Apusic 金蝶天燕

生成时间：2026-07-13

运维手册

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标准版的运维手册，为用户提供ALB运行期间的监控、安全加固、常见问题排查及日常运维操作说明，帮助用户保障ALB服务的稳定运行。

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 日志切割

由于日志文件会不断增长，定期清理和分割日志文件是非常重要的，针对日志切割，ALB支持：

- 管控台日志管理中，针对某个日志文件配置日志切割
- 使用日志切割工具，手工操作生成切割功能

下面是ALB日志切割工具的介绍，如需通过管控台，可查看对应管控台的手册。

ALB提供了默认的日志切割工具，根据 JSON 配置文件自动生成 日志切割脚本（Shell 脚本或 logrotate 配置），并自动注册到系统 crontab，实现日志轮转自动化。

支持以下日志：

- ✅ 单个日志文件（如 /var/log/app.log）
- ✅ 通配符日志路径（如 /var/log/alb/*.log）
- ✅ 两种日志轮转模式：shell 或 logrotate
- ✅ 自动压缩、保留天数控制、进程信号通知（USR1）
- ✅ 安装 / 列出 / 卸载任务全生命周期管理
- ✅ 安全的 crontab 自动注册与清理

2.1 工具结构说明

工具目录：`ALB安装目录/utils/logrotate`

```
.
├── logrotate-config.json          ← 配置文件（默认名）
├── shell_scripts/                 ← 生成的 Shell 脚本目录
│   └── <task-name>.sh
├── logrotate_configs/             ← 生成的 logrotate 配置目录
│   └── <task-name>
├── installed_tasks.json           ← 已安装任务的元数据（用于 list/remove）
```

字段	类型	必填	说明
name	string	✅	任务名称（用于文件名和注释）
log_file	string	✅	日志路径，支持通配符（如 *.log）；若为相对路径，会自动拼接到 albBaseDir

mode	string	✅	"shell" 或 "logrotate"
retain_days	int	❌	保留天数，默认 7 天
compress	bool	❌	是否压缩旧日志，默认 false
cron_schedule	string	❌	cron 表达式（如 "0 2 * * *"），默认 "0 1 * * *"（每天凌晨 1 点）

2.2 示例配置

```
[
  {
    "name": "alb-access",
    "log_file": "logs/access.log",
    "mode": "shell",
    "retain_days": 14,
    "compress": true,
    "cron_schedule": "0 2 * * *"
  },
  {
    "name": "alb-error",
    "log_file": "logs/error.log",
    "mode": "logrotate",
    "retain_days": 30,
    "compress": true
  },
  {
    "name": "all-app-logs",
    "log_file": "logs/app/*.log",
    "mode": "logrotate",
    "retain_days": 7,
    "compress": false
  }
]
```

> ✅ *相对路径 logs/access.log 会被自动转为绝对路径：/path/to/alb/logs/access.log* ✅ *通配符 logs/app/*.log 在 logrotate 和 shell 模式下均能正确处理*

2.3 使用方式

0. **权限要求**

- 需要对日志目录有**读写权限**
- 需要能执行 `crontab -l` 和 `crontab -e`（通常普通用户即可）
- 若日志由 root 写入，建议以相同用户身份运行本工具
- 若无法保证可执行权限，则切换至 root 用户运行此工具
- 切换至utils/logrotate目录下执行

1. **安装任务（生成脚本 + 注册 crontab）**

```
# 切换至utils/logrotate目录下（必须）
cd utils/logrotate
# 使用默认配置文件安装日志切割任务
./logrotate-gen --install
```

执行后：

- 生成 Shell 脚本到 ./shell_scripts/
- 生成 logrotate 配置到 ./logrotate_configs/
- 自动将任务加入当前用户的 crontab
- 保存安装记录到 installed_tasks.json

2. **预览生成内容（不写文件）**

```
# 预览所有生成内容
./logrotate-gen --dry-run

# 预览指定配置，前提是配置文件test.json在当前目录
./logrotate-gen --dry-run --config ./test.json
```

示例输出

```
=== SHELL SCRIPT: ./shell_scripts/alb-access.sh ===
#!/bin/bash
set -euo pipefail
shopt -s nullglob
```

```
...

=== LOGROTATE CONFIG: ./logrotate_configs/alb-error ===
# Auto-generated for task: alb-error
/path/to/alb/logs/error.log {
    daily
    rotate 30
    compress
    ...
}
```

