Apusic 金蝶天燕

生成时间：2026-05-19

用户手册-企业版

金蝶Apusic负载均衡器

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 前言

本文档为金蝶Apusic负载均衡器软件v2.0.5企业版的使用说明，为用户介绍金蝶Apusic负载均衡器产品，帮助用户快速上手。

2 面向对象

本用户手册主要面向对象为适用金蝶Apusic负载均衡器的开发人员，以及相关的管理人员和运维人员。

3 版本更新说明

本文根据实际情况进行更新，最新版本包含历史修改记录。

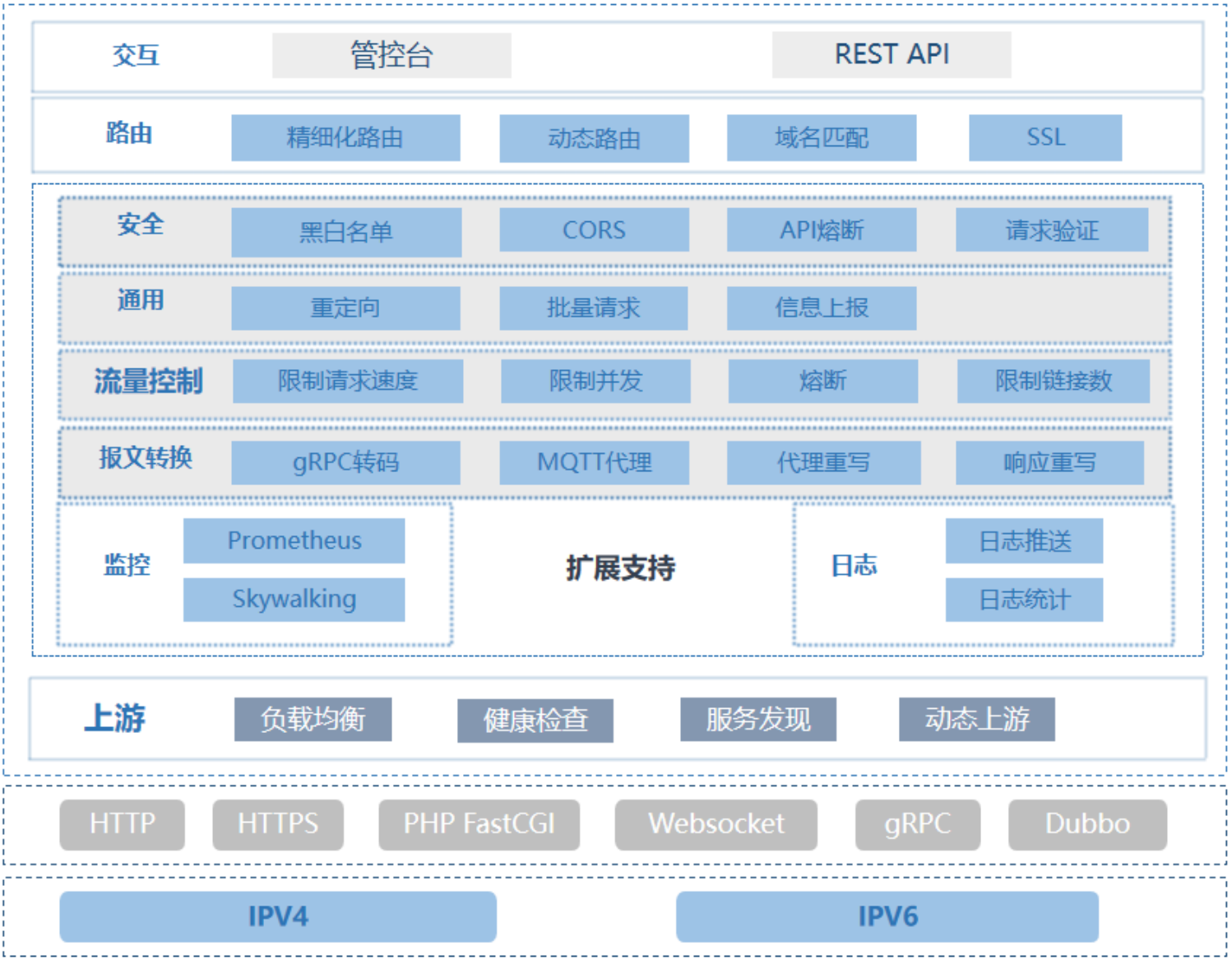
日期	手册版本	适用产品	更新说明
2024年12月	V2.0.2	ALB V2.0.2 企业版	修改ALB高可用和授权认证使用文档
2025年12月	V2.0.5	ALB V2.0.5 企业版	优化授权使用说明文档

4 用户手册

4.1 基本介绍

金蝶Apusic负载均衡软件企业版（Apusic Load Balance，ALB）是一款具备高性能、高可用性和可扩展性的流量治理软件。ALB能够应对大规模的集群、云平台在面向客户端提供服务时，对客户端访问请求和流量管理的需求，实现访问请求的验证、处理、转换和分发等操作，从而隔离客户端访问对提供服务的应用系统、平台以及资源的直接影响，达到对服务集群访问流量控制、访问管理和负载均衡的目的。

4.2 产品功能架构



5 产品安装

5.1 安装说明

相关资源

针对不同的操作系统及CPU架构平台，金蝶Apusic负载均衡软件提供不同的安装包。更多产品介质相关信息，可以访问金蝶天燕官方网站http://www.apusic.com获取。

基本概念

在正确使用应用服务器来部署、管理应用之前，需要先理解以下几个基本概念：

ALB：Apusic Load Balance，金蝶Apusic负载均衡软件。

- 安装与使用
- 安装前准备
- 获取安装包

> *从http://www.apusic.com/下载金蝶Apusic负载均衡软件v2.0安装包，或从金蝶Apusic负载均衡软件产品光盘中获得相应的安装包文件。*

支持的环境

平台类型	系统类型
芯片类型	鲲鹏、飞腾、兆芯等通用x86或arm架构cpu
国产操作系统	OpenEuler、统信UOS、银河麒麟系列、深度等
其他Linux系列	RedHat系列、CentOS、Suse Linux系列等

5.2 安装包说明

ALB企业版产品安装包名称结构是：

`ALB-版本号-EE-构建日期-CPU架构.tar.gz`，

目前仅支持linux平台下arm64和amd64架构:

- Linux平台arm64架构安装包： `ALB-V2.0.5-EE-构建日期-arm64.tar.gz`
- Linux平台amd64(x86_64)架构安装包: `ALB-V2.0.5-EE-构建日期-amd64.tar.gz`

注：构建日期为ALB产品包具体构建日期，同一个版本可能存在多个构建日期，获取安装包以最新日期为准。本手册全部演示内容中的安装包名称移除了构建日期。

5.2.1 安装介质

- tar.gz压缩包：`ALB-版本号-EE-构建日期-CPU架构.tar.gz` 解压到目标机器任意路径即可使用。
- RPM包：`ALB-版本号-EE-构建日期-CPU架构.rpm`,使用 `rpm -ivh 安装包名` 安装，安装完成后，其安装路径为 `/opt/ALB-EE-版本号`
- docker镜像：`ALB-版本号-EE-构建日期-CPU架构-docker.tar.gz` 或 或 `ALB-版本号-EE-Docker-构建日期-CPU架构-基础操作系统.tar.gz` 。

5.3 ALB安装部署

下面以ARM架构下Kylin系统上安装ALB为例，下载好安装文件ALB-V2.0.5-EE-arm64.tar.gz后，其安装步骤有：

1. 解压安装包
2. 导入license
3. 启动ALB

5.3.1 解压安装包到指定目录

1. 上传alb安装包至安装服务器的任意安装目录，（推荐/opt目录）。
2. 解压安装包：`tar -zvxf ALB-V2.0.5-EE-arm64.tar.gz` 获得ALB-V2.0.5-EE-arm64文件夹。
3. 进入解压后的文件夹: `cd ALB-V2.0.5-EE-arm64` 。

备注：下面演示的安装路径为/opt/目录（支持安装在任意目录）。

5.3.2 导入license

1. 将金蝶KBC授权或本地授权，把**授权文件放置在 【安装目录】/ 文件夹**下，如下：

```
【安装目录】tree -L 1
.
├── VERSION
├── acls.properties   # 授权中心配置文件
├── alb-dashboard
├── alb-deps
├── alb-standard
├── bin
├── license.xml       # 授权文件
└── utils
```

5.3.3 启动、停止、重新加载

- 启动： `./bin/start-alb.sh`
- 停止： `./bin/stop-alb.sh`
- 重新加载： `./bin/reload.sh`

5.4 Docker部署

alb企业版提供arm和x86的Docker系统安装包，可通过ALB Docker安装包直接部署ALB.

5.4.1 安装包说明

1. ALB压缩安装包名称

- `ALB-V2.0.5-EE-docker-amd64.tar.gz` 或 `ALB-V2.0.5-EE-Docker-20250227-amd64-kylin.tar.gz`
- `ALB-V2.0.5-EE-docker-arm64.tar.gz` 或 `ALB-V2.0.5-EE-Docker-20250227-arm64-kylin.tar.gz`

注： 安装包日期可能不同，请根据实际情况选择。

2. 安装包目录结构

```
tree -l 1 ALB-V2.0.5-EE-Docker-20250227-amd64-kylin
ALB-V2.0.5-EE-Docker-20250227-amd64-kylin            # docker安装包解压后目录
|-- ALB-V2.0.5-EE-Docker-20250227-amd64-kylin.tar.gz # docker镜像文件
|-- alb                                              # ALB启动配置文件、授权文件、日志等挂载目录
|   |-- alb.conf                                     # ALB配置文件
|   |-- license.xml                                  # 授权文件
-- alb-standard-docker-compose.yaml                  # docker compose配置文件
```

5.4.2 安装步骤

下面的操作以ALB-V2.0.5-EE-Docker-20250227-amd64-kylin.tar.gz为例

1. 解压和导入镜像

- 解压 `tar -zxvf ALB-V2.0.5-EE-Docker-20250227-amd64-kylin.tar.gz`
- 进入解压后的文件夹 `cd ALB-V2.0.5-EE-Docker-20250227-amd64-kylin`

- 加载镜像： `docker load < ALB-V2.0.5-EE-Docker-20250227-amd64-kylin.tar.gz`

2. 修改授权文件

ALB授权文件需要挂载到容器内部安装目录，当前样例是`alb/license.xml`，在启动容器前，请更换授权文件。

> *授权文件挂载说明：*
>
> - *license.xml支持旧授权*
> - *license.xml支持KBC授权（如果是KBC授权，把kbc授权内容复制写入license.xml即可）*
> - *acls.properties统一授权配置文件需要添加挂载到容器路径：* `/opt/ALB-V2.0.5-EE/alb-standard`

3. 镜像启动与停止

- 启动命令： `docker-compose -f alb-standard-docker-compose.yaml up -d`
- 停止命令： `docker-compose -f alb-standard-docker-compose.yaml down`

4. `alb-standard-docker-compose.yaml` 配置说明

```yaml
version: '3'
services:
  alb:
    image: harbor.apusic.com/apusic/alb:V2.0.5-ee.20250227-kylin-amd64  # ALB镜像名称
    ports:
      - 8080:80                # ALB企业版动态代理端口映射
      - 9443:443               # ALB企业版动态代理端口映射
      - 9000:9000              # AlB企业版管理控制台端口映射
    volumes:
      - ./alb/config.yaml:/opt/ALB-V2.0.5-EE/alb-standard/conf/config.yaml     # ALB 企业版配置文件映射
      - ./alb/license.xml:/opt/ALB-V2.0.5-EE/license.xml
# ALB 企业版授权文件映射
      - ./alb/logs:/opt/ALB-V2.0.5-EE/alb-standard/logs
# ALB 企业版日志文件映射
    command: bash /opt/ALB-V2.0.5-EE/bin/start-alb.sh
# 启动命令
```

5.5 K8s部署

alb企业版支持通过Kubernetes进行部署，以下是详细的部署步骤。

5.5.1 准备工作

- 环境要求
  - 已安装 kubectl 并配置好 Kubernetes 集群访问权限。
  - 若使用私有镜像仓库（如Harbor），确保集群有权限拉取镜像。
- 镜像准备
  - 将 Docker 镜像上传至镜像仓库（以 AMD64 镜像为例）

```
# 解压安装包
tar -zxvf ALB-V2.0.5-EE-Docker-20250227-amd64-kylin.tar.gz
# 切换到解压后的安装包
cd ALB-V2.0.5-EE-Docker-20250227-amd64-kylin
# 导入ALB镜像
docker load < ALB-V2.0.5-EE-Docker-20250227-amd64-kylin.tar.gz
# 重新标记镜像，请根据自己镜像仓库地址进行修改
docker tag [镜像ID] harbor.apusic.com/apusic/alb:V2.0.5-
se.20250227-kylin-amd64
# 推送镜像至镜像仓库
docker push [镜像ID]
```

5.5.2 配置文件与授权管理

- 创建 ConfigMap 将 `config.yaml` 配置文件挂载至容器：

```
kubectl create configmap alb-config --from-file=./alb/config.yaml
```

- 创建 Secret 将 `license.xml` 授权文件以 Secret 形式存储（敏感信息建议使用 Secret）：

```
kubectl create secret generic alb-license --from-
file=./alb/license.xml
```

5.5.3 部署ALB服务

1. 创建 Deployment 编写 alb-deployment.yaml 文件：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: alb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: alb
  template:
    metadata:
      labels:
        app: alb
    spec:
      containers:
      - name: alb
        image: harbor.apusic.com/apusic/alb:V2.0.5-se.20250227-
kylin-amd64 # 请根据仓库地址进行修改
        command: ["bash", "/opt/ALB-V2.0.5-EE/bin/start-alb.sh"]
        ports:
        - containerPort: 80
        - containerPort: 443
        - containerPort: 9000
        volumeMounts:
        - name: alb-config
          mountPath: "/opt/ALB-V2.0.5-EE/alb-
standard/conf/config.yaml"
          subPath: "config.yaml"
        - name: alb-license
          mountPath: "/opt/ALB-V2.0.5-EE/license.xml"
          subPath: "license.xml"
        - name: alb-logs
```

```
          mountPath: "/opt/ALB-V2.0.5-EE/alb-standard/logs"
      volumes:
      - name: alb-config
        configMap:
          name: alb-config
      - name: alb-license
        secret:
          secretName: alb-license
      - name: alb-logs
        hostPath:
          path: "/var/alb/logs"
          type: DirectoryOrCreate
```

参数说明

- image: 替换为实际镜像地址。
- hostPath: 日志目录挂载路径，可按需改为 PVC（如使用云存储）。

2. 创建 Service

```
apiVersion: v1
kind: Service
metadata:
  name: alb-service
spec:
  type: NodePort
  selector:
    app: alb
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30080
  - name: https
    port: 443
    targetPort: 443
    nodePort: 30443
```

```
  - name: console
    port: 9000
    targetPort: 9000
    nodePort: 30900
```

参数说明

- nodePort 默认范围为 30000-32767，若省略端口则由系统自动分配
- 若在云环境，可将 type 改为 LoadBalancer 直接获取外部 IP

3. 应用配置

```
kubectl apply -f alb-deployment.yaml
kubectl apply -f alb-service.yaml
```

5.5.4 部署验证

1. 检查 Pod 状态

```
kubectl get pods -l app=alb
```

2. 访问服务

- HTTP 服务：`http://<节点IP>:30080`
- HTTPS 服务：`http://<节点IP>:30080`
- 管控台：`http://<节点IP>:30900`

5.5.5 停止与清理

1. 删除部署

```
kubectl delete -f alb-deployment.yaml -f alb-service.yaml
```

2. 清理配置

```
kubectl delete configmap alb-config
kubectl delete secret alb-license
```

5.6 产品license使用说明

ALB支持本地授权和统一授权，其中本地授权采用把授权文件拷贝到产品安装目录进行授权，而统一授权则是通过在产品指定统一授权中心的连接信息进行连接授权。

5.6.1 授权文件放置目录

授权文件存放在 `ALB安装目录` 下，目录结构如下：

```
【安装目录】tree -L 1
.
├── VERSION
├── acls.properties   # 授权中心配置文件
├── alb-dashboard
├── alb-deps
├── alb-standard
├── bin
├── license.xml       # 授权文件
└── utils
```

5.6.2 本地授权

1. 获取本机特征码

```
./bin/alb-authcode
```

2. 获取授权码后，请与金蝶天燕联系，将特征码发回金蝶天燕进行授权文件获取，或者登录金蝶天燕授权平台获取授权文件。
3. 将获取到的授权文件修改为文件名license.xml，然后放到 `ALB安装目录` 下。

5.6.3 统一授权

1. 统一授权中心必须包含ALB企业版的有效授权信息。(请登录授权中心进行查看，若没有有效的产品授权信息，请参考授权中心手册，导入有效的产品授权信息。)
2. 产品配置连接授权中心的连接方式:

- acls.properties配置文件： acls.properties配置文件放置在 `ALB安装目录` 下。

```
apusic_acls_enable=false                    # 开启变量统一授权，这里false为关闭统一授权，true为开启统一授权
apusic_acls_authUrls=192.168.101.34:6869    # 统一授权中心地址
apusic_acls_ns=public                       # 命名空间
apusic_acls_tenant=public                   # 租户名称
```

- 系统环境变量配置

