



APUSIC
固若长城
睿比世界

開發手冊

金蝶Apusic負載均衡器

版權所有 © 深圳市金蝶天燕雲計算股份有限公司2026。保留所有權利。

版權聲明

本文件所涉及的軟件著作權、版權等知識產權已依法進行了註冊，由金蝶天燕雲計算股份有限公司合法擁有。受《中華人民共和國著作權法》《計算機軟件保護條例》《知識產權保護條例》和相關國際版權條約、法律、法規以及其它知識產權法律和條約的保護。未經授權許可，不得非法使用。

免責聲明

本文件包含的版權資訊由金蝶天燕雲計算股份有限公司合法擁有，受法律的保護，金蝶天燕雲計算股份有限公司對本文件可能涉及到的非金蝶天燕雲計算股份有限公司的資訊不承擔任何責任。在法律允許的範圍內，您可以查閱並僅能夠在《中華人民共和國著作權法》規定的合法範圍內複製和打印本文件。任何單位和個人未經金蝶天燕雲計算股份有限公司書面授權許可，不得使用、修改、再發佈本文件的任何部分和內容，否則將被視為侵權，金蝶天燕雲計算股份有限公司有依法追究其責任的權利。

本文件如有更新，不另行通知。對本文件中的問題您可向金蝶天燕雲計算股份有限公司告知或查詢。未經本公司明確授予的任何權利均予保留。

商標聲明

 是深圳市金蝶天燕雲計算股份有限公司向中華人民共和國國家商標局申請註冊的註冊商標，註冊商標專用權由金蝶天燕合法擁有，受法律保護。未經金蝶天燕的書面許可，任何單位及個人不得以任何方式或理由對該商標的任何部分進行使用、複製、修改、傳播、抄錄或與其它產品捆綁使用銷售。凡侵犯金蝶天燕商標權的，金蝶天燕將依法追究其法律責任。本文件提及的其他所有商標或註冊商標，由各自的所有人擁有。

目錄

- 1 前言
 - 1.1 適用對象
 - 1.2 相關文件
 - 1.3 技術支援
- 2 目錄
- 3 概述
 - 3.1 三種能力對比
 - 3.2 讀者路徑
- 4 外部 API 介面
 - 4.1 通用約定
 - 4.1.1 鑑權方式
 - 4.1.2 請求格式
 - 4.1.3 回應格式
 - 4.1.4 HA 叢集行為
 - 4.2 HTTP 站點介面
 - 4.2.1 建立 HTTP 站點
 - 4.2.2 更新 HTTP 站點
 - 4.2.3 刪除 HTTP 站點
 - 4.2.4 啟用 HTTP 站點
 - 4.2.5 停用 HTTP 站點
 - 4.2.6 取得 HTTP 站點清單
 - 4.2.7 取得 HTTP 站點詳情
 - 4.2.8 為 HTTP 站點繫結 SSL 憑證
 - 4.3 Stream 站點介面
 - 4.3.1 建立 Stream 站點
 - 4.3.2 更新 Stream 站點
 - 4.3.3 刪除 Stream 站點
 - 4.3.4 啟用 / 停用 Stream 站點
 - 4.3.5 取得 Stream 站點清單
 - 4.3.6 取得 Stream 站點詳情
 - 4.4 錯誤回應範例
 - 4.4.1 鑑權失敗 (401)

- 4.4.2 外部 API 未啟用 (403)
- 4.4.3 HA 備節點寫入 (10001)
- 4.4.4 業務錯誤 (7)
- 4.5 快速驗證指令碼
- 4.6 稽核查詢
- 5 alb-cli 命令列工具
 - 5.1 安裝與設定
 - 5.1.1 鑑權方式
 - 5.1.2 設定來源 (優先順序從高到低)
 - 5.1.3 驗證連線
 - 5.2 全域參數
 - 5.3 輸出格式
 - 5.3.1 JSON (預設)
 - 5.3.2 Table
 - 5.3.3 錯誤輸出
 - 5.4 命令參考
 - 5.4.1 `site` — 站點管理
 - 5.4.2 `status` — ALB 執行狀態
 - 5.4.3 `metrics` — 監控資料
 - 5.4.4 `config reload` — 重新載入 ALB 設定
 - 5.4.5 `log` — 日誌檢視
 - 5.5 安全策略
 - 5.5.1 危險操作確認
 - 5.5.2 非互動環境
 - 5.6 典型工作流程
 - 5.6.1 發佈一個 HTTP 站點
 - 5.6.2 為站點啟用 HTTPS
 - 5.6.3 日常巡檢
 - 5.6.4 故障排查
 - 5.7 後端介面對應
- 6 AI Agent 整合
 - 6.1 SKILL 用途
 - 6.2 SKILL 安裝
 - 6.2.1 在 Claude Code 中安裝

- 6.2.1.1 方式一：複製到專案 Skill 目錄（推薦）
- 6.2.1.2 方式二：手動建立
- 6.2.1.3 驗證安裝
- 6.3 在其他 Agent 工具中使用
 - 6.3.1 OpenCode
 - 6.3.2 通用相容工具
- 6.4 前置依賴
- 6.5 觸發方式
- 6.6 可用命令速查
- 6.7 預設輸出與解析
- 6.8 安全約束
- 6.9 標準工作流程
 - 6.9.1 工作流程 1：發佈 HTTP 站點
 - 6.9.2 工作流程 2：啟用 HTTPS
 - 6.9.3 工作流程 3：日常巡檢
 - 6.9.4 工作流程 4：故障排查
- 6.10 JSON 檔案範本
- 6.11 常見錯誤處理
- 6.12 最佳實踐
- 7 維運與監控介面
 - 7.1 ALB 執行狀態
 - 7.2 流量與節點監控
 - 7.3 設定重新載入
 - 7.4 日誌查詢
- 8 典型整合場景
 - 8.1 場景 1：CI/CD 自動發佈 HTTP 站點
 - 8.2 場景 2：與 ERP / 業務系統對接
 - 8.3 場景 3：日常巡檢機器人（AI Agent）
 - 8.4 場景 4：故障自動定位
- 9 整合建議
- 10 錯誤碼速查表
 - 10.1 外部 API 錯誤碼
 - 10.2 alb-cli 常見錯誤
- 11 最佳實踐與故障排查

