Apusic 金蝶天燕

生成时间：2026-07-13

安装手册

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V2.0.6标准版的安装手册，详细介绍如何在各操作系统上安装、部署和授权ALB，以及ALB服务的启停操作。

1.1 适用对象

本文档适用于IT信息化业务负责人、研发经理、软件项目经理、软件架构师、运维工程师。

1.2 相关文档

了解更多ALB V2.0.6产品相关的信息，请参阅以下ALB V2.0.6产品手册文档集：

序号	手册文档	说明
1	金蝶Apusic负载均衡器软件 V2.0.6 发版说明	介绍了ALB 产品版本更新内容 。
2	金蝶Apusic负载均衡器软件 V2.0.6 快速使用手册	简单介绍了如何快速上手使用ALB 。
3	金蝶Apusic负载均衡器软件 V2.0.6 安装手册	详细介绍如何在各操作系统上安装ALB，以及ALB服务启停操作，产品的注册过程。
4	金蝶Apusic负载均衡器软件 V2.0.6 代理功能手册	详细介绍 ALB 相关功能的使用、配置、管理及配套工具的使用方法。
5	金蝶Apusic负载均衡器软件 V2.0.6 管控台用户手册	详细介绍ALB管控台相关功能的使用和操作说明。
6	金蝶Apusic负载均衡器软件 V2.0.6 开发手册	详细介绍ALB外部API、alb-cli命令行工具及AI Agent集成方法。
7	金蝶Apusic负载均衡器软件 V2.0.6 迁移手册	详细介绍ALB历史版本迁移升级到V2.0.6版本的说明。
8	金蝶Apusic负载均衡器软件 V2.0.6 运维手册	详细介绍ALB的监控、运维、安全加固等运维说明。
9	金蝶Apusic负载均衡器软件 V2.0.6 性能优化手册	详细介绍ALB性能调优的说明。

1.3 技术支持

ALB产品提供全面的技术支持服务，您可以通过以下方式获得技术支持：

- 网址：www.apusic.com
- 电话：400-855-5800
- 邮箱：support@apusic.com
- 金蝶云社区：https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN

您在取得技术支持时，请提供如下信息：

1. 您的姓名
2. 公司信息与联系方式
3. 操作系统及其版本
4. 产品版本号
5. 出现异常及错误的日志、截图等详细信息

2 产品安装

2.1 安装包说明

ALB标准版安装包名称结构是：

`ALB-版本号-SE-构建日期-CPU架构.tar.gz` 或 `ALB-版本号-SE-构建日期-CPU架构-ebpf.tar.gz`

目前仅支持Linux平台的arm64和amd64架构:

- Linux-arm64架构安装包： `ALB-V2.0.6-SE-构建日期-arm64.tar.gz`
- Linux-amd64(x86_64)架构安装包: `ALB-V2.0.6-SE-构建日期-amd64.tar.gz`
- Linux-arm64架构 `ebpf` 特性安装包： `ALB-V2.0.6-SE-构建日期-arm64-ebpf.tar.gz`
- Linux-amd64(x86_64)架构 `ebpf` 特性安装包: `ALB-V2.0.6-SE-构建日期-amd64-ebpf.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。

注： `ebpf` 特性安装包，Linux内核版本不低于 `4.19` .

2.1.1 安装介质

- tar.gz压缩包： `ALB-版本号-SE-构建日期-CPU架构.tar.gz` 解压到目标机器任意路径即可使用。
- docker镜像： `ALB-版本号-SE-Docker-构建日期-CPU架构-docker.tar.gz` 或 `ALB-版本号-SE-Docker-构建日期-CPU架构-基础操作系统.tar.gz` （基础操作系统如 kylin、ubuntu 等，请以实际安装包为准）。
- RPM包： `ALB-版本号-SE-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-V2.0.6-SE-amd64` （路径中的版本号请以实际安装包为准）。

2.2 单节点部署

1. 获取软件安装包 ALB安装介质为: ALB-V2.0.6-SE-amd64.tar.gz或ALB-V2.0.6-SE-arm64.tar.gz
2. 复制安装包到目标主机
3. 解压: ALB-V2.0.6-SE-amd64.tar.gz到目标安装路径（示例为/opt目录）

