Apusic 金蝶天燕

生成时间：2026-05-19

运维手册

金蝶Apusic负载均衡器v2.0

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 ALB快速入门运维

本快速入门指南介绍了金蝶Apusic负载均衡软件v2.0标准版产品安装、启动、卸载、管理与使用等基本操作，为用户快速使用本产品提供指导

1.1 ALB产品介绍

金蝶Apusic负载均衡软件（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。

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、龙芯、兆芯、X86等
国产操作系统	银河麒麟系列、中标麒麟系列、中科红旗、深度等
其他Linux系列	RedHat系列、CentOS、Suse Linux系列等

1.2 产品安装

下面以ARM架构下Kylin系统上安装ALB为例，下载好安装文件alb-standard-2.0-arm.tar.gz后，其安装步骤有：

1. 解压安装包
2. 导入license
3. 启动

1.2.1 解压安装包到指定目录

1. 解压安装包并安装
2. 切换至root用户。
3. 上传安装包至安装服务器的/opt目录下。
4. 解压安装包：`tar -zvxf alb-standard-2.0-arm.tar.gz` 获得alb-standard-2.0文件夹。
5. 进入解压后的文件夹: `cd alb-standard-2.0-arm` 。

备注：默认的安装路径为/opt/目录（支持安装在任意目录）。

1.2.2 导入license

ALB标准版支持金蝶天燕认证、金蝶KBC认证、金蝶统一授权三种模式，默认授权类型为金蝶KBC授权。。

金蝶KBC授权和本地授权，请把**授权文件放置在 `安装目录/alb-standard` 文件夹**下，

1. 授权类型配置

alb授权类型的配置文件为：`安装目录/alb-standard/conf/alb_license.conf`

```
license kbc license.lic;  # 使用KBC授权模式，授权文件为license.lic
```

2. 授权类型配置格式

语句格式规范：`license [授权类型] [授权文件或地址];`

3. 不同授权类型及其配置说明

- local: 表示金蝶天燕本地授权，后面需要紧跟授权文件。
  - 如：`license local license.xml;`，
  - 把本地授权文件license.xml放置在`安装目录/alb-standard`文件夹下。
- kbc: 表示金蝶KBC授权，后面需要紧跟授权文件。如：`license kbc license.lic;`，
  - KBC特征码获取：1、直接启动，2、授权配置文件写：`license kbc;`
  - 执行ALB启动`./bin/start-alb.sh`脚本，获取授权码: `KBC auth: Auth Code is: SZTY2500879438` 。
  - 授权码为SZTY开头的内容,如上为：`SZTY2500879438`
  - 使用授权码在KBC系统中申请授权文件。
  - 获取授权文件后，放入`安装目录/alb-standard`，并确认授权文件名和alb_license.conf配置一致（如上为license.lic文件），重启ALB即可。
  - **注**：在多网卡多ip环境中，可以通过-ac参数指定网卡或ip，如`license kbc license-file.lic -ac eth0`

- center: 表示金蝶天燕统一授权中心，后面需要跟授权服务器地址(IP:端口)、租户名称和命名空间。如：
  `license center 172.21.33.33:6789 tenant namespace;`
  - 如果租户名称不确定，可以填写为public。

4. 通过环境变量设置统一授权配置 alb支持环境变量中设置统一授权的配置，如下关键项：

```
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
```

1.2.3 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-alb.sh`

2 常见问题解决

2.1 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。

2.1.1 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`

2.1.2 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`

2.2 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
```

2.3 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。

2.3.1 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`

2.3.2 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

```
upstream test_server {
    server 172.21.33.14:8999;
}

server {
    listen 8989;
    server_name test.com;

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

    location /api/ {
        proxy_pass http//test_server/api/;
    }
}
```

注:

1. 配置文件仅支持server和upstream块内容。

2.4 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

2.4.1 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