- 11.1 最佳實踐
- 11.2 故障排查
- 12 版本更新說明

1 前言

本文件為金蝶Apusic負載均衡器軟體V2.0.6標準版的開發手冊，面向維運與開發人員，詳細介紹 ALB 管控台對外提供的外部 API 介面、alb-cli 命令列工具 以及面向 AI Agent（Claude、OpenCode 等） 的整合能力，協助客戶基於 ALB 實現自動化維運、CI/CD 發佈、平台對接和 AI 輔助維運等場景。

1.1 適用對象

本文件適用於：

- ALB 二次開發與整合工程師
- 維運自動化平台 / DevOps 平台開發工程師
- 引入 AI Agent 進行日常維運的 SRE 工程師
- ERP、維運監控、業務發佈等業務系統的對接方
- 客戶單位的系統架構師、技術負責人

1.2 相關文件

瞭解更多ALB V2.0.6產品相關的資訊，請參閱以下ALB V2.0.6產品手冊文件集：

序號	手冊文件	說明
1	金蝶Apusic負載均衡器軟體 V2.0.6 發版說明	介紹了ALB 產品版本更新內容。
2	金蝶Apusic負載均衡器軟體 V2.0.6 快速使用手冊	簡單介紹了如何快速上手使用ALB。
3	金蝶Apusic負載均衡器軟體 V2.0.6 安裝手冊	詳細介紹如何在各作業系統上安裝ALB，以及ALB服務啟停操作，產品的註冊過程。
4	金蝶Apusic負載均衡器軟體 V2.0.6 代理功能手冊	詳細介紹 ALB 相關功能的使用、設定、管理及配套工具的使用方法。
5	金蝶Apusic負載均衡器軟體 V2.0.6 管控台使用者手冊	詳細介紹ALB管控台相關功能的使用和操作說明。
6	金蝶Apusic負載均衡器軟體 V2.0.6 開發手冊	詳細介紹ALB外部API、alb-cli命令列工具及AI Agent整合方法。
7	金蝶Apusic負載均衡器軟體 V2.0.6 遷移手冊	詳細介紹ALB歷史版本遷移升級到V2.0.6版本的說明。

8	金蝶Apusic負載均衡器軟體 V2.0.6 維運手冊	詳細介紹ALB的監控、維運、安全加固等維運說明。
9	金蝶Apusic負載均衡器軟體 V2.0.6 效能最佳化手冊	詳細介紹ALB效能調校的說明。

1.3 技術支援

ALB產品提供全面的技術支援服務，您可以透過以下方式獲得技術支援：

- 網址：www.apusic.com
- 電話：400-855-5800
- 信箱：support@apusic.com
- 金蝶雲社群：<https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN>

您在取得技術支援時，請提供如下資訊：

1. 您的姓名
2. 公司資訊與聯絡方式
3. 作業系統及其版本
4. 產品版本號
5. 出現異常及錯誤的日誌、截圖等詳細資訊

2 目錄

- [概述](#)
 - [外部 API 介面](#)
 - [alb-cli 命令列工具](#)
 - [AI Agent 整合](#)
 - [維運與監控介面](#)
 - [典型整合場景](#)
 - [整合建議](#)
 - [錯誤碼速查表](#)
 - [最佳實踐與故障排查](#)
-

3 概述

ALB V2.0.6 標準版在管控台基礎上，開放了三種面向開發與維運人員的能力：

能力	型態	適用場景
外部 Site API	HTTPS RESTful API（API Key 鑑權）	業務系統、維運平台、CI/CD 流水線對接
alb-cli	單一二進位命令列工具	維運指令碼、自動化任務、人工快速操作
AI Agent SKILL	面向 Claude、OpenCode 等 Agent 的 SKILL 描述	引入 AI 助理進行日常維運和故障排查

3.1 三種能力對比

維度	外部 API	alb-cli	AI Agent SKILL
呼叫方式	HTTP/JSON	命令列	自然語言 + 工具呼叫
鑑權	X-API-Key	ALB_API_KEY 環境變數	ALB_API_KEY 環境變數
適合整合方	後端服務、ERP	維運、DevOps	AI 助理場景
輸出格式	JSON	JSON / Table	解析 JSON 後給出回答
危險操作	需自行加確認	內建互動確認	內建使用者確認 + 非 TTY 拒絕

注意：以上三種能力底層共用同一套管控台介面，呼叫語義保持一致。開發人員可根據整合場景選擇最合適的型態。

3.2 讀者路徑

- 後端開發者：先閱讀 [外部 API 介面](#)，重點關注通用約定、HTTP 站點、Stream 站點、HA 叢集行為。
- 維運 / DevOps：先閱讀 [alb-cli 命令列工具](#) 和 [典型整合場景](#)。
- AI 整合方：先閱讀 [AI Agent 整合](#) 和 [典型整合場景](#) 中的故障排查流程。

4 外部 API 介面

版本: v1.0

基礎路徑: `/api/alb/console/external/v1`

適用版本: ALB V2.0.6+

4.1 通用約定

4.1.1 鑑權方式

所有外部 API 介面透過 HTTP Header 傳遞 API Key:

```
X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

API Key 在 **【ALB安裝目錄】** `/console/conf/config.yaml` 中設定:

```
external:
  enabled: true
  api_keys:
    - name: "erp-system"
      key: "alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

安全提示: 請妥善保管 API Key, 正式環境請透過 HTTPS 存取管控台; 建議為不同業務系統分配不同的 API Key, 便於稽核與按需撤銷。

4.1.2 請求格式

- 請求主體統一使用 `application/json`
- 路徑參數 `:id` 為站點 ID (數字)
- `locations`、`upstreams`、`backend_servers` 欄位為 JSON 字串 (後端按字串儲存), 請求時需將陣列序列化為 JSON 字串, 如 `"[{\"path\":\"/\", \"type\":\"proxy\"}]\"`

