# OpenClaw
# 新手入门宝典

北京九章云极智能研究院

2026年3月

# 目录

> ✨ *写在前面：*
> OpenClaw 已经被广大网友熟知了，但绝大部分网友还没有安装并上手一试，有些网友已经安装上了，但在使用上还不是很熟练，为了照顾广大网友，本文从 0 开始介绍 OpenClaw 的介绍、安装与使用，主要内容如下：
>
> **OpenClaw 简介 -> 安装环境准备 -> 安装部署 -> 接入飞书与微信 -> 安全增强 -> 大模型配置 ->Skill 的安装、创建与使用**
>
> Are you ready？go！

# 第一部分：关于 OpenClaw

> ***本章目标**：峰哥相信，学完这章，你就能跟朋友就 OpenClaw 侃侃而谈，OpenClaw 是什么、能做什么、不能做什么*

## 1.1 一句话解释：个人 AI 全能小助手

OpenClaw 就是**给你的大模型装上了一双手**，让 AI 从 “只会聊天” 变成 “能动手执行任务” 的全能助手，就像一只听话又能干的 AI 小龙虾，住在你的电脑里帮你干活。

更准确地说：

> *OpenClaw 是一个**AI 智能体平台**（Agent Platform），让你能在自己的电脑上运行 AI 助理，并把它接入到你日常使用的工具里——比如飞书、微信等聊天工具。*

## 1.2 它能做什么

按官网描述，它可以做以下事情：

- 文件管理—读写、编辑、组织文件
- 上网—搜索信息、抓取网页内容、操作浏览器
- 提醒和日程 — 设置定时任务、提醒事项
- 消息-帮你发送 WhatsApp、Telegram、飞书、Discord 等消息图片—分析图片内容
- 语音—文字转语音
- 文档 — 处理 Feishu 文档、云盘文件
- 设备—连接和管理你的其他设备（如果配置了）


## 1.3 它不能干什么

OpenClaw 很强大，但它不是万能的。以下几个误解，越早澄清越好。

OpenClaw 不是万能的，它的“不能”主要来自**物理限制、安全风控、模型依赖、执行边界、设计局限**五大类，以下是清晰分类与具体说明：

### 1.3.1 物理世界：完全无法突破

- **无法操作硬件实体**：不能插网线、换硬盘、接外设、按电源、插 U 盾、扫人脸。
- **无法突破物理隔离**：不能控制未联网、无网络通道的设备。
- **无法感知物理环境**：没有摄像头、麦克风、传感器，看不见、听不见、摸不到现实世界。

### 1.3.2 安全与风控：绕不过的“高墙”

- **无法绕过强人机验证**：银行 U 盾、人脸/指纹、短信二次验证、网站顶级反爬（如极验、429 限流）、支付密码、企业单点登录（SSO）。
- **无法突破权限与沙箱**：不能越权访问未授权文件、系统目录、其他用户数据；沙箱内无法逃逸到主机核心区域。
- **无法篡改加密/签名数据**：不能破解密码、篡改已签名文档、绕过 HTTPS 加密。

### 1.3.3 模型与智能：能力天花板

- **完全依赖底层大模型**：自身无推理能力，模型弱则执行差、易幻觉、不执行；模型强则成本极高。
- **复杂推理与深度洞察不足**：做不了真正的战略分析、投资决策、法律判断、深度科研；只能做表面信息整理。
- **上下文易失效/漂移**：长流程、多轮任务易“忘事”、逻辑混乱、步骤遗漏；压缩后关键约束可能丢失。

- **创造性与审美平庸**：做不出有独特风格的设计、艺术创作、深度文案；缺乏主观审美与创新。

### 1.3.4 执行与操作：明确的“不能做”

- **无法处理无 API/无入口的软件**：不能控制无接口、无命令行、无浏览器入口的封闭软件（如部分专用工业软件、老版桌面应用）。
- **无法处理动态/反自动化页面**：对频繁变动、强反爬、验证码密集的网站，极易失败或被封禁。
- **无法原生支持精细权限控制**：默认是“全权限或无权限”，没有操作级、金额级、联系人级的细粒度限制；不可逆操作（删邮件、发消息、转账）无原生二次确认。
- **无法保证 100% 可靠执行**：多步骤任务易卡死、死循环、工具调用错误；人工干预率高，不能完全“无人值守”。

