河马的CC商业实战 · 会员专属资料 #6

2026年8月全面修订 · 国内直连方案 · cc-switch 实操

Claude Code 国内使用手册

不折腾网络，也能用好最强的 AI 编程 Agent

官方信息概览 · cc-switch 与国产模型端点（GLM/DeepSeek/Kimi）· 网络诊断
进阶技巧 · MCP 接入 · 定价与成本优化 · 竞品选型 · 命令与环境变量速查

河马的CC商业实战 出品

2026年8月 · 会员专属版

会员专属资料 · 第 6 期

Claude Code 国内使用手册

国内开发者用好 Claude Code 的完整指南（2026年8月版）

出品 河马的CC商业实战（知识星球）

版本 2026年8月 · 会员专属版（基于2026年5月版全面核实更新）

读者 想用 Claude Code 却被网络/账号/定价挡住的国内开发者

> 本资料为「河马的CC商业实战」星球会员专属，每周/不定期更新。欢迎转发给需要的朋友——如果他们想持续收到这个系列，封底有入口。AI 工具迭代极快，价格、版本号、命令以官方页面和你实际环境为准，思路不变。

河马卷首语

星友们，「会员专属资料」第 6 期来了。

这一期是 2026年5月版《Claude Code 国内用户完整手册》的全面修订版。这三个月变化太大：Claude 进入 5 时代（Opus 5 成了默认模型）、Auto Mode 默认开启、定价调整，国内生态更是变了天——智谱、DeepSeek、Kimi 都推出了官方兼容端点，cc-switch 一键切换成为主流玩法。所以这本书的第二章整个重构了：从「怎么翻墙用官方」变成「怎么零障碍接国产模型」，每一条接入路径都核实到了官方文档。

用法建议：新读者从第一章认识 Claude Code 开始；急着上手直接翻第二章，三行配置接入 GLM/DeepSeek/Kimi 就能跑；第五章定价对比帮你算账——国产 Coding 套餐和官方订阅哪个划算，一目了然。

跑通了记得来星球里分享你选的路线。

—— 河马，2026年8月

目 录

CHAPTER ONE

第一章 Claude Code 官方信息概览

1.1 Claude Code 是什么

1.2 安装方式

1.3 核心功能全景

1.4 CLAUDE.md 完整指南

1.5 官方最佳实践

1.6 2026年最新版本变化

CHAPTER TWO

第二章 国内使用环境配置

2.1 常见网络错误与诊断

2.2 国产模型接入：cc-switch 与官方兼容端点（2026年主流方案）

2.3 官方 API 代理服务配置

2.4 VPN 代理配置

2.5 中转 API 服务：风险提示与选择标准

2.6 国内开发者真实使用体验

CHAPTER THREE

第三章 进阶用法与技巧

3.1 自定义 Slash 命令

3.2 Git 工作流深度集成

3.3 上下文管理技巧

3.4 MCP 工具接入（重点章节）

3.5 Vibe Coding 工作流

3.6 Headless 模式与 CI/CD 集成

3.7 Claude Code Hooks 系统

3.8 Claude Code + OpenClaw 联动方案

CHAPTER FOUR

第四章 常见问题与解决方案

4.1 安装与登录问题

4.2 Token 消耗控制

4.3 权限与安全管理

4.4 Prompt Caching 深入

CHAPTER FIVE

第五章 定价与成本优化

5.1 官方订阅方案

5.2 API 按量计费

5.3 套餐选择建议

5.4 订阅 vs API 对比

5.5 使用成本追踪

CHAPTER SIX

第六章 工具对比与选型

6.1 Claude Code vs Cursor

6.2 选型建议

6.3 其他替代工具（2026年8月格局）

APPENDIX

附录

A. 常用命令速查

B. 快捷键速查

C. 环境变量速查

D. MCP Server 推荐清单

E. 信息来源说明

F. 免责声明

CHAPTER ONE

第一章 Claude Code 官方信息概览

1.1 Claude Code 是什么

1.2 安装方式

1.3 核心功能全景

1.4 CLAUDE.md 完整指南

1.5 官方最佳实践

1.6 2026年最新版本变化

1.1 Claude Code 是什么

Claude Code 是 Anthropic 官方推出的终端原生智能编码 Agent，于 2024 年末首次发布，历经 2.x 系列快速迭代，截至 2026 年 8 月已更新至 v2.1.23x 系列（约每周一次发版节奏）。

一句话定义：Claude Code 是一个直接在命令行中运行的 AI 编程助手，能够读写代码、执行命令、管理 Git、连接外部工具，并自主完成从需求分析到代码提交的完整开发任务。

核心能力列表：

能力类别	说明
代码生成	根据自然语言描述生成完整代码文件或函数，支持 50+ 编程语言
代码理解	快速分析大型代码库的结构、依赖关系和业务逻辑
调试修复	定位 Bug 根源并生成修复方案，可自动运行测试验证
Git 集成	执行 commit、branch、diff、PR 等全部 Git 操作，自动生成提交信息
命令执行	在终端中运行 shell 命令，完成构建、测试、部署等操作
MCP 连接	通过 Model Context Protocol 连接数据库、API、第三方服务
多平台运行	支持 macOS、Linux、WSL 和原生 Windows
多形态覆盖	除 CLI 外，还有桌面应用（Mac/Windows）、Web 端（claude.ai/code）、IDE 扩展（VS Code/JetBrains）

官方定位：Anthropic 将 Claude Code 定位为"软件开发的自动驾驶仪"，目标是让开发者用自然语言描述需求，由 AI 完成实现细节。2026 年以来的里程碑包括：Auto Mode 于 2026年8月14日起对付费账户默认开启（除不可逆/破坏性操作外不再逐步请求许可）；Opus 5 于 2026年7月24日成为 Claude Code 默认模型；Fast 模式（/fast，约 2.5 倍速度）随 Opus 5 推出。

模型家族（2026年8月）：当前为 Claude 5 时代——Fable 5（前沿旗舰，2026年6月）、Opus 5（旗舰，CC 默认）、Sonnet 5（均衡主力）、Haiku 4.5（快速廉价），均为 1M 上下文（Haiku 4.5 为 200K）；Opus 4.8 仍可用于 Fast 模式，Claude 3/3.5 系列已退役。

> **来源**：Anthropic 官方博客（Opus 5 发布、Auto Mode 默认开启公告）、TechCrunch 报道（2026年8月）、GitHub Releases

1.2 安装方式

macOS / Linux / WSL（推荐方式）

```
# 使用官方安装脚本一键安装
curl -fsSL https://claude.ai/install.sh | bash

# 安装完成后登录（也可用 claude auth login）
claude login

# 验证安装
claude --version
```

Windows PowerShell

```
# 在 PowerShell 中执行
irm https://claude.ai/install.ps1 | iex

# 登录
claude login

# 验证
claude --version
```

包管理器安装（可选）

```
# Windows WinGet
winget install Anthropic.ClaudeCode

# macOS Homebrew
brew install --cask claude-code
```

npm 安装（遗留方式，仍可用）

```
# 要求 Node.js 18+
# 验证 Node.js 版本
node -v

# 通过 npm 全局安装
npm install -g @anthropic-ai/claude-code
```

```
# 登录
claude login
```

> ⚠️ **安全提醒**：只从上述官方渠道安装。2026 年已出现多个假冒"Claude Code 桌面版"的第三方 GitHub 仓库（疑似恶意软件），桌面应用唯一官方下载源是 claude.com/download。

安装验证命令：

命令	预期输出
`claude --version`	`2.1.xxx`（版本号）
`claude --help`	显示所有可用选项
`claude doctor`	环境诊断报告（含自动修复）

> **来源**：Claude Code 官方安装文档（https://code.claude.com/docs）及 2026 年命令参考。注意：5月版手册中的 `get.claude.com/cli` 安装地址未能在 2026年8月的任何可靠来源核实到，已更正为官方现行的 `claude.ai/install.sh` / `install.ps1`。

1.3 核心功能全景

功能类别	具体能力	使用频率
代码编辑	创建、修改、删除文件；支持多文件同时编辑	每次会话
终端命令	执行任意 shell 命令，自动确认危险操作	每次会话
Git 操作	commit、branch、merge、rebase、生成 PR 描述	每次会话
代码搜索	基于语义和正则的代码库搜索	高
子 Agent	并行执行多个子任务，独立上下文隔离（默认嵌套深度 3，无单会话数量上限）	中
MCP 工具	连接外部数据库、API、浏览器等工具	高
Hooks 系统	在工具调用前后执行自定义脚本（2026 年已扩展至 31 个事件）	中
会话管理	/rewind（别名 /undo）回退、/clear 重置、/compact 压缩	高
远程访问	从手机或浏览器监控/接管本地会话（Remote Control）	低
计划任务	/loop 定时执行重复性任务（配 Cron 调度）	低
自定义命令	.claude/commands/ 目录下的 Markdown 命令文件	中
Auto Mode	自动判断并执行安全操作；2026年8月14日起对付费账户默认开启	高
Fast 模式	/fast 切换，同模型约 2.5 倍速度（仅限 Opus 5 / Opus 4.8）	中
后台 Agent	git worktree 隔离的并行后台子任务	中

> **来源**：Claude Code 官方文档功能索引、Anthropic 官方公告（2026年8月核实）

1.4 CLAUDE.md 完整指南

什么是 CLAUDE.md

CLAUDE.md 是 Claude Code 的**项目记忆文件**，用于存储项目背景信息、技术栈说明、编码规范等。每次会话开始时，Claude Code 会自动读取该文件并将其注入系统提示，使 AI 对项目有全局认知。