4.1.3 回應格式

```
{
  "code": 0,
  "data": { },
}
```

```
"msg": "ok"
}
```

code	含義
0	成功
7	通用業務錯誤
401	鑑權失敗（缺少 API Key / Key 無效）
403	外部 API 未啟用
10001	目前節點為 HA 備節點，禁止寫入

4.1.4 HA 叢集行為

節點狀態	讀操作 (GET)	寫操作 (POST/PUT/DELETE)
未啟用 HA	✅ 正常	✅ 正常
啟用 HA + 主節點	✅ 正常	✅ 正常 + 自動同步到備節點
啟用 HA + 備節點	✅ 正常	❌ 回傳 code=10001

提示：HA 備節點的讀操作不會自動轉發到主節點，如需從備節點取得最新資料，請將請求發往主節點，或在用戶端做重試。

4.2 HTTP 站點介面

4.2.1 建立 HTTP 站點

POST `/sites/http`

請求參數

欄位	類型	必填	說明
name	string	是	站點名稱
domains	string	是	監聽網域名稱
listen_port	int	是	監聽埠
locations	string(JSON)	否	location 設定清單（JSON 字串）

upstreams	string(JSON)	否	upstream 設定清單 (JSON 字串) , locations 中使用 upstream_name 時需傳入
ssl_cert_id	int	否	SSL 憑證 ID
auto_reload	bool	否	儲存後自動重新載入, 預設 true
status	string	否	enabled / disabled, 預設 enabled

Location 欄位說明

欄位	類型	必填	說明
path	string	是	比對路徑, 如 / 或 /api
type	string	是	proxy / static / redirect
proxy_pass	string	否	直接填寫後端位址, 如 http://192.168.1.10:8080
upstream_name	string	否	引用已有的 upstream 名稱 (需同時在 upstreams 中定義)

注意: `proxy_pass` 和 `upstream_name` 二選一。使用 `upstream_name` 時必須同時在 `upstreams` 欄位中定義該 `upstream`, 否則 `nginx` 重新載入會失敗。

請求範例 1: 直接代理 (最簡單, 無需 upstream)

```
curl -X POST "http://alb-master:8887/api/alb/console/external/v1/sites/http" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
-H "Content-Type: application/json" \
-d '{
  "name": "test-api",
  "domains": "www.example.com",
  "listen_port": 8088,
  "locations": "
[[{\"path\":\"/\", \"type\":\"proxy\", \"proxy_pass\":\"http://192.168.1.1
[]}]",
  "upstreams": "[]",
  "custom_headers": "[]",
  "proxy_set_headers": "[]",
  "extra_directives": "[]",
```

```
"auto_reload": true,
"status": "enabled"
}'
```

請求範例 2: 使用 Upstream (支援負載均衡)

```
curl -X POST "http://alb-master:8887/api/alb/console/external/v1/sites/http" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
-H "Content-Type: application/json" \
-d '{
  "name": "test-upstream",
  "domains": "www.example.com",
  "listen_port": 8090,
  "upstreams": "[{\"name\": \"backend-
web\", \"balance_method\": \"round-robin\", \"servers\":
[\"address\": \"192.168.1.10:8080\", \"weight\": 1},
{ \"address\": \"192.168.1.11:8080\", \"weight\": 1}]]",
  "locations": "
[\"path\": \"/\", \"type\": \"proxy\", \"upstream_name\": \"backend-
web\", \"proxy_timeout\": 60, \"support_web_socket\": false, \"use_alias\": t
[]]",
  "custom_headers": "[]",
  "proxy_set_headers": "[]",
  "extra_directives": "[]",
  "auto_reload": true,
  "status": "enabled"
}'
```

回應範例

```
{
  "code": 0,
  "data": {
    "id": 42,
```

```

    "name": "test-upstream",
    "domains": "www.example.com",
    "listen_port": 8090,
    "status": "enabled"
  },
  "msg": "創建成功"
}

```

4.2.2 更新 HTTP 站點

PUT /sites/http

請求參數

欄位	類型	必填	說明
id	int	是	站點 ID
name	string	否	站點名稱
domains	string	否	監聽網域名稱
listen_port	int	否	監聽埠
locations	string(JSON)	否	location 設定清單 (JSON 字串)
upstreams	string(JSON)	否	upstream 設定清單 (JSON 字串)
ssl_cert_id	int	否	SSL 憑證 ID
auto_reload	bool	否	儲存後自動重新載入
status	string	否	enabled / disabled

請求範例

```

curl -X PUT "http://alb-master:8887/api/alb/console/external/v1/sites/http" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "id": 42,
    "name": "test-api-update",

```

```
"listen_port": 8080
}'
```

回應範例

```
{
  "code": 0,
  "data": null,
  "msg": "更新成功"
}
```

4.2.3 刪除 HTTP 站點

DELETE /sites/http

請求參數

欄位	類型	必填	說明
id	int	是	站點 ID

請求範例

```
curl -X DELETE "http://alb-master:8887/api/alb/console/external/v1/sites/http" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
-H "Content-Type: application/json" \
-d '{
  "id": 42
}'
```

回應範例

```
{
  "code": 0,
  "data": null,
}
```

```
{
  "msg": "刪除成功"
}
```

4.2.4 啟用 HTTP 站點

POST /sites/http/{id}/enable

```
curl -X POST "http://alb-master:8887/api/alb/console/external/v1/sites/http/42/enable" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

回應範例

```
{
  "code": 0,
  "data": null,
  "msg": "启用成功"
}
```

4.2.5 停用 HTTP 站點

POST /sites/http/{id}/disable

```
curl -X POST "http://alb-master:8887/api/alb/console/external/v1/sites/http/42/disable" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

回應範例

```
{
  "code": 0,
  "data": null,
  "msg": "禁用成功"
}
```

4.2.6 取得 HTTP 站點清單

GET /sites/http

查詢參數

欄位	類型	必填	說明
page	int	否	頁碼，預設 1
pageSize	int	否	每頁筆數，預設 10
name	string	否	按名稱模糊搜尋

請求範例

```
curl "http://alb-master:8887/api/alb/console/external/v1/sites/http?
page=1&pageSize=10" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

回應範例