```
php:
  php_enable: false
  php_root_path: "/var/www/html"
  php_listen_port: 9092
  php_server_name: "example.com"
  php_fastcgi_pass: "127.0.0.1:9001"

nginx_config_include:
  nginx_config_include_enable: true
  nginx_config_include_path: "/opt/alb-2.0-kylin-arm/alb_install_dir/alb/conf/nginx_conf/*.conf"
```

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

2.4.2 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` ，如下图为样例：

导入完后，重启ALB生效

2.5 非root启动ALB

很多客户要求使用非root启动alb，在使用root用户安装完毕alb标准版后，可以通过修改安装包的权限使得alb可以使用非root用户启动。

1. 使用root用户安装完毕alb标准版
2. 安装目录使用chown修改权限
3. 修改alb进程配置的用户为非root
4. 修改alb进程权限，支持非root监听80端口
5. 导入license并启动。

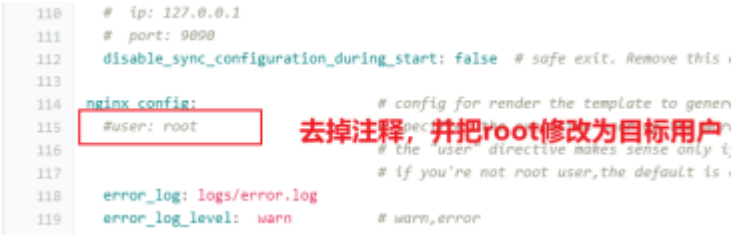
2.5.1 安装目录使用chown修改权限

1. 使用useradd创建用户alb_app，如果已经有用户，则使用已经有的用户名
2. 使用chown修改alb安装目录拥有者为目标用户

```
chown -R alb_app:alb_app alb-2.0-kylin-x86
chown -R alb_app:alb_app /opt/alb-apusic
```

2.5.2 修改alb进程配置的用户为非root用户名

1. 切换到非root用户alb_app， `su alb_app`
2. 使用vim修改文件： `vim` **安装目录**`/alb-standard/conf/config-default.yaml`



2.5.3 修改alb进程权限，支持非root监听80端口

1. 切换到root用户
2. 使用命令： `setcap cap_net_bind_service=+eip /opt/alb-apusic/openresty/nginx/sbin/nginx`
3. 切换回目标用户alb_app
4. 重启ALB即可

2.6 http配置项的导入

由于上面的ALB迁移nginx配置文件只能导入server、upstream块内容，如果需要修改http块里面的内容，可以通过以下方式实现。

1. 编辑配置 **安装目录**`/alb-standard/conf/config.yaml` 文件。
2. 添加nginx配置。
3. 写入指定的http配置内容，还需要注意不能添加的内容。
4. 不能添加的内容通过修改 **安装目录**`/alb-standard/conf/config-default.yaml` 直接修改。
5. 修改完毕后，重启ALB即可生效，如果有错误，则校验yaml文件格式或者添加内容冲突了

2.6.1 编辑并添加http内容

下面以添加http中的senfile、tcp_nodelay等配置为例

注意: 使用yaml格式进行格式化，缩进是两个空格.

2.6.2 不能添加和修改项

有些http的配置项是通过config-default.yaml修改的，而不能通过在上面的方式添加。下面是不能添加的项目:

- error_log: logs/error.log
- error_log_level: warn
- worker_processes: 3
- enable_cpu_affinity: true
- worker_rlimit_nofile: 20480
- worker_shutdown_timeout: 240s
- enable_access_log: true
- access_log: logs/access.log
- access_log_format:
- access_log_format_escape: default
- keepalive_timeout: 60s
- client_header_timeout: 60s
- client_body_timeout: 60s
- client_max_body_size: 0
- send_timeout: 10s
- underscores_in_headers: "on"
- real_ip_header: X-Real-IP
- real_ip_recursive: "off"
- real_ip_from:
- proxy_ssl_server_name: true
- upstream:
- keepalive: 320
- keepalive_requests: 1000
- keepalive_timeout: 60s
- charset: utf-8
- variables_hash_max_size: 2048

上面这些项需要修改，需要通过修改 **安装目录**`/alb-standard/conf/config-default.yaml` 文件中，对应的项目即可