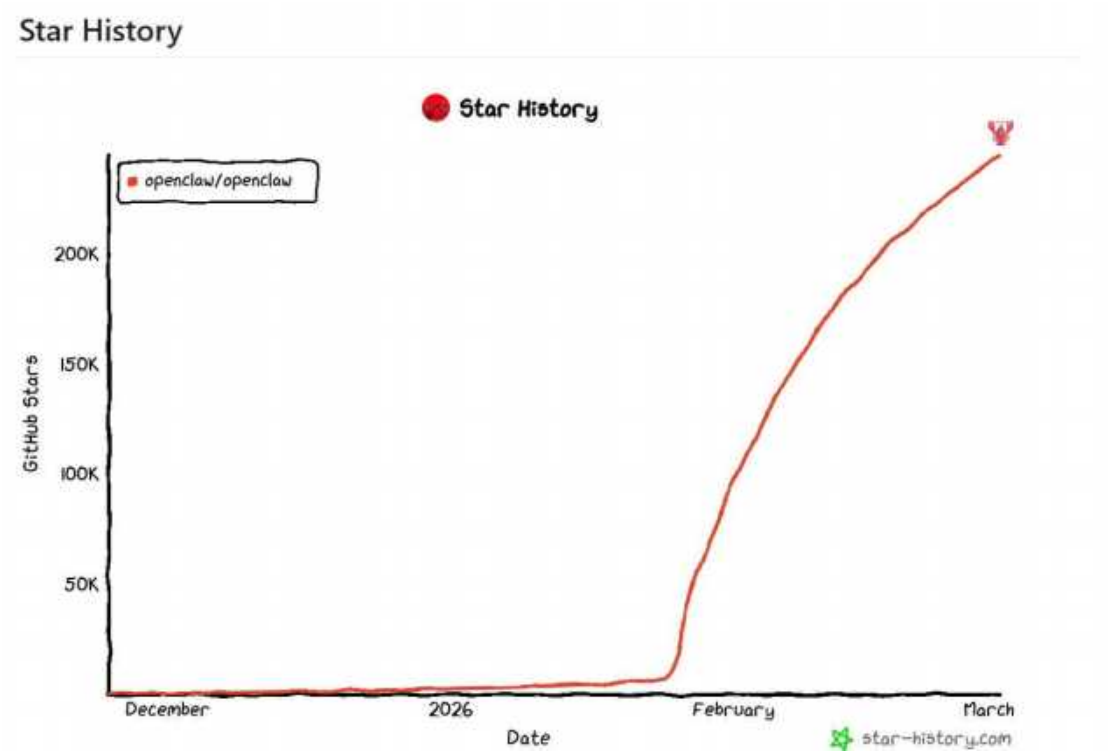
### 1.3.5 设计与架构：先天局限

- **无头无 UI，依赖外部渠道**：没有独立前端，必须通过 Telegram、飞书等聊天工具交互；不能像普通 App 直接操作。
- **跨平台兼容性问题**：Windows/macOS/Linux 命令、路径、权限差异大，技能易在不同系统失效。
- **无原生“人在回路”审核**：自主执行模式下，无内置人工审核流程；安全约束依赖 Prompt，易被绕过或遗忘。
- **成本与门槛高**：强模型 API 费用昂贵；部署、配置、写 Skill 对非技术用户不友好。

一句话总结：OpenClaw 是**数字世界的“执行手”**，但它**没有实体、绕不过风控、依赖模型大脑、执行有边界、安全有短板**——适合标准化、可自动化的任务，不适合需要物理操作、强安全验证、深度推理或高可靠无人值守的场景。

## 1.4 为什么 Openclaw 2026 年它突然火了

Openclaw 在 github 上的 Star History 在 2026 年 2 月份飙升：




OpenClaw 爆火，是因为它**用技术解决了 AI“只会聊天不会做事”的世纪难题**，**精准命中了职场与开发的效率刚需**，**靠开源社区实现了病毒式传播**，再加上**谷歌封号事件的破圈催化**，最终在 2026 年开年引爆全球。OpenClaw 在 2026 年开年迅速爆火，是**技术突破、市场刚需、社区爆发、事件催化**四大因素叠加的结果，精准踩中了 AI 从“聊天”到“做事”的时代拐点。

### 1.4.1 核心技术突破：解决 AI“只会说不会做”的痛点

OpenClaw 最核心的价值，是把 AI 从本章内容大纲“对话助手”**变成了**“数字执行者”本章内容大纲，这是它爆火的根本原因。

- **系统级权限突破**：突破传统 AI 的沙盒限制，获得 Shell 级系统访问权，能直接操作文件、运行终端、控制浏览器、接管键鼠，真正“动手干活”。
- **24 小时自主执行**：内置心跳机制，可后台持续运行、断点续跑，支持定时任务（Cron），实现无人值守自动化。
- **本地优先+记忆持久**：数据存在本地，隐私更安全；拥有长期记忆，上下文不丢失，能处理复杂长流程任务。
- **多端协同**：打通桌面（Windows/macOS/Linux）与移动端（Android），实现跨设备任务执行。

### 1.4.2 市场刚需爆发：打工人与开发者的“效率革命”

2026 年，AI 工具泛滥但本章内容大纲“落地难、执行弱”本章内容大纲成为普遍痛点，OpenClaw 精准满足了两类核心人群的刚需。

- **职场人/打工人**：需要一个能自动处理重复繁琐工作的“数字秘书”。
  - 自动整理文件、批量处理数据、监控邮件、自动回复消息、定时打卡。
  - 跨平台比价、自动下单、管理日程、生成周报，实现“摸鱼式高效”。
- **开发者/技术人员**：需要一个能接管运维、自动化开发流程的“数字员工”。
  - 自动监控代码仓库、运行测试、合并 PR、部署服务、处理服务器日志。
  - 快速构建自定义 Skill，将个人工作流自动化，解放双手。