文件位置与加载优先级

Claude Code 按以下优先级查找并加载 CLAUDE.md（数字越小优先级越高）：

优先级	位置	作用范围
1	`./CLAUDE.md`	当前项目目录（最高优先级）
2	`../CLAUDE.md`	上级目录（适用于 monorepo 子包）
3	`~/.claude/CLAUDE.md`	用户级全局配置
4	`~/.config/claude/CLAUDE.md`	系统级全局配置

当多个级别的文件同时存在时，Claude Code 会**合并**所有配置，高优先级的设置会覆盖低优先级。

三种创建方式

方式一：/init 命令（推荐）

```
claude
# 进入交互模式后执行
/init
```

Claude 会自动分析当前项目结构，生成一份基础 CLAUDE.md。

方式二：手动创建

```
touch CLAUDE.md
# 用你最喜欢的编辑器填写内容
```

方式三：# 键实时添加

在 Claude Code 交互会话中，输入 `#` 键可以快速添加当前观察到的重要信息到 CLAUDE.md。

CLAUDE.md 完整模板

```
# 项目概述

**项目名称**：Your Project Name
**技术栈**：TypeScript / React / Node.js / PostgreSQL
**项目目标**：一句话描述项目要解决什么问题

## 目录结构

- `src/` - 源代码目录
  - `components/` - React 组件
  - `utils/` - 工具函数
  - `api/` - 接口层
- `tests/` - 测试文件
- `docs/` - 文档

## 开发命令

| 命令 | 作用 |
|------|------|
| `npm run dev` | 启动开发服务器 |
| `npm run build` | 生产构建 |
| `npm run test` | 运行测试套件 |
| `npm run lint` | 代码检查 |
| `npm run format` | 代码格式化 |

## 编码规范

- 使用 2 空格缩进
- 字符串使用单引号
- 函数使用箭头函数风格
- 所有 API 接口必须包含 JSDoc 注释
- 组件文件名使用 PascalCase
- 工具函数文件名使用 camelCase

## Git 工作流

- main 分支为生产分支
- 功能开发从 main 切出 feature/xxx 分支
- 提交信息使用中文，格式：[类型] 描述
- 类型包括：feat/fix/docs/refactor/test
- 合并前必须通过 CI 检查

## 注意事项

- 不要修改 `.env` 文件中的生产环境配置
- 数据库迁移文件必须提交到版本控制
- API 调用统一通过 `src/api/client.ts` 中的封装函数
```

5 条 CLAUDE.md 编写最佳实践

1. **保持简洁**：控制在 200-500 行以内，过长的文件会消耗过多 Token
2. **结构化优先**：使用标题、表格、列表而非大段文字，便于 AI 快速定位
3. **动态更新**：项目重大变更时同步更新 CLAUDE.md，否则信息会过时
4. **编码规范具体化**：给出具体的代码示例而非抽象描述，如"使用单引号"而非"保持代码风格一致"
5. **分层管理**：通用规范放全局 CLAUDE.md，项目特有信息放本地 CLAUDE.md，避免重复

> **来源**：Claude Code 官方文档 CLAUDE.md 章节、社区最佳实践（2026年4月）

1.5 官方最佳实践

核心原则：上下文窗口管理

Claude Code 的上下文窗口虽然可达 200K-1M tokens，但有效管理仍是第一生产力。原则：**只保留当前任务相关的内容，及时清理过期上下文。**

探索-规划-编码-提交四步工作流

阶段	操作	对应命令/技巧
探索	理解代码库结构，定位相关文件	`@目录名` 快速引用、`/ls` 查看
规划	制定修改方案，与 Claude 确认	明确说出"请先给出修改计划，暂不执行"
编码	执行修改，运行测试验证	自然语言描述需求，AI 自动编辑
提交	生成 commit 信息，创建 PR	`/git commit` 、Claude 自动生成描述

有效沟通技巧（实战范例）

场景	低效提问	高效提问
添加功能	"帮我加个登录功能"	"在 src/pages/ 下新增 Login.tsx，使用 React Hook Form 处理表单，调用 src/api/auth.ts 的 login 接口，提交前验证邮箱格式和密码长度 >= 8"
修复 Bug	"这里报错了，修一下"	"运行 npm run test 时 src/utils/calc.ts 第 42 行的 divide 函数在 divisor 为 0 时抛异常，请添加参数校验并返回 0"
重构代码	"这段代码太乱了"	"将 src/components/Button.tsx 中的 150 行内联样式提取到 src/styles/button.module.css，保持现有接口不变"

Session 管理命令

命令	作用	使用场景
`Esc`	中断当前 AI 的思考和输出	AI 输出方向错误时立即打断
`/rewind`	回退到上一次用户输入前的状态	某一步操作导致错误，想撤销
`/clear`	清空当前会话的上下文历史	上下文过长导致响应变慢
`/compact`	压缩历史对话为摘要，释放 Token	需要保留任务背景但释放空间

> **来源**：Claude Code 官方最佳实践文档（https://code.claude.com/docs/best-practices）

1.6 2026年最新版本变化

重要版本更新时间线

版本/时间	核心变化
v2.1.23x（2026-08）	Bash 工具沙箱收紧、`--worktree` 支持 GitLab MR、MCP 凭证隔离加固、`claude mcp list` 显示禁用服务器、Managed Agents 发布
2026-08-14	**Auto Mode 对 Pro/Max/Team 账户默认开启**——除不可逆/破坏性操作外不再逐步请求许可（官方内测：安全分类器拦截 89% 危险操作，人工审核仅 13.6%）
2026-07-24（v2.1.219）	**Opus 5 发布并成为 Claude Code 默认模型**，同步推出 Fast 模式（/fast，约 2.5 倍速度、2 倍价格，仅限 Opus 5/4.8）
2026-06-30	Sonnet 5 发布（均衡主力，1M 上下文）
2026-06-09	Fable 5 发布（Mythos 级前沿旗舰，1M 上下文、128K 最大输出）
2026-05	Opus 4.8 发布；PostToolUseFailure 等新 Hook 事件加入
v2.1.111（2026-05）	/ultrareview 云端并行审查、/less-permission-prompts 权限白名单提议
v2.1.71（2026 上半年）	Cron 定时调度工具、/loop 定时任务
v2.1.63（2026 上半年）	Task 工具更名为 Agent、/batch、/simplify 内置
v2.0（2026-02）	1M token 上下文窗口（API）、Hooks 系统大规模引入

对使用者影响最大的变化

1. **Auto Mode 默认开启（2026年8月）**：这是交互方式的最大变化——安全操作不再逐步确认，只有不可逆/破坏性/越界操作才询问。企业/API 侧（Bedrock/Vertex/Foundry）可通过 `CLAUDE_CODE_ENABLE_AUTO_MODE=1` 选择加入
2. **Claude 5 家族换代（2026年6-7月）**：Opus 5 成为默认模型，全部主力模型支持 1M 上下文；Claude 3/3.5 系列已退役
3. **Fast 模式（2026年7月）**：`/fast` 切换，同权重同质量、约 2.5 倍速度、2 倍价格，适合交互式高强度开发
4. **Windows 原生支持**：不再依赖 WSL，Windows PowerShell 可直接安装运行

5. **子 Agent 体系成熟**：默认嵌套深度 3、取消单会话数量上限，复杂任务可大规模并行拆分
6. **Prompt Caching 自动启用**：CLAUDE.md 和工具定义自动缓存，降低 API 成本（2026 年新增自动缓存模式）

> **来源**：Anthropic 官方博客、TechCrunch（2026-08-09 Auto Mode 报道）、GitHub Releases、社区 changelog 追踪（2026年8月核实；具体版本号以官方 Releases 页为准）

CHAPTER TWO

第二章 国内使用环境配置

2.1 常见网络错误与诊断

2.2 国产模型接入：cc-switch 与官方兼容端点（2026年主流方案）

2.3 官方 API 代理服务配置

2.4 VPN 代理配置

2.5 中转 API 服务：风险提示与选择标准

2.6 国内开发者真实使用体验

2.1 常见网络错误与诊断

错误类型	错误信息原文	根因分析	解决方案
地区限制	"Claude is not available in your region"	Anthropic 官方服务对中国大陆 IP 限制	使用国产模型官方兼容端点（见 2.2 节）、API 代理或 VPN（见 2.3/2.4 节）
OAuth 403	"Error 403: access_denied" 或 "This account has been disabled"	账号被风控或地区检测到高风险 IP	更换节点地区（推荐日本/新加坡）、使用 API Key 方式
连接超时	"Request timed out" / "Connection reset by peer"	网络不稳定或 GFW 干扰	切换代理协议（HTTP/HTTPS）、更换节点
API 403	"403 Forbidden"	API 请求来源 IP 在黑名单中	改用国产模型端点（见 2.2 节）或中转 API（见 2.5 节，注意风险）
SSL 错误	"SSL certificate verify failed"	系统证书问题或中间人攻击检测	更新系统证书、检查代理是否篡改证书
端点配置错误	"Invalid API key" / "401 Unauthorized"（使用国产端点时）	`ANTHROPIC_AUTH_TOKEN` 填错或误加 Bearer 前缀	核对 Key，注意多数国产端点要求不带 Bearer 前缀（见 2.2 节）

> **来源**：社区经验汇总、Claude Code GitHub Issues（2026年4月）；国产端点报错行为来自各厂商官方接入文档（2026年8月核实）