```
nginx_config:                   # config for render the template to generate nginx
  #user: root                   # specifies the execution user of the worker proce
                                # the "user" directive makes sense only if the mas
                                # if you're not root user,the default is current u
  error_log: logs/error.log
  error_log_level:  warn        # warn,error
  worker_processes: 3           # if you want use multiple cores in container, you ca
  enable_cpu_affinity: true     # enable cpu affinity, this is just work well only
  worker_rlimit_nofile: 20480   # the number of files a worker process can open, s
  worker_shutdown_timeout: 240s # timeout for a graceful shutdown of worker proces
  event:
    worker_connections: 10620
  #envs:                        # allow to get a list of environment variables
  #  - TEST_ENV

  stream:
    lua_shared_dict:
      etcd-cluster-health-check-stream: 10m
      lrucache-lock-stream: 10m
      plugin-limit-conn-stream: 10m
```

```
  http:
    enable_access_log: true        # enable access log or not, default true
    access_log: logs/access.log
    access_log_format: "$remote_addr - $remote_user [$time_local] $http_host \"$request\" $status $body_bytes_sent $request_tim
    access_log_format_escape: default       # allows setting json or default characters escaping in variables
    keepalive_timeout: 60s         # timeout during which a keep-alive client connection will stay open on the server side.
    client_header_timeout: 60s     # timeout for reading client request header, then 408 (Request Time-out) error is returned t
    client_body_timeout: 60s       # timeout for reading client request body, then 408 (Request Time-out) error is returned to
    client_max_body_size: 0        # The maximum allowed size of the client request body.
                                   # If exceeded, the 413 (Request Entity Too Large) error is returned to the client.
                                   # Note that unlike Nginx, we don't limit the body size by default.

    send_timeout: 10s              # timeout for transmitting a response to the client.then the connection is closed
    underscores_in_headers: "on"   # default enables the use of underscores in client request header fields
    real_ip_header: X-Real-IP      # http://nginx.org/en/docs/http/ngx_http_realip_module.html#real_ip_header
    real_ip_recursive: "off"       # http://nginx.org/en/docs/http/ngx_http_realip_module.html#real_ip_recursive
    real_ip_from:                  # http://nginx.org/en/docs/http/ngx_http_realip_module.html#set_real_ip_from
      - 127.0.0.1
      - "unix:"
    #lua_shared_dicts:             # add custom shared cache to nginx.conf
    #  ipc_shared_dict: 100m       # custom shared cache, format: `cache-key: cache-size`

    # Enables or disables passing of the server name through TLS Server Name Indication extension (SNI, RFC 6066)
    # when establishing a connection with the proxied HTTPS server.
    proxy_ssl_server_name: true
    upstream:
      keepalive: 320               # Sets the maximum number of idle keepalive connections to upstream servers that are preser
                                   # When this number is exceeded, the least recently used connections are closed.
      keepalive_requests: 1000     # Sets the maximum number of requests that can be served through one keepalive connection.
                                   # After the maximum number of requests is made, the connection is closed.
      keepalive_timeout: 60s       # Sets a timeout during which an idle keepalive connection to an upstream server will stay
    charset: utf-8                 # Adds the specified charset to the "Content-Type" response header field, see
                                   # http://nginx.org/en/docs/http/ngx_http_charset_module.html#charset
```

2.7 修复ALB出现服务宕机或不响应(旧版，非2.0.1版本)

旧版ALB标准会出现长时间运行后服务不响应的问题，解决办法可以通过**升级最新版本**或者**直接修改旧版ALB代码**避免该问题的出现。

如果不能升级服务，那么可以通过一下方式，修改alb的代码，实现问题的修复，下面是修改代码修复问题的操作：

1. 进入ALB服务器，并切换到ALB安装目录
2. 编辑alb_install_dir/alb/apisix/alb_extra/alb_license.lua文件

3. 使用--注释第86行至91行内容，如下图所示