### 1.4.3 社区与传播：开源+免费+病毒式扩散

OpenClaw 的传播速度，得益于**开源模式+社区生态+极简上手**的组合拳。

- **完全开源免费**：MIT 协议，GitHub 上免费获取，无订阅费，仅承担模型 API 成本，降低了使用门槛。
- **GitHub 热度爆炸**：2026 年 1-2 月，星标数从 0 飙升至 **21.5 万+**，Fork 4 万+，贡献者 700+，成为现象级开源项目。
- **Skill 生态爆发**：社区快速产出本章内容大纲 3000+本章内容大纲现成技能插件，覆盖办公、开发、生活等场景，开箱即用。
- **部署极简**：支持一键部署（尤其 Mac），配置简单，非技术用户也能快速上手。

### 1.4.4 事件催化：争议与话题引爆全网

2026 年 2 月的**谷歌大规模封号事件**，成为 OpenClaw 破圈传播的关键催化剂。

- **事件经过**：2 月 24 日，谷歌无预警封禁大量使用 OpenClaw 的 Google 账号（含 Gmail、Workspace），理由是“违反服务条款”。
- **传播效应**：事件引发全球开发者与用户的激烈讨论，OpenClaw 谷歌封号 登上多国科技热搜，让 OpenClaw 从技术圈彻底破圈，成为全民话题。
- **反向背书**：争议反而证明了 OpenClaw 的**强大执行力**——它能真正操作用户的数字资产，触及了平台的核心利益，也让更多人意识到其颠覆性。

### 1.4.5 时代背景：AI Agent 从概念走向落地

2026 年是本章内容大纲 AI Agent（自主智能体）本章内容大纲的元年，行业共识是：AI 的下一阶段是“能自主执行复杂任务的智能体”。

- OpenClaw 是**首个大规模落地、可直接使用**的 Agent 框架，完美契合了行业趋势与资本热点。
- 它证明了“本地+自主+执行”的 AI 模式可行，为整个 AI 行业指明了新方向。

## 1.5 Openclaw 与之前爆火的 Manus 有何异同

OpenClaw 与 Manus 是 2025–2026 年 AI Agent 领域最具代表性的两条路线：**Manus 是云端托管、开箱即用的商业智能体**，**OpenClaw 是本地优先、高度可定制的开源执行框架**。两者核心目标都是让 AI“真正做事”，但在**架构、控制权、隐私、成本、上手门槛**上差异巨大。

### 1.5.1 核心对比表（最关键差异）

| 对比维度 | **Manus** | **OpenClaw** |
| --- | --- | --- |
| **产品性质** | 商业 SaaS 平台（闭源） | 开源项目（MIT 协议） |
| **部署方式** | 云端托管，登录即用 | 本地部署（电脑/服务器） |
| **运行环境** | 云端沙箱（厂商服务器） | 你的本地设备（或自行部署到云环境） |
| **控制权** | 平台完全掌控 | 用户 100% 掌控 |
| **数据隐私** | 数据上传云端，依赖厂商安全 | 数据本地存储，隐私自主可控 |
| **模型选择** | 内置专用模型，不可更换 | 可自由切换 GPT‑ 4、Claude、DeepSeek 等 |
| **权限边界** | 沙箱隔离，无法逃逸 | 可获得宿主机完整权限（风险更高） |
| **上手门槛** | 极低（零配置、开箱即用） | 中高（需部署、配置模型、写 Skill） |
| **成本** | 订阅付费（月费/积分） | 免费开源，仅承担模型 API 费用 |
| **生态** | 官方技能库，审核严格 | 社区驱动，3000+ 开源 Skill 自由使用 |
| **代表理念** | “让 AI 替你包办一切” | “给你工具，自己打造专属 AI 员工” |

### 1.5.2 相同点（为什么常被放在一起比）

1. **核心定位一致**：都是**通用 AI Agent**，目标是从“聊天”升级为“执行”，能自主规划、调用工具、完成复杂任务。
2. **能力范围重叠**：都能操作浏览器、读写文件、执行代码、处理邮件/日历、生成报告、自动化办公。
3. **都解决“AI 只会说不会做”**：突破传统聊天机器人局限，实现端到端任务交付。
4. **都在 2025–2026 年爆火**：Manus 2025 年 3 月引爆国内，OpenClaw 2026 年初席卷全球开源社区。

### 1.5.3 关键差异详解（为什么路线完全不同）

（1）架构哲学：云端托管 vs 本地优先

- **Manus**：**中心化云端架构**。所有计算、规划、执行都在厂商服务器的沙箱里完成。
  - 优点：稳定、安全、无需维护、跨设备一致。
  - 缺点：数据必须上传、隐私不可控、成本高、功能受平台限制。
- **OpenClaw**：**本地优先架构**。核心运行在你的电脑/服务器上。
  - 优点：数据不出本地、隐私安全、可深度定制、无平台限制。
  - 缺点：需自行部署、维护、承担模型成本、跨设备同步复杂。

