7.1. 证书列表

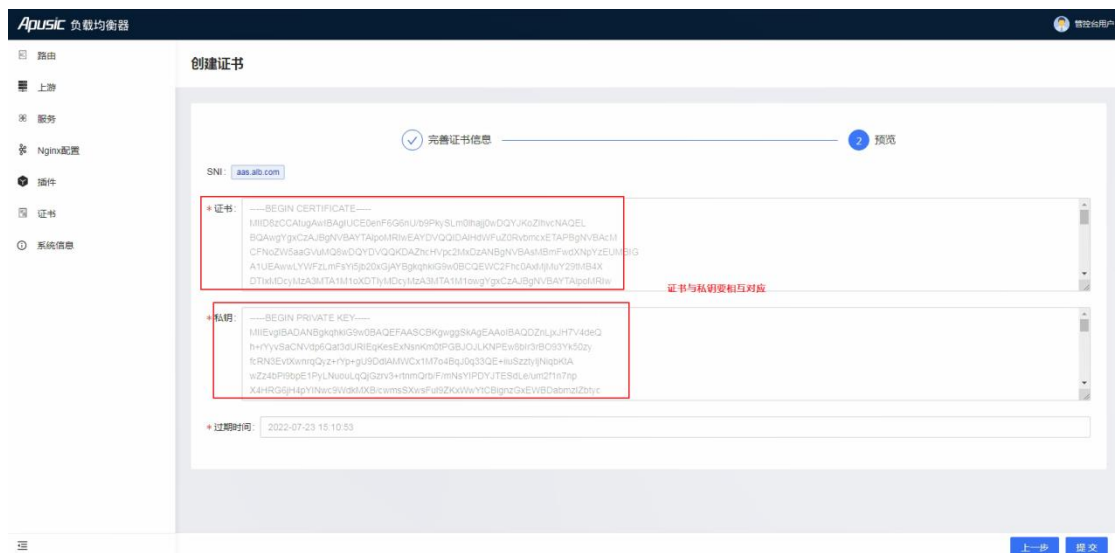
管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮，进入配置负载均衡器页面点击【证书】TAB 进入证书页面。



4.7.2. 创建证书

在证书页面点击【创建】按钮，可以输入证书或导入证书，需要注意的是证书和私钥需相互匹配





4.7. 系统信息

管控台用户登录 MICC,进入【负载均衡器】，点击负载均衡器列表右侧【配置】按钮，点击管理控制台系统信息 TAB 进入系统信息页面，这里展示控制台信息、license 信息以及 ALB 节点信息。



第5章 静态资源代理、PHP、TCP 代理

静态资源代理、PHP 代理、TCP 代理可由 Nginx 配置文件实现，请根据需要，编写 Nginx 配置文件，由 Nginx 配置管理模块实现。

第6章 REST API 接口

ALB 支持通过 REST API 对路由、上游、插件、服务进行管理
其中 REST ful api 接口的地址为单个的 ALB 实例，如 alb 部署在 192.168.1.100 机器中，使用 REST ful api 请求的地址是： http://192.168.1.100:9080

6. 1. REST API 请求的方法和参数

1.请求的方法

名字	请求 uri	请求 body	说明
GET	/alb/admin/routes	无	获取资源列表
GET	/alb/admin/routes/{id}	无	获取资源
PUT	/alb/admin/routes/{id}	{...}	根据 id 创建资源
POST	/alb/admin/routes	{...}	创建资源, id 由后台服务自动生成
DELETE	/alb/admin/routes/{id}	无	删除资源
PATCH	/alb/admin/routes/{id}	{...}	标准 PATCH ， 修改已有 Route 的部分属性，其他不涉及的属性会原样保留；如果你要删除某个属性，将该属性的值设置为 null 即可删除；特别地，当需要修改属性的值为数组时，该属性将全量更新
PATCH	/alb/admin/routes/{id}/{path}	{...}	SubPath PATCH ， 通过 {path} 指定 Route 要更新的属性，全量更新该属性的数据，其他不涉及的属性会原样保留。两种 PATCH 的区别可以参考后面的示例