```
{
  "code": 0,
  "data": [
    {
      "id": 42,
      "name": "官网",
      "domains": "www.example.com",
      "listen_port": 80,
      "status": "enabled",
      "created_at": "2024-01-15 10:30:00"
    }
  ],
  "msg": "ok"
}
```

4.2.7 取得 HTTP 站點詳情

GET /sites/http/{id}

```
curl "http://alb-master:8887/api/alb/console/external/v1/sites/http/42" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

回應範例

```
{
  "code": 0,
  "data": {
    "id": 42,
    "name": "官网",
    "domains": "www.example.com",
    "listen_port": 80,
    "locations": [
      {
        "path": "/",
        "type": "proxy",
        "upstream_name": "backend-web"
      }
    ],
    "status": "enabled",
    "created_at": "2024-01-15 10:30:00"
  },
  "msg": "ok"
}
```

4.2.8 為 HTTP 站點繫結 SSL 憑證

POST /sites/http/bind-ssl

請求參數

欄位	類型	必填	說明
site_id	int	是	站點 ID

cert_id	int	是	憑證 ID
---------	-----	---	-------

```
curl -X POST "http://alb-master:8887/api/alb/console/external/v1/sites/http/bind-ssl" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
-H "Content-Type: application/json" \
-d '{
  "site_id": 42,
  "cert_id": 2
}'
```

4.3 Stream 站點介面

Stream 站點用於四層負載均衡（TCP/UDP），介面結構與 HTTP 站點一致。

4.3.1 建立 Stream 站點

POST /sites/stream

請求參數

欄位	類型	必填	說明
name	string	是	站點名稱
listen_port	int	是	監聽埠
protocol	string	是	TCP / UDP
backend_servers	string(JSON)	是	後端伺服器清單（JSON 字串）
auto_reload	bool	否	儲存後自動重新載入，預設 true
status	string	否	enabled / disabled, 預設 enabled

BackendServer 欄位說明

欄位	類型	必填	說明
address	string	是	後端位址，如 192.168.1.10:3306
weight	int	否	權重，預設 1

max_fails	int	否	最大失敗次數
fail_timeout	int	否	失敗逾時（秒）
backup	bool	否	是否為備份伺服器

請求範例

```
curl -X POST "http://alb-master:8887/api/alb/console/external/v1/sites/stream" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
-H "Content-Type: application/json" \
-d '{
  "name": "数据库代理",
  "listen_port": 3306,
  "protocol": "TCP",
  "backend_servers": "
[{\\"address\\":\\"192.168.1.10:3306\\",\\"weight\\":1}],
  "auto_reload": true,
  "status": "enabled"
}'
```

回應範例

```
{
  "code": 0,
  "data": {
    "id": 15,
    "name": "数据库代理",
    "listen_port": 3306,
    "protocol": "TCP",
    "status": "enabled"
  },
  "msg": "创建成功"
}
```

4.3.2 更新 Stream 站點

PUT /sites/stream

```
curl -X PUT "http://alb-master:8887/api/alb/console/external/v1/sites/stream" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
-H "Content-Type: application/json" \
-d '{
  "id": 15,
  "backend_servers": "
[{"address":"192.168.1.11:3306","weight":1}]'
}
```

4.3.3 刪除 Stream 站點

DELETE /sites/stream

```
curl -X DELETE "http://alb-master:8887/api/alb/console/external/v1/sites/stream" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
-H "Content-Type: application/json" \
-d '{"id": 15}'
```

4.3.4 啟用 / 停用 Stream 站點

POST /sites/stream/{id}/enable

POST /sites/stream/{id}/disable

```
curl -X POST "http://alb-master:8887/api/alb/console/external/v1/sites/stream/15/enable" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

```
curl -X POST "http://alb-
```

```
master:8887/api/alb/console/external/v1/sites/stream/15/disable" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

4.3.5 取得 Stream 站點清單

GET /sites/stream?page=1&pageSize=10

```
curl "http://alb-
master:8887/api/alb/console/external/v1/sites/stream?
page=1&pageSize=10" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

4.3.6 取得 Stream 站點詳情

GET /sites/stream/{id}

```
curl "http://alb-
master:8887/api/alb/console/external/v1/sites/stream/15" \
-H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

4.4 錯誤回應範例

4.4.1 鑑權失敗 (401)

```
{
  "code": 401,
  "msg": "invalid api key"
}
```

4.4.2 外部 API 未啟用 (403)

```
{
  "code": 403,
```

```
"msg": "外部 API 未启用"
}
```

4.4.3 HA 備節點寫入 (10001)

```
{
  "code": 10001,
  "msg": "当前节点为 HA 备节点, 请前往主节点执行"
}
```

4.4.4 業務錯誤 (7)

```
{
  "code": 7,
  "msg": "站点名称已存在"
}
```

4.5 快速驗證指令碼

```
#!/bin/bash

API_KEY="alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
BASE_URL="http://alb-master:8887/api/alb/console/external/v1"

echo "=== 1. 建立 HTTP 站點 (直接代理) ==="
curl -s -X POST "$BASE_URL/sites/http" \
  -H "X-API-Key: $API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "测试站点",
    "domains": "test.example.com",
    "listen_port": 80,
    "locations": "
```

```
[{"path":"/", "type":"proxy", "proxy_pass":"http://192.168.1.1:80"},
{"path":"/", "type":"proxy", "proxy_pass":"http://192.168.1.1:80"}],
  "upstreams": [],
  "custom_headers": [],
  "proxy_set_headers": [],
  "extra_directives": [],
  "auto_reload": true,
  "status": "enabled"
}' | jq .
```

echo "=== 2. 取得清單 ==="

```
curl -s "$BASE_URL/sites/http?page=1&pageSize=5" \
-H "X-API-Key: $API_KEY" | jq .
```

echo "=== 3. 停用站點 ==="

```
curl -s -X POST "$BASE_URL/sites/http/1/disable" \
-H "X-API-Key: $API_KEY" | jq .
```

echo "=== 4. 刪除站點 ==="

```
curl -s -X DELETE "$BASE_URL/sites/http" \
-H "X-API-Key: $API_KEY" \
-H "Content-Type: application/json" \
-d '{"id": 1}' | jq .
```