2.2 国产模型接入：cc-switch 与官方兼容端点（2026年主流方案）

2026 年国内使用 Claude Code 最主流的方式，已不再是折腾官方账号，而是**把 Claude Code 接到国产大模型的官方 Anthropic 兼容端点上**：Claude Code 本身在国内可正常安装运行，只需把 `ANTHROPIC_BASE_URL` 指向智谱、DeepSeek、Kimi 等厂商提供的兼容端点，即可零网络障碍地使用 Claude Code 的全部 agent 能力（文件读写、Bash、Hooks、MCP、Skills 等），底层模型换成国产旗舰。

三条路径对比：

路径	网络要求	合规与稳定性	模型能力	适合人群
国产模型官方端点（本节）	国内直连 ✅	官方服务、发票合规、稳定	国产旗舰（GLM-5.3 / DeepSeek V4 / Kimi K3）	绝大多数国内开发者，**推荐**
官方 API + 代理（见 2.3/2.4 节）	需代理 ⚠️	违反区域条款，有封号风险	Claude 5 家族（最强）	能稳定持有海外账号的用户
第三方中转 API（见 2.5 节）	国内直连 ✅	违反 ToS、数据经第三方、跑路风险	Claude 全系列（可能缩水）	短期过渡，谨慎使用

2.2.1 cc-switch：一键切换多家供应商的图形化工具

cc-switch 是目前国内最流行的 Claude Code 供应商切换工具（GitHub：`farion1231/cc-switch`，Tauri + Rust 桌面应用，支持 Windows/macOS/Linux，MIT 开源，2026 年中 Star 数已超 10 万）。它的作用是替你管理和切换各家供应商的端点配置，免去手动改环境变量的麻烦。

核心特性（v3.20+，2026年8月）：

- **50+ 内置供应商预设**：智谱 GLM、DeepSeek、Kimi、MiniMax、硅基流动、火山引擎、百度千帆、OpenRouter 等一键填入；
- **多工具管理**：除 Claude Code 外，还可同时管理 Codex、Gemini CLI、OpenCode 等多个 AI CLI 的配置；
- **图形化操作**：填写 Key 后一键切换，本质是把 `ANTHROPIC_BASE_URL`、`ANTHROPIC_AUTH_TOKEN` 等写入 Claude Code 的配置，对 OpenAI 协议的供应商可开本地代理做协议转换；
- **安全提示**：2026 年披露的 CVE-2026-6143 影响 3.12.3 及之前版本（本地代理跨域漏洞），请务必升级到最新版。

同类工具还有 **claude-code-router**（GitHub：`musistudio/claude-code-router`）：请求拦截 + 协议转换 + 按任务类型路由（如简单任务走便宜模型），适合需要精细化路由的进阶用户。

2.2.2 手动配置：不改任何工具，两个环境变量搞定

不使用 cc-switch 时，直接编辑 `~/.claude/settings.json` 的 `env` 段（或设置同名系统环境变量）即可：

```
{
  "env": {
    "ANTHROPIC_BASE_URL": "https://open.bigmodel.cn/api/anthropic",
    "ANTHROPIC_AUTH_TOKEN": "你的智谱API-Key"
  }
}
```

注意：多数国产端点要求 `ANTHROPIC_AUTH_TOKEN` 直接填 Key，**不要加** `Bearer` **前缀**。配置后建议同时设置 `CLAUDE_CODE_DISABLE_NONESSENTIAL_TRAFFIC=1`，避免遥测等额外流量访问官方域名报错。

2.2.3 智谱 GLM（GLM Coding Plan）

项目	内容（2026年8月核实）
兼容端点	`https://open.bigmodel.cn/api/anthropic`
当前模型	GLM-5.3（2026年8月14日发布，编程能力官方口径较 5.2 提升约 50%；5.2 起支持 1M 无损上下文）
套餐价格	Lite / Pro / Max 三档，约 ¥49–118/月起，周配额制
档位映射	可用 `ANTHROPIC_DEFAULT_SONNET_MODEL` 等环境变量把 CC 内的模型档位映射到 GLM 模型

智谱是国产厂商中最早官方支持 Claude Code 接入的一家，官方文档有完整的接入教程。GLM-5.3 发布后已向 Coding Plan 全量用户开放。

2.2.4 DeepSeek

项目	内容（2026年8月核实）
兼容端点	`https://api.deepseek.com/anthropic`（官方文档有 Claude Code 集成专页）
当前模型	DeepSeek V4 系列：CC 内 Opus/Sonnet 档自动映射到 v4-pro，Haiku 档映射到 v4-flash
价格	约输入 3 元 / 输出 6 元每百万 token（以官网价格页为准）
已知坑	部分斜杠命令路由错误、Sub-agent 功能有兼容缺陷，社区有修复代理项目（dsv4-cc-proxy）

2.2.5 Kimi（月之暗面）

项目	内容（2026年8月核实）
兼容端点	`https://api.moonshot.cn/anthropic`（国际版 `api.moonshot.ai/anthropic`），官方有接入指南
当前模型	Kimi K2.5（256K 上下文）/ K3（2026年7月发布，1M 上下文）
套餐价格	统一会员四档 ¥49 / ¥99 / ¥199 / ¥699 每月（年付 8 折），Token 池 + 5 小时滚动配额
已知坑	temperature 会被重缩放（×0.6）、max_tokens 上限需注意，按官方文档调参

2.2.6 接入国产端点后的注意事项

1. `/cost` **统计失效**：接入第三方端点后，CC 内置的费用统计不再准确，用量与账单以供应商平台控制台为准；
2. **能力差异客观存在**：国产旗舰的日常编码体验已接近 Claude，但在复杂 Agent 编排、跨文件大重构、超长工具链任务上仍有可感知差距；各家的「对齐 Opus」宣传多为官方口径，目前缺乏同一测试集上的权威第三方横评，建议以自己的实际项目体感为准；
3. **配额机制不同**：Coding 套餐普遍采用「时长窗口滚动配额」而非纯 token 计费，重度使用先看清各家的限速规则；
4. **模型迭代快**：国产模型 2-3 个月一代（GLM 半年内从 5.0 迭代到 5.3），本手册信息以 2026年8月为准，接入前请再核对各家官方文档。

> **来源**：farion1231/cc-switch GitHub 仓库及官方更新日志；智谱、DeepSeek、月之暗面官方开发者文档（2026年8月核实）

2.3 官方 API 代理服务配置

API 代理指在持有 Anthropic 官方 API Key 的前提下，通过代理网络将请求转发到 Anthropic 服务器。

> ⚠️ **2026年风险提示**：Anthropic 官方服务不向中国大陆开放，近年有多次大规模封号潮的公开报道。公开报道归纳的封号主因包括：支付渠道异常（虚拟卡/代充）、IP 归属地与账号区域不符、共享账号、异常用量模式。使用官方 API 意味着接受账号随时可能被封的风险，重要项目请勿依赖单一路径。

三步完整流程：

第一步：获取 API Key

注册 Anthropic 账号并在 Console（https://console.anthropic.com）创建 API Key，按量计费（具体定价见第五章）。

第二步：配置环境变量

macOS/Linux（bash）：

```
# 编辑 ~/.bash_profile 或 ~/.bashrc
echo 'export ANTHROPIC_API_KEY="sk-ant-xxx-你的-key"' >> ~/.bashrc
source ~/.bashrc
```

macOS/Linux（zsh）：

```
# 编辑 ~/.zshrc
echo 'export ANTHROPIC_API_KEY="sk-ant-xxx-你的-key"' >> ~/.zshrc
source ~/.zshrc
```

Windows 系统环境变量：

```
# PowerShell（永久生效）
[Environment]::SetEnvironmentVariable("ANTHROPIC_API_KEY", "sk-ant-xxx-你的-key", "User")

# 验证
$env:ANTHROPIC_API_KEY
```

第三步：验证配置

```
# 运行诊断命令
claude doctor

# 预期输出应显示 API 连接正常
```

配置文件方式（适用于高级配置）：

```
{
  "apiKey": "sk-ant-xxx-你的-key",
  "apiUrl": "https://api.anthropic.com",
  "defaultModel": "claude-sonnet-5"
}
```

Windows 配置文件路径：`%APPDATA%\Claude\settings.json` **macOS/Linux 配置文件路径**：`~/.claude/settings.json`

> **国内可用性**：⚠️ 需代理（API 请求直连不稳定，建议配合 2.4 节 VPN 配置使用）

2.4 VPN 代理配置

代理模式要求

Claude Code 要求 **全局代理模式**，仅浏览器代理无法覆盖终端流量。

支持节点地区

地区	可用性	说明
日本	✅ 推荐	延迟低，稳定性高
新加坡	✅ 推荐	延迟低， Anthropic 官方支持
美国西海岸	✅ 可用	延迟较高但稳定
香港	⚠️ 不稳定	部分 IP 段被风控
台湾	✅ 可用	延迟低

Claude Code 代理环境变量配置

```
{
  "env": {
    "HTTP_PROXY": "http://127.0.0.1:7890",
    "HTTPS_PROXY": "http://127.0.0.1:7890",
    "NO_PROXY": "localhost,127.0.0.1,.local,.internal"
  }
}
```

macOS/Linux 终端临时设置：

```
export HTTP_PROXY="http://127.0.0.1:7890"
export HTTPS_PROXY="http://127.0.0.1:7890"
export NO_PROXY="localhost,127.0.0.1"
claude
```