```
79  main()
80
81
82  function _M.set_alb_linlin()
83      -- ngx.timer.every(query_etcd_timer_interval, hello)
84      -- 先执行一次校验license是否有效
85      check_deaddate()
86      -- local ok, err = ngx.timer.every(query_etcd_timer_interval, check_deaddate)
87      -- if not ok then
88      --     -- core.log.warn(ngx.ERR, "failed to create timer check license deadline#", err)
89      --     print(ngx.ERR, "failed to create timer check license deadline#", err)
90      --     return
91      -- end
92  end
93  return _M
```

在代码前面使用--注释内容

2.8 ALB开机自启动

1、修改启动脚本安装目录的start.sh的变量cur_dir修改为脚本的绝对路径

```
#!/bin/bash

#cur_dir=$(pwd)
cur_dir=/opt/alb-2.0-centos-x86
alb_packages="$cur_dir/alb_packages"
# 默认安装在当前目录下alb_install_dir
install_package=${1:-"$cur_dir/alb_install_dir"}
install_soft="alb"

if [[ ! -d $alb_packages || ! -d $install_package ]];then
    alb_packages="$cur_dir/asg_packages"
    install_package="$cur_dir/asg_install_dir"
    install_soft="asg"
fi
```

2、在/etc/rc.d/rc.loacl里添加alb启动脚本路径

```
#!/bin/bash
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
#
# It is highly advisable to create own systemd services or udev rules
# to run scripts during boot instead of using this file.
#
# In contrast to previous versions due to parallel execution during boot
# this script will NOT be run after all other services.
#
# Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure
# that this script will be executed during boot.
touch /var/lock/subsys/local
/opt/alb-2.0-centos-x86/start.sh
~
~
~
```

3、reboot重新启动即可。

2.9 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。

```
[root@linux-3-107 alb-2.0-centos-x86]# ./start.sh
已经安装: alb
found supported os: centos
found supported arch: x86_64
pkg manager is: yum
启动alb 实例失败，启动错误内容如下:
/opt/alb-apusic/openresty/luajit/bin/luajit ALB start
request etcd endpoint 'http://127.0.0.1:2379/version' error, connection refused
Warning! Request etcd endpoint 'http://127.0.0.1:2379/version' error, connection refused, retry time=1
Warning! Request etcd endpoint 'http://127.0.0.1:2379/version' error, connection refused, retry time=2
```

2.9.1 问题解决

```
332 # etcd_path不存在时就不启动etcd
333 if [ -f $etcd_path ];then
334     # 启动etcd
335     check_port_used_and_kill $etcd_listen_port
336     nohup run_etcd
337 fi
338 sleep 15
339 # 启动ALB
340 if [[ $alb_running -eq 1 ]];then
341     ps_kill "nginx"
342 fi
343
```

1、修改启动脚本start.sh，等待etcd启动完成。

如上图所示，在338行插入sleep 15，保存退出后，重启ALB

2.10 Prometheus监控

ALB标准版提供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。

2.10.1 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)
```

2.10.2 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。

3 ALB主备搭建

3.1 前提准备

两台机器好ALB和keepalived软件，其中ALB的安装路径为：/opt/alb-2.0-centos-x86 本文操作的两台机器分别为：

- A： 172.21.32.118、
- B： 172.21.32.11
- IP：172.21.32.198为keepalived的虚拟IP地址。
- 其中A节点作为主节点，B节点作为备节点。

3.2 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

3.3 修改A节点启动方式脚本

切换到安装目录 /opt/alb-2.0-centos-x86，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。

3.4 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

3.5 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
    }
}
```

3.6 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" |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" |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
```

3.7 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
```

```
    }
}
```

3.8 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" |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" |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
```

3.9 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。

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