4.6 稽核查詢

外部 API 呼叫記錄在 `sys_operation_logs` 資料表中，可透過以下 SQL 查詢：

```
SELECT
  created_at,
  operator_name,
  oper_type,
  request_method,
  request_path,
  status
```

```
FROM sys_operation_logs
WHERE operator_name LIKE 'external:%'
ORDER BY id DESC
LIMIT 20;
```

範例輸出：

created_at	operator_name	oper_type	request_method	request_path	
2024-01-15 10:30:01	external:erp-system	外部API: 建立HTTP 站點	POST	/api/alb/console/external/v1/sites/http	

5 alb-cli 命令列工具

版本： V2.0.6

定位： 面向 Claude、OpenCode 等 AI Agent 的 ALB 管控台命令列入口

5.1 安裝與設定

ALB V2.0.6 產品包自帶 `alb-cli` 可執行程式，在 **【ALB安裝路徑】** `/utils/alb-cli` 中。

5.1.1 鑑權方式

`alb-cli` 透過 API Key 與管控台通訊，請求標頭為 `X-API-Key`。

設定 API Key：在後端設定檔 **【ALB安裝目錄】** `/console/conf/config.yaml` 中設定 `external.api_keys`，或透過環境變數 `ALB_API_KEY` 設定。

5.1.2 設定來源（優先順序從高到低）

1. 環境變數
2. 設定檔 `~/.alb/config.yaml`

環境變數

```
export ALB_CONSOLE_URL=https://172.21.61.46:8887
export ALB_API_KEY=your-api-key
```

設定檔

```
# ~/.alb/config.yaml
url: https://172.21.61.46:8887
api_key: your-api-key
```

5.1.3 驗證連線

```
alb-cli status
```

5.2 全域參數

參數	說明	預設值
--output	輸出格式: json 或 table	json
-h, --help	檢視說明	-

全域參數可放在子命令前後：

```
alb-cli --output table site list --type http
alb-cli site list --type http --output table
```

5.3 輸出格式

5.3.1 JSON (預設)

所有查詢命令預設輸出 JSON，方便 Agent 解析。

```
alb-cli site list --type http
```

範例輸出：

```
[
  {
    "id": 1,
    "name": "example",
    "domains": "example.com",
    "listen_port": 80,
    "enable_ssl": false,
    "status": "enabled"
  }
]
```

5.3.2 Table

適合人工閱讀。

```
alb-cli site list --type http --output table
```

範例輸出：

```
ID    NAME      PORT  STATUS  DOMAINS
1     example  80    enabled example.com
```

5.3.3 錯誤輸出

所有錯誤統一輸出到 `stderr`：

```
missing API key: set ALB_API_KEY or ~/.alb/config.yaml api_key
```

伺服器端回傳的錯誤也會直接列印訊息。

5.4 命令參考

5.4.1 `site` — 站點管理

```
alb-cli site [子命令] --type http|stream [參數]
```

全域旗標：

旗標	說明	預設值
<code>--type</code>	站點類型：http 或 stream	http
<code>--status</code>	按狀態過濾：enabled / disabled	-

`site list` — 列出站點

```
alb-cli site list --type http
alb-cli site list --type http --status enabled
alb-cli site list --type stream --output table
```

`site get` — 檢視站點詳情

```
alb-cli site get 1 --type http
```

site create — 從 JSON 檔案建立站點

```
alb-cli site create --type http --file site.json
```

site.json 範例 (HTTP) :

```
{
  "name": "example",
  "domains": "example.com www.example.com",
  "listen_port": 80,
  "enable_ssl": false,
  "locations": "[{\"path\":\"/\", \"type\":\"proxy\", \"proxy_pass\":\"http://127.0.0.1:8080\"}]"
}
```

site.json 範例 (Stream) :

```
{
  "name": "tcp-example",
  "listen_port": 9090,
  "protocol": "TCP",
  "backend_servers": "[{\"address\":\"127.0.0.1:8080\", \"weight\":1}]"
}
```

site update — 從 JSON 檔案更新站點

```
alb-cli site update 1 --type http --file site.json
```

JSON 檔案中不需要包含 `id`，CLI 會自動將命令列傳入的 ID 寫入請求主體。

site enable / **site disable** — 啟用/停用站點

```
alb-cli site enable 1 --type http
```

```
alb-cli site disable 1 --type stream
```

`site delete` — 刪除站點

```
alb-cli site delete 1 --type http
```

`site bind-ssl` — 為 HTTP 站點繫結 SSL 憑證

```
alb-cli site bind-ssl --site-id 1 --cert-id 2
```

5.4.2 `status` — ALB 執行狀態

```
alb-cli status
```

回傳目前 ALB 執行狀態、監聽埠、server 數量等。

5.4.3 `metrics` — 監控資料

`metrics traffic` — 流量趨勢

```
alb-cli metrics traffic --metric-type min    # 過去 1 小時
alb-cli metrics traffic --metric-type hour   # 過去 24 小時
```

`metrics nodes` — 多節點狀態快照

```
alb-cli metrics nodes
```

5.4.4 `config reload` — 重新載入 ALB 設定

```
alb-cli config reload
```

執行後 ALB 會重新載入設定檔，使站點變更生效。

5.4.5 `log` — 日誌檢視

`log list` — 日誌檔案清單

```
alb-cli log list
```

`log show` — 檢視完整日誌內容

```
alb-cli log show access.log
```

`log tail` — 檢視日誌尾部

```
alb-cli log tail access.log -n 100
```

5.5 安全策略

5.5.1 危險操作確認

以下命令預設需要終端機互動確認：

- `site create`
- `site update`
- `site enable`
- `site disable`
- `site delete`
- `site bind-ssl`
- `config reload`

範例互動：

```
$ alb-cli config reload
reload ALB configuration? [y/N]: y
```

5.5.2 非互動環境

如果 `stdout` 或 `stdin` 不是 TTY，寫操作會被拒絕：

```
destructive operation requires an interactive terminal
```