```
export apusic_acls_enable=true                  # 开启变量统一授权
export apusic_acls_authUrls=172.24.3.116:6869   # 统一授权中心地址
export apusic_acls_ns=后付费                     # 命名空间
export apusic_acls_tenant=user_env中文           # 租户名称
```

注： 2种授权方式如果都进行了配置，则优先级为系统环境变量配置 > acls.properties文件配置;如果同时配置了本地授权方式和统一授权方式，则使用统一授权方式，若授权验证失败，将不会执行后续的授权验证。

6 产品快速入门

本快速入门指南介绍了金蝶Apusic负载均衡软件v2.0产品的功能和使用，为用户快速使用本产品提供指导。

6.1 登录管理控制台

打开浏览器（推荐Chrome、firefox），输入地址 `http://serverIP:serverPort/` 进入到管理控制台登录页面。

例如：将ALB部署在172.20.140.137端口默认为9000，登录地址则为： `http://172.20.140.137:9000`

输入用户名密码登录管理控制台。

- 用户名为： `admin`
- 初始密码为： `apusic$alb`

6.2 使用ALB创建反向代理

前提条件：

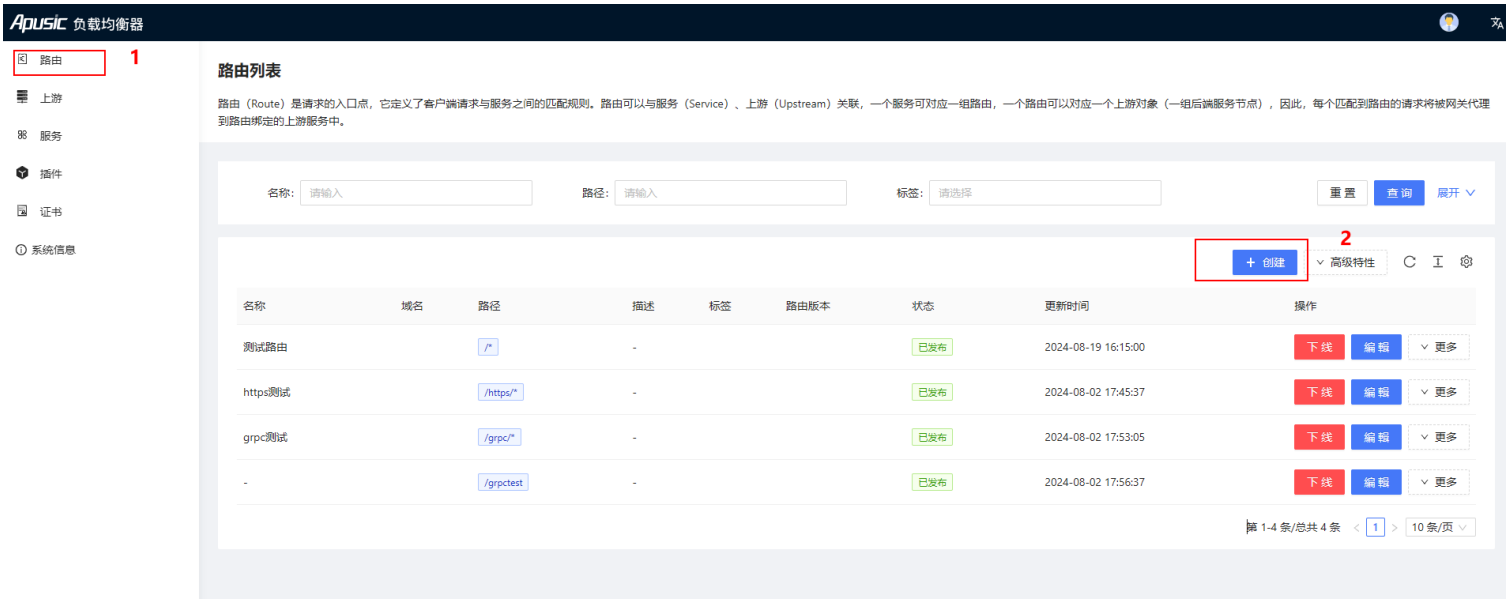
- 已经部署好的ALB
- 已经部署好的后端服务

操作步骤：

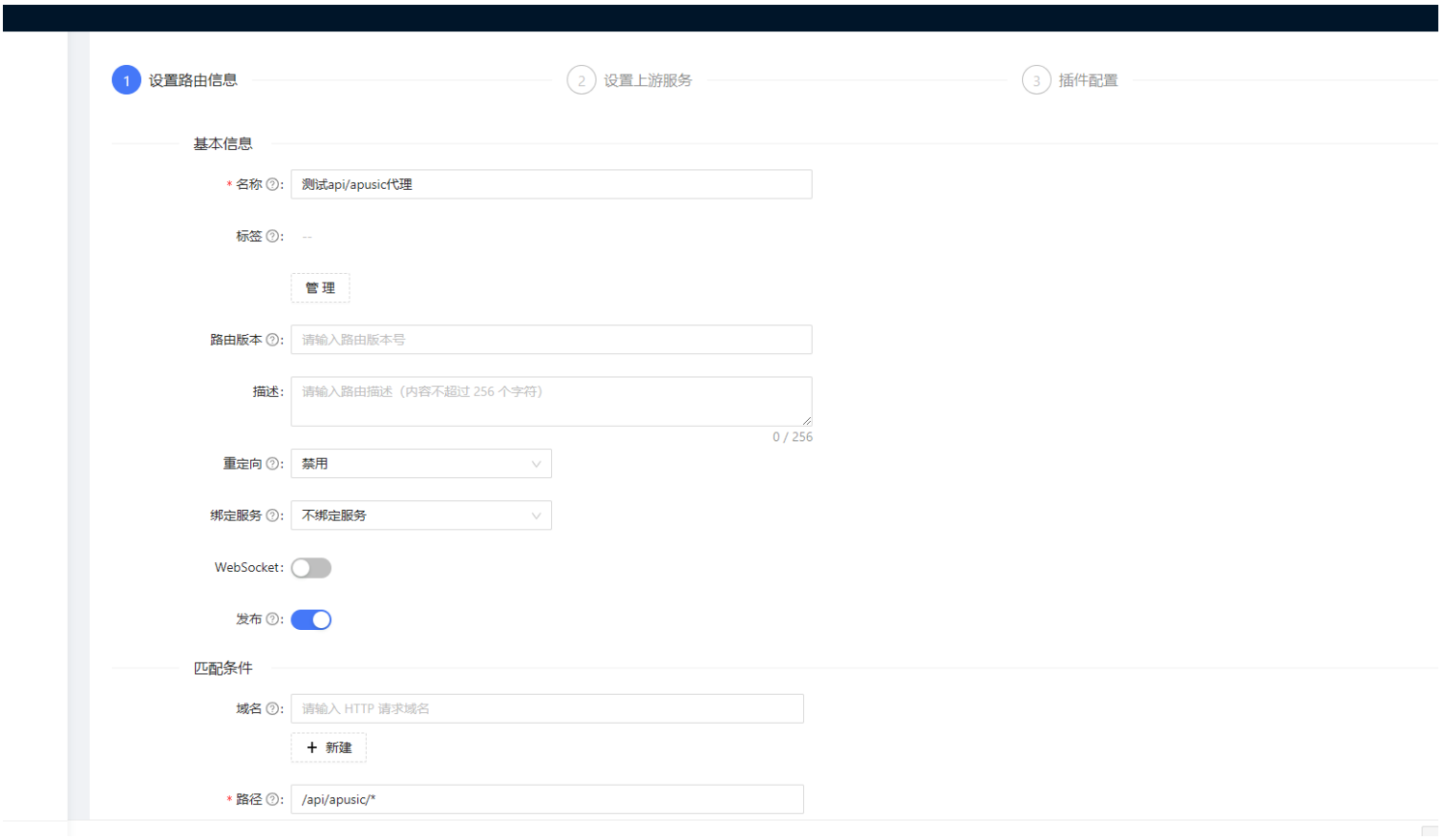
1. 登录ALB管理页面，点击“创建路由”按钮。
2. 填写路由信息，包括路由名称、代理url前缀等信息，点击下一步
3. 填写后端服务信息，包括节点地址和端口、负载均衡算法等信息，点击下一步
4. 插件配置这块点击下一步
5.

如： 配置代理url前缀为 `/api/apusic` ，后端服务地址为 `http://172.21.32.42:8000` ，其配置过程如下：

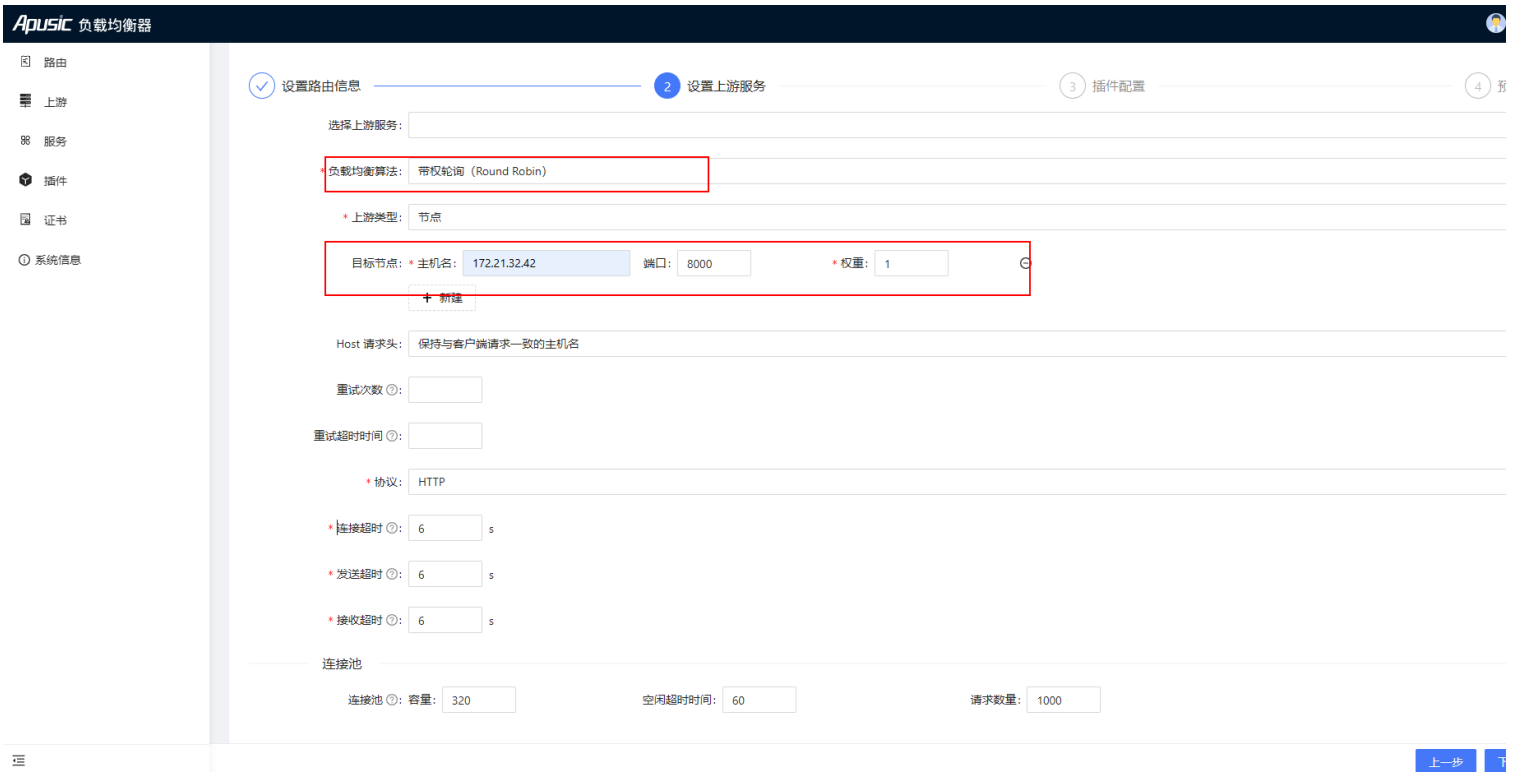
1、点击创建路由



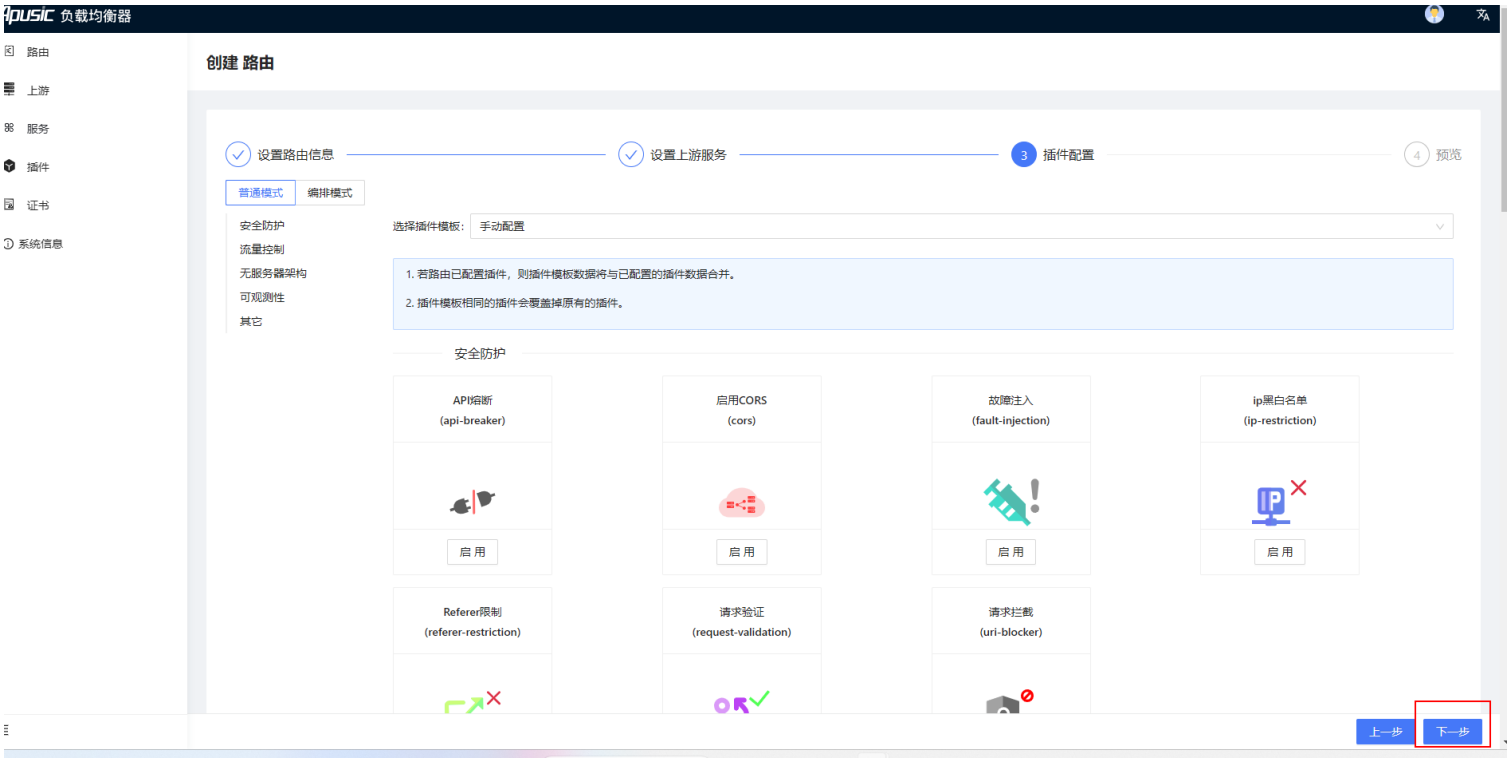
2、填写路由信息，并点击下一步



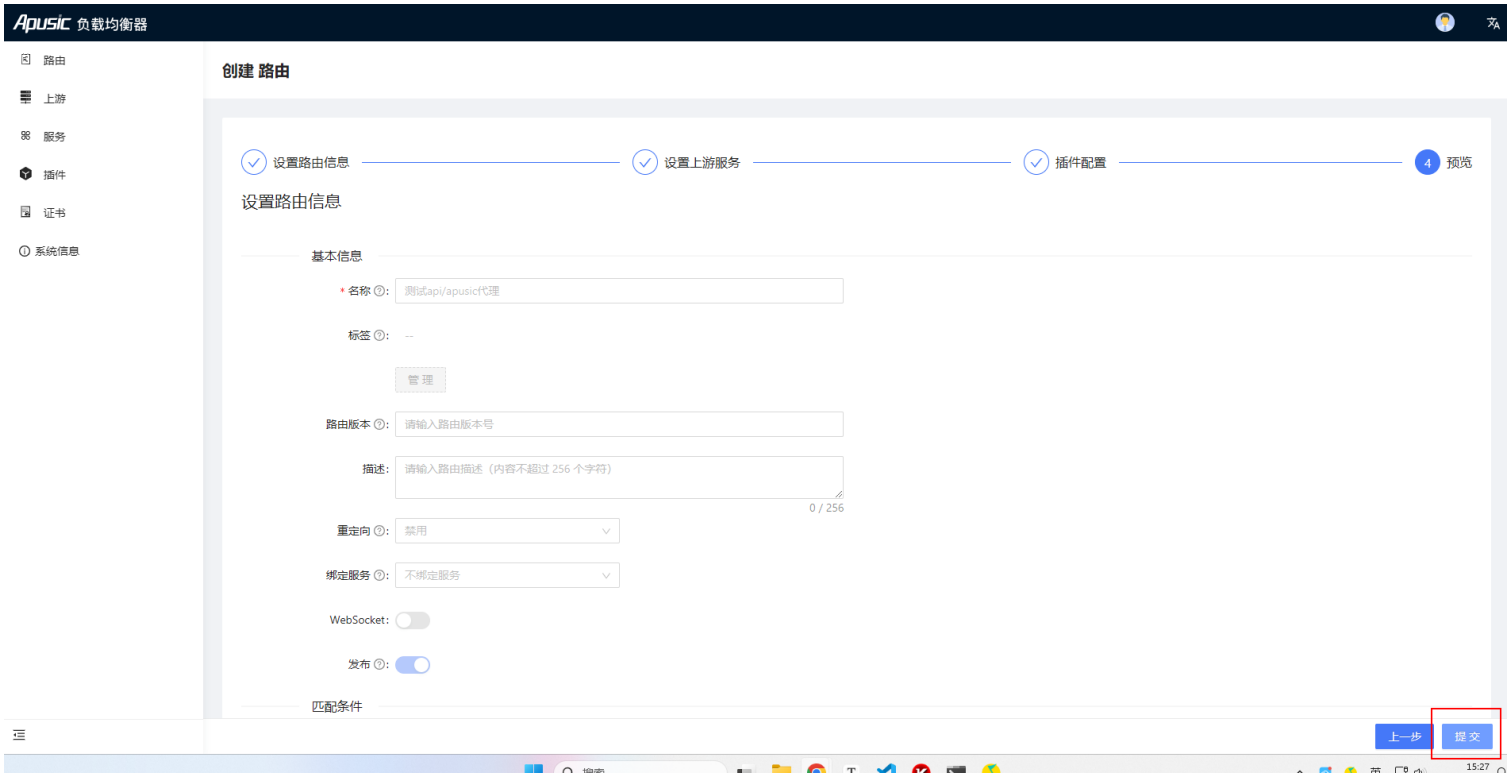
3、填写后端服务信息，并点击下一步



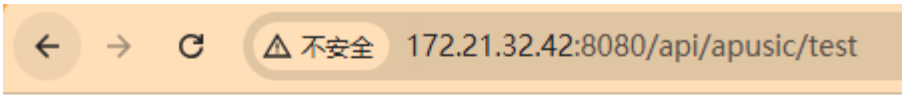
4、插件跳过，点击下一步


Apusic 负载均衡器
路由
上游
服务
插件
证书
系统信息
创建 路由
设置路由信息
设置上游服务
3 插件配置
4 预览
普通模式
编排模式
安全防护
流量控制
无服务器架构
可观测性
其它
选择插件模板:
手动配置
1. 若路由已配置插件，则插件模板数据将与已配置的插件数据合并。
2. 插件模板相同的插件会覆盖掉原有的插件。
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)
启 用
Referer限制
(referer-restriction)
请求验证
(request-validation)
请求拦截
(uri-blocker)
上一步
下一步

5、预览点击提交即可


Apusic 负载均衡器
路由
上游
服务
插件
证书
系统信息
创建 路由
设置路由信息
设置上游服务
插件配置
4 预览
设置路由信息
基本信息
* 名称:
测试api/apusic代理
标签:
管 理
路由版本:
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:
禁用
绑定服务:
不绑定服务
WebSocket:
发布:
匹配条件
上一步
提 交

6、使用ALB代理入口测试访问后端 注：当前演示环境的ALB代理的入口修改为8080端口。



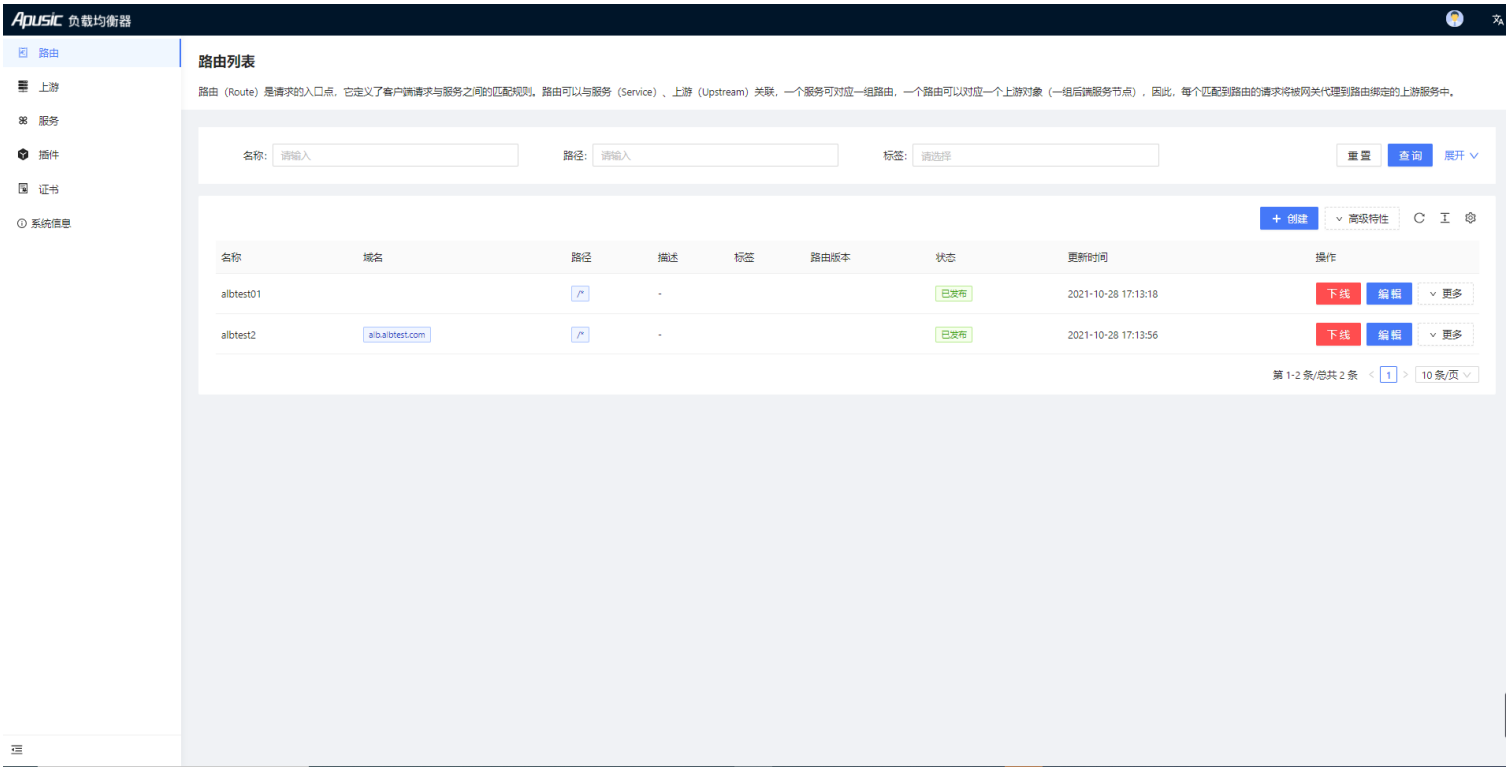