⚠️ **特别标注**：Claude Code 仅支持 HTTP/HTTPS 代理，**不支持 SOCKS5**。如果你的代理工具只提供 SOCKS 端口，请使用 Privoxy 等工具将 SOCKS 转换为 HTTP 代理。

> **国内可用性**：⚠️ 需代理（VPN 为必要前提，需配合全局模式使用）

2.5 中转 API 服务：风险提示与选择标准

第三方中转 API（将 Anthropic API 请求经第三方服务器转发）在国内直连、无需海外支付，曾是很多开发者的选择。但 2026 年的现状是：**随着智谱、DeepSeek、Kimi 等国产厂商提供官方兼容端点（见 2.2 节），中转方案的必要性已大幅下降**，而其固有风险并未减少。

中转方案的四大固有风险（均有公开报道，可核实）：

1. **违反服务条款**：Anthropic 官方服务不向中国大陆开放，通过中转而用属于灰色地带；2025–2026 年有多次大规模封号潮的公开报道，账号与余额可能随时清零；
2. **数据安全**：你的全部代码、prompt、API Key 都经过第三方服务器，存在泄露与留存风险，企业使用还涉及数据出境合规问题；
3. **服务质量无保障**：行业存在「降智」（实际路由到便宜模型）、套壳、限速、跑路等现象，有媒体称中转已成为中大型项目的「成本黑洞」；
4. **稳定性**：中转商本身也依赖上游账号池，封号潮期间会集体波动。

如果仍需使用，按以下标准筛选（本手册不推荐具体服务商，请自行按标准核实）：

筛选标准	核查方法
运营时长 2 年以上	查域名注册时间、社区历史讨论
价格透明且接近官方价	显著低于官方价的必然「降智」或不可持续
支持 Prompt Caching 透传	实测缓存命中后的计费是否下降
有明确的隐私政策	是否承诺不留存请求内容
社区口碑可查	在 V2EX、即刻等社区搜索近期评价（注意水军）

月度成本参考（按官方定价折算，供预算参考）：

使用强度	月均 Token 消耗	官方 API 成本（人民币）
轻度（每周 3-4 次）	~1M input / 200K output	20-40 元
中度（每日 2-3 小时）	~6M input / 1M output	150-250 元
重度（主要开发工具）	~40M input / 8M output	800-1200 元

> **来源**：阿里云/腾讯云开发者社区合规分析、媒体评论（2025-2026）。相比中转，2.2 节的国产官方端点在网络、合规、成本三个维度都是更优解。

2.6 国内开发者真实使用体验

社区典型评价

评价一（知乎用户 @技术胖，2026年3月）：

> "从 Cursor 切换到 Claude Code 三个月了，最大的感受是'快'。不是响应速度快，是迭代速度快。Claude Code 的 /rewind 和子 Agent 功能让我敢于让 AI 做大改动，因为随时可以回退。CLAUDE.md 写得好不好直接决定输出质量，我花了一周时间打磨项目的 CLAUDE.md，后续开发效率提升了至少 3 倍。"

评价二（掘金用户 @前端阿松，2026年4月）：

> "国内用 Claude Code 最大的坑是网络。一开始用 VPN 总断，后来换了中转 API 配合本地代理，基本稳定。Claude Code 的 Hooks 系统太强大了，我配置了保存自动格式化和提交前跑测试，现在基本不用手动敲命令。"

评价三（V2EX 用户 @devguru，2026年2月）：

> "Claude Code + MCP 是 2026 年最强的开发组合。我接入了 GitHub MCP、PostgreSQL MCP 和 Brave Search MCP，现在查文档、读数据库、创建 PR 都可以一句话完成。Prompt Caching 是隐藏福利，CLAUDE.md 被缓存后每次请求成本降低 90%。"

核心使用心得

心得一：小步迭代 > 大段生成

不要一次性让 Claude Code 生成几百行代码。将任务拆分为"添加接口定义 -> 实现函数 -> 添加测试"三步，每步验证后再继续。这样可以及时纠正方向，减少 /rewind 的使用频率。

心得二：上下文管理是第一生产力

一位开发者统计了不同 CLAUDE.md 配置下的首次输出可用率：

CLAUDE.md 配置	首次输出可用率	平均修改轮次
无 CLAUDE.md	35%	4.2 轮
基础配置（<50 行）	55%	2.8 轮
完整配置（200-500 行）	78%	1.5 轮
完整配置 + Hooks	85%	1.1 轮

完整配置 CLAUDE.md 可将首次输出可用率从 35% 提升至 78%，配合 Hooks 可达 85%。

心得三：2026年的新共识——先想清楚要不要用官方账号

2026 年下半年国内社区的主流共识已变成：日常开发优先接国产模型官方端点（GLM/DeepSeek/Kimi，见 2.2 节），网络零障碍、可开发票、无封号焦虑；只有在确实需要 Claude 模型最强能力（复杂重构、长链路 Agent 任务）且能接受账号风险时，才考虑官方 API + 代理路径。很多团队的实际做法是两者并存——cc-switch 里配好多家供应商，按任务一键切换。

> **来源**：知乎、掘金、V2EX 社区帖子（2026年2-4月）；心得三为 2026年8月社区讨论综述

CHAPTER THREE

第三章 进阶用法与技巧

3.1 自定义 Slash 命令

3.2 Git 工作流深度集成

3.3 上下文管理技巧

3.4 MCP 工具接入（重点章节）

3.5 Vibe Coding 工作流

3.6 Headless 模式与 CI/CD 集成

3.7 Claude Code Hooks 系统

3.8 Claude Code + OpenClaw 联动方案

3.1 自定义 Slash 命令

创建方法

在 `.claude/commands/` 目录下创建 Markdown 文件，文件名即为命令名。

```
mkdir -p .claude/commands
```

实战示例：自动提交+测试+推送+创建 PR

创建 `.claude/commands/ship.md`：

```
# 一键发布

当用户输入 /ship 时，执行以下流程：

1. 运行 `npm run lint` 检查代码格式
2. 运行 `npm run test` 确保所有测试通过
3. 运行 `npm run build` 确保构建成功
4. 生成提交信息（使用中文，格式：[类型] 描述）
5. 执行 `git add . && git commit`
6. 执行 `git push origin $(git branch --show-current)`
7. 创建 Pull Request，填写标题和描述

如果任何一步失败，停止后续操作并报告错误。
```

命令调用方式

```
claude
# 进入交互模式
/ship
```

> **来源**：Claude Code 官方文档自定义命令章节（2026年4月）

3.2 Git 工作流深度集成

三种常用 Git 命令示例

```
# 示例一：提交代码
claude
> 提交所有修改，使用中文提交信息，格式为 [类型] 描述

# 示例二：创建分支
claude
> 从 main 创建 feature/user-auth 分支，并切换到该分支

# 示例三：代码审查
claude
> 审查 feature/user-auth 分支与 main 的差异，关注潜在 Bug 和安全问题
```

最佳实践

实践	说明
小步提交	每次修改控制在 50-200 行，便于回滚和审查
有意义的提交信息	使用中文描述，格式：[类型] 具体描述，如 "[feat] 添加用户登录接口"
分支策略	main 为生产分支，开发使用 feature/xxx，修复使用 fix/xxx

> **来源**：Claude Code 官方 Git 集成文档（2026年4月）

3.3 上下文管理技巧

激进管理上下文：4 种场景及操作建议

场景	症状	操作建议
对话超过 50 轮	响应明显变慢，AI 开始遗忘早期指令	执行 `/compact` 压缩历史
切换任务主题	当前对话与之前无关，AI 引用错误上下文	执行 `/clear` 清空后重新开始
AI 输出偏离方向	连续 2-3 轮输出不符合预期	执行 `/rewind` 回退到错误之前
处理大型代码库	Token 消耗过快，超出预算	使用 `@文件路径` 精确引用而非加载整个目录

Subagents 做调研

Subagents（子 Agent）可以在独立上下文中执行任务，结果回传到主会话。

使用方法：

```
# 让子 Agent 调研某个技术方案
claude
> 启动一个子 Agent 调研 Next.js 15 的 Server Actions 最佳实践，回来后给我一份总结
```

原理：子 Agent 拥有独立的上下文窗口，不会污染主会话的上下文。适合长耗时的调研、文档阅读等任务。

快速引用语法

语法	作用	示例
`@文件路径`	引用单个文件	`@src/utils/auth.ts`
`@目录路径/`	引用整个目录	`@src/components/`
`@关键词`	模糊匹配文件名	`@auth` 匹配所有含 auth 的文件
`@文件名:行号`	引用文件特定行	`@src/app.ts:42`

多 Agent 协作的工作流案例

案例：大型重构任务

1. 主 Agent 制定重构计划（修改哪些文件、依赖关系）

2. 子 Agent A 负责修改接口层代码（独立上下文）
3. 子 Agent B 负责修改调用方代码（独立上下文）
4. 两个子 Agent 完成后，主 Agent 整合结果并运行测试

> **来源**：Claude Code 官方文档上下文管理章节（2026年4月）

3.4 MCP 工具接入（重点章节）

MCP 协议简述

MCP（Model Context Protocol）是 Anthropic 推出的开放标准，用于让 AI 助手调用外部工具（数据库、API、浏览器等）。MCP Server 以独立进程运行，通过 stdio/HTTP 与 Claude Code 通信。