`tar -zxvf ALB-V2.0.6-SE-amd64.tar.gz -C /opt/` 4. ALB使用参考《快速使用手册》。

注：如果没有权限，请使用root账户或sudo

2.3 集群部署

在主备节点上分别执行单节点安装步骤安装完毕后，参考本文档中的"高可用集群搭建"章节进行配置即可。

2.4 Docker部署

ALB标准版提供amd64和arm64架构的Docker系统安装包，可通过ALB Docker安装包直接部署ALB。

2.4.1 安装包说明

1. ALB压缩安装包名称

- `ALB-V2.0.6-SE-docker-amd64.tar.gz` 或 `ALB-V2.0.6-SE-Docker-20260421-amd64-kylin.tar.gz`
- `ALB-V2.0.6-SE-docker-arm64.tar.gz` 或 `ALB-V2.0.6-SE-Docker-20260421-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-V2.0.6-SE-Docker-20260421-amd64-kylin
ALB-V2.0.6-SE-Docker-20260421-amd64-kylin           # docker安装包解压后目录
|-- ALB-V2.0.6-SE-Docker-20260421-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-dockercompose.yaml                    # docker compose配置文件
```

2.4.2 安装步骤

下面的操作以ALB-V2.0.6-SE-Docker-20260421-amd64-kylin.tar.gz为例

1. 解压和导入镜像

- 解压 `tar -zxvf ALB-V2.0.6-SE-Docker-20260421-amd64-kylin.tar.gz`
- 进入解压后的文件夹 `cd ALB-V2.0.6-SE-Docker-20260421-amd64-kylin`

- 加载镜像：`docker load < ALB-V2.0.6-SE-Docker-20260421-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.lic：本地旧版授权文件*
> - *license.xml：本地授权文件，支持旧授权和 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，需要挂载到容器路径：`/opt/ALB-V2.0.6-SE/acls.properties`*

3. 镜像启动与停止

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

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

```
version: '3'
services:
  alb:
    image: harbor.apusic.com/apusic/alb:V2.0.6-se.20260421-kylin-amd64    # ALB 镜像
    ports:
      - 8080:8080           # ALB http对外端口
      - 8887:8887           # ALB 管控台端口
    volumes:
      - ./alb/alb.conf:/opt/ALB-V2.0.6-SE/conf/alb.conf       # alb配置文件
      - ./alb/license.xml:/opt/ALB-V2.0.6-SE/license.xml      # alb授权文件
      - ./alb/logs:/opt/ALB-V2.0.6-SE/logs                    # alb访问和错误日志存储目录
    command: bash /opt/ALB-V2.0.6-SE/bin/start-alb-and-console.sh
```

2.5 K8s部署

alb标准版支持通过Kubernetes进行部署，以下是详细的部署步骤。

2.5.1 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-V2.0.6-SE-Docker-20260421-amd64-kylin.tar.gz
# 切换到解压后的安装包
cd ALB-V2.0.6-SE-Docker-20260421-amd64-kylin
# 导入ALB镜像
docker load < ALB-V2.0.6-SE-Docker-20260421-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:V2.0.6-
se.20260421-kylin-amd64
# 推送镜像至镜像仓库
docker push harbor.apusic.com/apusic/alb:V2.0.6-se.20260421-
kylin-amd64
```

2.5.2 配置文件与授权管理

- 创建 ConfigMap 将 `alb.conf` 配置文件挂载至容器：

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

- 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
```

2.5.3 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:V2.0.6-se.20260421-kylin-amd64 # 请根据仓库地址进行修改
        command: ["bash", "/opt/ALB-V2.0.6-SE/bin/start-alb-and-console.sh"]
        ports:
        - containerPort: 8080
        - containerPort: 8887
        volumeMounts:
        - name: alb-config
          mountPath: "/opt/ALB-V2.0.6-SE/conf/alb.conf"
          subPath: "alb.conf"
        - name: alb-license
          mountPath: "/opt/ALB-V2.0.6-SE/license.xml"
          subPath: "license.xml"
        - name: alb-logs
          mountPath: "/opt/ALB-V2.0.6-SE/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: 8080
    targetPort: 8080
    nodePort: 30080
  - name: console
    port: 8887
    targetPort: 8887
    nodePort: 30887
```

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
```

2.5.4 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`
- 管控台：`http://<节点IP>:30887`

2.5.5 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
```