```
Current IP Address: 172.21.32.42
Requested URL: /api/apusic/test
Request Method: GET
```

7 产品功能介绍

7.1 路由

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，一个服务可对应一组路由，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。

路由列表如下图所示：



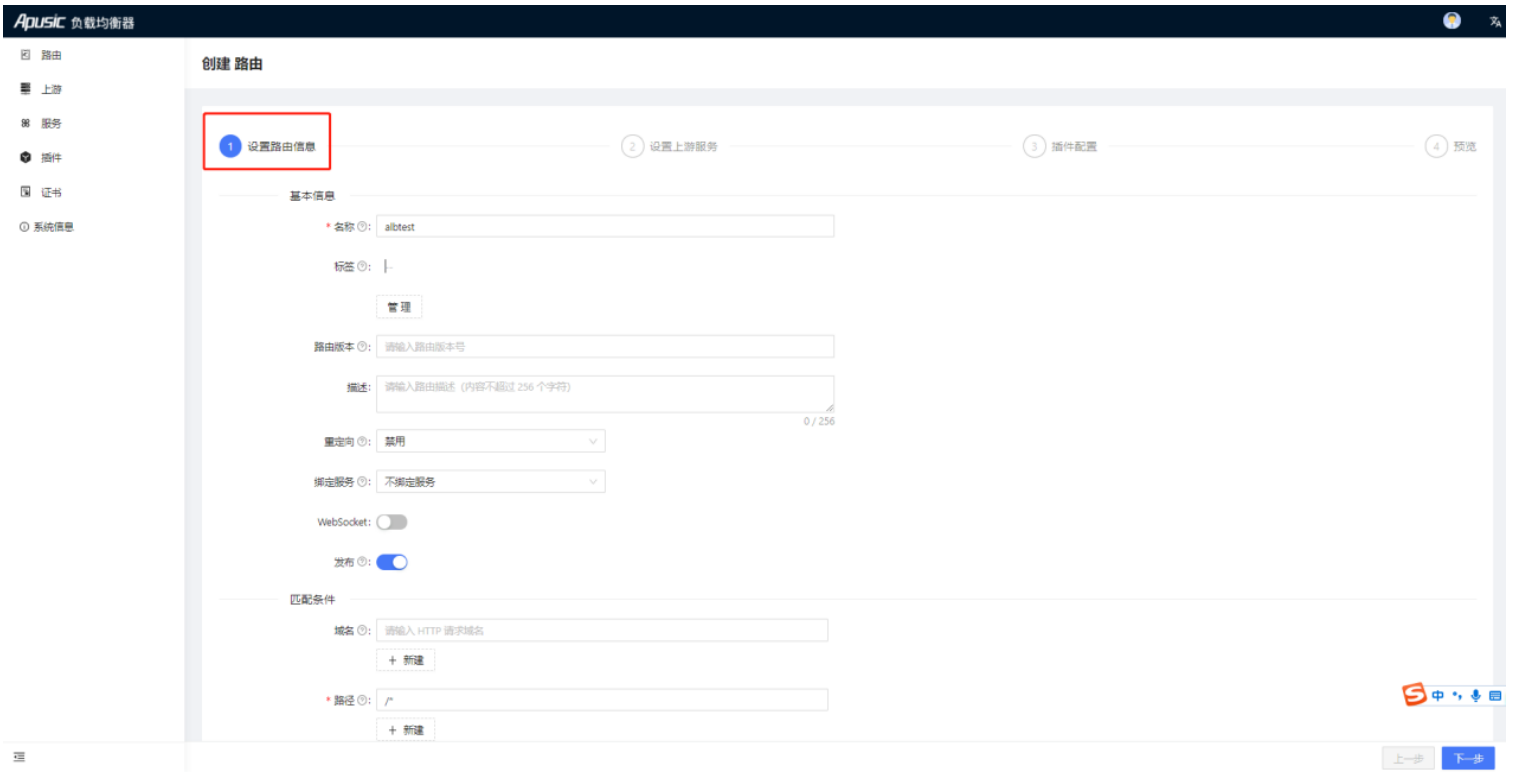
7.1.1 创建路由

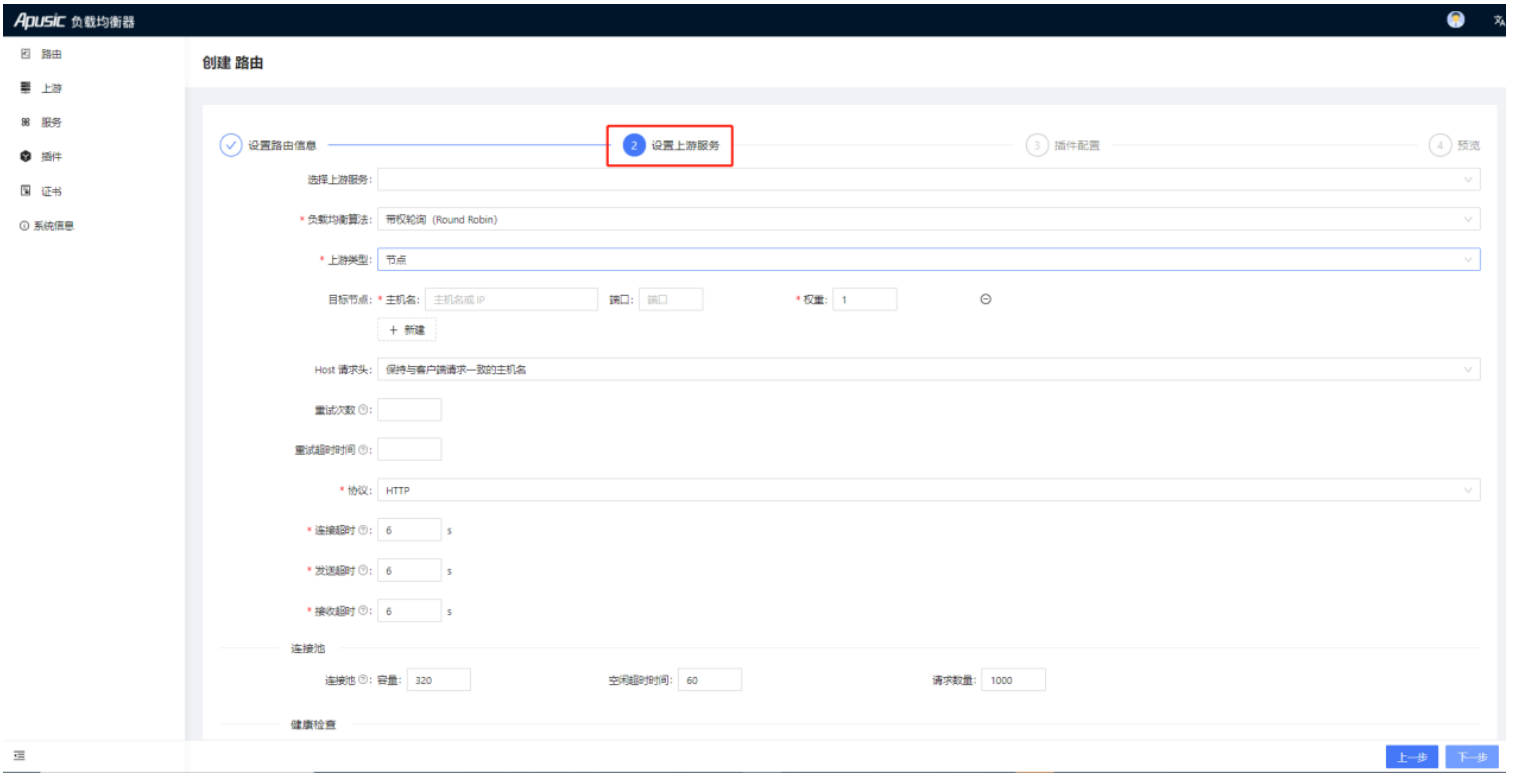
可通过两种方式创建路由：

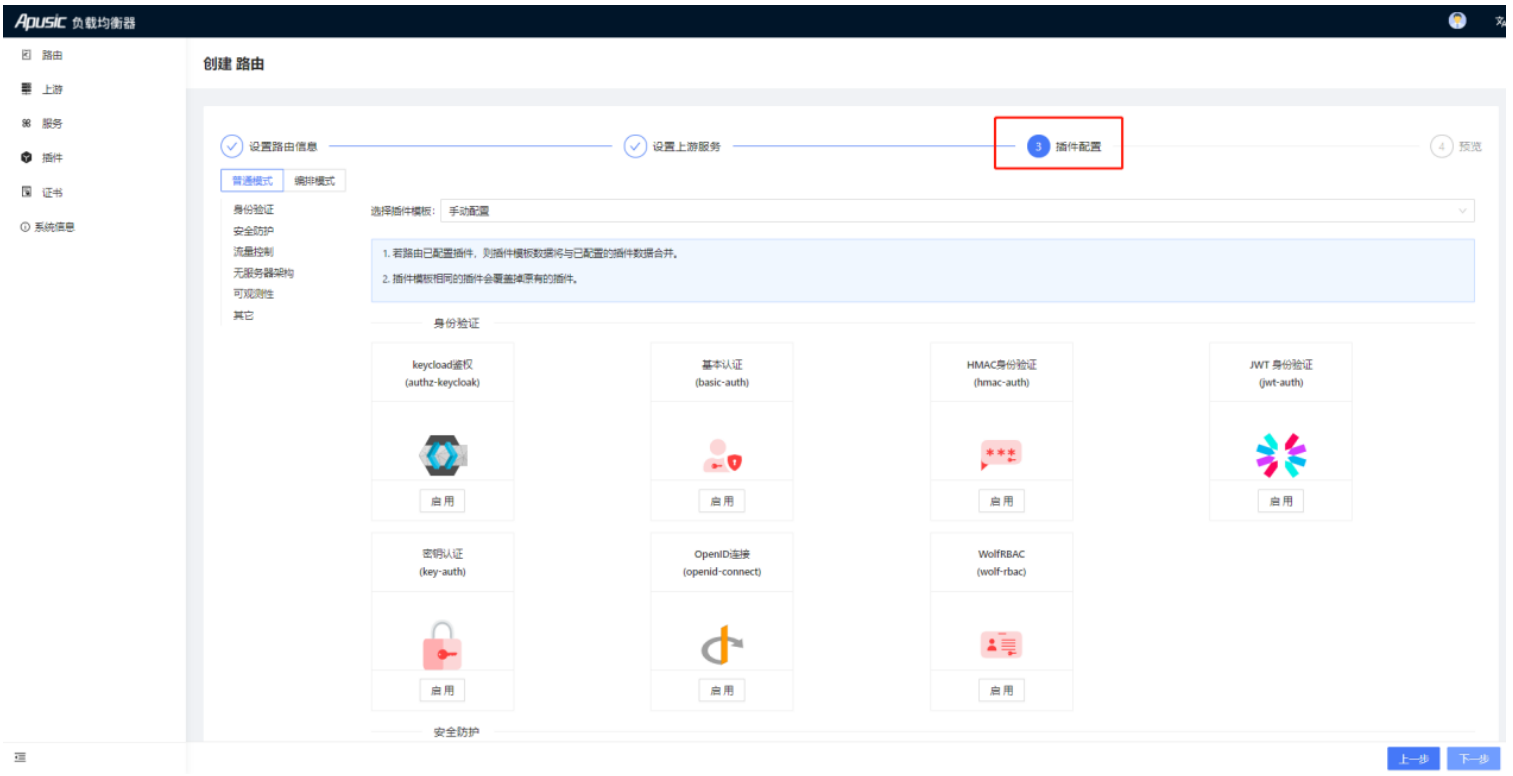
- 界面创建
- 使用编辑器创建

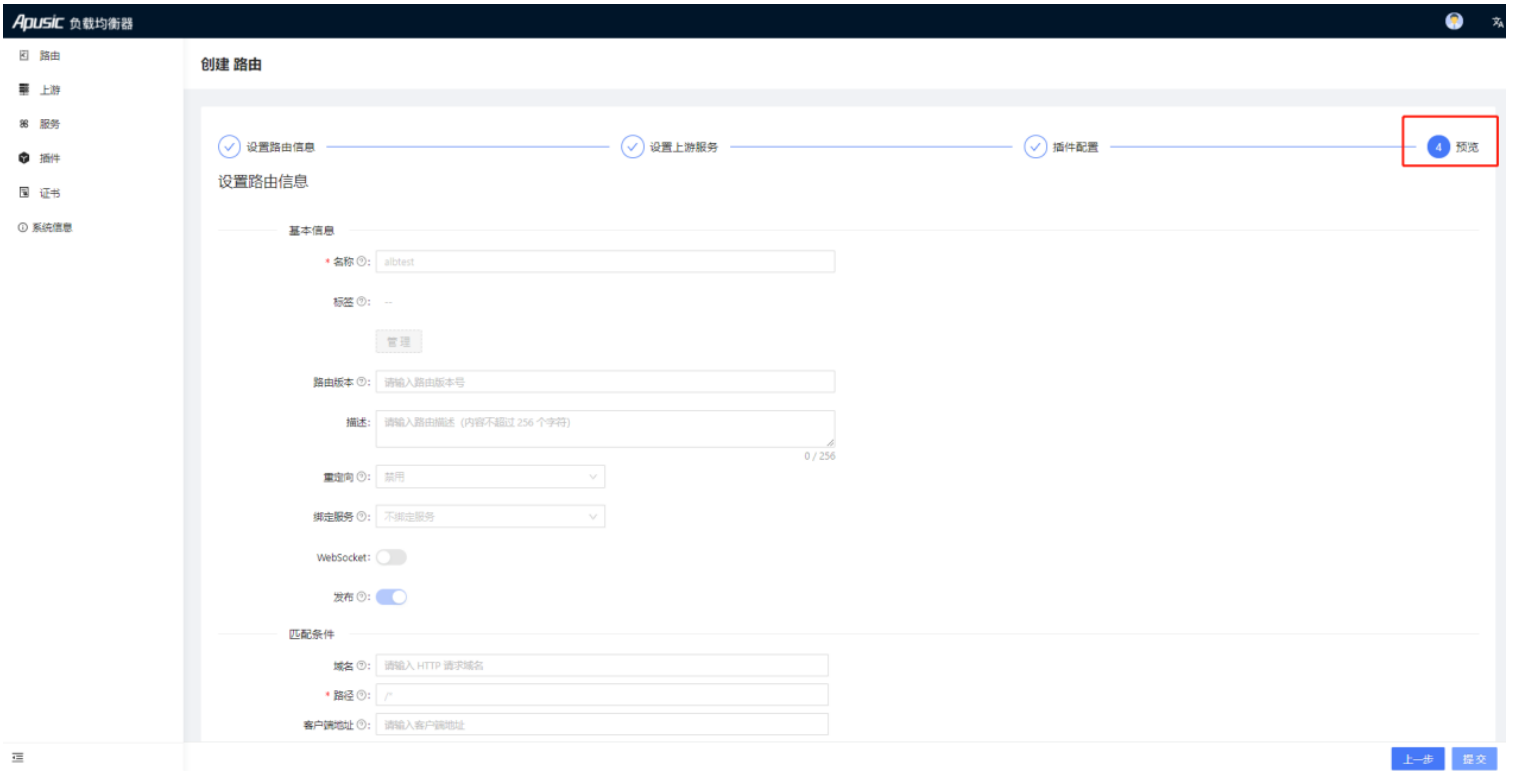
分为四个步骤：

1. 定义API请求
2. 定义API后端服务
3. 插件配置
4. 预览


Apusic 负载均衡器
路由
上游
服务
插件
证书
系统信息
创建 路由
1 设置路由信息
2 设置上游服务
3 插件配置
4 预览
基本信息
* 名称：allbtest
标签：
管理
路由版本：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：禁用
绑定服务：不绑定服务
WebSocket：
发布：
匹配条件
域名：请输入 HTTP 请求域名
+ 新建
* 路径：/*
+ 新建
上一步
下一步


Apusic 负载均衡器
路由
上游
服务
插件
证书
系统信息
创建 路由
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：
* 负载均衡算法：带权轮询（Round Robin）
* 上游类型：节点
目标节点：* 主机名：主机名或 IP 端口：端口 * 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：
重试超时时间：
* 协议：HTTP
* 连接超时：6 s
* 发送超时：6 s
* 接收超时：6 s
连接池
连接池：容量：320 空闲超时时间：60 请求数量：1000
健康检查
上一步
下一步





7.1.2 界面创建路由

以界面创建为例介绍路由的创建过程

1. **设置路由信息**：

填写路由基本信息

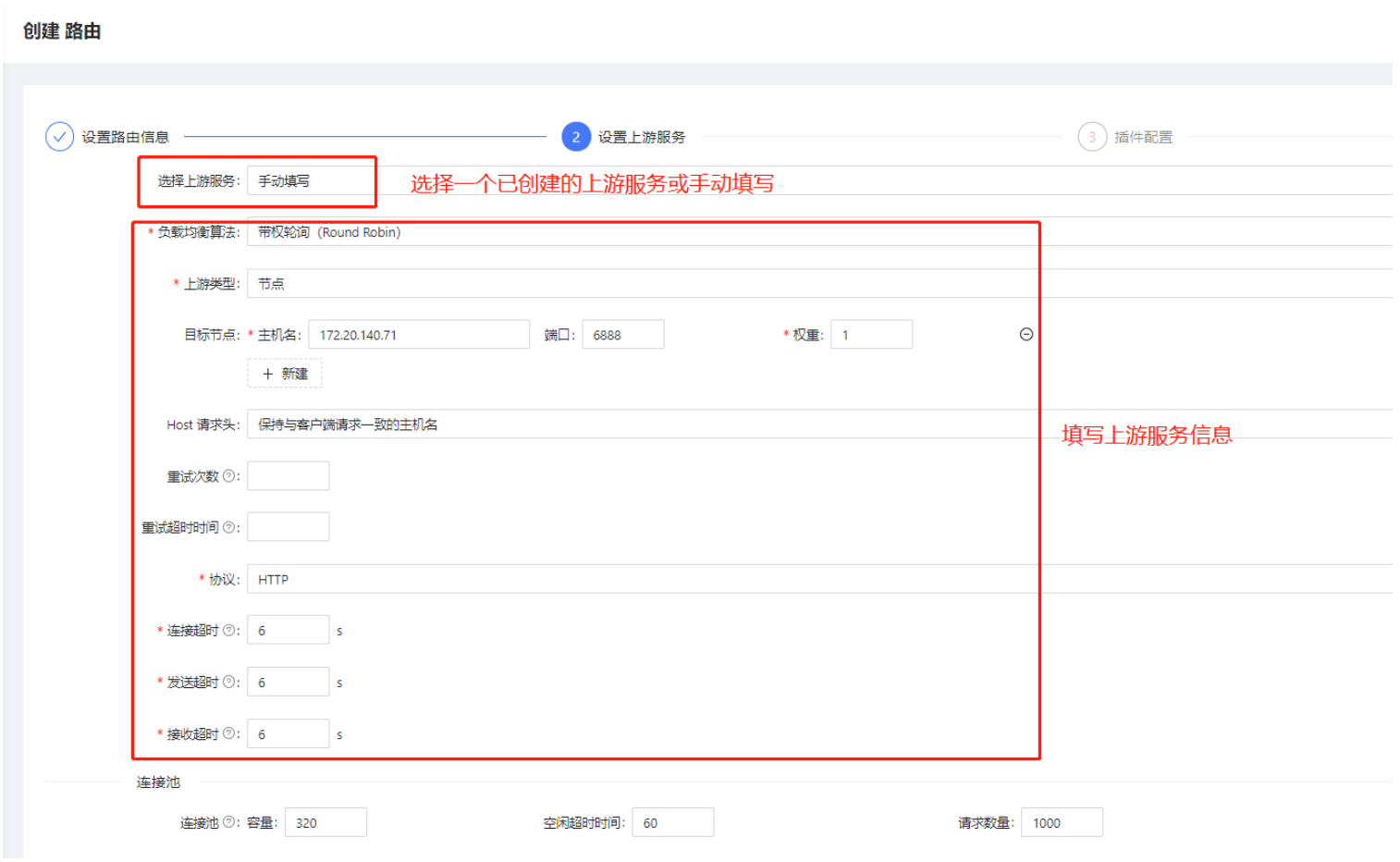
- 名称：必输项，自定义路由名称，不允许配置重名路由
- 发布：默认开启，开启状态下路由才会生效

填写请求信息：

- 域名：alb服务器域名
- 路径：必输项，默认所有路径/*，可以是路由的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：必输项，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