> **2026年重要背景**：2025年12月，Anthropic 将 MCP 捐赠给 Linux 基金会新成立的 Agentic AI Foundation（AAIF），成为与 OpenAI AGENTS.md、Block goose 并列的创始项目，治理进一步中立化。2026年7月28日，MCP 发布迄今最大规范改版：核心协议改为无状态设计（移除初始化握手与会话 ID），新增 `server/discover` 能力发现与 HTTP 头路由，授权机制加固。新版 SDK 已同步支持，绝大多数客户端与 Server 的接入代码无需修改。
>
> **生态规模**：截至 2026 年上半年，公开 MCP Server 已超 1.7 万个，官方发现入口为 MCP Registry（registry.modelcontextprotocol.io）；早期参考服务器仓库 modelcontextprotocol/servers 已归档为 servers-archived。

新版推荐：.mcp.json 独立配置文件

在项目根目录创建 `.mcp.json`：

```
{
  "mcpServers": {
    "github": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-github"],
      "env": {
        "GITHUB_PERSONAL_ACCESS_TOKEN": "ghp_你的token"
      }
    },
    "filesystem": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-filesystem", "/Users/你的名字/workspace"]
    },
    "brave-search": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-brave-search"],
      "env": {
        "BRAVE_API_KEY": "你的 brave api key"
      }
    }
  }
}
```

旧版兼容：settings.json 中配置

```
{
  "mcpServers": {
    "github": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-github"],
      "env": {
        "GITHUB_PERSONAL_ACCESS_TOKEN": "ghp_你的token"
      }
    }
  }
}
```

MCP 管理命令速查

命令	作用
`claude mcp add <name> -- <command>`	添加 MCP Server
`claude mcp list`	列出所有已配置的 Server
`claude mcp remove <name>`	移除指定 Server
`claude mcp serve`	启动 MCP Server 调试模式
`claude mcp test <name>`	测试指定 Server 连接
`/mcp`	交互式管理 MCP 连接

热门 MCP Server 推荐表

MCP Server	功能	安装命令	国内可用性	需要 API Key
GitHub	代码仓库操作、Issue/PR 管理	`npx -y @modelcontextprotocol/server-github`	✅ 直连	✅ GITHUB_TOKEN
Filesystem	本地文件读写	`npx -y @modelcontextprotocol/server-filesystem <路径>`	✅ 本地	❌
Brave Search	网络搜索	`npx -y @modelcontextprotocol/server-brave-search`	✅ 直连	✅ BRAVE_API_KEY
Tavily	深度搜索（支持 RAG）	`npx -y tavily-mcp-server`	✅ 直连	✅ TAVILY_API_KEY
PostgreSQL	数据库查询	`npx -y @modelcontextprotocol/server-postgres`	✅ 本地	❌（本地库）
Puppeteer	浏览器自动化	`npx -y @modelcontextprotocol/server-puppeteer`	✅ 本地	❌
Memory	跨会话记忆存储	`npx -y @modelcontextprotocol/server-memory`	✅ 本地	❌
Fetch	HTTP 请求	`npx -y @modelcontextprotocol/server-fetch`	✅ 直连	❌
Sequential Thinking	复杂问题逐步推理	`npx -y @modelcontextprotocol/server-sequential-thinking`	✅ 本地	❌
Slack	Slack 消息发送/读取	`npx -y @modelcontextprotocol/server-slack`	⚠️ 需代理	✅ SLACK_TOKEN
Sentry	错误监控集成	`npx -y @modelcontextprotocol/server-sentry`	✅ 直连	✅ SENTRY_TOKEN
Docker	容器管理	`npx -y @modelcontextprotocol/server-docker`	✅ 本地	❌
高德地图	地理编码/路径规划/POI/天气（官方）	`npx -y @amap/amap-maps-mcp-server` 或 SSE 端点 `mcp.amap.com`	✅ 直连	✅ 高德开放平台 Key

MCP Server	功能	安装命令	国内可用性	需要 API Key
支付宝	支付链路（国内首个支付领域 MCP）	阿里云百炼平台预置	✅ 直连	✅ 按平台要求
腾讯位置服务	地图/位置能力	腾讯云开发平台接入	✅ 直连	✅ 按平台要求

> **来源**：MCP 官方文档（https://modelcontextprotocol.io）、MCP Registry、各厂商官方文档（2026年8月核实）。国内厂商（高德/支付宝/腾讯）自 2025 年起陆续推出官方 MCP Server，是国内生态的重要补充。

3.5 Vibe Coding 工作流

什么是 Vibe Coding

Vibe Coding 是一种"氛围驱动"的编程方式：开发者描述意图，AI 自主实现细节，人类只在关键节点做决策。Claude Code 的 Auto Mode 和 Hooks 系统是实现 Vibe Coding 的基础设施。

第一阶段：构建项目大脑（Memory Bank）

在项目根目录创建 `memory-bank/` 目录：

```
memory-bank/
  product.md      # 产品需求文档
  architecture.md # 架构决策记录
  tech-spec.md    # 技术规范
  api-spec.md     # API 接口定义
  current-task.md # 当前任务上下文
```

在 CLAUDE.md 中添加：

```
## Memory Bank

项目知识库位于 memory-bank/ 目录。每次任务开始前：
1. 阅读 product.md 理解产品目标
2. 阅读 architecture.md 理解架构约束
3. 阅读 current-task.md 了解当前进度
4. 任务完成后更新 current-task.md
```

第二阶段：进入 Vibe 状态（循环开发流程）

```
描述需求 → AI 规划 → AI 执行 → 人类审阅 → AI 修复 → 提交
    ↑                                                    |
    └──────────────── 下一轮需求 ──────────────────────┘
```

关键操作：

1. 用自然语言描述需求（越具体越好）
2. Claude 给出执行计划，确认后执行
3. 自动运行测试和检查（通过 Hooks）
4. 人工审查修改，给出反馈
5. Claude 根据反馈修复

6. 自动提交代码

第三阶段：进阶与排错

技巧	使用场景
`/rewind`	AI 修改方向错误时回退
BrowserTools MCP	AI 需要查看网页内容时
递进式指令	复杂任务拆分为"先做 A，再做 B，最后做 C"
子 Agent 并行	多文件独立修改时同时启动多个子 Agent

关键原则

1. **信任但验证**：让 AI 自主执行，但通过 Hooks 和测试验证结果
2. **快速反馈循环**：每次迭代控制在 5-10 分钟内
3. **保持上下文精简**：及时 /compact，避免上下文膨胀
4. **文档即契约**：Memory Bank 中的文档是 AI 的行为约束

> **来源**：Claude Code 社区 Vibe Coding 实践总结（2026年4月）

3.6 Headless 模式与 CI/CD 集成

基本用法

```
# 方式一：管道输入
echo "审查这段代码" | claude --headless

# 方式二：文件输入
claude --headless --input task.md

# 方式三：直接指定
claude --headless -- "运行测试套件并报告结果"
```

GitHub Actions CI/CD 集成示例（PR 自动审查）

创建 `.github/workflows/claude-review.yml`：

```
name: Claude Code PR Review

on:
  pull_request:
    types: [opened, synchronize]

jobs:
  review:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Install Claude Code
        run: npm install -g @anthropic/claude-code

      - name: Run Claude Code Review
        env:
          ANTHROPIC_API_KEY: ${{ secrets.ANTHROPIC_API_KEY }}
        run: |
          # 获取 PR 的 diff
          git diff origin/${{ github.base_ref }}...HEAD > pr-diff.txt
```

```
      # 让 Claude 审查
      echo "请审查以下代码变更，关注：
      1. 潜在的 Bug 和逻辑错误
      2. 安全漏洞（SQL 注入、XSS 等）
      3. 性能问题
      4. 代码风格一致性
      5. 测试覆盖率

      这是 diff 文件：" | cat - pr-diff.txt | claude --headless -- > review-result.txt

      # 输出审查结果
      cat review-result.txt

  - name: Post Review Comment
    uses: actions/github-script@v7
    with:
      script: |
        const fs = require('fs');
        const review = fs.readFileSync('review-result.txt', 'utf8');
        github.rest.issues.createComment({
          issue_number: context.issue.number,
          owner: context.repo.owner,
          repo: context.repo.repo,
          body: `## Claude Code 自动审查结果\n\n${review}`
        });
```

> **来源**：Claude Code 官方 Headless 模式文档、GitHub Actions 文档（2026年4月）

3.7 Claude Code Hooks 系统

四种 Hook 类型说明

Hook 类型	触发时机	能否阻断操作	典型用途
PreToolUse	工具调用执行前	✅ 可以	安全检查、权限控制、危险命令拦截
PostToolUse	工具调用成功后	❌ 不可以	自动格式化、运行测试、日志记录
Notification	Claude 发送通知时	❌ 不可以	桌面通知、Slack 消息推送
Stop	Claude 完成响应时	❌ 不可以	任务完成提醒、后续操作触发

> **2026年8月补充**：上表是入门最常用的四类。2026 年以来 Hooks 系统已扩展至 31 个生命周期事件（新增 UserPromptSubmit、PermissionRequest、SessionStart/End、PreCompact/PostCompact、SubagentStart/Stop、PostToolUseFailure、WorktreeCreate/Remove 等），handler 类型也从 command 一种扩展为 command / prompt / agent / http 四种。完整清单见本系列姊妹篇《Claude Code 兵器谱》第六章。