這可以避免 Agent 在 CI / 指令碼中誤觸發破壞性操作。

5.6 典型工作流程

5.6.1 發佈一個 HTTP 站點

```
# 1. 建立站點
alb-cli site create --type http --file site.json

# 2. 重新載入設定使其生效
alb-cli config reload

# 3. 驗證站點狀態
alb-cli site list --type http --output table
```

5.6.2 為站點啟用 HTTPS

```
# 1. 繫結憑證
alb-cli site bind-ssl --site-id 1 --cert-id 2

# 2. 重新載入設定
alb-cli config reload
```

5.6.3 日常巡檢

```
alb-cli status
alb-cli metrics traffic --metric-type min
alb-cli metrics nodes
```

5.6.4 故障排查

```
alb-cli log list
alb-cli log tail error.log -n 50
```

5.7 後端介面對應

CLI 命令	後端 API
site list --type http	GET /api/alb/console/external/v1/sites/http
site get 1 --type http	GET /api/alb/console/external/v1/sites/http/1
site create --type http	POST /api/alb/console/external/v1/sites/http
site update 1 --type http	PUT /api/alb/console/external/v1/sites/http
site enable 1 --type http	POST /api/alb/console/external/v1/sites/http/1/enable
site disable 1 --type http	POST /api/alb/console/external/v1/sites/http/1/disable
site delete 1 --type http	DELETE /api/alb/console/external/v1/sites/http?id=1
site bind-ssl	POST /api/alb/console/external/v1/sites/http/bind-ssl
status	GET /api/alb/console/external/v1/alb/status
metrics traffic	GET /api/alb/console/external/v1/alb/flow?type=min
metrics nodes	GET /api/alb/console/external/v1/analytic/nodes
config reload	POST /api/alb/console/external/v1/config/reload
log list	GET /api/alb/console/external/v1/alb/logger/name
log show	GET /api/alb/console/external/v1/alb/logger/text?logger_name=...

6 AI Agent 整合

ALB 提供面向 AI Agent（Claude、OpenCode 等）的標準 SKILL 描述，使 Agent 可以安全、可控地呼叫 ALB 管控台能力。

6.1 SKILL 用途

透過 `alb-cli` 與 ALB 管控台互動，AI Agent 可以完成：

- 站點管理（建立、更新、啟用/停用、刪除、SSL 繫結）
- 狀態監控（執行狀態、流量趨勢、節點健康度）
- 設定重新載入（修改後即時生效）
- 日誌排查（list / show / tail）

6.2 SKILL安裝

【ALB安裝路徑】`/utils/alb-cli/alb-cli/SKILL.md` 是面向 Claude Code、OpenCode 等 AI Agent 的 Skill 定義檔案，安裝後 Agent 可以透過自然語言直接呼叫 `alb-cli` 完成 ALB 維運任務。

6.2.1 在 Claude Code 中安裝

Claude Code 會自動載入專案根目錄 `.claude/skills/<skill-name>/SKILL.md` 下的 Skill。

6.2.1.1 方式一：複製到專案 Skill 目錄（推薦）

```
# 從原始碼目錄複製到專案根目錄的 .claude/skills/  
cp -r 【ALB安裝路徑】/utils/alb-cli/alb-cli/alb-cli .claude/skills/
```

複製後目錄結構：

```
.claude/skills/  
└─ alb-cli/  
    └─ SKILL.md
```

6.2.1.2 方式二：手動建立

在專案根目錄執行：

```
mkdir -p .claude/skills/alb-cli
cp ALB【安裝路徑】/utils/alb-cli/alb-cli/SKILL.md
.claude/skills/alb-cli/SKILL.md
```

6.2.1.3 驗證安裝

重新啟動 Claude Code 或在對話中輸入：

```
/alb-cli 查看状态
```

如果 Claude Code 辨識了該 Skill，會自動執行 `alb-cli status` 並回傳 ALB 執行狀態。

6.3 在其他 Agent 工具中使用

6.3.1 OpenCode

OpenCode 同樣遵循 `.claude/skills/<skill-name>/SKILL.md` 的目錄約定。按方式一複製後即可使用，觸發詞與 Claude Code 相同。

6.3.2 通用相容工具

任何支援 Markdown Skill 定義的 Agent 工具，只要讀取 `SKILL.md` 中的 `description` 和 `allowed-tools`，即可呼叫其中定義的命令。安裝方式通常是：

1. 將 `cli/docs/alb-cli/SKILL.md` 複製到該工具指定的 skill 目錄
2. 確保工具能夠存取 `alb-cli` 二進位檔（已隨管控制台打包到 `console/tools/`，或單獨建置到 `cli/alb-cli`）
3. 設定環境變數或 `~/.alb/config.yaml`

6.4 前置依賴

使用Skill 前，確保目標環境滿足：

1. ALB 管控制台可存取：設定 `ALB_CONSOLE_URL`，例如 `http://127.0.0.1:8887`
2. API Key 有效：設定 `ALB_API_KEY`，該 key 需與後端 `config/config.yaml` 中 `external.api_keys` 的某一項相符

範例環境變數：

```
export ALB_CONSOLE_URL=http://127.0.0.1:8887
export
ALB_API_KEY=9489ebf25ac6664c242e0b2f2b406164d4175ae227bd7b09db762d66c58fa32b
```

或設定檔 `~/.alb/config.yaml`：

```
url: http://127.0.0.1:8887
api_key:
9489ebf25ac6664c242e0b2f2b406164d4175ae227bd7b09db762d66c58fa32b
```

6.5 觸發方式

使用者輸入 `/alb-cli` 或類似指令時觸發，例如：

- "幫我看看 ALB 目前狀態"
- "列出所有 HTTP 站點"
- "建立一個新站點"
- "重新載入 ALB 設定"
- "檢視最近 100 行 error.log"

6.6 可用命令速查

任務	命令
檢視執行狀態	<code>alb-cli status</code>
列出 HTTP 站點	<code>alb-cli site list --type http</code>
列出 Stream 站點	<code>alb-cli site list --type stream</code>
檢視站點詳情	<code>alb-cli site get <id> --type http</code>
建立站點	<code>alb-cli site create --type http --file site.json</code>
更新站點	<code>alb-cli site update <id> --type http --file site.json</code>
啟用/停用站點	<code>alb-cli site enable <id> --type http / alb-cli site disable <id> --type http</code>
刪除站點	<code>alb-cli site delete <id> --type http</code>
繫結 SSL 憑證	<code>alb-cli site bind-ssl --site-id <id> --cert-id <id></code>