7.1.3 定义API后端服务

配置上游信息：



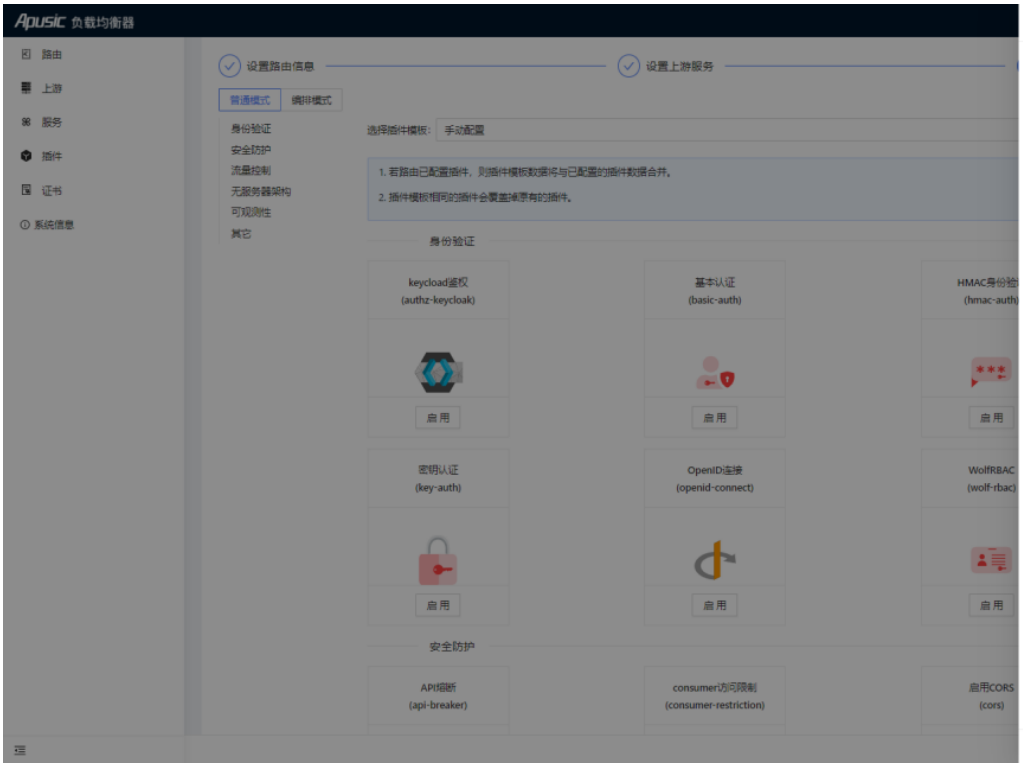
- 选择上游服务：选择已经添加的上游服务时不可编辑上游，手动填写才能编辑上游服务
- 负载均衡算法：必选项，默认带权轮询
- 上游类型：必选项，默认为节点
- 目标节点：必输项，输入目标服务器主机名或者IP地址、端口
- 权重：必输项，输入上游服务器的权重，权重高的节点承受的流量多，权重配置为0时熔断该节点
- 协议类型：必选项，选择客户端请求的类型
- 超时时间：必输项，默认超时时间为6S

7.1.4 插件配置

插件配置根据自定义可选择普通方式或插件编排方式选择插件：

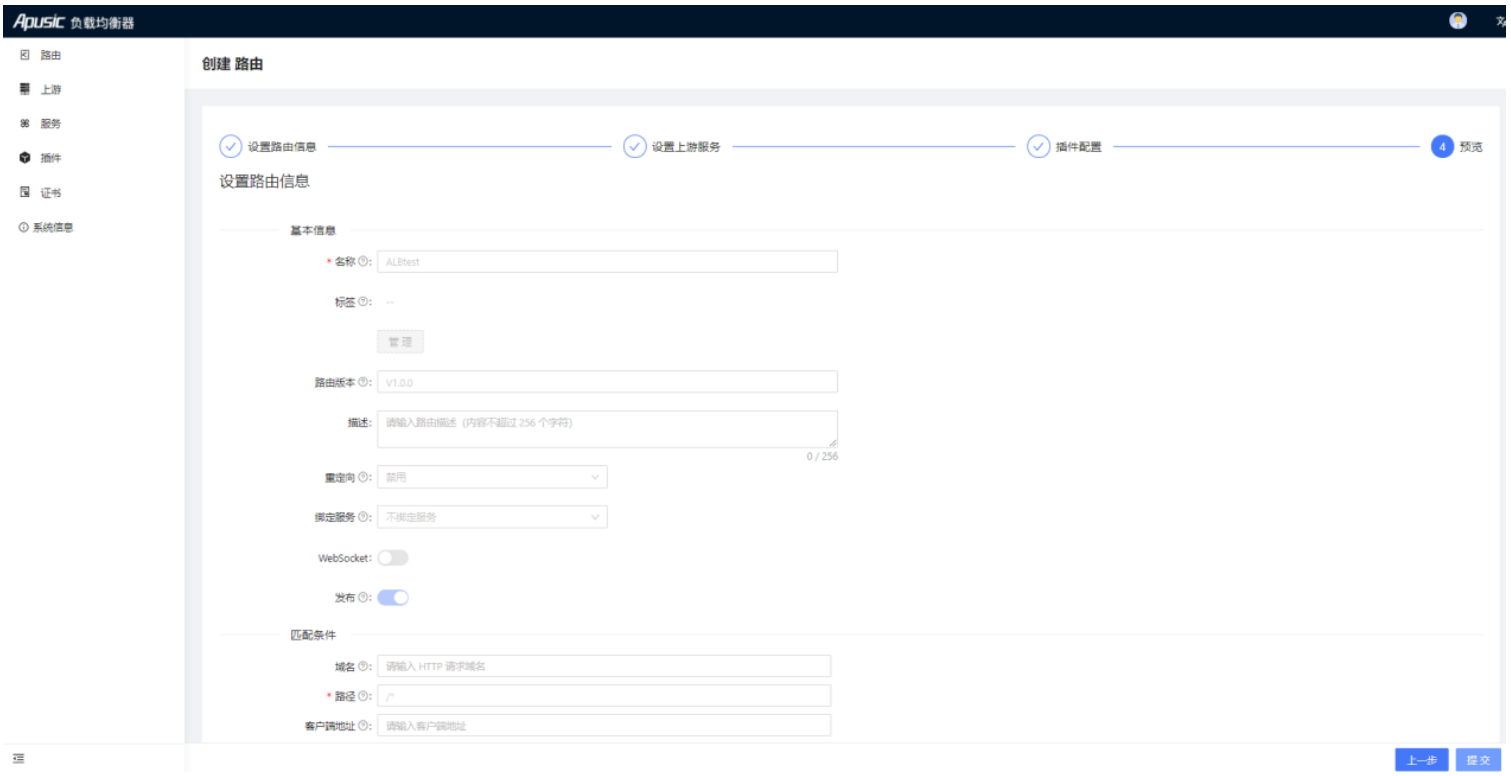

定义 API 请求
定义 API 后端服务
3 插件配置
4 预览
普通模式
插件编排
两种模式
1.普通方式
2.插件编排
选择插件模板：
手动配置
1. 若路由已配置插件，则插件模板数据将与已配置的插件数据合并。
2. 插件模板相同的插件会覆盖掉原有的插件。
认证
通用
日志
监控
协议转换
安全
流量控制
报文转换
插件分类
认证
keycload鉴权
(authz-keycloak)
OpenID连接
(openid-connect)
点击启用
点击启用
选择对应插件点击启用
通用
批量请求
(batch-requests)
回显
(echo)
系统信息上报
(server-info)
后附加函数
(serverless-post-functio
上一步
下一步

启用插件：


Apusic 负载均衡器
路由
上游
服务
插件
证书
系统信息
设置路由信息
设置上游服务
普通模式
编排模式
身份验证
安全防护
流量控制
无服务器架构
可观测性
其它
选择插件模板：
手动配置
1. 若路由已配置插件，则插件模板数据将与已配置的插件数据合并。
2. 插件模板相同的插件会覆盖掉原有的插件。
身份验证
keycload鉴权
(authz-keycloak)
基本认证
(basic-auth)
HMAC身份验证
(hmac-auth)
启用
密钥认证
(key-auth)
OpenID连接
(openid-connect)
WolfRBAC
(wolf-rbac)
安全防护
API熔断
(api-breaker)
consumer访问限制
(consumer-restriction)
启用CORS
(cors)

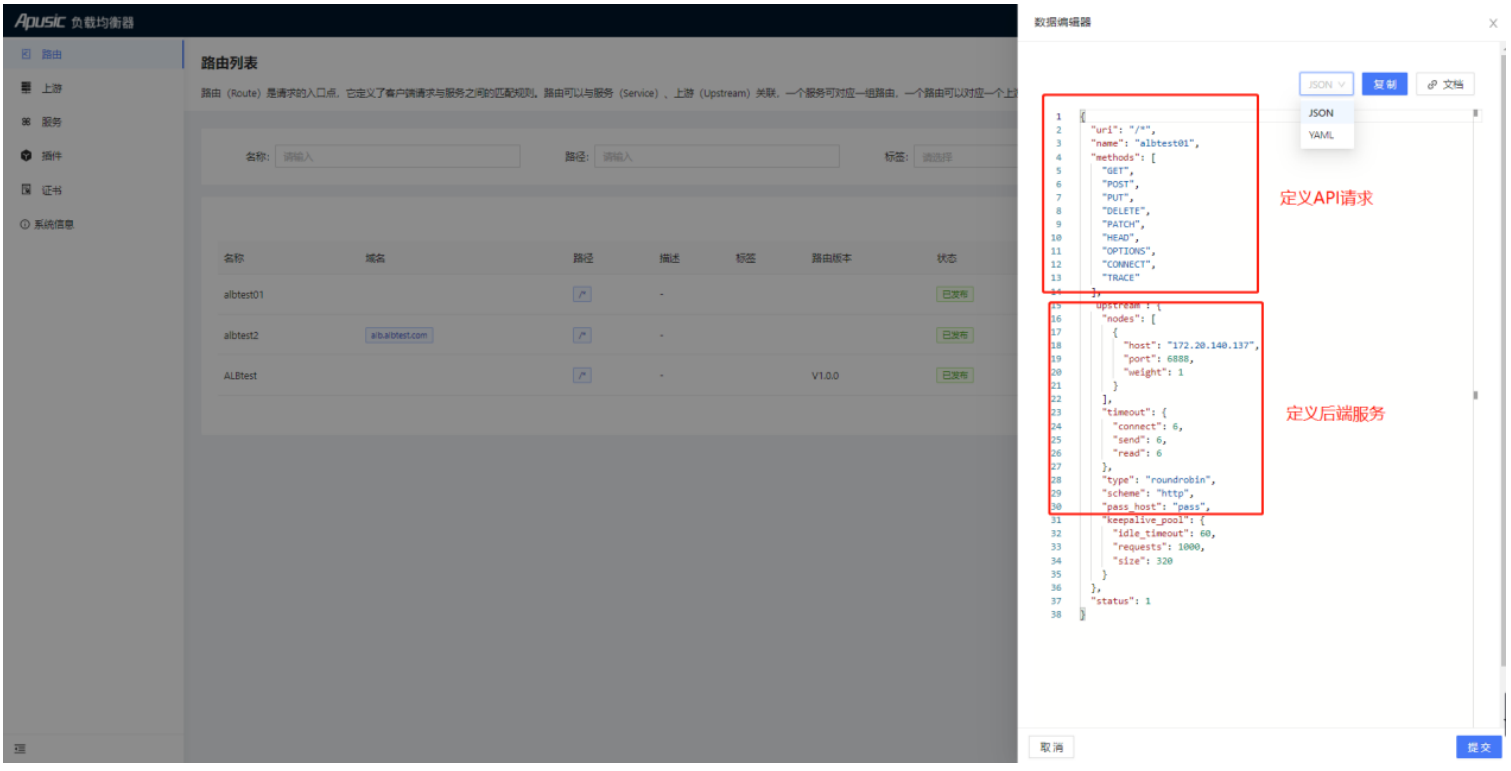

插件配置
名称：
basic-auth
启用：
数据编辑器
表单
本插件无需配置
取消
提交

7.1.5 预览



7.1.6 查看路由

点击路由列表右侧【查看】操作按钮，通过JSON格式或YAML格式查看创建的路由详细信息

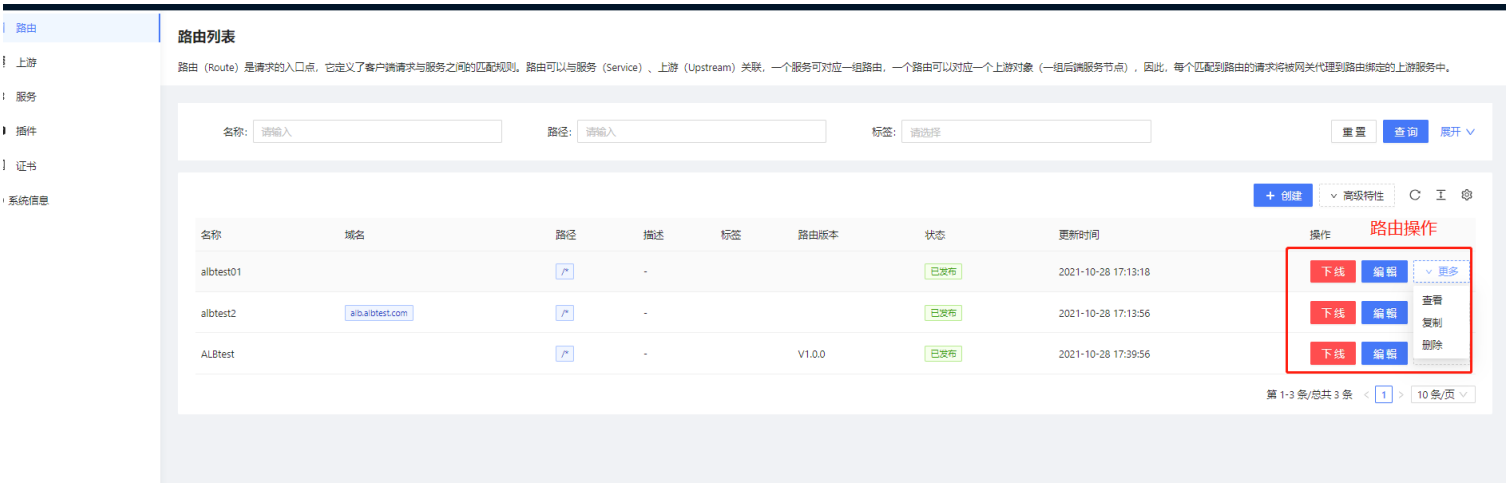


示例Json数据:

```
{
    "uri":"/\*",
    "name":"testALB",
    "methods":[
        "GET",
        "POST",
        "PUT",
        "DELETE",
        "PATCH",
        "HEAD",
        "OPTIONS",
        "CONNECT",
        "TRACE"
    ],
    "upstream":{
        "nodes":{
            "172.18.100.159":1
        },
        "timeout":{
            "connect":6,
            "send":6,
            "read":6
        },
        "type":"roundrobin",
        "scheme":"http",
        "pass_host":"pass",
        "keepalive_pool":{
            "idle_timeout":60,
            "requests":1000,
            "size":320
        }
    },
    "status":1
}
```

7.1.7 路由操作

在路由列表中实现对路由的发布、下线、编辑、查看、删除、复制

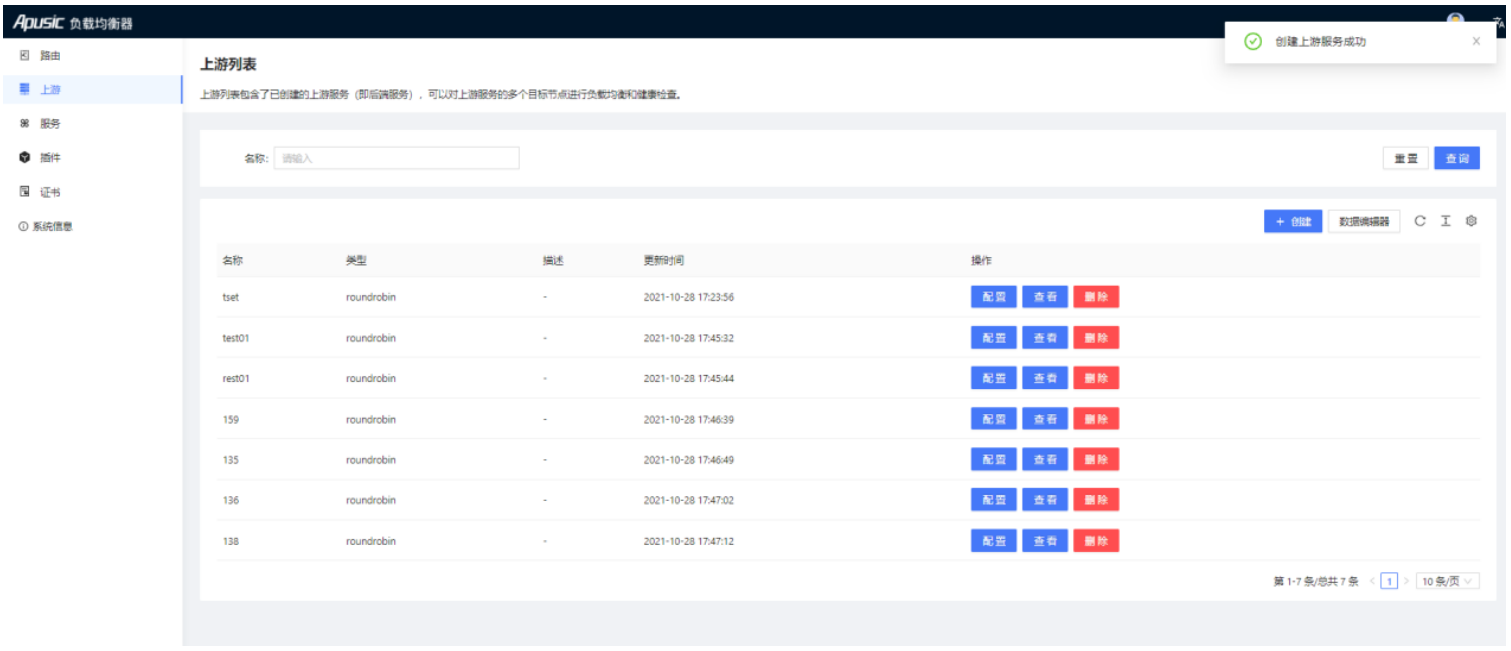


- 编辑：编辑路由基本信息或上游服务节点信息，动态生效无需重启
- 查看：通过JSON格式或YUML格式查看路由
- 复制：复制路由所有信息，名称不可与原名称相同
- 下线\发布：路由下线和发布动态生效，无需重启ALB
- 删除：删除路由信息不保存

7.2 上游

上游列表包含了已创建的上游服务（即后端服务），可以对上游服务的多个目标节点进行负载均衡和健康检查。

7.2.1 上游列表

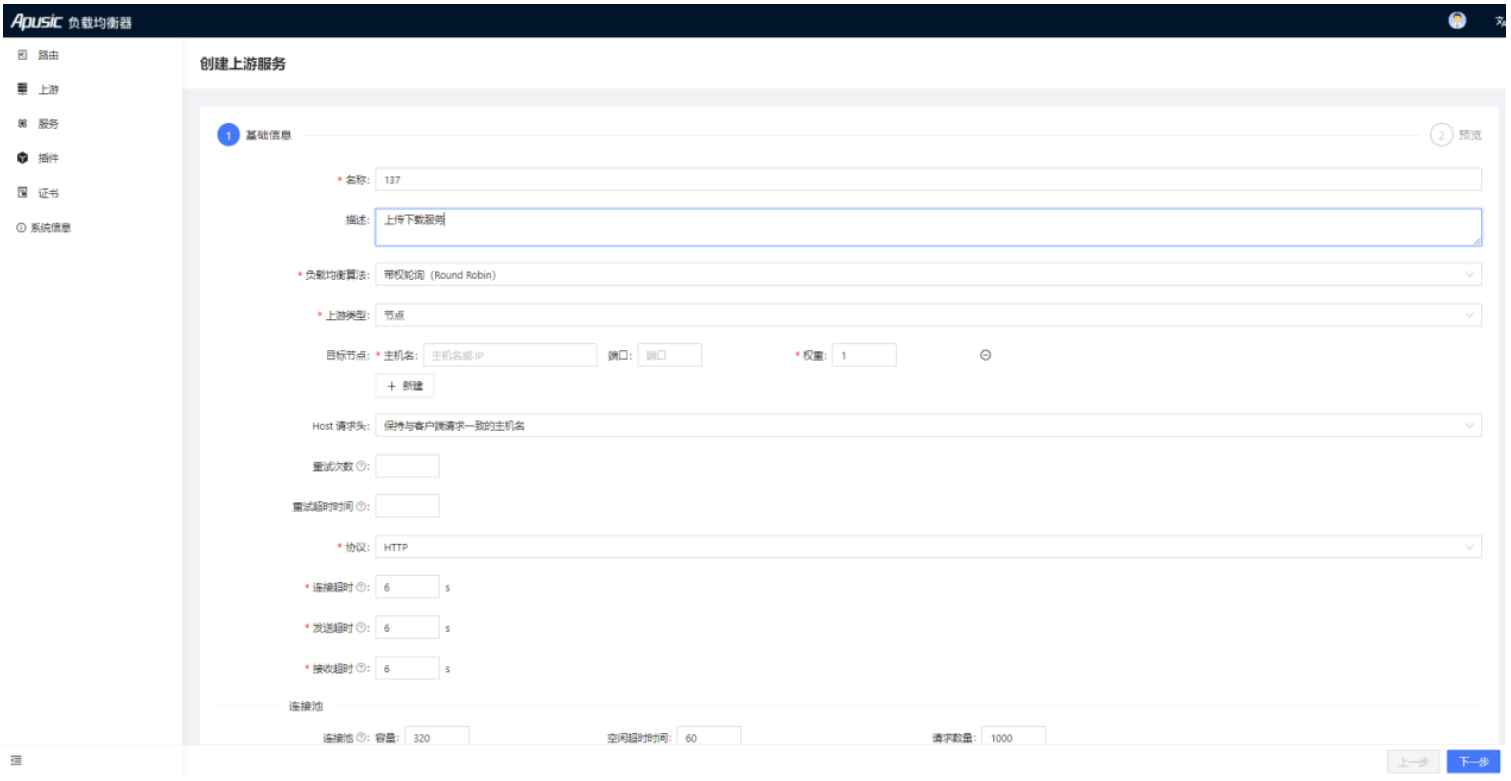


7.2.2 创建上游

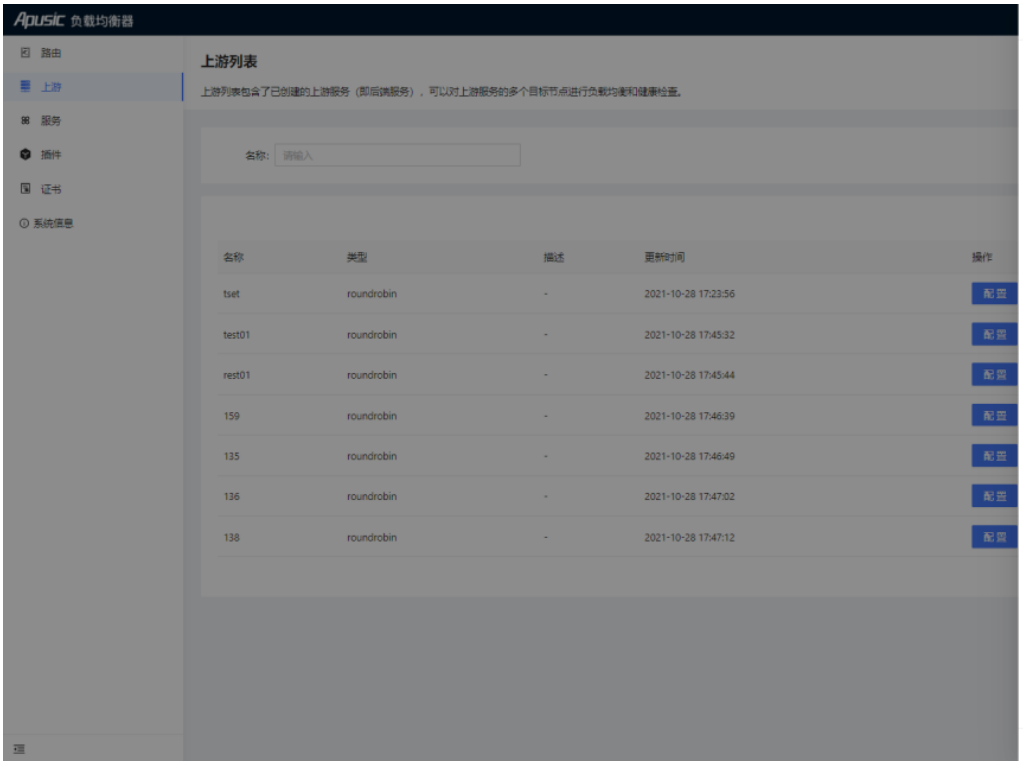
可以通过两种方式进行上游的创建：

- 界面创建
- 使用编辑器创建

下图是使用界面创建：



下图是使用编辑器创建：



7.3 服务

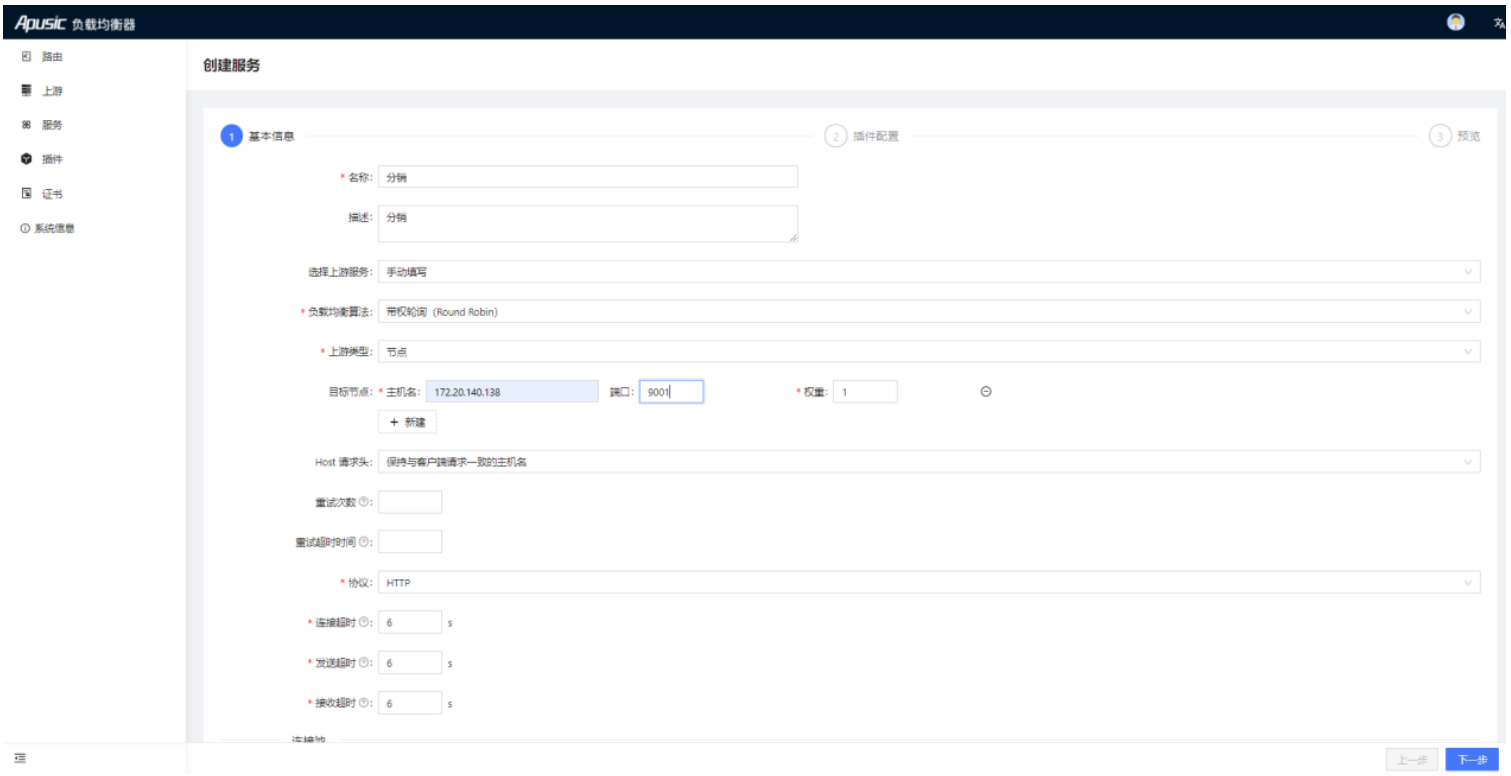
服务由路由中公共的插件配置、上游目标信息组合而成。服务与路由、上游关联，一个服务可对应一组上游节点、可被多条路由绑定。