（2）智能体模式：多 Agent 闭环 vs 单 LLM 执行引擎

- **Manus**：**多智能体协同**（规划师 + 执行官 + 验收员），形成“思考- 行动- 验证”闭环。
  - 更像一个团队在协作，任务成功率高、容错强。
- **OpenClaw**：**单 LLM 驱动的执行框架**。自身是编排层，把任务交给底层大模型做规划。
  - 能力上限取决于你选的模型：用 GPT- 4/Claude 很强，用小模型则弱。

（3）控制权与安全：平台沙箱 vs 用户全权

- **Manus**：**平台掌控一切**。你只有使用权，没有控制权。
  - 安全：沙箱隔离，不会破坏你的本地环境。
  - 风险：平台可封禁、可查看你的数据、可随时改规则。
- **OpenClaw**：**用户全权掌控**。你拥有所有权限。
  - 安全：可直接操作你的文件、系统，配置不当有风险。
  - 自由：可改代码、换模型、加技能、完全自定义。

（4）成本与门槛：付费即用 vs 免费但需动手

- **Manus**：**付费订阅**，但**零门槛**。
  - 适合：非技术用户、不想折腾、追求稳定的人。
- **OpenClaw**：**免费开源**，但**有技术门槛**。
  - 适合：开发者、极客、重视隐私、需要高度定制的人。

一句话总结：

**Manus 是“五星级酒店式 AI 服务”**：你只动口，它包办一切，但你要付费、数据要交给它、不能自己改菜单。

**OpenClaw 是“DIY 厨房式 AI 工具”**：厨房（电脑）是你的、食材（模型）自己选、菜谱（Skill）自己写，完全自由，但你得自己动手、自己维护。

怎么选？（快速决策）

- 选 **Manus**：如果你是**非技术用户**、追求**开箱即用**、**不想折腾**、**能接受付费和数据上传**。
- 选 **OpenClaw**：如果你是**开发者/极客**、重视**隐私与控制权**、需要**高度定制**、愿意**自己部署和维护**。

## 1.6 OpenClaw 技术原理

官网地址：https://openclaw.ai/

官网文档：https://docs.openclaw.ai/zh-CN

OpenClaw 的技术本质是一套**本地优先的 AI Agent 执行引擎**，核心通过“网关调度+LLM 决策+技能执行+分层记忆”的闭环架构，让 AI 从“对话式建议”升级为“系统级执行”，实现自主完成文件操作、终端命令、浏览器自动化等复杂任务。

其核心架构采用三层解耦设计，以网关为中枢，打通“思考”与“行动”的全链路。最上层是 **LLM 大模型层**，作为“智能大脑”，支持 GPT-4、Claude 等云端模型与 Llama、Mistral 等本地模型的混合部署，通过标准化接口与网关联动，用户可按需切换以平衡隐私与推理能力。中间层是**网关层（Gateway）**，基于 Node.js 构建的常驻守护进程，默认监听本地 WebSocket 端口（18789），是系统的“神经中枢”。它负责消息标准化、会话路由、技能调度、安全认证与状态维护，将来自微信、Telegram 等多渠道的指令统一分发，并同步执行结果。最下层是**渠道与执行层**，包含交互渠道与技能（Skills）插件，技能作为 AI 的“手脚”，通过标准化协议封装 Shell 执行、文件读写、浏览器驱动等能力，社区贡献的 3000+插件实现了开箱即用的场景覆盖。

OpenClaw 的核心运行逻辑是 **Agent 循环（思考-行动-反馈-记忆）**，这是其实现自主执行的关键。当用户发送指令后，网关首先加载智能体的人格定义（Soul）、分层记

忆与可用技能清单，按需注入技能以控制 Token 消耗。随后将指令、上下文、技能 Schema 打包发送给 LLM，LLM 完成意图解析与任务规划，生成工具调用指令（而非自然语言回复）。网关接收到调用指令后，通过权限策略管线与沙箱隔离（如 Docker）执行对应技能，将执行结果（成功/失败、返回值、日志）实时反馈给 LLM。若任务未完成，LLM 基于反馈调整规划，触发新一轮工具调用，直至生成最终结果或达到最大循环次数（默认约 20 次）。任务结束后，网关自动更新会话记忆与长期记忆，实现“越用越精准”的持续优化。

为支撑本地长期运行，OpenClaw 设计了两大关键保障：**分层记忆体系**与**并发安全机制**。记忆分为四级：不可变的系统人格（SOUL）、动态工具注册表（TOOLS）、基于向量的用户长期偏好（USER）、实时会话记忆（Session），通过本地文件存储确保隐私可控。并发安全则依赖基于文件的分布式锁，通过锁文件、指数退避重试、过期锁自动回收与看门狗线程，避免多 Agent 并发操作导致的会话数据损坏。