3 高可用集群搭建

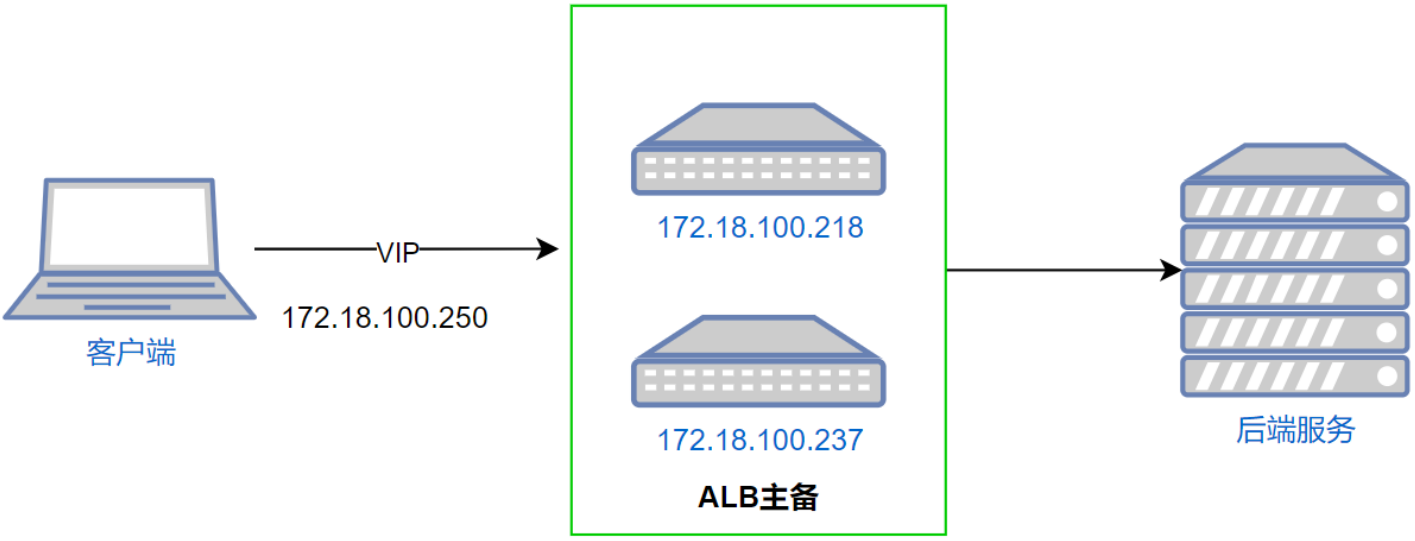
ALB支持原生高可用和keepalived两种高可用方式，搭建高可用集群，当集群节点发生故障时，自动将流量切换至备节点。其中原生高可用可通过ALB管控台配置，也可以通过命令行进行配置。