檢視流量	<code>alb-cli metrics traffic --metric-type min</code>
檢視節點狀態	<code>alb-cli metrics nodes</code>
重新載入設定	<code>alb-cli config reload</code>
日誌清單	<code>alb-cli log list</code>
檢視日誌	<code>alb-cli log show <name></code>
追蹤日誌	<code>alb-cli log tail <name> -n <lines></code>

6.7 預設輸出與解析

- 預設輸出 `json`，Agent 應優先解析 JSON。
- 需要人工可讀表格時可加 `--output table`。
- 錯誤訊息會輸出到 `stderr`。

6.8 安全約束

寫操作需要終端機確認。以下命令在執行時會提示 `[y/N]`：

- `site create`
- `site update`
- `site enable`
- `site disable`
- `site delete`
- `site bind-ssl`
- `config reload`

如果目前環境不是互動式終端機（`stdout/stdin` 不是 TTY），這些命令會失敗並回傳：

```
destructive operation requires an interactive terminal
```

Agent 在呼叫這些命令前，必須先向使用者確認，再讓使用者在終端機執行，或要求使用者明確授權。

6.9 標準工作流程

6.9.1 工作流程 1：發佈 HTTP 站點

```
# 1. 建立站點 (使用者確認後執行)
alb-cli site create --type http --file site.json

# 2. 重新載入設定 (使用者確認後執行)
alb-cli config reload

# 3. 驗證
alb-cli site list --type http
```

6.9.2 工作流程 2: 啟用 HTTPS

```
# 使用者確認後執行
alb-cli site bind-ssl --site-id 1 --cert-id 2
alb-cli config reload
```

6.9.3 工作流程 3: 日常巡檢

```
alb-cli status
alb-cli metrics traffic --metric-type min
alb-cli metrics nodes
```

6.9.4 工作流程 4: 故障排查

```
alb-cli status
alb-cli log list
alb-cli log tail error.log -n 50
```

6.10 JSON 檔案範本

建立/更新 HTTP 站點時，JSON 檔案範例：

```
{
  "name": "example",
```

```
"domains": "example.com www.example.com",
"listen_port": 80,
"enable_ssl": false,
"locations": "
[{"path\":"\"/\",\"type\":"\"proxy\", \"proxy_pass\":"\"http://127.0.0.1:8080\""}]
```

Stream 站點範例：

```
{
  "name": "tcp-example",
  "listen_port": 9090,
  "protocol": "TCP",
  "backend_servers": "
[{"address\":"\"127.0.0.1:8080\", \"weight\":"\"1\""}]
```

Agent 產生 JSON 檔案後，可呼叫：

```
alb-cli site create --type http --file /tmp/site.json
```

6.11 常見錯誤處理

錯誤	處理
missing console URL	提示使用者設定 ALB_CONSOLE_URL 或 ~/.alb/config.yaml
missing API key	提示使用者設定 ALB_API_KEY
invalid api key	提示使用者在管控台重新產生 API Key 並啟用外部 API
destructive operation requires an interactive terminal	說明該命令需要使用者確認，請使用者手動執行或切換到互動終端機
alb状态模块未开启	提示使用者在管控台開啟狀態模組後再查詢流量

6.12 最佳實踐

1. 先查詢再修改：建立/更新/刪除站點前，先 `site list` 確認現狀。

2. **修改後重新載入**：站點變更類操作（create/update/enable/disable/delete/bind-ssl）後，建議執行 `config reload`。
 3. **優先 JSON**：Agent 解析時使用 `--output json` 預設值。
 4. **避免批次危險操作**：不要在沒有使用者確認的情況下迴圈執行 delete/disable/reload。
 5. **檔案清理**：暫存 JSON 檔案使用 `/tmp/` 目錄，任務完成後可刪除。
-

7 維運與監控介面

本節彙總 ALB 對外提供的維運與監控相關能力，便於維運平台和監控系統整合。

7.1 ALB 執行狀態

介面/命令	說明
GET /api/alb/console/external/v1/alb/status	目前 ALB 節點執行狀態、監聽埠、server 數量
alb-cli status	CLI 入口，等同上述 API

7.2 流量與節點監控

介面/命令	說明
GET /api/alb/console/external/v1/alb/flow?type=min	過去 1 小時流量趨勢
GET /api/alb/console/external/v1/alb/flow?type=hour	過去 24 小時流量趨勢
GET /api/alb/console/external/v1/analytic/nodes	多節點狀態快照（HA 叢集下使用）
alb-cli metrics traffic / alb-cli metrics nodes	CLI 入口

提示：呼叫 `metrics traffic` 前，請確認 ALB 狀態模組 (`stub_status`) 已開啟。

7.3 設定重新載入

介面/命令	說明
POST /api/alb/console/external/v1/config/reload	通知 ALB 重新載入設定
alb-cli config reload	CLI 入口，會觸發互動確認

注意：HA 叢集中寫操作會自動從主節點同步到備節點，無需在備節點重複呼叫。

7.4 日誌查詢

介面/命令	說明
GET /api/alb/console/external/v1/alb/logger/name	日誌檔案清單
GET /api/alb/console/external/v1/alb/logger/text?logger_name=...	讀取指定日誌內容

alb-cli log list / alb-cli log show / alb-cli log tail
--

CLI 入口

8 典型整合場景

8.1 場景 1：CI/CD 自動發佈 HTTP 站點

在持續整合流水線中，透過 alb-cli 把建置產物發佈到 ALB：

```
# 1. 準備站點 JSON
cat > /tmp/site.json <<EOF
{
  "name": "web-${BUILD_ID}",
  "domains": "web.example.com",
  "listen_port": 80,
  "locations": "
[[{\"path\":\"/\",\"type\":\"proxy\",\"proxy_pass\":\"http://${APP_HOST}
  "auto_reload": true,
  "status": "enabled"
}
EOF

# 2. 由維運在終端機執行（避免 CI 誤觸發）
export ALB_CONSOLE_URL=https://alb-master:8887
export ALB_API_KEY=${ALB_API_KEY}
alb-cli site create --type http --file /tmp/site.json
alb-cli config reload
```