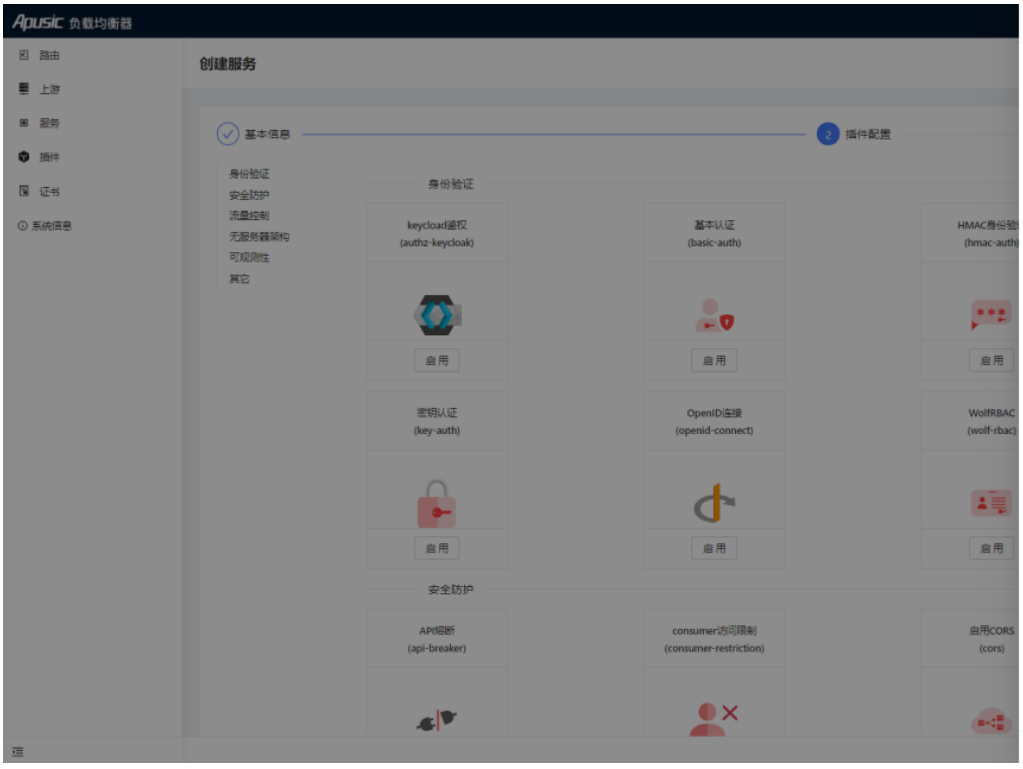
服务列表

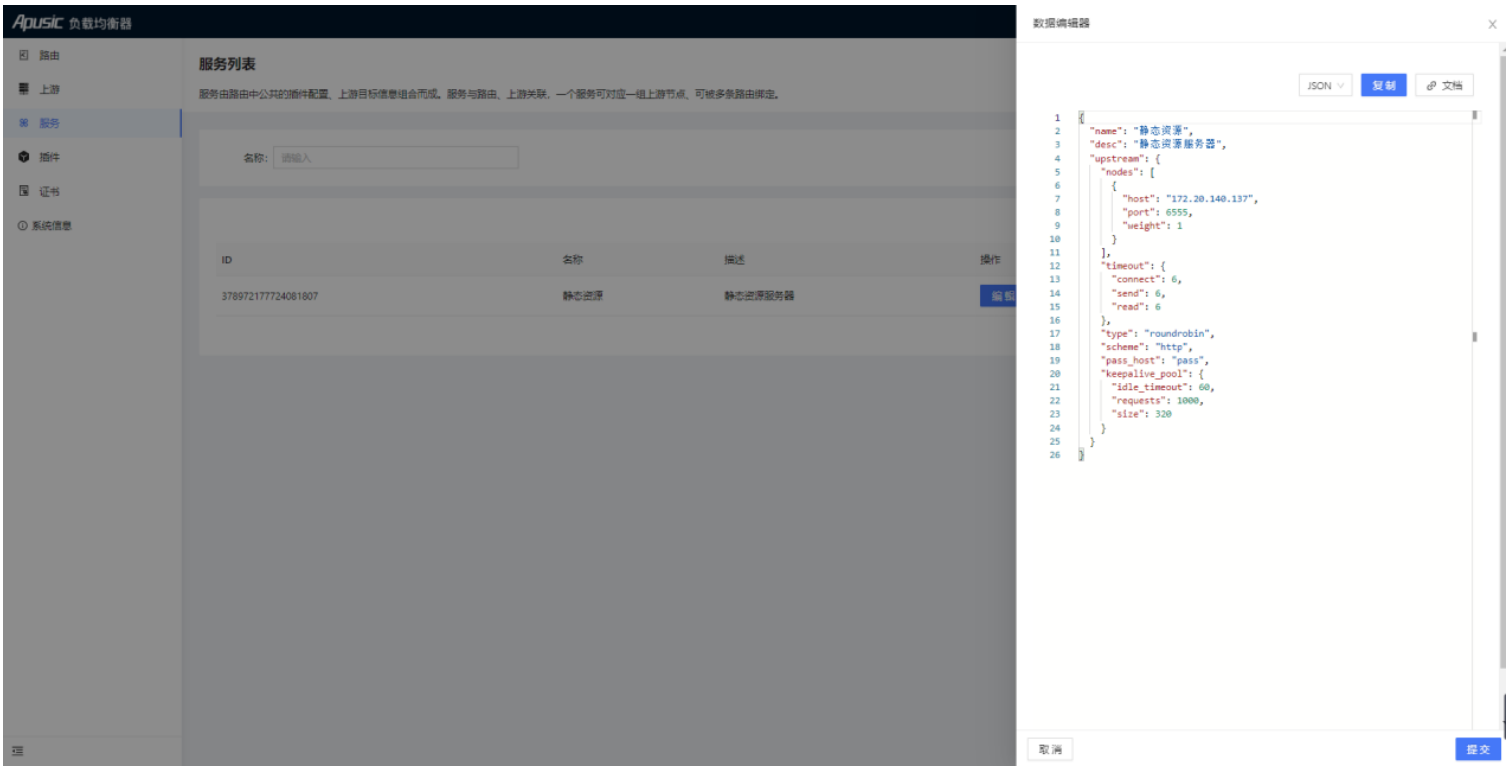
创建服务

可通过两种方式创建服务。

- 界面创建
- 使用编辑器创建







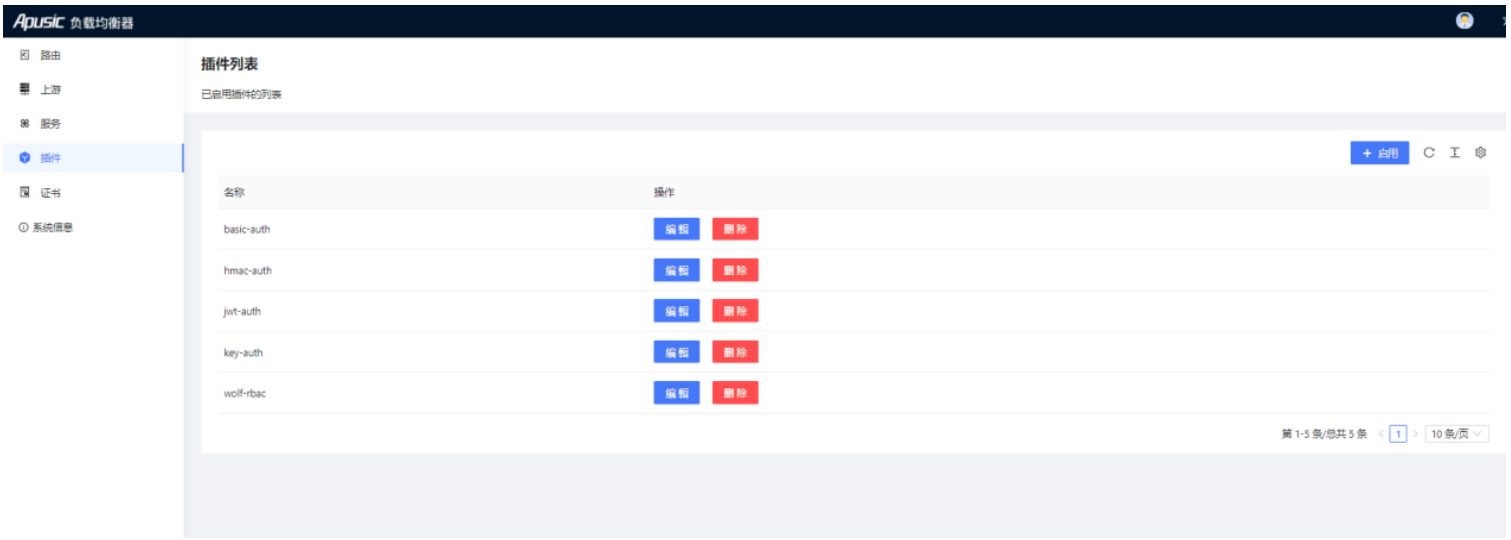
7.4 插件

已启用插件的列表

- 内置插件：
  - 认证：keycloak鉴权、openID连接
  - 通用：批量请求、回显、重定向、前后附加函数等
  - 日志：错误日志、HTTP日志、kafka日志、阿里云日志、TCP日志等
  - 监控：节点状态、prometheus、skywalking等
  - 协议转换：dubbo代理
  - 安全：启用CORS、ip黑白名单、referer限制、请求拦截等
  - 流量控制：API熔断、限制并发连接、限制请求次数等
  - 报文转换：故障注入、gPRC转码、响应信息重写等
- 自定义插件：

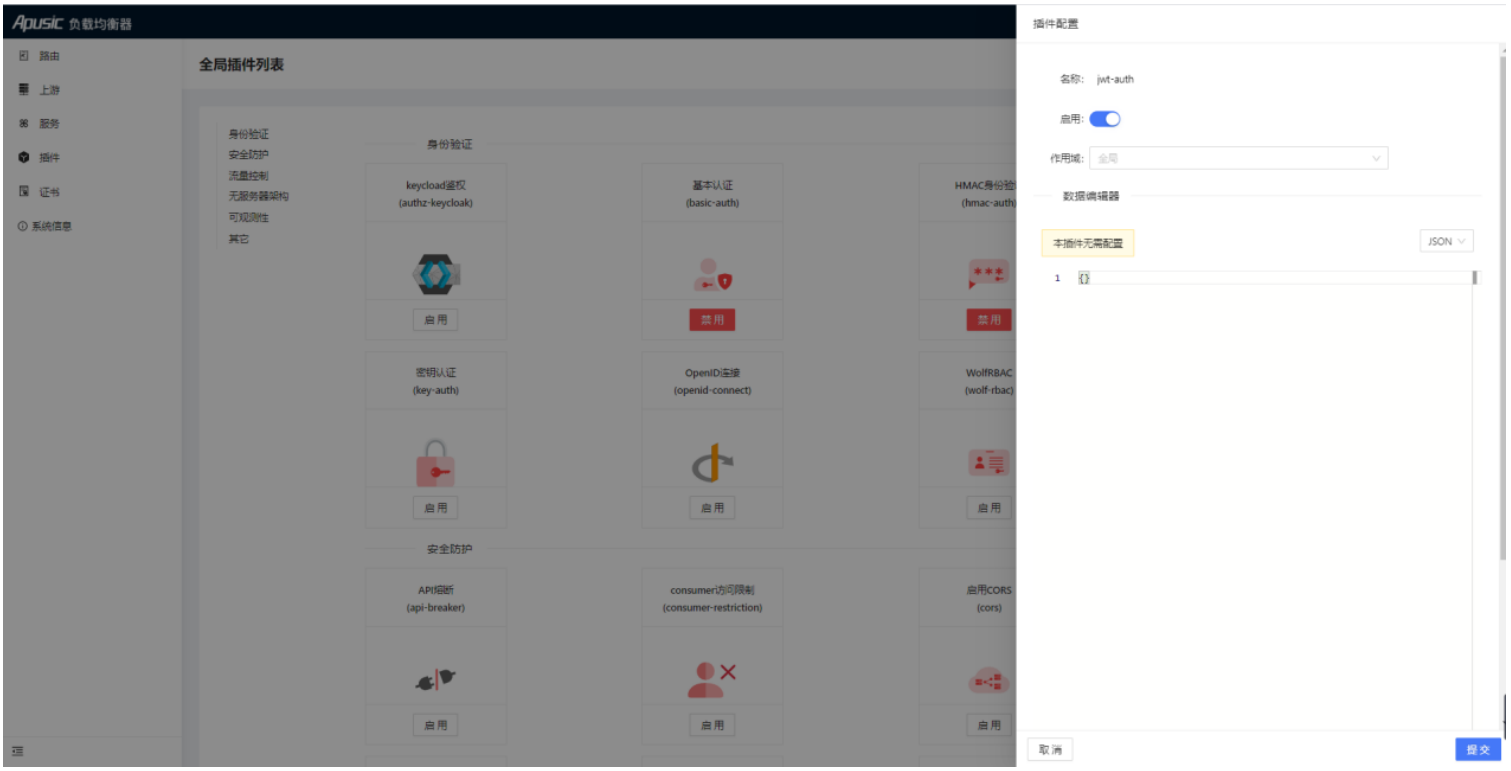
可以根据实际业务编写插件，插件允许在常见阶段进行挂载生效，例如init, rewrite，access，balancer,header filter，body filter 和 log 阶段。

插件列表



启用插件

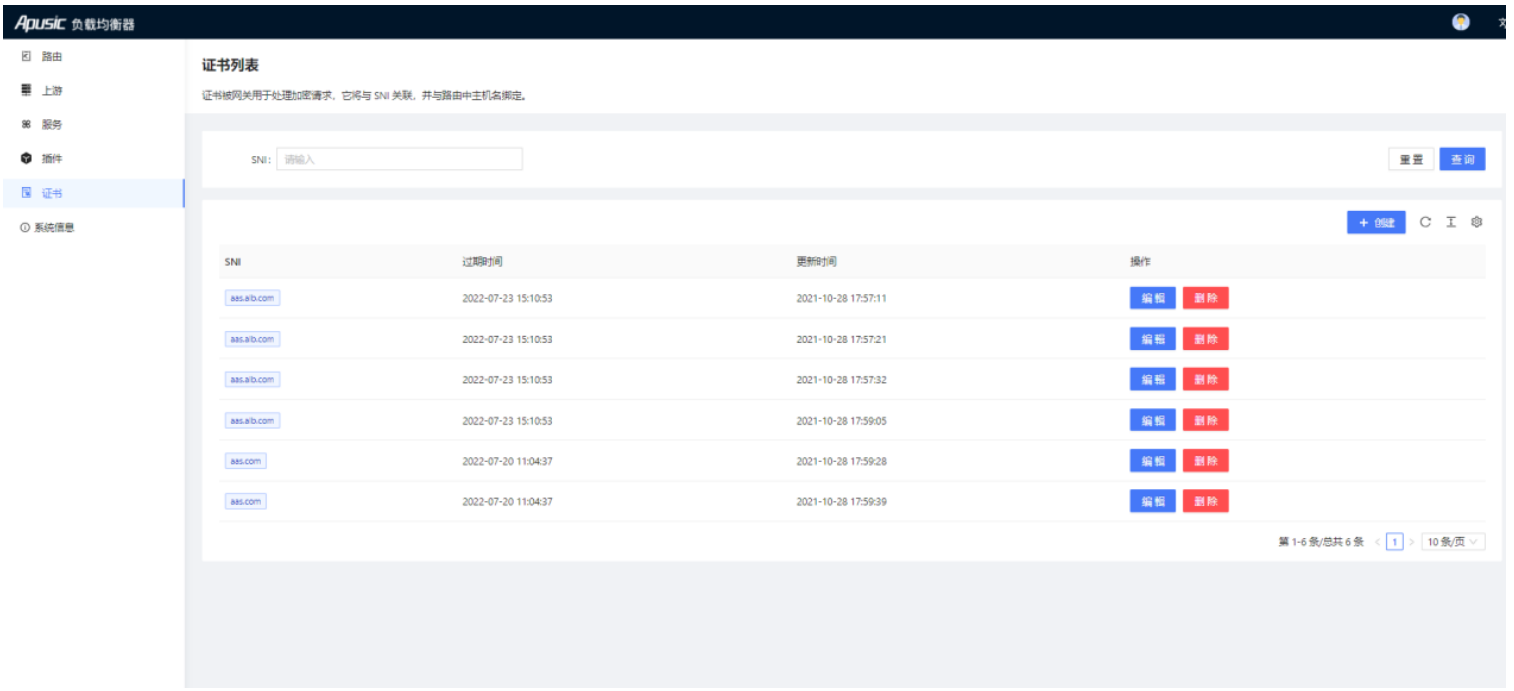
在配置列表中可以选择插件进行启用



7.5 证书

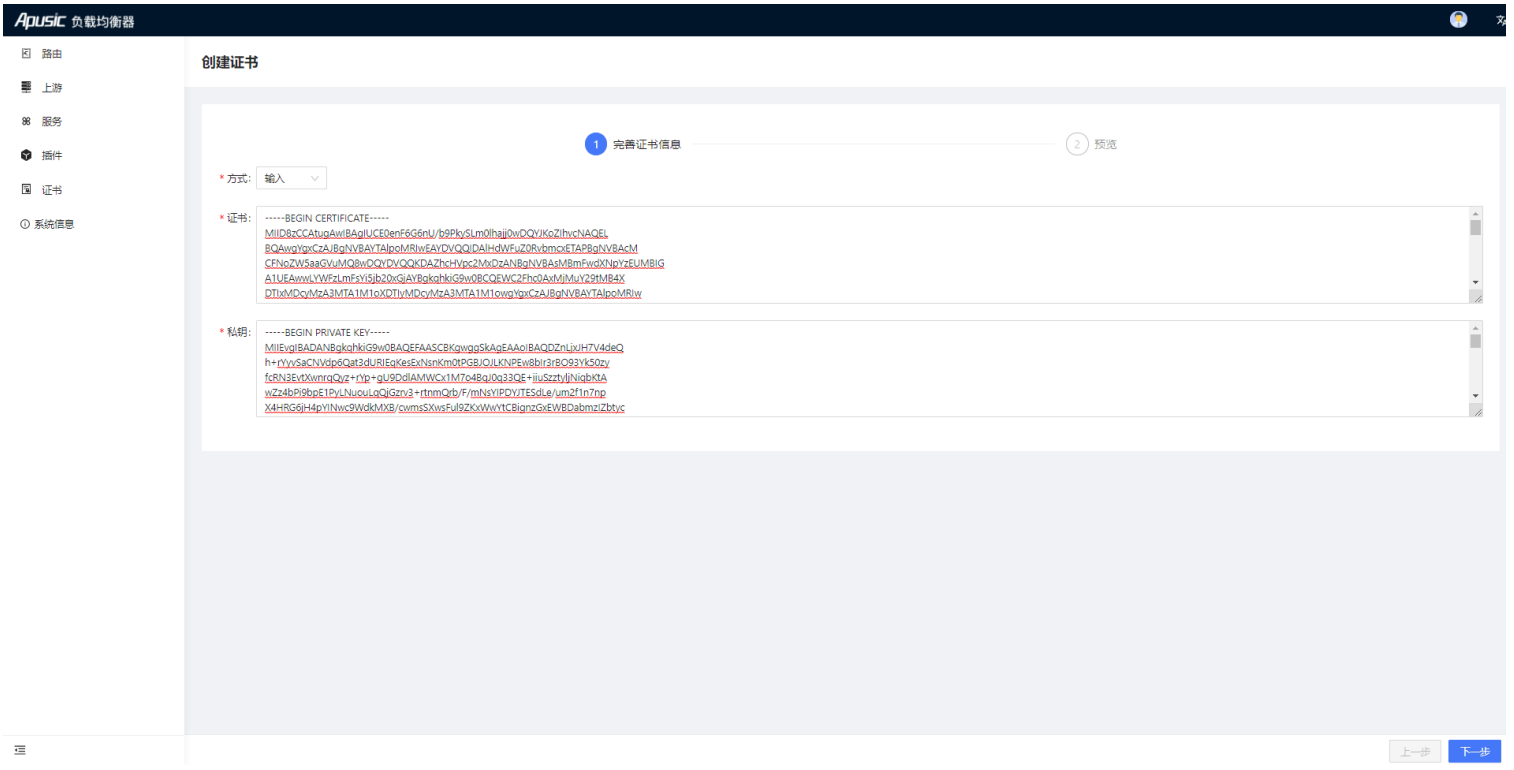
证书被网关用于处理加密请求，它将与 SNI 关联，并与路由中主机名绑定。

** 证书列表**

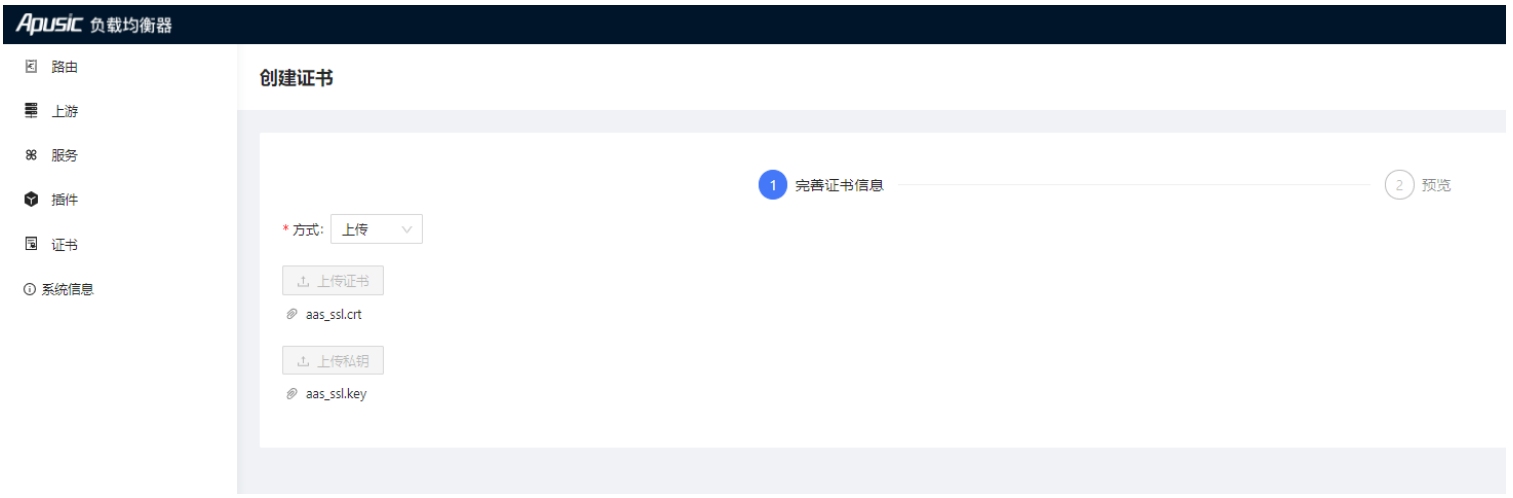


创建证书

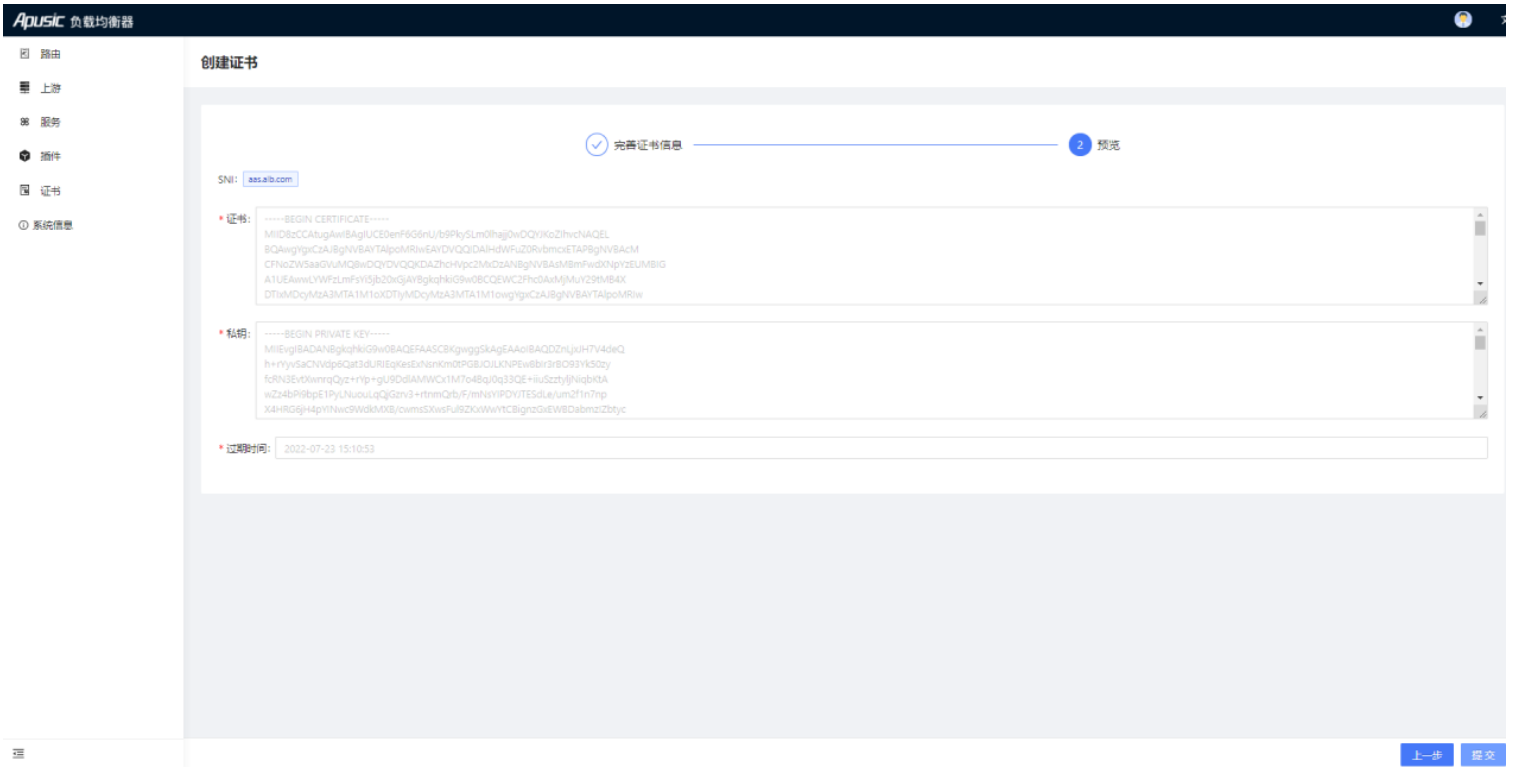
- 输入证书



- 上传证书



- 预览



7.6 系统信息

显示控制台以及ALB节点的信息。

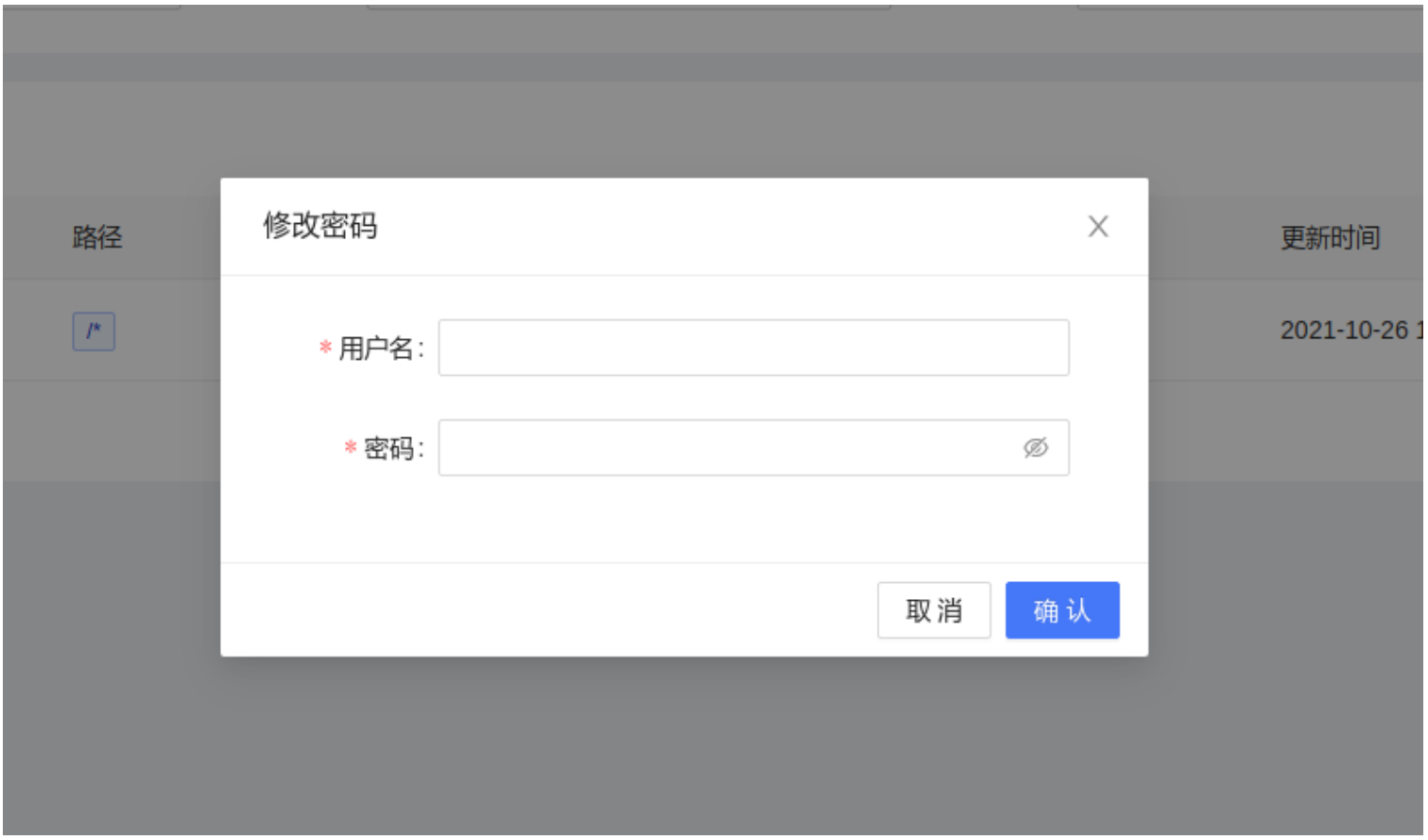
信息页

7.7 密码与安全

密码修改说明：为了保证系统的安全，要求密码**长度需6位**及以上，并且必须包含**特殊字符**。

管控台密码修改

点击右上角头像，选择修改密码，输入登录用户名和新密码即可。



忘记密码

若忘记密码，则需要通过修改配置文件中的密码设置项进行修改，修改步骤为：

1. 用编辑器打开配置文件：`安装目录/alb-dashboard/conf/conf.yaml`
2. 跳转至57行，修改password为新密码项，如下图示：