2.URL 请求参数

名字	可选项	类型	说明	示例
ttl	可选	辅助	超过这个时间会被自动删除，单位：秒	ttl=1

3.body 请求参数

名字	可选项	类型	说明	示例
uri	必选，不能与 uris 一起使用	匹配规则	除了如 /foo/bar、/foo/gloo 这种全量匹配外，使用不同 Router 还允许更高级匹配，更多见 Router。	"/hello"
uris	必选，不能与 uri 一起使用	匹配规则	非空数组形式，可以匹配多个 uri	["/hello", "/world"]
plugins	可选	Plugin	详见 Plugin	
script	可选	Script	详见 Script	
upstream	可选	Upstream	启用的 Upstream 配置，详见 Upstream	
upstream_id	可选	Upstream	启用的 upstream id，详见 Upstream	
service_id	可选	Service	绑定的 Service 配置，详见 Service	
plugin_config_id	可选，无法跟 script 一起配置	Plugin	绑定的 Plugin config 配置，详见 Plugin config	
name	可选	辅助	标识路由名称	route-xxxx
desc	可选	辅助	标识描述、使用场景等。	路由 xxxx
host	可选，不能与 hosts 一起使用	匹配规则	当前请求域名，比如 foo.com；也支持泛域名，比如 *.foo.com。	"foo.com"
hosts	可选，不能与 host 一起使用	匹配规则	非空列表形态的 host，表示允许有多个不同 host，匹配其中任意一个即可。	["foo.com", "*.bar.com"]
remote_addr	可选，不能与 remote_addrs 一起使用	匹配规则	客户端请求 IP 地址：192.168.1.101、192.168.1.102 以及 CIDR 格式的支持 192.168.1.0/24。特别的，APISIX 也完整支持 IPv6 地址匹配：::1，fe80::1，fe80::1/64 等。	"192.168.1.0/24"
remote_addrs	可选，不能与 remote_addr 一起使用	匹配规则	非空列表形态的 remote_addr，表示允许有多个不同 IP 地址，符合其中任意一个即可。	["127.0.0.1", "192.0.0.0/8", "::1"]

名字	可选项	类型	说明	示例
methods	可选	匹配规则	如果为空或没有该选项，代表没有任何 method 限制，也可以是一个或多个的组合：GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS, CONNECT, TRACE, PURGE。	["GET", "POST"]
priority	可选	匹配规则	如果不同路由包含相同 uri，根据属性 priority 确定哪个 route 被优先匹配，值越大优先级越高，默认值为 0。	priority = 10
vars	可选	匹配规则	由一个或多个 [var, operator, val] 元素组成的列表，类似这样：[[var, operator, val], [var, operator, val], ...]]。例如：["arg_name", "=", "json"]，表示当前请求参数 name 是 json。这里的 var 与 Nginx 内部自身变量命名是保持一致，所以也可以使用 request_uri、host 等。更多细节请参考 lua-resty-expr	[[{"arg_name", "=", "json"}, {"arg_name", ">", 18}]
filter_func	可选	匹配规则	用户自定义的过滤函数。可以使用它来实现特殊场景的匹配要求实现。该函数默认接受一个名为 vars 的输入参数，可以用它来获取 Nginx 变量。	function(vars) return vars["arg_name"] == "json" end
labels	可选	匹配规则	标识附加属性的键值对	{"version": "v2", "build": "16", "env": "production"}
timeout	可选	辅助	为 route 设置 upstream 的连接、发送消息、接收消息的超时时间（单位为秒）。这个配置将会覆盖在 upstream 中配置的 timeout 选项	{"connect": 3, "send": 3, "read": 3}
enable_websocket	可选	辅助	是否启用 websocket (boolean)，缺省 false。	
status	可选	辅助	是否启用此路由，缺省 1。	1 表示启用, 0 表示禁用
create_time	可选	辅助	单位为秒的 epoch 时间戳，如果不指定则自动创建	1602883670
update_time	可选	辅助	单位为秒的 epoch 时间戳，如果不指定则自动创建	1602883670