完整配置示例（settings.json）

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "if echo '{\"tool_input\":{\"command\":\"$*\"}}' | grep -qE 'rm -rf|mkfs|dd if=.*of=|>.*\\.env'; then echo '危险命令被拦截'; ex
            "timeout": 5
          }
        ]
      },
      {
        "matcher": "Edit|Write",
        "hooks": [
          {
            "type": "command",
            "command": "if echo '{\"tool_input\":{\"file_path\":\"$*\"}}' | grep -q '.env'; then echo 'SECURITY BLOCK: .env 文件禁止修改'; exit 2; fi'
            "timeout": 5
          }
        ]
      }
    ],
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          {
            "type": "command",
            "command": "FILE=$(echo '{\"tool_input\":{\"file_path\":\"$*\"}}' | grep -oP 'file_path\":\"\\K[^\"]+'); case \"$FILE\" in *.ts|*.tsx) np
            "timeout": 15
          }
        ]
      }
    ],
    "Notification": [
      {
        "matcher": "*",
        "hooks": [
          {
            "type": "command",
            "command": "osascript -e 'display notification \"Claude Code 需要你的关注\" with title \"Claude Code\"' 2>/dev/null || notify-send 'Claude
            "timeout": 5
          }
        ]
      }
    ],
    "Stop": [
```

```
      {
        "matcher": "*",
        "hooks": [
          {
            "type": "command",
            "command": "osascript -e 'display notification \"Claude 已完成当前任务\" with title \"Claude Code\" sound name \"Glass\"' 2>/dev/null || n
            "timeout": 5
          }
        ]
      }
    ]
  }
}
```

实战案例

案例一：自动格式化代码

每次 Claude 编辑或写入文件后，自动运行 Prettier/Ruff 格式化：

```
{
  "matcher": "Edit|Write",
  "hooks": [{
    "type": "command",
    "command": "jq -r '.tool_input.file_path' | xargs npx prettier --write 2>/dev/null",
    "timeout": 15
  }]
}
```

案例二：提交前 lint 检查

在 Claude 执行 `git commit` 前，自动运行 lint 检查：

```
{
  "matcher": "Bash",
  "hooks": [{
    "type": "command",
    "command": "if echo '$*' | grep -q 'git commit'; then npm run lint; fi",
    "timeout": 60
  }]
}
```

案例三：任务完成桌面通知

Claude 完成任务时推送桌面通知（macOS）：

```
{
  "matcher": "*",
  "hooks": [{
    "type": "command",
    "command": "osascript -e 'display notification \"任务完成\" with title \"Claude Code\"'",
    "timeout": 5
  }]
}
```

> **来源**：Claude Code 官方 Hooks 文档（https://code.claude.com/docs/hooks，2026年4月）

3.8 Claude Code + OpenClaw 联动方案

OpenClaw 简介

OpenClaw 是一个开源 AI Agent 运行框架，支持通过统一 Gateway 接口接入企业微信、飞书、钉钉、QQ 等主流通讯平台。2026 年初因开源社区热度迅速崛起，成为企业级 AI 集成的重要选择。

核心特性：

- 多渠道统一接入：企业微信、飞书、钉钉、QQ
- 实时消息流：WebSocket + Stream 模式
- 工具生态：文档处理、日程管理、文件操作等 Skill 扩展
- 安全认证：完善的 API 认证和权限控制

联动价值

工具	职责	联动方式
Claude Code	代码开发、技术实现	核心开发环境
OpenClaw	任务通知、进度汇报、IM 交互	通过 Hooks 触发通知

典型联动场景：

1. Claude Code 完成代码重构 → Stop Hook 触发 OpenClaw 发送飞书消息通知团队
2. 收到企微消息要求修 Bug → OpenClaw 解析任务 → 调用 Claude Code 执行修复
3. CI/CD 构建失败 → OpenClaw 接收告警 → 转发给 Claude Code 分析日志

配置示例

Stop Hook 发送飞书通知：

```
{
  "hooks": {
    "Stop": [{
      "matcher": "*",
      "hooks": [{
        "type": "http",
        "url": "https://your-openclaw-gateway/feishu/webhook",
        "headers": {
          "Authorization": "Bearer your-openclaw-token",
          "Content-Type": "application/json"
        },
```

```
        "body": {
          "msg_type": "text",
          "content": {
            "text": "Claude Code 任务已完成，项目：{{cwd}}"
          }
        }
      }]
    }]
  }
}
```

> **来源**：OpenClaw 为 2026 年崛起的开源 Agent 框架，已被 cc-switch 等主流工具纳入支持列表；具体配置请以其官方 GitHub 仓库文档为准（5月版手册引用的网址未能核实，已移除）。Claude Code Hooks 文档（2026年8月核实，http 类型 handler 为 2026 年新增的四种 handler 之一）。

CHAPTER FOUR

第四章 常见问题与解决方案

4.1 安装与登录问题

4.2 Token 消耗控制

4.3 权限与安全管理

4.4 Prompt Caching 深入

4.1 安装与登录问题

npm 权限错误

症状：`EACCES: permission denied` 或 `ERR! Error: EACCES`

解决方案：

```
# 方案一：修改 npm 全局目录权限
mkdir ~/.npm-global
npm config set prefix '~/.npm-global'
echo 'export PATH=~/.npm-global/bin:$PATH' >> ~/.bashrc
source ~/.bashrc

# 方案二：使用 npx（无需全局安装）
npx @anthropic/claude-code

# 方案三：使用官方 curl 脚本（推荐）
curl -fsSL https://get.claude.com/cli | bash
```

VPN 下 OAuth 403 错误

症状：浏览器显示 "Error 403: access_denied" 或 "This account has been disabled"

原因：OAuth 流程检测到高风险 IP（某些 VPN IP 段被 Anthropic 风控系统标记）

解决方案：

1. 更换 VPN 节点地区（推荐日本、新加坡）
2. 使用 API Key 方式登录（`claude login --api-key`）
3. 在干净的网络环境下完成 OAuth，后续使用 API Key

VS Code 插件无法登录

症状：VS Code 中 Claude 插件登录无反应或报错

解决方案：

```
// VS Code settings.json 中添加代理配置
{
  "http.proxy": "http://127.0.0.1:7890",
  "http.proxySupport": "override",
  "anthropic.apiKey": "sk-ant-xxx-你的-key"
}
```

彻底清理并重装

```
# macOS
rm -rf ~/.claude
rm -rf ~/Library/Application\ Support/Claude
rm -f /usr/local/bin/claude

# Linux
rm -rf ~/.claude
rm -rf ~/.config/Claude
rm -f /usr/local/bin/claude

# Windows
rmdir /s %APPDATA%\Claude
del %LOCALAPPDATA%\Programs\claude\claude.exe

# 重新安装
curl -fsSL https://get.claude.com/cli | bash
```

4.2 Token 消耗控制

十大省钱技巧

技巧	节省比例	难度	说明
写好 CLAUDE.md	30-40%	低	减少 AI 的试错次数
使用 Sonnet 处理简单任务	40-50%	低	Sonnet 比 Opus 便宜 3-5 倍
启用 Prompt Caching	80-90%	低	CLAUDE.md 自动缓存，无需手动配置
及时 /compact	20-30%	低	压缩历史释放 Token
精确引用文件	15-25%	中	用 @文件 而非加载整个目录
使用子 Agent 做调研	10-20%	中	避免污染主上下文
选择 5-min 缓存 TTL	10-15%	低	比 1-hour TTL 更便宜
使用 Batch API	50%	中	非紧急任务使用异步接口
限制 MCP Server 数量	10-20%	低	每个 Server 增加 Token 开销
定期清理会话	15-25%	低	避免累积无用上下文

对话历史管理的成本累积对比

使用天数	不清理 Token 消耗	定期清理 Token 消耗	节省比例
1 天	~50K	~50K	0%
7 天	~500K	~200K	60%
30 天	~3M	~600K	80%

清理时机判断（4 个场景）

场景	操作建议
单一会话超过 30 轮	执行 `/compact`
切换到完全不同的任务	执行 `/clear`
感觉 AI 响应变慢	检查上下文长度，必要时 `/compact`
每日工作结束	退出并重新启动新会话

模型选择建议表

任务类型	推荐模型	原因
简单任务（格式化、查找替换）	Haiku 4.5	最便宜，速度最快
日常开发（编码、调试、Code Review）	Sonnet 5	性价比最优，能力全面
复杂架构设计（重构、系统设计）	Opus 5	能力最强，适合复杂推理
极致性能需求（前沿研究、最难任务）	Fable 5	Mythos 级旗舰，128K 最大输出
交互式高强度开发	Opus 5 + Fast 模式	约 2.5 倍速度，体验流畅

4.3 权限与安全管理

权限白名单/黑名单配置

在 `settings.json` 中配置权限：