3. **列出已安装的任务**

```
./logrotate-gen -list
```

输出示例：

```
📅 Installed tasks (installed at: Mon Oct 27 10:00:00 CST 2025):
1. Name: alb-access
   Mode: shell
   Log File: /path/to/alb/logs/access.log
   Schedule: 0 2 * * *
   Retain Days: 14
   Compress: true

2. Name: alb-error
   Mode: logrotate
   Log File: /path/to/alb/logs/error.log
   Schedule: 0 1 * * * (default)
   Retain Days: 30
   Compress: true

📋 Current crontab entries for logrotate:
0 2 * * * /path/to/tool/shell_scripts/alb-access.sh # alb-access
```

```
0 1 * * * logrotate /path/to/tool/logrotate_configs/alb-error # alb-error
```

4. **卸载所有任务**

```
./logrotate-gen -remove-all
```

执行后：

- 从 crontab 中移除所有本工具添加的条目
- 删除 shell_scripts/ 和 logrotate_configs/ 目录
- 删除 installed_tasks.json

> ⚠ *此操作不可逆，请谨慎使用！*

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（以下命令通常需要 root 权限）

```
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
systemctl start alb                      # 启动ALB
```

3.1 移除系统启动项

```
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/remove_startup.sh        # 移除开机启动项
```

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=+ep ./sbin/alb`
3. 切换回**普通用户**
4. 启动ALB： `./sbin/alb` （或 `./bin/start-alb.sh` ，若ALB已在运行，请先执行 `./bin/stop-alb.sh` 停止后再启动）

5 ALB代理安全配置

5.1 通用安全配置

5.1.1 隐藏ALB版本信息

```
http {
    server_tokens off;   # 在http块中添加这行
}
```

5.1.2 限制请求方法

只允许必要的 HTTP 方法（如 GET、POST、HEAD、PUT）：

```
http {
    server {
        location / {
            if ($request_method !~ ^(GET|HEAD|POST|PUT)$) {
                return 405;
            }
        }
    }
}
```

5.1.3 限制客户端请求大小

防止 DoS 攻击（如大文件上传、超长 URL）：

```
http {
    client_max_body_size 1M;          # 限制请求体，根据实际业务调整
    client_header_buffer_size 1k;   # 请求头缓冲区
    large_client_header_buffers 4 8k;
}
```

5.1.4 设置合理的超时时间

防止慢速攻击（Slowloris 等）

```
http {
    client_body_timeout 10s;
    client_header_timeout 10s;
    keepalive_timeout 60s;
    send_timeout 10s;
    lingering_timeout 5s;
    lingering_close on;
}
```

5.1.5 启用 HTTPS 并强制跳转

```
server {
    listen 80;
    server_name example.com;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    ssl_certificate /path/to/cert.pem;
    ssl_certificate_key /path/to/privkey.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers
ECDHE+AESGCM:DHE+AESGCM:AES256+EECDH:AES256+EDH:!aNULL:!MD5:!DSS;
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 1d;
    ssl_session_tickets off;

    # HSTS 配置（首次部署建议先测试，确认无误后再启用）
    add_header Strict-Transport-Security "max-age=31536000;
```

```
includeSubDomains" always;
}
```

5.2 反向代理安全配置

5.2.1 清理/重写请求头

防止头注入或信息泄露：

```
http {
    server {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # 清除客户端可能伪造的内部头（设置为空字符串并不能完全阻止头传递，建议后端增加验证）
        proxy_set_header X-Original-URI "";
        proxy_set_header X-Internal-Secret "";
    }
}
```

5.2.2 限制后端响应头

避免后端泄露敏感信息（如 Server、X-Powered-By）： 可通过 proxy_hide_header 隐藏：

```
http {
    server {
        proxy_hide_header Server;
        proxy_hide_header X-Powered-By;
        proxy_hide_header X-AspNet-Version;
    }
}
```

5.2.3 限制代理请求速率

防CC攻击

```
http {
    limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;

    server {
        location /api/ {
            limit_req zone=api burst=20 nodelay;
            proxy_pass http://backend;
        }
    }
}
```

5.3 静态资源服务专用安全配置

5.3.1 禁止目录列表

确保未配置 `autoindex on;`，默认关闭即可。

5.3.2 禁止危险文件类型访问

根据静态资源类型设置，禁止危险扩展名（如 .php, .pl, .py, .jsp, .asp, .sh, .cgi）：

```
location ~ \.(php|pl|py|jsp|asp|sh|cgi)$ {
    return 403;
}
```

5.3.3 防止敏感文件泄露

```
http {
    server {
        # 禁止访问以点开头的隐藏目录（如 .git、.svn）
        location ~ /\. {
            deny all;
        }
```

```
        # 禁止访问 .env 文件
        location ~ /\.env {
            deny all;
        }
    }
}
```