```
54  expire_time: 3600    # jwt token expire time, in second
55  users:               # yamllint enable rule:comments-indentation
56    - username: admin  # username and password for login `manager api`
57      password: apusic$alb
58    - username: user
59      password: user
```

8 产品使用介绍

8.1 产品端口修改

若系统已有服务占用了80、443、9000端口，会导致alb启动失败，可以通过配置修改端口。

8.1.1 修改http或https的端口

1. 修改配置文件：`安装目录/alb-standard/conf/config.yaml`
2. 修改配置文件内容：

```
soft: ALB/2.0.1
alb:
  admin_key:
    - name: "admin"
      key: edd1c9f034335f136f87ad84b625c8f1  # using fixed API token
has security risk, please update it when you deploy to production
environment
      role: admin
  node_listen: 80        # http默认的网关访问入口

  ssl:
    listen_port: 443     # https访问网关的入口
```

3. 如上，修改对应的 `node_listen` 、 `ssl.listen_port` 端口即可。
4. 重新启动ALB： `./bin/start-alb.sh`

8.1.2 修改WEB管理控制台端口

1. 修改配置文件：`安装目录/alb-dashboard/conf/config.yaml`
2. 修改配置文件内容：

```
conf:
  listen:
    host: 0.0.0.0
    port: 9000   # WEB管理控制台默认端口
```

3. 如上，修改对应的 `listen.port` 端口即可。
4. 重新启动ALB： `./bin/start-alb.sh`

8.2 ALB迁移nginx配置文件

ALB支持兼容nginx的server、upstream等配置内容，可以直接把对应匹配提取问单独的配置，并开启alb导入nginx配置，进行导入即可。

8.2.1 开启支持导入nginx的配置文件

- 修改配置文件，支持导入nginx配置文件

```
nginx_config_include:
  nginx_config_include_enable: true      # 开启nginx配置导入
  nginx_config_include_path: "安装目录/alb-
standard/conf/nginx_conf/*.conf"   #待导入的nginx配置文件存放文件夹
```

- 创建 `安装目录/alb-standard/conf/nginx_conf` 目录， `mkdir conf/nginx_conf`

8.2.2 导入nginx配置文件

根据上面创建的nginx配置存放文件夹： `安装目录/alb-standard/conf/nginx_conf/` ，把nginx的upstream或者server块内容的配置文件即可。如下配置 nginx_test.conf

```
server {
    listen       4545;       #监听端口
    server_name  127.0.0.1; #监听地址
    location  ~*^.+$ {       #请求的url过滤，正则匹配，~为区分大小写，~*为不
区分大小写。
        #root path;           #根目录
        #index vv.txt;        #设置默认页
        proxy_pass  http://mysvr;   #请求转向mysvr 定义的服务器列表
        deny 127.0.0.1;   #拒绝的ip
        allow 172.18.5.54; #允许的ip
    }
}
```

注:

1. 配置文件仅支持server和upstream块内容。

8.3 国密支持

国密算法仅可通过Nginx配置文件实现，具体操作方法为：

1. 开启ALB支持Nginx Server配置块导入
2. 编写国密证书的Server内容文件并放入配置文件夹
3. 重启ALB

8.3.1 开启ALB支持Nginx Server配置块导入

1. 进入ALB安装目录，编辑 `安装目录/alb-standard/conf/config.yaml` 文件，设置 nginx_conf_include_enable 修改为true，并确定nginx配置文件目录

```
nginx_config_include:
  nginx_config_include_enable: true      #开启nginx配置导入
  nginx_config_include_path: "/opt/ALB-V2.0.5-EE-arm64/alb-
standard/conf/nginx_conf/*.conf"     #nginx配置导入的位置
```

2. 创建配置导入目录： `mkdir 安装目录/alb-standard/conf/nginx_conf`

8.3.2 编写国密证书的Server内容文件并放入配置文件夹

编写国密双证书的server.conf文件，并放入 `安装目录/alb-standard/conf/nginx_conf` ，如下为样例：

```
server {
    listen 443 ssl;    # 使用SSL标记当前端口开启https
    server_name your_domain.com;    #  SSL证书对应的域名

    # 国密服务端签名证书和密钥，一般名称中携带sig字样（sig证书必须先配置）
    ssl_certificate /path/to/your_domain_name.sig.crt.pem;       # 服
务端签名证书文件
    ssl_certificate_key /path/to/your_domain_name.sig.key.pem;   # 服
务端加密证书私钥文件

    # 国密服务端证书和密钥，一般名字中携带enc（enc证书放在sig证书后面）
    ssl_certificate /path/to/your_domain_name.enc.crt.pem;       # 服
务端加密证书文件
```

```
    ssl_certificate_key /path/to/your_domain_name.enc.key.pem;   # 服务端加密证书私钥文件

    ssl_protocols TLSv1.2 TLSv1.3; # 使用更安全的协议版本
    ssl_prefer_server_ciphers on;   # 优先使用服务器定义的加密套件
    ssl_ciphers HIGH:!aNULL:!MD5;   # 使用更安全的加密算法

    location / {
        root /var/www/html;
        index index.html index.htm;
    }
}
```

导入完后，重启ALB生效

8.4 非root启动ALB

ALB企业版支持通过修改程序权限，使得在普通用户下可以监听80端口。ALB企业版提供两种方式实现非root用户监听80端口，一种是使用ALB自带脚本，一种是使用shell命令。

8.4.1 方法一：使用ALB自带脚本开启非root监听80端口

1. 切花到root用户。
2. 切换到ALB安装目录下的utils/normal-user目录
3. 执行 `./set-normal-user-listen80.sh`
4. 使用vim修改文件： `vim 安装目录/alb-standard/conf/config-default.yaml`

```
nginx_config:     # 114行
  user: alb       # 去掉注释，修改为目标用户
```

5. 执行完后，切换到普通用户启动ALB即可。

8.4.2 方法二：使用shell命令

注： 请严格按照顺序操作，若失败，删除安装包，重新部署再试。

1. 在**普通用户**下，切换到ALB安装目录。
2. 使用ALB企业版自带工具patchelf添加动态连库地址： `./alb-deps/bin/patchelf --add-rpath ./alb-deps/luajit/lib ./alb-deps/alb-core/sbin/alb`

3. 创建标记文件: `touch ./alb-deps/first`
4. 切换到**root用户**，并切换到ALB安装目录。
5. 授权监听80端口： `setcap cap_net_bind_service=+eip ./alb-deps/alb-core/sbin/alb`
6. 切换回**普通用户**
7. 使用vim修改文件： `vim 安装目录/alb-standard/conf/config-default.yaml` 的#user root，去掉注释，把root改为目标用户，如: `user alb`

```
nginx_config:     # 114行
  user: alb       # 去掉注释，修改为目标用户
```

7. 执行完后后，使用普通目标普通用户启动ALB即可。

8.5 ALB开机自启动和系统服务

ALB默认不开启开机自动启动，如需开机启动，切换到安装目录，执行