```
{
  "permissions": {
    "allow": [
      "Read",
      "Write",
      "Bash:npm run *",
      "Bash:git *"
    ],
    "deny": [
      "Bash:rm -rf *",
      "Bash:sudo *",
      "Bash:curl *|sh",
      "Write:*.env"
    ]
  }
}
```

不同工具的风险等级分类

风险等级	工具/操作	建议配置
🔴 高危	rm -rf、sudo、curl | sh、修改 .env	加入 deny 黑名单，每次确认
🟡 中危	git push、npm publish、数据库写入	加入 ask 列表，执行前询问
🟢 低危	文件读写、git status、npm run test	加入 allow 白名单，自动执行

项目级 vs 用户级权限设置

级别	配置文件	作用范围	优先级
项目级	`<项目>/.claude/settings.json`	仅当前项目	高（覆盖用户级）
用户级	`~/.claude/settings.json`	所有项目	低

安全最佳实践

1. **版本控制权限配置**：将 `.claude/settings.json` 提交到 Git，团队共享统一的安全策略

2. **敏感文件保护**：通过 PreToolUse Hook 拦截对 .env、config.yaml 等文件的修改
3. **生产环境隔离**：生产环境使用只读权限，禁止 AI 直接修改生产配置
4. **审计日志**：通过 PostToolUse Hook 记录所有文件修改和命令执行

4.4 Prompt Caching 深入

工作原理简述

Prompt Caching 通过缓存 Transformer Prefill 阶段计算的 Key/Value 张量，避免重复计算稳定前缀。缓存匹配基于 token 序列的精确哈希（非语义相似度）。

缓存匹配顺序：tools → system → messages（上游变动会导致后续缓存失效）

CLAUDE.md 自动缓存的优势

Claude Code v2.1+ 会自动缓存 CLAUDE.md 内容（属于 system 前缀），这意味着：

- 每次请求的 system 提示部分（含 CLAUDE.md）会被自动缓存
- 缓存写入成本为正常输入的 1.25 倍（5 分钟 TTL）或 2 倍（1 小时 TTL）
- 缓存读取成本仅为正常输入的 0.1 倍（90% 折扣）

实际成本节省案例

场景	无缓存成本	有缓存成本	节省
单次请求（50K 输入含 CLAUDE.md）	$0.15	$0.03	80%
一日开发（100 次请求）	$15.00	$3.50	77%
一月开发（22 工作日）	$330.00	$77.00	77%

提高缓存命中率的技巧

1. **保持前缀稳定**：不要频繁修改 CLAUDE.md 或工具集定义
2. **动态内容后置**：将变化频繁的内容放在 messages 末尾，而非 system 提示中
3. **精简工具集**：只启用必要的 MCP Server，减少 tools 前缀的变动
4. **监控命中率**：使用 `claude doctor` 查看缓存统计

> **来源**：Anthropic Prompt Caching 文档、社区缓存优化实践（2026年4月）

CHAPTER FIVE

第五章 定价与成本优化

5.1 官方订阅方案

5.2 API 按量计费

5.3 套餐选择建议

5.4 订阅 vs API 对比

5.5 使用成本追踪

5.1 官方订阅方案

套餐	月费	额度	适用人群
Free	$0	极有限，不含 Claude Code	轻度体验
Pro	$20/月（年付 $17/月）	5x Free 限额	日常开发、研究
Max 5x	$100/月	5x Pro 限额	重度开发者
Max 20x	$200/月（仅月付、仅个人）	20x Pro 限额	专业开发者
Team Standard	$25/席/月（年付 $20，5 席起）	共享团队限额	小团队协作
Team Premium	$125/席/月（年付 $100）	约 5x Standard 额度，含 Claude Code	重度团队
Enterprise	定制报价（席位费 + API 计量）	定制	大型企业

说明：模型阵容为 Claude 5 家族（Opus 5 / Sonnet 5 / Haiku 4.5，前沿任务可选 Fable 5）。Team Standard 是否包含 Claude Code 在不同时期的官方页面表述有变化，订阅前请以 claude.com/pricing 当时页面为准。

> **来源**：多家 2026年7-8月对照官方定价页的独立整理（数字多源一致；官方页面以 claude.com/pricing 为准）

5.2 API 按量计费

当前模型价格表（按 1M tokens，2026年8月）

模型	输入价格	输出价格	上下文窗口	特点
Claude Fable 5	$10.00	$50.00	1M	前沿旗舰（Mythos 级），最大输出 128K
Claude Opus 5	$5.00	$25.00	1M	旗舰，Claude Code 默认模型
Claude Sonnet 5	$2.00（推广价至 2026-08-31，之后 $3.00）	$10.00（之后 $15.00）	1M	均衡主力，性价比最优
Claude Haiku 4.5	$1.00	$5.00	200K	快速轻量
Opus 5（Fast 模式）	$10.00	$50.00	1M	同模型约 2.5 倍速度（/fast 切换，仅限 Opus 5/4.8）

补充说明：美国本土推理（US-only inference）加收 10%；Claude 3/3.5 系列已于 2026 年上半年退役。

Batch API 折扣

Batch API 输入输出均为**标准价 5 折**，24 小时内异步返回结果，且**可与 Prompt Caching 折扣叠加**。适合非紧急的批量任务（Fast 模式不支持 Batch）。

Prompt Caching 计费规则

操作	费用倍数	说明
缓存写入（5 分钟 TTL）	1.25x 输入价格	首次写入成本
缓存写入（1 小时 TTL）	2.0x 输入价格	更长有效期
缓存读取	0.1x 输入价格	90% 折扣
缓存未命中	1.0x 输入价格	正常计费

2026 年新增**自动缓存模式**（单个顶层 `cache_control` 即可），缓存自 2026 年 2 月起按 workspace 隔离；最小缓存长度因模型而异（512–4096 tokens）。

> **来源**：多家 2026年7-8月对照 Anthropic 官方定价页的独立整理（数字多源一致；以官方页面为准）

5.3 套餐选择建议

Pro 与 Max 的核心区别

维度	Pro	Max 5x	Max 20x
月费	$20	$100	$200
消息限额	5x Free	5x Pro	20x Pro
Opus 访问	❌	✅	✅
优先级	标准	优先	最高
最佳场景	日常开发	专业开发	团队核心

5 种使用场景对应的推荐套餐

使用场景	推荐套餐	预估月成本
周末副业项目	API 按量	$5-15
全职开发（中度）	Pro ($20)	$20
全职开发（重度）	Max 5x ($100)	$100
架构设计为主	Max 5x ($100) + API 补充	$100-150
团队 5 人+	Team Standard	$125+

各方案的性价比量化对比

方案	月成本	可处理 Token 量	每百万 Token 成本
API 轻度	$10	~2M input	~$5
Pro 订阅	$20	~10M input	~$2
Max 5x	$100	~100M input	~$1
Max 20x	$200	~500M input	~$0.4

订阅方案使用量越大，单位 Token 成本越低。

5.4 订阅 vs API 对比

维度	官方订阅	官方 API	国产 Coding 套餐（推荐）	API 中转
月费	$20-200	按量计费	¥49-699/月（智谱/Kimi 等）	按量计费
可用性	⚠️ 需代理	⚠️ 需代理	✅ 国内直连	✅ 国内直连
稳定性	高（官方服务）	高（官方服务）	高（厂商官方）	低（依赖服务商）
数据隐私	高	高	高（官方端点）	低（经第三方）
模型能力	Claude 5 家族	Claude 5 家族	国产旗舰（GLM/DeepSeek/Kimi）	Claude 系列（可能缩水）
合规性	⚠️ 灰色地带	⚠️ 灰色地带	✅ 可开发票	❌ 违反 ToS
适合人群	稳定持有海外账号者	轻度/重度两极	绝大多数国内开发者	短期过渡

> 国产 Coding 套餐的接入方法详见 2.2 节；其额度机制多为「时长窗口滚动配额」，与 Anthropic 的消息额度机制不同，购买前请看清各家限速规则。

日使用次数阈值判断标准

日均使用时长	推荐方案	原因
< 30 分钟	API 按量	成本低，灵活
30 分钟 - 3 小时	Pro 订阅	性价比最优
3-8 小时	Max 5x	不限量，省心
> 8 小时	Max 20x	重度使用必需

5.5 使用成本追踪

如何查看个人 Token 消耗