5.3.4 设置安全的 Content Security Policy (CSP) 和其他安全头

```
server {
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-Frame-Options "DENY" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin"
always;
    add_header Permissions-Policy "geolocation=(), microphone=(),
camera=()" always;

    # HSTS 配置（首次部署建议先测试，确认无误后再启用）
    add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains" always;

    # CSP 示例（根据实际调整，生产环境应避免使用 'unsafe-inline'）
    add_header Content-Security-Policy "default-src 'self'; script-
src 'self' 'unsafe-inline' https://trusted.cdn.com; style-src 'self'
'unsafe-inline'; img-src 'self' data:; font-src 'self'; object-src
'none'; base-uri 'self'; form-action 'self';" always;
}
```

6 产品常见问题

注： 性能相关错误参考【性能优化】 手册。

6.1 启动失败： getgrnam("nobody") failed

启动中出现下面错误：

```
root@t16:/home/lyan/桌面/temp/alb/alb-agile-2.0-x86/bin# ./start-alb.sh
Starting ALB...
alb: [emerg] : getgrnam("nobody") failed
Start ALB failed, please check logs/error.log
root@t16:/home/lyan/桌面/temp/alb/alb-agile-2.0-x86/bin# ls
reload-alb.sh  start-alb.sh  start-all.sh  stop-alb.sh  stop-all.sh
```

修复方式：

修改conf/alb.conf文件中的user指令，把 `user nobody;` 改为对应静态资源有访问权限的用户名（建议创建专用用户如 `alb` 并赋予权限，而非直接使用root）：

```
user alb;
```

6.2 Docker容器启动失败

Docker 容器启动失败后，可以使用 `docker logs <container_id>` 查看容器日志。

1. 获取ALB容器ID

```
docker ps -a
CONTAINER ID        IMAGE                  COMMAND
CREATED             STATUS
     PORTS               NAMES
xxxxxxxxxxxx        harbor.apusic.com/apusic/alb:V2.0.6-ae.20260421-
kylin-amd64    "bash /opt/ALB-V2...."   17 seconds ago      Exited
(1) 16 seconds ago                       alb-v206-se-docker-amd64-
alb-1
```

如上ALB容器的ID是 `xxxxxxxxxxxx`

2. 查看容器日志

使用docker logs 查看日志，如下样例：

```
# docker logs 48c60b6fced6
Starting ALB...
alb: [emerg] : open() "/opt/ALB-V2.0.6-SE-amd64/conf/alb.conf"
failed (13: Permission denied)
Start ALB failed, please check logs/error.log
```

3. docker启动Permission denied错误

错误原因：容器内部的进程没有足够的权限来访问挂载的卷中的文件和目录。

解决方案：

- 修改 `alb-standard-dockercompose.yaml`，标记使用 `:z` 标记挂载路径（ `:z` 标记仅适用于 SELinux 环境，非 SELinux 系统无需添加）

```
services:
  alb:
    image: harbor.apusic.com/apusic/alb:V2.0.6-ae.20260421-kylin-
amd64
    ports:
      - 8080:8080
      - 8887:8887
    volumes:
      - ./alb/alb.conf:/opt/ALB-V2.0.6-SE/conf/alb.conf:z
      - ./alb/license.xml:/opt/ALB-V2.0.6-SE/license.xml:z
      - ./alb/logs:/opt/ALB-V2.0.6-SE/logs:z
    command: bash /opt/ALB-V2.0.6-SE/bin/start-alb-and-console.sh
```