此外，**权限与安全架构**是其落地的核心。系统采用九层策略过滤，对工具调用进行身份校验、渠道策略限制、敏感操作授权等多重审查。同时支持“本地模型优先+云端模型降级”的隐私策略，敏感任务可完全离线运行，复杂任务则通过加密通道调用云端模型并脱敏数据，实现自由与安全的平衡。

综上，OpenClaw 通过解耦架构、闭环执行循环、分层记忆与严格安全策略，解决了传统 AI“只会说不会做”的痛点，成为 2026 年开年现象级 AI Agent 的技术根基。

OpenClaw 的术语体系围绕“网关-智能体-技能-记忆”的核心架构设计，以下按**核心架构、智能体与记忆、技能与工具、交互与运行、安全与扩展**五大类，用结构化表格清晰呈现，兼顾定义、作用与关键细节，适配开发与使用双视角。

(1)核心架构术语（系统骨架）

| 术语 | 中文释义 | 核心作用 | 关键细节 |
| --- | --- | --- | --- |
| **Gateway** | 网关 | 中央协调器与进程管家 | 管理所有 Channels、路由消息、控制 Agent 生命周期；支持本地/远程部署，提供 RPC 入口 |
| **Agent** | 智能体 | 任务执行的核心实例 | 由工作区文件（Identity、Soul、Skills 等）定义能力与人格；负责意图拆解、工具调用与结果汇总 |
| **Channel** | 渠道 | 用户交互入口 | 对接 Telegram、Discord、飞书等 23+ 平台；支持单聊/群聊，可配置“提及模式”触发 |

| | | | |
|---|---|---|---|
| **Node** | 节点 | 跨设备执行单元 | 用于 iOS/Android 等终端，提供 Canvas、摄像头等能力；支持分布式任务执行 |

(2)智能体与记忆术语（核心能力）

| 术语 | 中文释义 | 核心作用 | 关键细节 |
|---|---|---|---|
| **Workspace** | 工作区 | Agent 配置目录 | 存放 Identity、Soul、Skills 等文件；默认路径 `~/.openclaw/workspace/` |
| **Identity** | 身份 | Agent 核心标识 | 用 `identity.md` 定义角色（如“数据分析师”）、职责与权限；决定 Agent 的“核心人设” |
| **Soul** | 灵魂 | 人格与风格定义 | 补充 Identity，细化沟通语气、专业背景；让 Agent 更具“个性化” |
| **Memory** | 记忆系统 | 持久化与上下文管理 | 分 4 类（见下文）；通过向量检索+Markdown 持久化，避免“会话即遗忘” |
| **Session Memory** | 会话记忆 | 对话上下文 | 记录当前会话的交互内容；持久化，用于上下文连贯 |
| **Semantic Memory** | 语义记忆 | 知识与概念 | 存储长期事实、偏好；通过向量检索召回，支撑“长期记忆” |
| **Procedural Memory** | 程序记忆 | 技能与模式 | 记录常用任务的执行流程；加速重复任务的自动化 |
| **Working Memory** | 工作记忆 | 当前任务焦点 | 临时存储任务中间状态；任务结束后不持久化，降低资源占用 |
| **Compaction** | 会话压缩 | 记忆优化策略 | 压缩会话记忆，保留关键信息；减少 Token 消耗，避免上下文溢出 |

(3)技能与工具术语（执行能力）

| 术语 | 中文释义 | 核心作用 | 关键细节 |
| --- | --- | --- | --- |
| **Skill** | 技能 | 可复用的能力包 | 以文件夹为单元，核心是 `SKILL.md`（YAML 元数据 + Markdown 指令）；支持脚本/资源扩展 |
| **Tool** | 工具 | 原子操作能力 | Agent 执行任务的“手脚”；如文件操作、终端执行、浏览器控制（Playwright）、网页搜索 |
| **SKILL.md** | 技能定义文件 | 技能的“说明书” | 必需文件；元数据声明依赖（OS/环境变量），正文用自然语言教 Agent 如何执行 |
| **Progressive Disclosure** | 渐进式信息披露 | 技能加载策略 | 启动时仅加载元数据，触发后加载正文，执行时按需读取资源；降低上下文成本 |

(4)交互与运行术语（工作流）

| 术语 | 中文释义 | 核心作用 | 关键细节 |
| --- | --- | --- | --- |
| **Agent Loop** | 智能体循环 | 核心执行流程 | 接收消息 → 上下文组装 → 模型推理 → 工具执行 → 流式回复 → 记忆持久化；单次会话序列化运行 |
| **Bootstrap** | 引导 | 系统提示构建 | 在模型推理前，组装 Identity、Soul、Skills 等上下文；支持 `agent:bootstrap` 钩子自定义 |
| **Hook** | 钩子 | 扩展与拦截点 | 分内部钩子（Gateway 命令/生命周期）和插件钩子（Agent/工具生命周期）；用于自定义流程 |

| | | | |
|---|---|---|---|
| **Provider** | 模型提供商 | AI 模型配置 | 对接 OpenAI、Anthropic、Ollama（本地模型）、国产模型等；通过 `models.json` 配置 |
| **Headless** | 无头架构 | 运行模式 | 无独立前端，以后台守护进程运行；通过现有聊天工具交互，专注“执行”而非“对话” |