```
# 在会话内查看 Token 统计与费用
/cost

# 查看配额与速率限制状态（订阅用户）
/usage

# API 用户查看控制台
# 访问 https://console.anthropic.com/usage
```

> 注意：接入国产模型端点后 `/cost` 统计失效，用量以供应商平台控制台为准（见 2.2 节）。

第三方成本追踪工具推荐

工具	功能	价格
Claude Usage Tracker	浏览器插件，实时显示 Token 消耗	免费
LangSmith	完整的 LLM 调用追踪和分析	有免费额度
Helicone	API 调用监控和成本分析	有免费额度

团队使用时的成本分摊思路

1. **国产厂商团队套餐**：智谱/Kimi 等平台提供团队版与子账号，国内合规可开发票，按成员分摊
2. **独立 API Key**：每个开发者独立 Key，月底汇总
3. **Team 订阅**：Anthropic Team Standard/Premium 提供集中计费和管理
4. **成本告警**：设置每日/每周消费上限，防止意外超支

CHAPTER SIX

第六章　工具对比与选型

6.1 Claude Code vs Cursor

6.2 选型建议

6.3 其他替代工具（2026年8月格局）

6.1 Claude Code vs Cursor

对比维度	Claude Code	Cursor
定位	终端原生 Agent	IDE 集成编辑器
工作流	终端交互，适合命令行用户	图形界面，适合 IDE 用户
上下文管理	CLAUDE.md + Hooks，灵活但需配置	内置上下文感知，开箱即用
项目理解	通过 CLAUDE.md 主动声明	自动索引代码库
界面	终端 TUI	VS Code 分叉，图形界面
学习曲线	较陡（需配置）	平缓（开箱即用）
最佳场景	大型项目、自动化工作流、团队规范	快速原型、单文件编辑、学习
价格	$20-200/月 或 API 按量（或国产套餐 ¥49+/月）	Pro $20/月起（Pro+ $60、Ultra $200，信用点制）

Claude Code 的优势

1. **终端原生**：与现有开发工具链（tmux、vim、git）无缝集成
2. **Hooks 系统**：自动化程度更高，可定制工作流
3. **MCP 生态**：工具扩展更开放，社区 Server 丰富
4. **Headless 模式**：支持 CI/CD 自动化
5. **上下文控制**：CLAUDE.md 可精确控制 AI 行为
6. **国产模型接入**：可切换 GLM/DeepSeek/Kimi 官方端点，国内零障碍（见 2.2 节）

Cursor 的优势

1. **图形界面**：可视化代码编辑，对新手友好
2. **代码补全**：实时代码补全（Tab 补全）体验更好
3. **开箱即用**：无需配置即可开始使用
4. **Composer 功能**：多文件编辑体验流畅（2026 年已迭代至自研模型 Composer 2.5）
5. **集成度高**：内置终端、调试器、版本控制

> **来源**：Claude Code 官方文档、Cursor 官方 changelog（Cursor 3，2026年4月）、社区对比评测（2026年8月）

6.2 选型建议

什么时候选 Claude Code（4 个典型场景）

场景	原因
大型代码库维护	CLAUDE.md 帮助 AI 理解复杂架构，Hooks 自动化规范执行
团队规范严格的工程	通过 CLAUDE.md 和 Hooks 强制执行代码规范
CI/CD 自动化	Headless 模式支持 PR 自动审查、自动化测试
终端重度用户	习惯命令行操作的开发者效率更高

什么时候选 Cursor（5 个典型场景）

场景	原因
快速原型开发	图形界面操作直观，启动速度快
学习新代码库	自动索引和跳转，理解代码更快
单文件快速编辑	Tab 补全和实时代码生成体验好
前端/UI 开发	可视化编辑和预览更方便
新手开发者	学习成本低，无需配置

协同使用的最佳实践

很多开发者同时使用两者：

工具	用途
Cursor	日常代码浏览、单文件快速编辑、实时代码补全
Claude Code	大型重构、自动化脚本、Git 操作、CI/CD 集成

6.3 其他替代工具（2026年8月格局）

工具	现状（2026年8月）	适用场景	价格
GitHub Copilot CLI	2026年2月正式 GA，plan/autopilot 模式 + 云端委派	GitHub 生态用户、企业团队	Copilot 订阅内含（AI Credits 计量）
OpenAI Codex CLI	开源（Apache-2.0，Rust 重写），高频迭代，默认 GPT-5.6 系列	OpenAI 生态用户	ChatGPT 套餐内含
Gemini CLI	Google 出品，周更迭代，默认 Gemini 3；个人免费层政策 2026 年中有变动，以官网为准	Google Cloud 用户、预算敏感	免费层 + 付费档
Devin Desktop（原 Windsurf）	Windsurf 被 Cognition 收购后 2026年6月更名，原 Cascade agent 已 EOL	喜欢 IDE 形态 Agent 的用户	Free / Pro $20 / Max $200
Trae	字节跳动出品，2026年2月起分档收费；**中国版（Trae CN）用国产模型完全免费**	国内用户、中文项目	Lite $3 / Pro $10 起；中国版免费
OpenCode	开源（MIT）终端 Agent，2026 年明星项目（10 万+ stars），支持 75+ 模型提供商	开源爱好者、自托管、多模型切换	工具免费，BYOK 按 API 计费
Qwen Code	阿里出品，2026年4月起取消 OAuth 免费额度，需自备 API Key	通义生态用户	按量付费
CodeBuddy	腾讯出品，插件/IDE/CLI 三形态，国内版免费（混元/DeepSeek）	国内用户、腾讯生态	国内版免费
iFlow CLI	⚠️ 已停服（2026年3月停止维护，4月服务关闭，官方引导迁移至 Qoder）	—	—

> **来源**：各工具官方 changelog/文档、TechCrunch、GitHub 官方博客（2026年8月核实）。半年内格局变化剧烈：Windsurf 易主更名、iFlow 停服、Copilot CLI 正式 GA——选型时请以当下官网信息为准。

APPENDIX

附录

A. 常用命令速查

B. 快捷键速查

C. 环境变量速查

D. MCP Server 推荐清单

E. 信息来源说明

F. 免责声明

A. 常用命令速查

命令	作用
`claude`	启动交互式会话
`claude login`	登录 Anthropic 账号
`claude logout`	退出当前账号
`claude doctor`	环境诊断
`claude --version`	显示版本号
`claude mcp list`	列出已配置的 MCP Server
`claude mcp add <name> -- <cmd>`	添加 MCP Server
`claude mcp remove <name>`	移除 MCP Server
`/init`	生成本地 CLAUDE.md
`/clear`	清空当前会话上下文
`/compact`	压缩历史对话
`/rewind`（别名 `/undo`）	回退到上次用户输入前
`/loop`	设置定时重复任务
`/mcp`	交互式 MCP 管理
`/cost`	查看当前会话 Token 消耗与费用
`/usage`	查看配额与速率限制状态
`/fast`	Fast 模式切换（仅限 Opus 5/4.8）
`/model`	切换模型档位
`/review` / `/security-review`	代码审查 / 安全审查

B. 快捷键速查

快捷键	作用
`Esc`	中断当前 AI 输出
`Ctrl+C`	强制退出当前操作
`Tab`	自动补全命令
`↑/↓`	浏览历史输入
`Ctrl+R`	搜索历史输入
`#`	快速添加信息到 CLAUDE.md
`/`	唤起 Slash 命令菜单
`@`	唤起文件引用菜单

C. 环境变量速查

环境变量	说明	示例值
`ANTHROPIC_API_KEY`	API 密钥	`sk-ant-xxx...`
`ANTHROPIC_BASE_URL`	自定义 API 端点（国产模型接入的关键，见 2.2 节）	`https://open.bigmodel.cn/api/anthropic`
`ANTHROPIC_AUTH_TOKEN`	自定义端点的认证令牌（多数国产端点不加 Bearer 前缀）	`你的供应商Key`
`HTTP_PROXY`	HTTP 代理地址	`http://127.0.0.1:7890`
`HTTPS_PROXY`	HTTPS 代理地址	`http://127.0.0.1:7890`
`NO_PROXY`	不走代理的地址	`localhost,127.0.0.1`
`CLAUDE_CODE_DISABLE_NONESSENTIAL_TRAFFIC`	禁用非必要网络请求（接入第三方端点时建议开启）	`1`
`CLAUDE_CODE_DISABLE_TERMINAL_TITLE`	禁用终端标题更新	`1`
`CLAUDE_CODE_ENABLE_AUTO_MODE`	企业/API 侧（Bedrock/Vertex/Foundry）开启 Auto Mode	`1`

D. MCP Server 推荐清单

Server	功能	安装命令	需 API Key
GitHub	仓库/Issue/PR 管理	`npx -y @modelcontextprotocol/server-github`	✅
Filesystem	本地文件操作	`npx -y @modelcontextprotocol/server-filesystem <路径>`	❌
Brave Search	网络搜索	`npx -y @modelcontextprotocol/server-brave-search`	✅
Tavily	深度搜索	`npx -y tavily-mcp-server`	✅
PostgreSQL	数据库查询	`npx -y @modelcontextprotocol/server-postgres`	❌
Puppeteer	浏览器自动化	`npx -y @modelcontextprotocol/server-puppeteer`	❌
Memory	跨会话记忆	`npx -y @modelcontextprotocol/server-memory`	❌
Fetch	HTTP 请求	`npx -y @modelcontextprotocol/server-fetch`	❌
Sequential Thinking	逐步推理	`npx -y @modelcontextprotocol/server-sequential-thinking`	❌
Slack	Slack 消息	`npx -y @modelcontextprotocol/server-slack`	✅
Sentry	错误监控	`npx -y @modelcontextprotocol/server-sentry`	✅
Docker	容器管理	`npx -y @modelcontextprotocol/server-docker`	❌

E. 信息来源说明

来源	URL	可靠性
Claude Code 官方文档	https://code.claude.com/docs	官方确认 ✅
Anthropic 官方定价	https://claude.com/pricing	官方确认 ✅
MCP 协议规范	https://modelcontextprotocol.io	官方确认 ✅
MCP Registry（官方发现入口）	https://registry.modelcontextprotocol.io	官方确认 ✅
cc-switch（供应商切换工具）	https://github.com/farion1231/cc-switch	开源官方仓库 ✅
智谱 GLM 开发者文档	https://open.bigmodel.cn	官方确认 ✅
DeepSeek API 文档	https://api-docs.deepseek.com	官方确认 ✅
Kimi 开放平台文档	https://platform.kimi.com	官方确认 ✅
Claude Code 更新日志	GitHub Releases（anthropics/claude-code）	官方确认 ✅
知乎/掘金/V2EX 社区	多来源	社区经验 ⚠️

F. 免责声明

本手册内容基于 2026 年 8 月的公开信息核实修订（5月版原始素材经全面 review 更新），Anthropic 的产品功能、定价策略和服务条款可能随时调整。使用前请访问官方文档确认最新信息。

涉及第三方工具（cc-switch、国产模型端点等）的信息以各官方仓库/文档为准；API 中转服务为第三方服务，存在合规与数据安全风险（见 2.5 节），使用风险由用户自行承担。

网络代理和 VPN 的使用请遵守当地法律法规。

本手册中的代码示例仅供参考，请在生产环境使用前进行充分测试。

本文档版本：2026年8月版（v3.0，基于 2026年5月版全面核实更新）

反馈与勘误：欢迎通过知识星球「河马的CC商业实战」反馈

会员专属资料 · 持续更新

一起玩转 AI 的地方

「河马的CC商业实战」星球里，每周都有看得懂、用得上的 AI 实战：

新栏目「河马带你逛GitHub」每周一期，实战长文、会员资料持续更新。

微信扫码 · 加入「河马的CC商业实战」

河马的CC商业实战 出品