在ALB的管控台中一键配置和启动高可用，下面介绍在命令行中的安装配置方式。

3.1 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。

下面以两台服务器（192.168.2.10和192.168.2.11）为例，选择VIP为192.168.2.30。

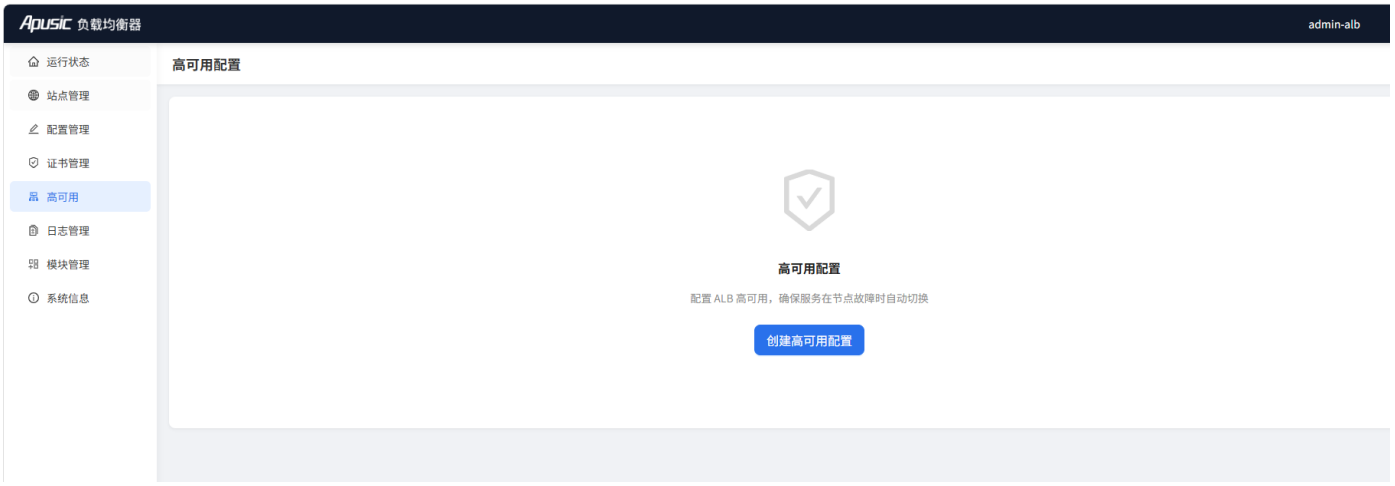


通过搭建两套ALB，并配置VIP，实现客户端通过VIP访问ALB，当主节点宕机时，VIP自动切换至备节点。

3.2 使用管控台配置高可用

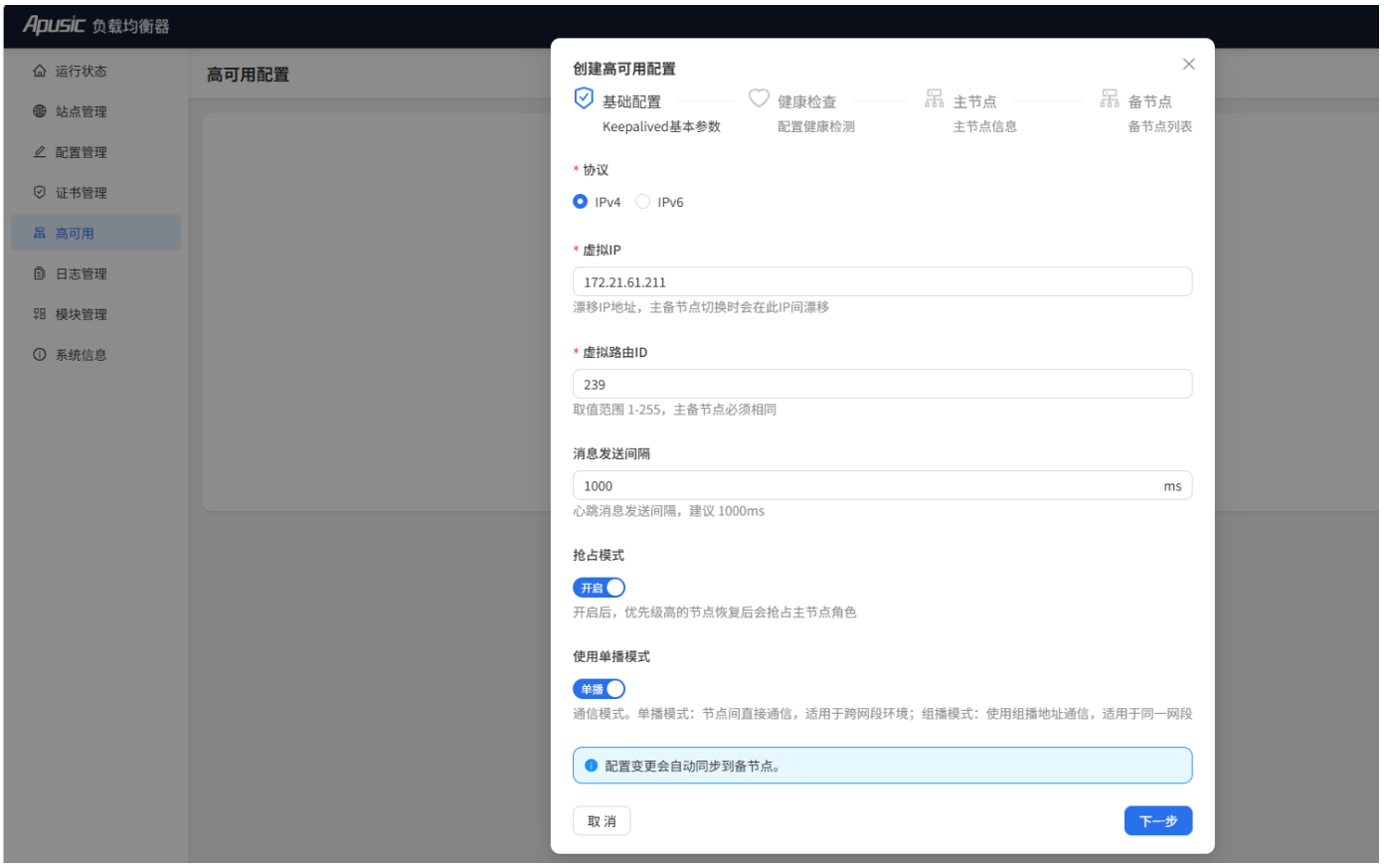
1. 登录ALB管控台，进入高可用配置页面。

2. 点击“高可用”