(5)安全与扩展术语（保障与进阶）

| 术语 | 中文释义 | 核心作用 | 关键细节 |
|---|---|---|---|
| **Sandbox** | 沙箱 | 安全执行环境 | 隔离工具/脚本的执行权限；防止恶意操作，保护系统安全 |
| **RPC** | 远程过程调用 | 跨进程通信 | Gateway 与 Agent 的通信方式；支持 `agent`/`agent.wait` 等指令 |
| **Cron** | 定时任务 | 自动化调度 | 内置工具，支持按时间规则触发任务（如“每天 8 点整理邮件”） |

一句话总结重要内容：

Gateway 是“门卫”，Channel 是“电话”，Agent 是“CEO”，Skill 是“员工手册”，Tool 是“工具台”，Memory 是“档案室”，Sandbox 是“安全车间”——整个系统像一家 24 小时运转的“数字外包公司”，帮你完成电脑上的所有实操任务。

# 第二部分：安装环境准备

> ***本章目标***：*学完这章，你能确认自己具备开始的所有条件，并准备好 API Key*

OpenClaw 的安装部署环境要求以**官方规范**为核心，按“系统、硬件、软件、网络、安全”五大维度分层明确，兼顾云端模型与本地模型两种核心场景：

## 2.1 官方支持的操作系统

| 系统类型 | 最低版本 / 要求 | 推荐选择 | 关键说明 |
| --- | --- | --- | --- |
| **Linux** | Ubuntu 20.04+ LTS、Debian 11+ | Ubuntu 22.04 LTS | 生产环境首选，原生支持所有功能 |
| **macOS** | macOS 12+ (Monterey) | macOS 14+ (Sonoma) | Intel/Apple Silicon 均原生兼容 |
| **Windows** | Windows 10 21H2+ / 11 | WSL2 (Ubuntu 22.04) | 官方强烈建议用 WSL2；原生需 C++ 编译环境。WSL2 为 Windows 自带的 Linux 环境 |
| **云容器** | 九章智算云云容器实例 | 底层操作系统为 Ubuntu | 提供公网地址与端口，可 24 小时伺服，并接入公网应用，如微信，本地安装的话需要机器常开且不能远程访问。 |

## 2.2 核心软件依赖（必选）

| 依赖项 | 最低版本 | 安装说明 | 作用 |
| --- | --- | --- | --- |
| **Node.js** | 22.x | 官方源安装，避免系统默认低版本 | 框架核心运行时 |
| **Git** | 2.30+ | 系统包管理器或官网安装 | 拉取源码与版本管理 |
| **模型后端** | — | 二选一：<br>1. 云端 API（OpenAI/Claude/MiniMax/Kimi 等）<br>2. 本地服务（Ollama ≥0.15.4 / LM Studio） | 提供 LLM 推理能力 |

| | | | |
|---|---|---|---|
| **Windows 原生额外** | — | 安装 Visual C++ Build Tools（勾选“C++ 桌面开发”） | 编译 node-llama-cpp 等原生模块 |

## 2.3 网络与端口要求

1. **网络连通性**
   - 云端模型：需稳定访问海外 API（国内用户需配置代理）。
   - 本地模型：可完全离线，仅首次安装需联网拉取依赖。
   - 容器化：可通过联网拉取 Docker 镜像或直接安装部署并保存为镜像。
2. **端口放行（默认）**
   - 主网关端口：**18789**（WebSocket，必须开放）。
   - 管理/调试：按插件需求开放（如 8080 等）。
   - 远程访问：需配合反向代理（如 Nginx）并做好认证（九章智算云云容器实例默认已经配置）。

## 2.4 安全与权限要求

1. **权限基础**
   - 本地部署：需管理员/root 权限以安装系统依赖、配置守护进程。
   - 容器化：遵循“最小权限”原则，避免使用 root 运行容器。
2. **安全策略**
   - 启用官方九层权限过滤，限制敏感操作（如系统命令、文件写入）。
   - 本地模型：通过 Ollama/LM Studio 隔离模型进程，避免直接暴露端口。
   - 云端 API：妥善保管 API Key，避免明文写入配置文件。

## 2.5 快速部署建议

1. **新手优先**：使用官方一键脚本，自动检测并满足所有环境要求。
   - Linux/macOS：`curl -fsSL https://openclaw.ai/install.sh | bash`
   - Windows (PowerShell)：`iwr -useb https://openclaw.ai/install.ps1 | iex`
2. **Windows 用户**：优先安装 WSL2（Ubuntu 22.04），避免原生环境的编译与兼