```
cd /opt/ALB-V2.0.5-EE-amd64            # 进入安装目录
bash utils/systemd/enable_startup.sh    # 自动设置开机启动项。
```

上述命令脚本执行后，可以通过 `systemctl start alb.service` 启动ALB，或者通过 `systemctl stop alb.service` 停止ALB。

8.6 Prometheus监控

ALB企业版提供Prometheus监控功能，默认不启用，如需启用，请切换到utils/exporter目录，执行： `./start-exporter.sh` ，启动ALB的exporter。

8.6.1 启动说明

启动ALB的exporter必须要在alb配置文件中开启stub_status功能，如下：

```
server {
  listen 8080;              # node_status的端口
  location /node_status {   # 必须要使用node_status
        stub_status on;     # 开启stub_status
        access_log off;
        allow 127.0.0.1;
```

```
    }
}
```

在执行start-exporter脚本后，要求输入

- node_status对应的server监听端口，默认为8080.
- exporter监听端口，默认为9113

如下为输入过程：

```
root@root:/utils/expoter$ ./start-exporter.sh
input alb server listening port(请输入node_status监听端口)：
alb server listening port(ALB node_status端口)：  8080
input exporter listening port(请输入监听端口)：
exporter listening port is 9113
exporter are running now!(exporter启动成功)
```

8.6.2 Prometheus监控数据获取

通过start-exporter成功后，可以通过浏览器或者prometheus访问：`http://IP:9113/metrics`获取监控数据。

注： 如果上面修改了exporter的监听端口，上面url中的9113也一并修改。

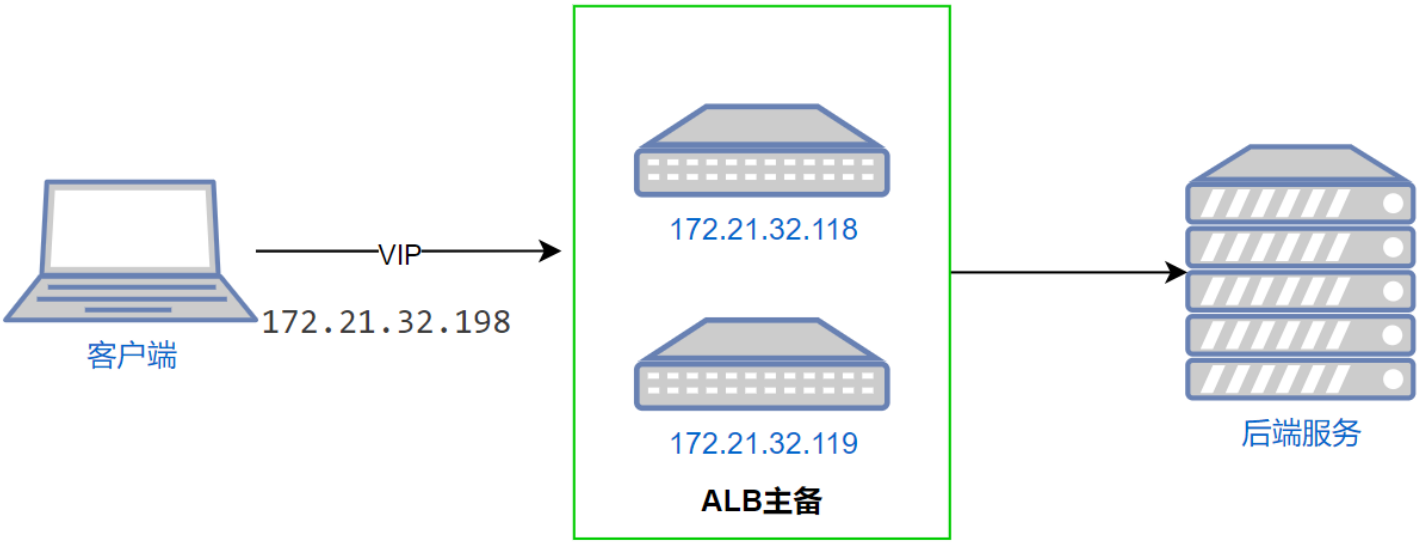
9 ALB主备搭建

ALB支持原生高可用和keepalived两种高可用方式，搭建高可用集群，当集群节点发生故障时，自动将流量切换至备节点。

9.1 前提准备

1. 两台服务器，一台作为主节点（Master），一台作为备用节点（Backup）。
2. 在两台服务器上部署 ALB 服务。（安装过程忽略）
3. 在两台服务器的局域网中，准备一个未使用的IP地址作为虚拟IP（VIP）。
4. 确保两台服务器之间的网络通信正常。
5. 确保两台服务器上的防火墙允许 keepalived 和 ALB 服务的通信。
6. 安装并配置 keepalived。
7. 验证VIP漂移。

下面以两台服务器（172.21.32.118、172.21.32.119）为例，选择VIP为172.21.32.198。



如上图，通过搭建两套ALB，并配置VIP，实现客户端通过VIP访问ALB，当VIP指向的ALB节点宕机的时候，VIP自动切换至备节点。

9.2 原生高可用

ALB自带高可用组件，可一键部署和启动ALB。

9.2.1 查看物理网卡名称

1. 在主节点和备用节点上执行以下命令，查看当前机器IP的物理网卡名称：

```
ip addr show
```

2. 记录物理网卡名称，例如：eth0。

9.2.2 配置alb-ha

主节点

1. 角色配置为: master
2. 修改VIP地址
3. 修改网卡名称

备节点

1. 角色配置为: backup
2. 修改VIP地址
3. 修改网卡名称

9.2.2.1 主节点

1. 切换至utils/alb-ha目录，编辑config.yaml文件，如下样例

```
keepalived:
  # 虚拟路由ID
  vroute_id: 239

  # 虚拟IP绑定网卡名称
  interface: eth0

  # 虚拟IP地址
  vip: 172.21.32.198

  # 协议类型，可选值为ipv4和ipv6
  protocol: ipv4

  # 是否抢占模式
  #如果启用了抢占模式（Preempt Mode），当一个更高优先级的路由器加入到 VRRP 组中时，它可以取代当前的活动路由器成为新的活动路由器。
  #如果未启用抢占模式，则即使有更高优先级的路由器加入，当前的活动路由器也不会被取代
```

```
  preempt: true

  # 发送消息间隔(毫秒)
  msg-send-interval: 800

  # 优先级, 0-255
  priority: 100

  # 角色，可选值为master和backup
  role: master

alb-checker:

  # 检查alb的状态时间间隔(毫秒)
  check-interval: 1000

  # 重试次数
  retry-count: 3

  # 重试间隔(毫秒)
  retry-interval: 2000

  # 当前alb状态检查url，用于检查alb是否健康
  health-url: http://localhost:8080/node_status

  # 自定义健康检查返回的url状态码
  health-codes:
    - 200
    - 404

  # 当节点故障时候，是否尝试重启alb(默认为false)
  restart-alb-on-failure: false

alb-ha:
```

```
  # 日志文件路径
  log-file: alb-ha.log
```

9.2.2.2 备节点

```
keepalived:
  # 虚拟路由ID
  vroute_id: 239

  # 虚拟IP绑定网卡名称
  interface: eth0

  # 虚拟IP地址
  vip: 172.21.32.198

  # 协议类型，可选值为ipv4和ipv6
  protocol: ipv4

  # 是否抢占模式
  #如果启用了抢占模式（Preempt Mode），当一个更高优先级的路由器加入到 VRRP 组中时，它可以取代当前的活动路由器成为新的活动路由器。
  #如果未启用抢占模式，则即使有更高优先级的路由器加入，当前的活动路由器也不会被取代
  preempt: true

  # 发送消息间隔(毫秒)
  msg-send-interval: 800

  # 优先级，0-255
  priority: 100

  # 角色，可选值为master和backup
  role: backup

alb-checker:
```

```
  # 检查alb的状态时间间隔(毫秒)
  check-interval: 1000

  # 重试次数
  retry-count: 3

  # 重试间隔(毫秒)
  retry-interval: 2000

  # 当前alb状态检查url，用于检查alb是否健康
  health-url: http://localhost:8080/node_status

  # 自定义健康检查返回的url状态码
  health-codes:
    - 200
    - 404

  # 当节点故障时候，是否尝试重启alb(默认为false)
  restart-alb-on-failure: false

alb-ha:
  # 日志文件路径
  log-file: alb-ha.log
```

9.2.3 注册系统服务并启动系统服务

主备节点均需要执行以下操作：

1. 切换到sys-service目录
2. 执行：`./setup-alb-ha-service.sh`自动注册系统服务
3. 启动：`systemctl start alb_ha.service`

9.2.4 验证VIP是否自动飘逸测试步骤

1. 使用VIP 172.21.32.198 访问ALB正常。
2. 停掉主节点所在的ALB，继续使用VIP访问ALB，验证是否正常。

9.2.5 其他配置说明：

1. 自定义健康检查URL： 修改配置文件中的health-url: http://localhost:8080/node_status

2. 自定义URL的健康HTTP状态码： 修改配置中的：health-codes: 200, 404
3. 配置alb-ha日志文件路径： 修改配置文件中的log-file: alb-ha.log

9.3 使用keepalived实现高可用

9.3.1 操作步骤汇总

1. 修改ALB启动方式和数据库连接
2. 配置keepalived软件
3. 启动ALB、和keepalived

9.3.2 修改A节点启动方式脚本

切换到安装目录 /opt/ALB-V2.0.5-EE-arm64，编辑start.sh脚本，如下所示，修改150行为下面注释所示

```
nohup_run_etcd() {
    # etcd 不支持arm架构
    if [ "$system_arch" = "arm" ];then
        export  ETCD_UNSUPPORTED_ARCH=arm64
    fi
    # 注释下面一样,
    # nohup etcd --data-dir $etcd_data_dir >> $etcd_log_path 2>&1 &
    # 改为这一行,
    nohup etcd --data-dir $etcd_data_dir  --listen-client-
urls="http://172.21.32.118:2379" --advertise-client-
urls="http://172.21.32.118:2379" >> $etcd_log_path 2>&1 &
}
```

这是一行内容，不能换行

上面的IP地址：172.21.32.118需要改为主节点的IP地址。

9.3.3 修改AB两个节点的数据库配置

- 修改alb连接数据库的配置 `vim 安装目录/alb-dashboard/conf/config.yaml` ,在文件结尾加入下面三行：

```
etcd:
  host:
    - "http://172.21.32.118:2379"
```

- 修改alb管控台的数据库配置 `vim 安装目录/alb-dashboard/conf/conf.yaml` , 修改第10行的配置：为节点A的IP地址,如下所示

```
etcd:
  endpoints:           # 支持多个地址，支持集群
   - 172.21.32.118:2379  # 主节点的IP地址
```

修改完成后，执行stop.sh然后start.sh启动两个alb节点

9.3.4 主节点配置keepalived（A节点）

创建 `/etc/keepalived/keepalived.conf` 文件，在主节点A节点添加下面内容（附件文件keepalivedmaster.conf）

- interface enp0s31f6这个网卡enp0s31f6修改为当前节点物理网卡名称。
- 在virtual_ipaddrss中，把虚拟IP填上。

```
! Configuration File for keepalived

global_defs {
   router_id alb
}

vrrp_script chk_http_port {
   script "/etc/keepalived/checkalb.sh"
   interval 2 #（检测脚本执行的间隔）
   weight 2
}

vrrp_instance VI_1 {
    # state BACKUP
    state MASTER
    # 备份服务器上将 MASTER 改为 BACKUP
    interface enp0s31f6
    virtual_router_id 51
    # 主、备机的 virtual_router_id 必须相同
    priority 100
    # 主、备机取不同的优先级，主机值较大，备份机值较小
    advert_int 1
    authentication {
        auth_type PASS
```

```
        auth_pass 1111
    }
    virtual_ipaddress {
        172.21.32.198 # 请根据需要，修改这个虚拟IP地址
    }
    track_script {
        chk_http_port
    }
}
```

9.3.5 主节点（A节点）创建健康检测脚本

创建健康检查脚本 `/etc/keepalived/checkalb.sh` ，并添加下面内容

```
#!/bin/bash
alb_count=$(ps -ef|grep "alb: main process" | grep -v grep |wc -l)
#1.判断ALB是否存活,如果不存活停止keepalived
if [ $alb_count -eq 0 ];then
    sleep 3
    #2.等待3秒后再次获取一次alb状态
    alb_count=$(ps -ef|grep "alb: main process" | grep -v grep |wc -
l)
    #3.再次进行判断， 如alb还不存活则停止Keepalived,让地址进行漂移,并退出脚本
    if [ $alb_count -eq 0 ];then
        echo "alb is down"
        exit 1
    fi
fi
```

9.3.6 从节点B配置keepavlied（B节点）

创建 `/etc/keepalived/keepalived.conf` 文件，在从节点B节点添加下面内容（附件文件keepalived-backup.conf）

- 注： interface enp0s31f6这个网卡enp0s31f6修改为当前节点物理网卡名称。

```
! Configuration File for keepalived
global_defs {
    router_id alb
}
vrrp_script chk_http_port {
    script "/etc/keepalived/checkalb.sh"
    interval 2 # (检测脚本执行的间隔)
    weight 2
}
vrrp_instance VI_1 {
    state BACKUP
    # state MASTER
    # 备份服务器上将 MASTER 改为 BACKUP
    interface enp0s31f6
    virtual_router_id 51
    # 主、备机的 virtual_router_id 必须相同
    priority 90
    # 主、备机取不同的优先级，主机值较大，备份机值较小
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        172.21.32.198 # 请根据需要，修改这个虚拟IP地址
    }
    track_script {
        chk_http_port
    }
}
```

9.3.7 备节点创建健康检测脚本

创建健康检查脚本`/etc/keepalived/checkalb.sh`，并添加下面内容

```
#!/bin/bash
alb_count=$(ps -ef|grep "alb: main process" | grep -v grep |wc -l)
#1.判断ALB是否存活,如果不存活停止keepalived
if [ $alb_count -eq 0 ];then
    sleep 3
    #2.等待3秒后再次获取一次alb状态
    alb_count=$(ps -ef|grep "alb: main process" | grep -v grep |wc -
l)
    #3.再次进行判断, 如alb还不存活则停止Keepalived,让地址进行漂移,并退出脚本
    if [ $alb_count -eq 0 ];then
        echo "alb is down"
        exit 1
    fi
fi
```

9.3.8 启动keepalived

- 在A、B两个节点使用命令启动keepalived： keepalived -f /etc/keepalived/keepalived.conf
- 使用虚拟IP访问ALB的管理控制台和代理地址。

10 常见问题与解决

10.1 启动错误： ETCD连接失败

当执行start脚本后，出现以下etcd连接失败问题，可以通过修改启动脚本，延时启动alb。

```
Warning! Request etcd endpoint 'http://127.0.0.1:2379/version'
error, closed, retry time=1
Warning! Request etcd endpoint 'http://127.0.0.1:2379/version'
error, closed, retry time=2
reqeust etcd endpoint 'http://127.0.0.1:2379/version' error, closed
```

10.1.1 问题解决

1、修改启动脚本 `【安装目录】/bin/start-alb.sh` ，等待etcd启动完成。

```
# 略
 nohup_run_etcd     # 68行

 sleep 15            # 将sleep 5改为sleep 15
 # 启动管控台
 nohup_run_alb_dashboard()
```

如上图所示，在70行修改 `sleep 15` ，保存退出后，重启ALB

2、排查本机是否已经存在etcd数据库，如果有，则是端口和服务冲突。

如果存在etcd，则需要修改etcd启动监听端口

- 修改启动脚本bin/start-alb.sh 的第10行，将2379修改为2479

```
# etcd配置项,
etcd_data_dir=$parent_dir/alb-deps/etcd-data
etcd_log_path=$parent_dir/alb-deps/logs/etcd.log
etcd_listen_port=2479    # 修改此端口
```

```
alb_dashboard_service="manager-api"
etcd_service="etcd"
```

- 修改alb配置文件中的etcd端口: 【安装目录】/alb-standard/conf/config-default.yaml

```
# 修改第215行的端口为2479
etcd:
  host:
    - "http://127.0.0.1:2479"
```

- 重新启动ALB

10.2 启动错误：DNS is Empty（缺失dns服务器）

所在机器环境无DNS，则需要手动开启内置的DNS配置。

> *配置文件地址：* *安装目录*/alb-standard/conf/config.yaml

去掉前面的注释，即可开启DNS。

注意dns_resolver对齐到上面的enable_control

```
alb:
  enable_control: false   #开启会监听9090端口。

# 如果出现 local DNS is empty，那么把下面三行注释去掉即可，注意对齐到上面的
enable_control。
  dns_resolver:
    - 8.8.8.8
    - 114.114.114.114
php:
  php_enable: false
```

10.3 启动错误： getgrnam("nobody") failed

```
root@user-PC:/opt/ALB-V2.0.5-EE-arm64# ./bin/start-alb.sh
alb: [emerg] : getgrnam("nobody") failed
./bin/start-alb.sh: line 29: 11777 Killed                  nohup
./manager-api >> dashboard.log 2>&1   (wd: /opt/ALB-V2.0.5-EE-
arm64/alb-dashboard)
Start ALB failed, please check alb-stardard-2.0/logs/error.log
```

出现上面错误，是ALB进程的启动用户授权失败导致。

10.3.1 问题解决

修改alb配置文件：`安装目录/alb-standard/conf/config-default.yaml`的第115行，将`user: nobody`改为`user: root`

```
nginx_config
  user: root    # 取消注释，并且修改为root
```

修改完成后，重启ALB即可。

10.4 启动错误： failed to load the 'resty.core' module

```
[root@localhost ALB-V2.0.5-EE-arm64]# ./bin/start-alb.sh
alb: [alert] : failed to load the 'resty.core' module
(https://github.com/openresty/lua-resty-core); ensure you are using
an OpenResty release from https://openresty.org/en/download.html
(reason: /usr/local/openresty/lualib/resty/core/base.lua:24:
ngx_http_lua_module 0.10.27 required) in /root/temp/alb-test/alb-
ee/ALB-V2.0.5-EE-arm64/alb-standard/conf/alb.conf:345
./bin/start-alb.sh: line 29: 3217055 Killed                nohup
./manager-api >> dashboard.log 2>&1   (wd: ~/temp/alb-test/alb-
ee/ALB-V2.0.5-EE-arm64/alb-dashboard)
Start ALB failed, please check alb-stardard-2.0/logs/error.log
```

出现上面错误，是因为和本机原有的openresty或者旧版ALB冲突

10.4.1 问题解决

1、检查本机是否安装过旧版ALB，如果安装过，请卸载。

如openresty默认安装位置 `/usr/local/openresty` ，请删除该目录。

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com