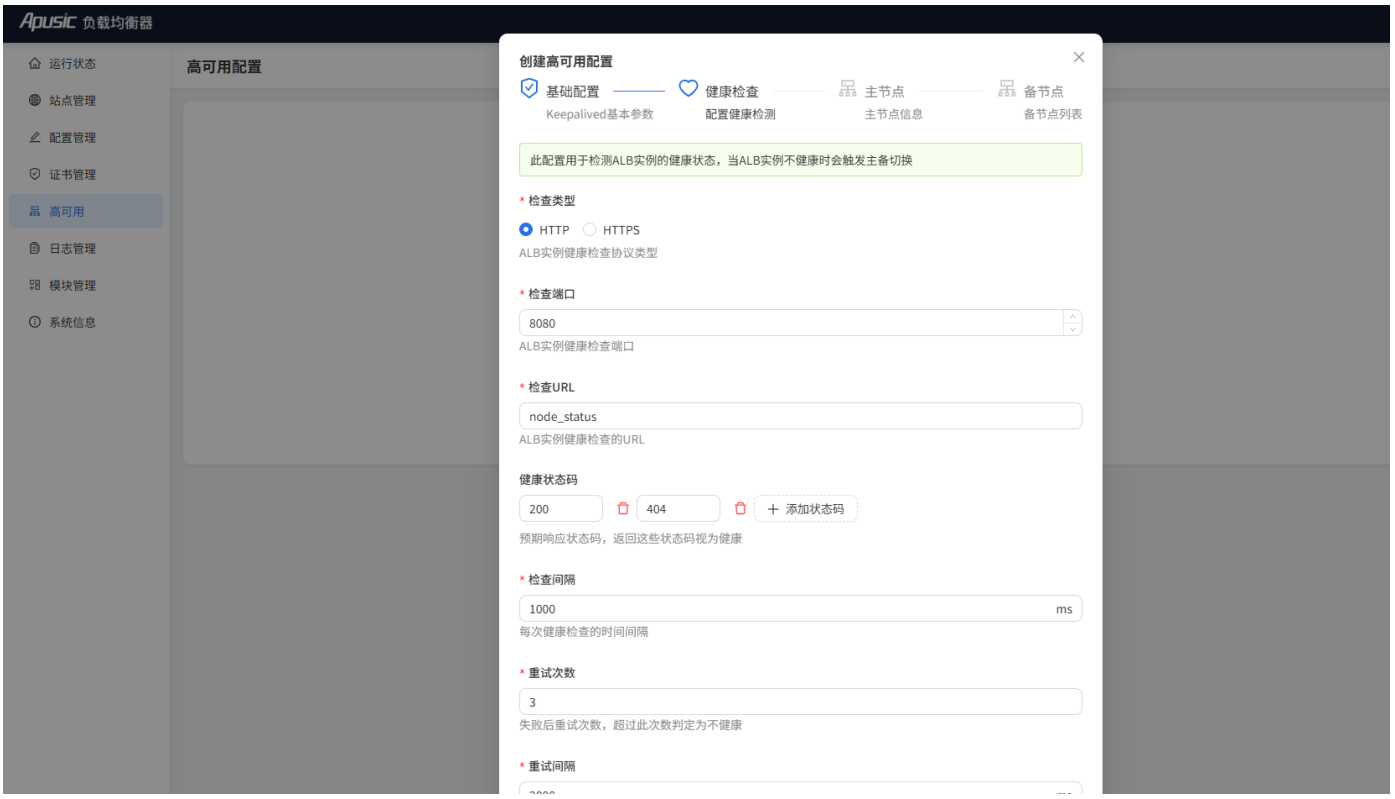


3. 点击“创建高可用”