如上，每个挂载路径结尾添加 `:z` 标记。

- 重启容器

```
docker compose -f alb-standard-dockercompose.yaml up -d
```

6.3 启动ALB时，显示授权信息后，停住了

执行ALB启动后，终端显示完授权信息后，停住了，没有自动退出启动脚本，如下所示：

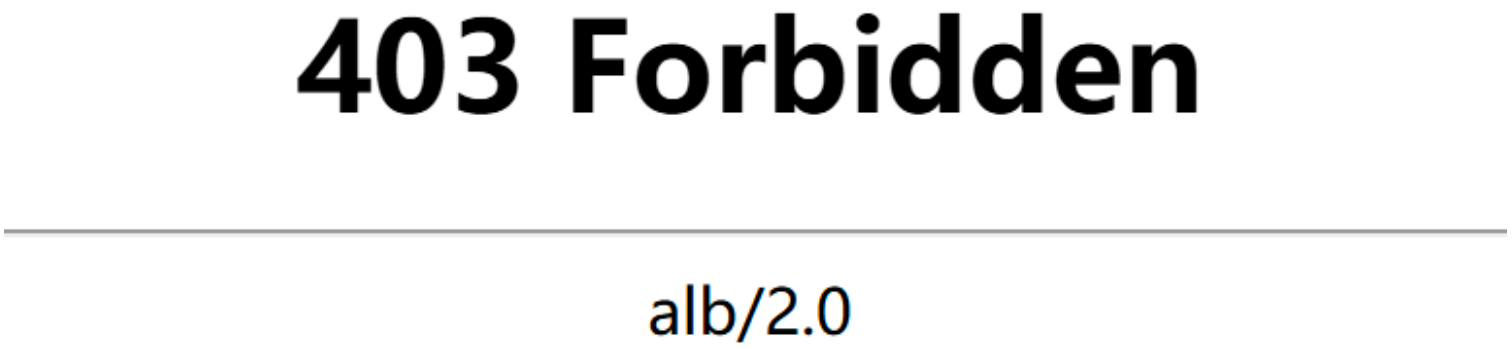
```
./bin/start-alb.sh
Starting ALB...
      ___                       _          Apusic Load Balancer
     /   |  ____  __  _______(_)____      Edition:Agile Edition
    / /| | / __ \/ / / / ___/ / ___/      License start: 2025-02-21
   / ___ |/ /_/ / /_/ (__  ) / /__        License end: 2025-06-31
  /_/  |_/ .___/\__,_/____/_/\___/        IP granted: 255.255.255.255
        /_/                               Licensee: 金蝶天燕研发测试

  version: V2.0.6
```

- 现象： 执行启动脚本后，并没有退出脚本的执行，也没有输出`ALB is running now!`等提示
- 原因分析： ALB前台启动，并没有后台启动
- 问题解决： 在conf/alb.conf配置文件中，检查是否存在`daemon off;`这个配置，如果是，注释掉（`#daemon off;`）即可，重新启动ALB。

```
./bin/stop-alb.sh &&  ./bin/start-alb.sh
```

6.4 访问页面出现403错误

在配置好ALB后，访问页面出现403错误页面，如下所示：



- 现象： 访问任意的页面（静态资源）均出现403错误。
- 原因分析： ALB启动的用户，没有对应静态资源的读权限，导致403错误。
- 问题解决： 首选方案是修改静态资源目录权限（`chown -R nobody:nobody /path/to/static`），使ALB启动用户能够访问静态资源。次选方案是在conf/alb.conf配置文件中修改`user`指令为对应静态资源有访问权限的用户名（建议创建专用用户如`alb`），最后重启ALB即可。

6.5 AAS登录失败

- 现象： 使用ALB代理多个AAS服务，在输入用户名和密码等点击登录页面后，然后出现登录失败或者跳转到登录页面。
- 原因分析：
  - 会话不一致： 浏览器访问的时候，不同请求发送到不同的AAS节点上，而登录的会话只在其中一个AAS服务器中。
  - 请求头或cookie丢失： AAS服务器中的服务依赖请求头或者cookie验证客户端是否登录，如果请求头或者cookie丢失，则无法验证客户端是否登录。
  - WebSocket连接失败： 某些服务在登录后，会建立WebSocket连接，如果WebSocket连接失败，则导致实时功能不可用。
- 问题解决：
  - 会话不一致： ALB使用ip_hash或者sticky负载均衡算法。
  - 请求头或cookie丢失： 配置转发全部请求头和cookie

```
location / {

  proxy_pass http://backend/;
```

```
    # 以下内容为新增的配置
    proxy_cookie_path / /;  # 确保 Cookie 路径正确
    proxy_pass_header Set-Cookie;


    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;

}
```

- WebSocket连接失败： 单独为WebSocket连接地址编写location配置代理

```
location /websocket_url {
    proxy_pass http://backend/websocket_url;

      # WebSocket 必要配置
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "upgrade";

      # 设置WebSocket连接读超时时间为10分钟，根据业务修改时间
      proxy_read_timeout 600s;
      # 设置WebSocket连接写超时时间为10分钟，根据业务修改时间
      proxy_send_timeout 600s;

      proxy_cookie_path / /;  # 确保 Cookie 路径正确
      proxy_pass_header Set-Cookie;

      proxy_set_header Host $host;
```

```
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
    $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
```