8.2 場景 2：與 ERP / 業務系統對接

透過外部 API，在 ERP 上錄入「業務系統」時自動在 ALB 建立反向代理：

```
import requests

API_KEY = "alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
BASE = "http://alb-master:8887/api/alb/console/external/v1"

def create_proxy(name, domain, port, upstream):
```

```

return requests.post(
    f"{BASE}/sites/http",
    headers={"X-API-Key": API_KEY, "Content-Type":
"application/json"},
    json={
        "name": name,
        "domains": domain,
        "listen_port": port,
        "locations":
f'[[{"path":"/", "type":"proxy", "proxy_pass":"{upstream}"}]]',
        "auto_reload": True,
        "status": "enabled",
    },
).json()

```

8.3 場景 3：日常巡檢機器人 (AI Agent)

將 SKILL 接入 Claude / OpenCode，讓 AI 進行日常巡檢：

使用者：幫我看下 ALB 現在的狀態，並檢視最近 50 行 error.log

AI 執行：

1. alb-cli status
2. alb-cli log list
3. alb-cli log tail error.log -n 50

AI 輸出 (結合 JSON 解析)：目前 ALB 主節點執行正常，監聽 3 個埠，最近 50 行 error.log 中未發現 ERROR 級別日誌。

8.4 場景 4：故障自動定位

使用者：192.168.1.100:8080 這個後端存取不了，幫我看看

AI 執行：

```
1. alb-cli site list --type http      (找到對應站點 ID)
2. alb-cli site get <id> --type http
3. alb-cli log tail error.log -n 100
4. alb-cli metrics nodes
```

AI 輸出：站點 test-api 的 upstream 後端 192.168.1.100:8080 健康檢查失敗，請檢查後端服務是否存活。

9 整合建議

- 不要把 API Key 寫進程式碼倉庫，建議透過環境變數、K8s Secret、Vault 等外部方式注入。
 - 正式環境使用 HTTPS 存取管控台，避免 API Key 在網路上明文傳輸。
 - 為不同業務系統分配獨立 API Key，便於按需撤銷、稽核與配額隔離。
 - HA 叢集下呼叫寫操作前，建議先 `alb-cli status` 確認主節點身分，再將請求發往主節點。
-

10 錯誤碼速查表

10.1 外部 API 錯誤碼

code	含義	建議處理
0	成功	-
7	通用業務錯誤	讀取 msg 欄位，根據業務提示處理
401	鑑權失敗	檢查 X-API-Key 是否正確、是否過期
403	外部 API 未啟用	在管控台 系統管理 > 開放 API 中開啟
10001	目前節點為 HA 備節點	將請求發往主節點

10.2 alb-cli 常見錯誤

錯誤訊息	原因	建議處理
missing console URL	未設定 ALB_CONSOLE_URL 或 ~/.alb/config.yaml	檢查環境變數或設定檔
missing API key	未設定 ALB_API_KEY	檢查環境變數或設定檔
invalid api key	API Key 錯誤或未啟用外部 API	在管控台重新產生並啟用
destructive operation requires an interactive terminal	非 TTY 環境下執行寫操作	在終端機中手動執行
alb状态模块未开启	呼叫 metrics traffic 時 ALB 狀態 模組未啟用	在管控台開啟 stub_status 或狀態模組

11 最佳實踐與故障排查

11.1 最佳實踐

1. **先查詢再修改**：寫操作前先透過 `site list` 確認目標 ID 與目前狀態。
2. **修改後重新載入**：所有站點變更類操作後呼叫 `config reload`。
3. **寫操作走互動確認**：CI / 指令碼中不要繞過 alb-cli 的安全確認，建議由維運在終端機手動觸發。
4. **使用 HTTPS**：正式環境透過 HTTPS 存取管控制台，避免 API Key 明文傳輸。
5. **分級 API Key**：不同業務系統分配獨立 API Key，並按需在管控制台撤銷。
6. **稽核追溯**：所有外部 API 呼叫都會寫入 `sys_operation_logs`，可透過 SQL 查詢 `operator_name LIKE 'external:%'`。
7. **HA 叢集**：寫操作發往主節點；備節點寫操作會被拒絕（code=10001）。

11.2 故障排查

現象	可能原因	建議處理
呼叫 API 回傳 401	API Key 錯誤或過期	在管控制台重新產生 API Key
呼叫 API 回傳 403	外部 API 未啟用	在管控制台 系統管理 > 開放 API 中啟用
寫操作回傳 10001	目前為 HA 備節點	將請求發往主節點
alb-cli 提示 missing api key	環境變數 / 設定檔未設定	設定 <code>ALB_CONSOLE_URL</code> 與 <code>ALB_API_KEY</code>
alb-cli 提示 requires an interactive terminal	非 TTY 環境	切換到終端機或由維運手動執行
metrics traffic 提示狀態模組未開啟	ALB stub_status 未啟用	在管控制台開啟狀態模組
建立站點後存取 502	後端服務不可達	檢查 upstream 節點連通性

12 版本更新說明

日期	手冊版本	適用產品	更新說明
2026年06月	V2.0.6	ALB V2.0.6 標準版	全新發佈：外部 API 介面、alb-cli、AI Agent SKILL

全國統一服務熱線
4008-555-800

金蝶天燕雲計算股份有限公司（簡稱“金蝶天燕”）成立於2000年，前身為“金蝶中間件公司”，是金蝶集團旗下基礎軟件平台提供商，雲計算國家標準制定企業，國家信創產業核心軟件企業。金蝶天燕是國家863重點研發計劃與核高基重大專項承接企業，也是“兩網一站四庫十二金”國家重點工程的基礎平台提供商，產品廣泛應用於政府、軍工、金融、能源等關鍵行業，累計服務客戶總數超過10萬家。



雲計算國家標準制定企業
金蝶集團旗下基礎軟件企業
資訊技術應用創新核心企業
官網：www.apusic.com