4. 填写VIP地址，点击下一步

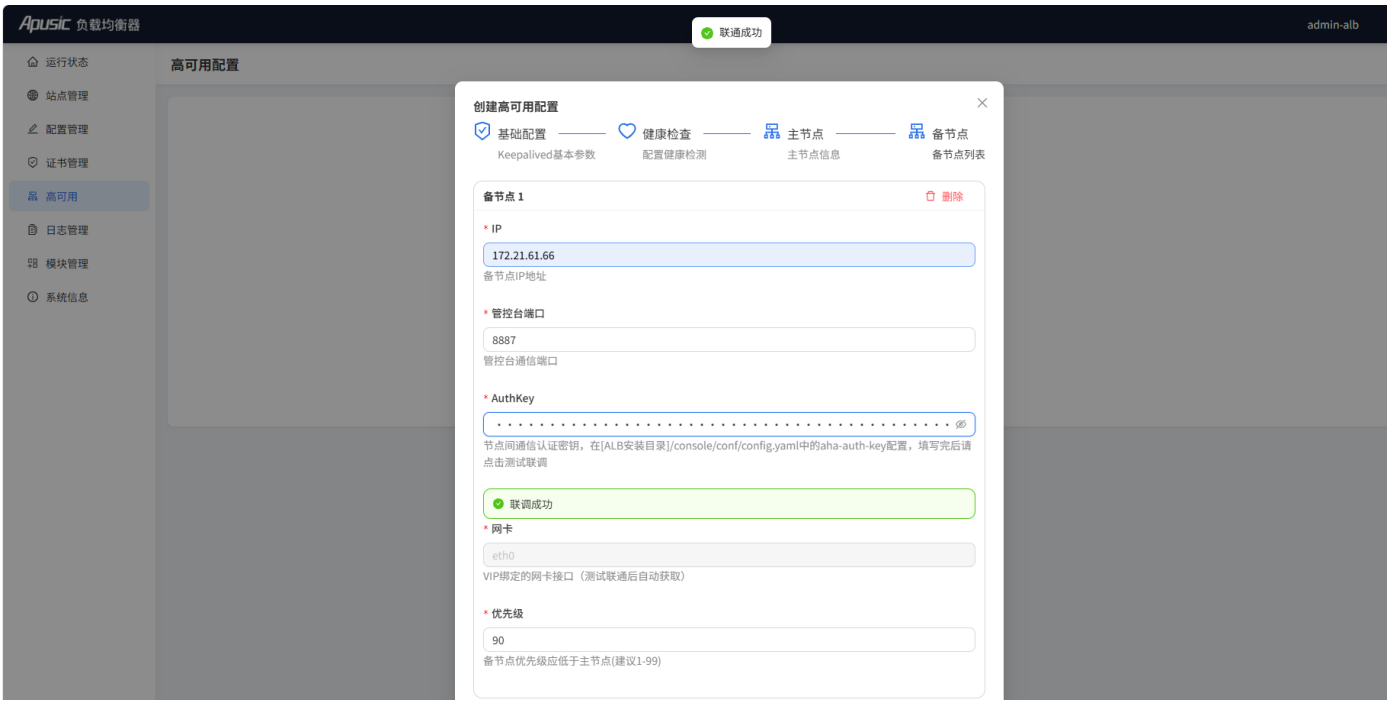


5. 填写健康检查内容（默认无需更改），点击下一步

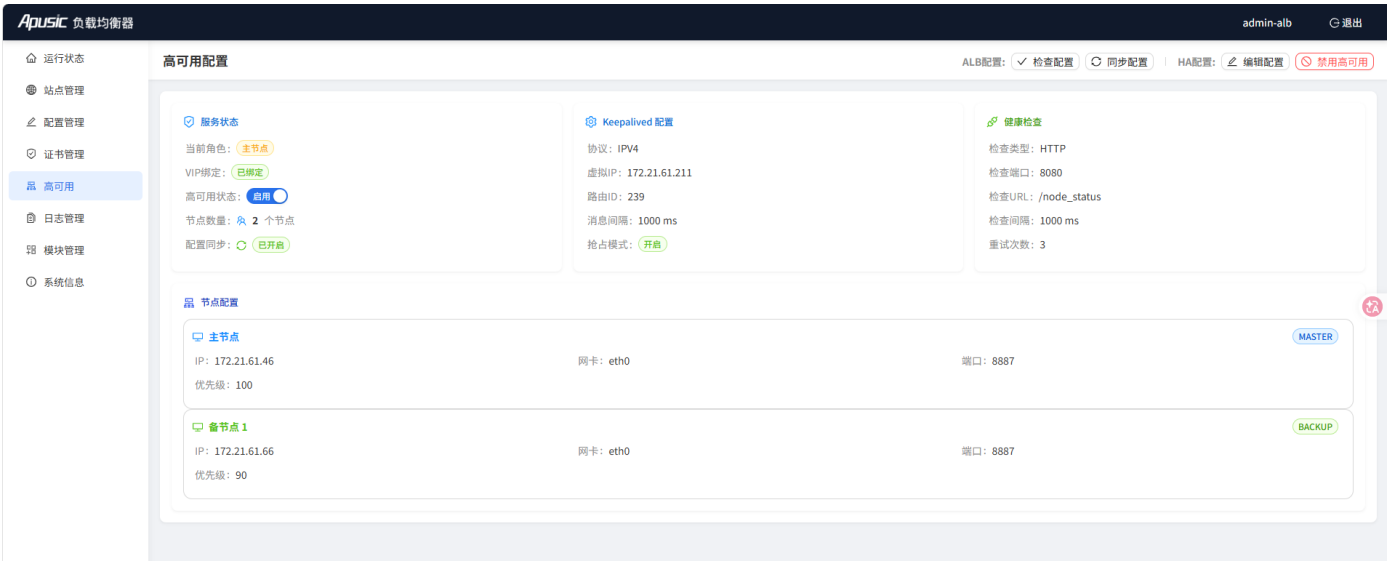


6. 选择主节点IP和网卡，点击下一步

7. 点击添加备节点， 输入备节点的IP和端口和AuthKey，点击联调通过后点击创建



8. 创建成功



3.3 手工配置ALB高可用服务

ALB自带高可用组件，可一键部署和启动高可用服务。在ALB管控台中配置高可用时，默认使用自带高可用组件。

3.3.1 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。

3.3.2 配置aha

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

3.3.2.1 主节点

1. 切换至utils/aha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: 192.168.2.30

  # 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

checker:

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

  # 重试次数
  retry-count: 3

  # 重试间隔(毫秒)
  retry-interval: 2000

  # 健康检查方式，可以url或者shell，url为http或https，shell为shell命令
  health-type: url

  # 当前目标服务状态检查url，用于检查目标服务是否健康
  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

  # shell健康检查命令，shell命令执行返回0表示健康，其他值表示异常
  health-cmd:
```

```
  # 当节点故障时候，是否尝试重启目标服务(默认为false)
  restart-on-failure: false
  # 重启目标服务命令
  restart-cmd:

aha:
  # 日志文件路径
  log-file: aha.log

  # 日志级别，可选值为debug, info, warn, error
  log-level: info
```

3.3.2.2 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: 192.168.2.30

  # 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（备节点需低于主节点）
  priority: 90

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

checker:

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

  # 重试次数
  retry-count: 3

  # 重试间隔（毫秒）
  retry-interval: 2000

  # 健康检查方式，可以url或者shell，url为http或https，shell为shell命令
  health-type: url

  # 当前目标服务状态检查url，用于检查目标服务是否健康
  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

  # shell健康检查命令，shell命令执行返回0表示健康，其他值表示异常
  health-cmd:

  # 当节点故障时候，是否尝试重启目标服务（默认为false）
  restart-on-failure: false