6.6 后端反向代理服务的一个节点宕机后，服务出现无法访问，但其他节点正常

- 现象： ALB反向代理服务的ABC三个服务节点，其中A宕机或挂了，但BC服务正常，此时访问ALB出现短时间无法访问。
- 原因分析：ALB连接已经宕机的A服务超时导致请求等待时间过长。ALB默认的被动健康检查机制会在 `max_fails` 次失败后标记节点不可用，但默认超时时间较长，期间新的请求仍可能转发到故障节点。建议启用主动健康检查，让ALB提前感知节点状态，不再把请求转发给宕机服务。
- 问题解决：修改ALB连接上游服务的超时时间，默认为60秒，修改为10秒或者更少，同时启用主动健康检查，让ALB不再把请求转发给宕机服务，下面是参考样例

```
http {
  upstream backend {
    sticky; # 使用会话亲和负载均衡算法（与ip_hash二选一，不可同时使用）
    server 192.168.1.100:8080 max_fails=3 fail_timeout=10s;
    server 192.168.1.2:8080 max_fails=3 fail_timeout=10s;
    server 192.168.1.3:8080 max_fails=3 fail_timeout=10s;

    # 配置主动健康检查（如ALB未内置该模块，请参考官方文档确认）
    check timeout=1000 fall=3 rise=2 type=http interval=3000;
    check_http_expect_alive http_2xx  http_3xx;
  }

  server {
    location / {
      proxy_pass http://backend/;
      # 核心配置，缩短服务连接超时（建议至少5s，高并发场景根据实际调整）
      proxy_connect_timeout 5s;
    }
```

```
    }
}
```

6.7 使用统一授权中心授权方式启动失败

- 现象： 使用统一授权中心授权方式启动失败，失败错误提示如下：

```
alb: [emerg]: center auth: request to center fault, err: send auth
info to authserver error
```

- 原因分析： 授权中心更新导致ALB解析请求失败
- 问题解决： 更新至2025年发版的ALB即可。

6.8 502错误： no live upstreams while connecting to upstream

- 现象： 前端访问出现502错误，并且ALB错误日志出现大量 `no live upstreams while connecting to upstream`
- 原因分析： 在ALB反向代理中，如果上游节点在 `proxy_connect_timeout` 、 `proxy_send_timeout` 、 `proxy_read_timeout` 指定的时间内未响应，那么将会标记节点不可用。 `max_fails` 默认值为1，配合 `fail_timeout` 使用，达到阈值后ALB会暂时将该节点标记为不可用，导致出现上述错误。
- 问题解决： 修改 `proxy_connect_timeout` 、 `proxy_send_timeout` 、 `proxy_read_timeout` 为更长的时间（超过60s），或者将 `max_fails` 设置为3，表示不标记节点不可用。
- 配置样例：

```
upstream gateway_api {
    # 增加 max_fails和fail_timeout参数
    server 127.0.0.1:9099 max_fails=3 fail_timeout=15s;
    server 127.0.0.2:9099 max_fails=3 fail_timeout=15s;
}

server {
  location /api/ {
    proxy_pass http://gateway_api/;

    # 增加超时时间参数
    proxy_connect_timeout 15s;
```

```
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }
}
```

6.9 管控台三员账号密码重置

- 现象： 系统中所有三员账号（admin-system、admin-security、admin-audit）均不可用，密码遗忘或被锁定，无法通过管控台界面登录。
- 原因分析： 密码长期未修改、初始随机密码未保存，或账号被安全策略锁定。
- 问题解决： 通过服务器端命令行工具重置指定账号的密码，重置后会输出新的随机密码，登录后系统会强制要求修改密码。

```
# 切换到管控台安装目录（以实际安装路径为准）
cd /opt/ALB-V2.0.6-SE-amd64/console

# 重置指定账号密码（可重置 admin-system、admin-security、admin-audit、admin-alb、user-alb）
[root@localhost console]# ./alb-console --reinit admin-system
[alb-agile-console]2026/06/24 - 04:24:04.462    info .../sqlite.go:64     register table success
[alb-agile-console]2026/06/24 - 04:24:04.463    info .../sqlite.go:95     数据库中已存在用户，跳过默认用户初始化
[alb-agile-console]2026/06/24 - 04:24:04.520    info .../sqlite.go:204    用户密码重置成功          {"username": "admin-system", "new_password": "%B4H%VKDf#yLj8oG"}
用户 admin-system 密码已重置为新密码： %B4H%VKDf#yLj8oG
```

- 操作说明：
  - `--reinit` 参数后接需要重置的用户名，可重置任意已存在的管控台账号。
  - 命令执行成功后，终端会输出新生成的随机密码，请立即记录并在登录后强制修改。
  - 该命令仅重置密码，不会重置账号的其他配置信息（如角色、状态）。
  - 如需批量重置多个账号，请分别执行上述命令，每次仅能重置一个账号。
- 注意事项： 该命令输出的新密码为系统自动生成的强密码，请妥善保管并立即在管控台登录后修改为自定义密码。如多次重置后仍无法登录，请联系金蝶天燕技术支持。

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