6.2. REST API 示例

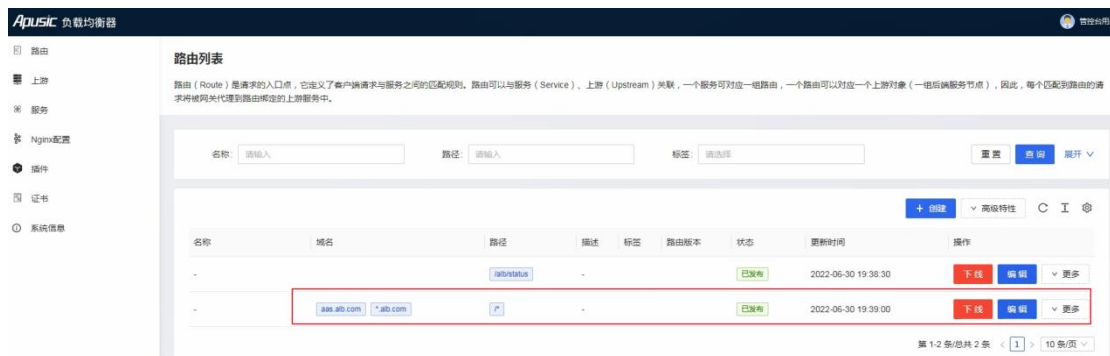
示例 12: 创建一个路由

root 权限用户登录 ALB 部署服务器输入 curl 命令:

```
curl http://127.0.0.1:80/alb/admin/routes/1 -H 'X-API-KEY: edd1c9f034335f136f87ad84b625c8f1' -X PUT -i -d '{ "uri": "/", "hosts": ["aas.alb.com", "*.alb.com"], "methods": ["POST", "GET"], "enable_websocket": true, "upstream": { "type": "roundrobin", "nodes": { "172.24.4.212:6888": 1 } } }'
```

```
[root@linux-4-190 ~]# curl http://127.0.0.1:80/alb/admin/routes/1 -H 'X-API-KEY: edd1c9f034335f136f87ad84b625c8f1' -X PUT -i -d '{
> {
>   "uri": "/",
>   "hosts": ["aas.alb.com", "*.alb.com"],
>   "methods": ["POST", "GET"],
>   "enable_websocket": true,
>   "upstream": {
>     "type": "roundrobin",
>     "nodes": {
>       "172.24.4.212:6888": 1
>     }
>   }
> }'
HTTP/1.1 201 Created
Date: Thu, 30 Jun 2022 11:39:00 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
Server: ALB/2.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: *
Access-Control-Max-Age: 3600

{"node":{"key":"/alb/routes/46/v1","value":{"priority":0,"uri":"/","status":1,"id":1,"create_time":1656589140,"methods":["POST","GET"],"hosts":["aas.alb.com","*.alb.com"],"update_time":1656589140,"upstream":{"scheme":"http","pass_host":"pass","nodes":{"172.24.4.212:6888":1},"hash_on":"vars","type":"roundrobin"},"enable_websocket":true},"action":"set"}}
```



示例 13: 创建一个上游

```
curl http://127.0.0.1:80/alb/admin/upstreams/100 -H 'X-API-KEY: edd1c9f034335f136f87ad84b625c8f1' -i -X PUT -d '{
{
  "type": "roundrobin",
  "nodes": {
    "172.24.4.212:6888": 1
  }
}'
```



```
[root@linux-4-190 ~]# curl http://127.0.0.1:80/alb/admin/upstreams/100 -H 'X-API-KEY: eddic9f034335f136f87ad84b625c8f1' -i -X PUT -d '{
>   "type": "roundrobin",
>   "nodes": {
>     "172.24.4.212:6888": 1
>   }
> }'
HTTP/1.1 201 Created
Date: Thu, 30 Jun 2022 11:44:08 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
Server: ALB/2.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: *
Access-Control-Max-Age: 3600

{"node":{"key":"\\alb\\upstreams\\46\\100","value":{"update_time":1656589448,"id":"100","hash_on":"vars","scheme":"http","pass_host":"pass","nodes":{"172.24.4.212:6888":1},"create_time":1656589448,"type":"roundrobin"},"action":"set"}
```



示例 14：创建一个服务

```
curl http://127.0.0.1:80/alb/admin/services/201 -H 'X-API-KEY: edd1c9f034335f136f87ad84b625c8f1' -X PUT -i -d '{
```

```
{
    "plugins": {
        "limit-count": {
            "count": 2,
            "time_window": 60,
            "rejected_code": 503,
            "key": "remote_addr"
        }
    },
    "enable_websocket": true,
    "upstream": {
        "type": "roundrobin",
        "nodes": {
            "172.24.4.212:6888": 1
        }
    }
}
```