  # 重启目标服务命令
  restart-cmd:
```

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

  # 日志级别，可选值为debug, info, warn, error
  log-level: info
```

3.3.3 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. 切换到安装目录下的 `utils/aha/sys-service` 目录
2. 切换到root用户，或者使用sudo执行下面命令
3. 执行：`./setup-aha-service.sh` 自动注册系统服务
4. 启动：`systemctl start aha.service`

3.3.4 验证VIP是否自动漂移测试步骤

1. 使用VIP 192.168.2.30 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。

3.3.5 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aha.log日志文件路径： 修改配置文件中的log-file: aha.log
4. 通过自定义shell命令或脚本实现健康检查，要求返回0表示健康，其余值在ALB中标记不健康，如下样例：

```
# 使用shell命令进行健康检查
health-type: shell
# 使用shell命令检查alb进程存在则健康
health-cmd: ps aux | grep alb | grep -v grep
```

3.4 使用keepalived实现高可用

3.4.1 安装 keepalived

```
# kylin、opensuler、centos操作系统
yum install -y keepalived

# uos、ubuntu操作
apt-get install -y keepalived
```

3.4.2 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，查看物理网卡名称：

```
ip addr show
```

2. 记录物理网卡名称，例如：eth0。

3.4.3 配置 keepalived

1. 编辑主节点（Master）的 keepalived.conf 文件：

```
vi /etc/keepalived/keepalived.conf
```

在主节点/keepalived.conf文件中添加以下内容：

注：

- 把 `interface eth0` 替换为主节点的物理网卡名称。
- 把 `192.168.2.30` 替换为准备使用的虚拟IP地址。

```
! 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 MASTER
    interface eth0
    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   # 示例值，生产环境请更换为强密码
    }
    virtual_ipaddress {
        # 请根据需要，修改这个虚拟IP地址
        192.168.2.30
    }
    track_script {
        chk_http_port
    }
}
```

2. 编辑备用节点（Backup）的 keepalived.conf 文件：

```
vi /etc/keepalived/keepalived.conf
```

在备用节点的 keepalived.conf 文件中添加以下内容：

注：

- 把 `interface eth0` 替换为备用节点的物理网卡名称。
- 把 `192.168.2.30` 替换为准备使用的虚拟IP地址。