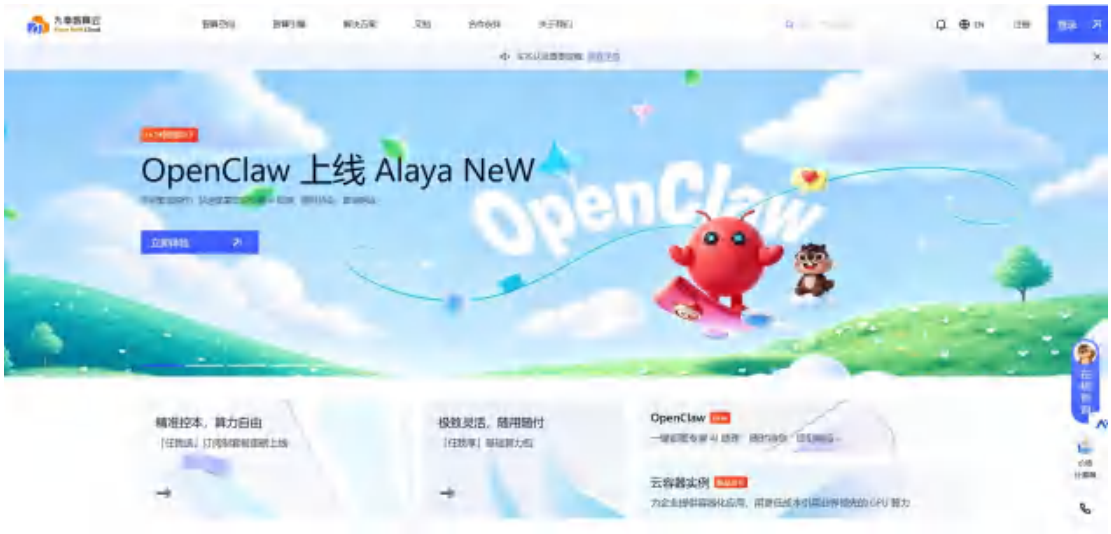
容性问题。

3. **本地模型（可选，前期用大模型厂商的 API）**：搭配 Ollama 部署，推荐模型如 `ministral-3:8b`（8GB VRAM）或 `glm-4.7-flash`（20GB VRAM）。

OpenClaw 的环境门槛灵活：**云端模型仅需 Node.js 22+、4GB 内存与网络；本地模型则需 16GB+ 内存、8GB+ VRAM 与更大存储**。Windows 建议用 WSL2（Windows 中的虚拟 linux 系统），不会折腾 WSL2 也可以在 windows powershell 中安装，有部分组件不兼容，不影响常规操作，容器化是跨平台最省心的选择。

## 2.6 免部署使用方案

**如果看完本文档，发现自己还是部署不上，也不想花几百块请人部署 openclaw，也可以不自己部署，直接选第三方 Openclaw 供应方提供的在线产品，好巧，我们公司正好提供这个产品，而且正在做活动，首月 45 元即可体验 openclaw，地址：https://www.alayanew.com/product/openClaw?utm_source=official04。**




# 第三部分：openclaw 安装部署

> ***本章目标***：*学完这章，你能完成 OpenClaw 安装并发出第一条消息*
>
> 本示例采用云容器实例 CCI。如何开通云容器实例，请参见：
>
> https://docs.alayanew.com/docs/documents/quickStart/createCCI
>
> 视频教程：https://www.bilibili.com/video/BV1VjkCBSEE4