```
! 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 eth0
    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  # 示例值，生产环境请更换为强密码
    }
    virtual_ipaddress {
        192.168.2.30 # 请根据需要，修改这个虚拟IP地址
    }
    track_script {
        chk_http_port
    }
}
```

3. 在**两个节点**创建检测脚本checkalb.sh：

```
vi /etc/keepalived/checkalb.sh
```

在checkalb.sh文件中添加以下内容：

```
#!/bin/bash
# 检测ALB主进程是否存活（根据实际安装路径调整PID文件路径）
alb_pid=$(cat /opt/ALB-V2.0.6-SE-amd64/logs/alb.pid 2>/dev/null)
alb_count=0
if [ -n "$alb_pid" ]; then
    alb_count=$(ps -p "$alb_pid" --no-headers | wc -l)
fi
#1.判断ALB是否存活,如果不存活停止keepalived
if [ $alb_count -eq 0 ];then
    sleep 3
    #2.等待3秒后再次获取一次alb状态
    alb_pid=$(cat /opt/ALB-V2.0.6-SE-amd64/logs/alb.pid 2>/dev/null)
    alb_count=0
    if [ -n "$alb_pid" ]; then
        alb_count=$(ps -p "$alb_pid" --no-headers | wc -l)
    fi
    #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"
        systemctl stop keepalived
        exit 1
    fi
fi
```

在创建检测脚本后，给checkalb.sh添加执行权限：

```
chmod +x /etc/keepalived/checkalb.sh
```

3.4.4 主备节点均启动keepalived服务

```
systemctl start keepalived.service
```

3.4.5 验证VIP是否指向主节点

在主节点上查看虚拟IP地址（把eth0改为主节点网卡名称，192.168.2.30替换为实际VIP）：

```
ip addr show eth0 | grep '192.168.2.30'
```

3.4.6 停止主节点alb，验证VIP是否漂移到备节点

```
# 切换到alb安装目录
./bin/stop-alb.sh
```

在备节点上查看虚拟IP地址（把eth0改为备节点网卡名称）：

```
ip addr show eth0 |grep inet|grep -v inet6|awk '{print $2}'
```

如果VIP漂移到备节点，则说明keepalived配置成功。

4 ALB授权与服务配置

4.1 license使用说明

ALB标准版支持本地授权（`license.lic`或`license.xml`）和统一授权（`acls.properties`），授权文件需要放到产品安装目录下。

- `license.lic`：本地授权文件（旧版授权格式）。
- `license.xml`：本地授权文件（支持旧授权和 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，需要放到 ALB 安装目录下。

4.1.1 本地授权

1. 获取本机特征码

```
# 获取本机特征码
./bin/alb-authcode

# 指定网卡获取特征码
./bin/alb-authcode -ac eth0

# 指定IP地址获取特征码
./bin/alb-authcode -ac 192.168.1.100
```

> *注： -ac参数仅支持alb标准版构建日期大于等于2025年6月及以后版本 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.lic`或`license.xml`，然后放到 ALB 安装目录下。若使用统一授权，请将`acls.properties`放到 ALB 安装目录下。*

4.1.2 统一授权

1. 统一授权中心必须包含ALB标准版的有效授权信息。(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=true                          # 开启变量统一授权
apusic_acls_authUrls=192.168.4.10:6869     # 示例统一授权中心地址，请替换为实际地址
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=192.168.5.10:6869   # 示例统一授权中心地址，请替换为实际地址
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。

4.2 产品端口说明

ALB默认端口如下：

- 8080：http服务端口。
- 8887: ALB管控台端口

如需修改默认http代理端口，可以编辑 `ALB安装目录/conf/alb.conf` 文件中的 `listen 8080` 行。

如需修改管控台的端口，可以编辑 `ALB安装目录/console/conf/config.yaml` 文件中的system中的addr字段端口。

4.3 开机启动项与系统服务

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-V2.0.6-SE-amd64              # 进入安装目录（以实际安装路径为准）
./utils/systemd/enable_startup.sh        # 自动设置开机启动项（通常需要root
```

```
权限)
systemctl start alb                    # 启动ALB
```

4.3.1 移除系统启动项

```
cd /opt/ALB-V2.0.6-SE-amd64            # 进入安装目录
./utils/systemd/remove_startup.sh      # 移除开机启动项
```

4.4 普通用户启动ALB并且开启监听80端口

ALB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。具体操作如下：

1. 切换到**root用户**，并切换到ALB安装目录。
2. 授权监听80端口： `setcap cap_net_bind_service=+eip ./sbin/alb`
3. 切换回**普通用户**

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