## 3.1 环境检查：Node.js 是什么？

### 3.1.1 Node.js 简介

**Node.js** 是一个让 JavaScript 能在电脑本地运行的环境。简单说：

> *Node.js 就像 JavaScript 的"翻译官"，让它能在浏览器之外的地方工作。*

你不需要深入理解它，只需要确认电脑上已经安装了。

### 3.1.2 检查 Node.js 版本

打开你的终端（Terminal），输入：

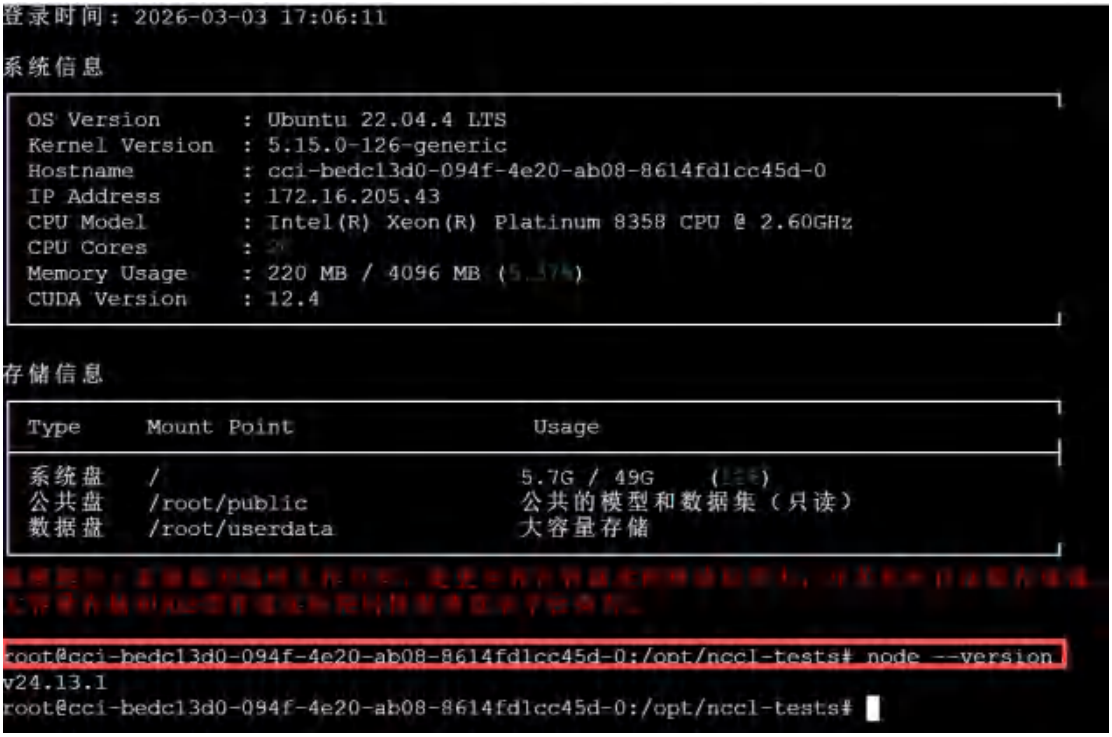
```
Plain Text
node --version
```

**期望看到的结果**：

```
Plain Text
v22.x.x
```

**判断标准**：

- • ✅ 版本 >= v22：可以继续
- • ❌ 版本 < v22：需要升级
- • ❌ 提示"command not found"：需要安装




### 3.1.3 如果 Node.js 不符合要求

macOS 安装/升级 node.js

```
Plain Text
# 使用 Homebrew 安装（推荐）
brew install node

# 如果已安装但版本低，升级
brew upgrade node
```

Windows 安装/升级 node.js

**推荐方式：使用 winget**

```
Plain Text
winget install OpenJS.NodeJS.LTS
```

**或者手动下载**:

1. 访问 https://nodejs.org
2. 下载 LTS 版本（长期支持版）


3. 按向导安装

> ***Windows 用户注意***：*官方推荐在 WSL2 中运行 OpenClaw，能避免很多奇怪问题。WSL2 安装指南：https://docs.microsoft.com/zh-cn/windows/wsl/install*

Linux 安装/升级 node.js

**Ubuntu/Debian**:

```
Plain Text
```

```
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt-get install -y nodejs
```

**CentOS/RHEL/Fedora**:

```
Plain Text
sudo dnf install nodejs
```

### 3.1.4 验证安装

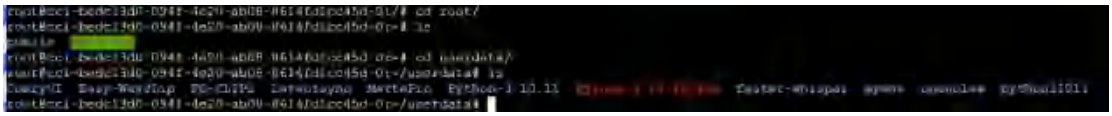
安装完成后，再次检查:

```
Plain Text
node --version
npm --version
```


两个命令都应该返回版本号。

## 3.2 安装命令：就一行，复制粘贴

进到 userdata 目录下（这个目录是共享目录，不同的云容器实例可以共享），创建 openclaw 目录，我们装相关软件都安装到这里。



确认 Node.js >= v22 后，执行安装命令:

```
Plain Text
npm install -g openclaw@latest
```

**这行命令在做什么?**

- `npm`: Node.js 的包管理器
- `install`: 安装

- `-g`：全局安装（在任何目录都能用 `openclaw` 命令）
- `openclaw@latest`：安装 npm 上当前发布版

验证安装成功

```
Plain Text
openclaw --version
```

期望看到类似输出：

```
Plain Text
2026.2.22
```

如果提示 git 没有安装成功，应该是网络问题，可以先不安装。

## 3.3 运行向导：openclaw onboard

安装完成后，运行初始化向导进行配置，后续要变更相关内容也可以再次运行：

```
Plain Text
openclaw onboard --install-daemon
```

**参数说明**：

- `onboard`：运行初始化向导
- `--install-daemon`：同时安装后台服务（推荐）

（1）第一步：风险提示

启动向导后，通常会先看到：

```
Plain Text
? I understand this is powerful and inherently risky. Continue?
❯  Yes
  No
```

这里选 `Yes` 继续。

（2）第二步：选择配置模式（Onboarding mode）

向导会问你：

```
Plain Text
? Onboarding mode
❯  QuickStart - Minimal setup, get running fast
  Manual - Full control over all settings
```

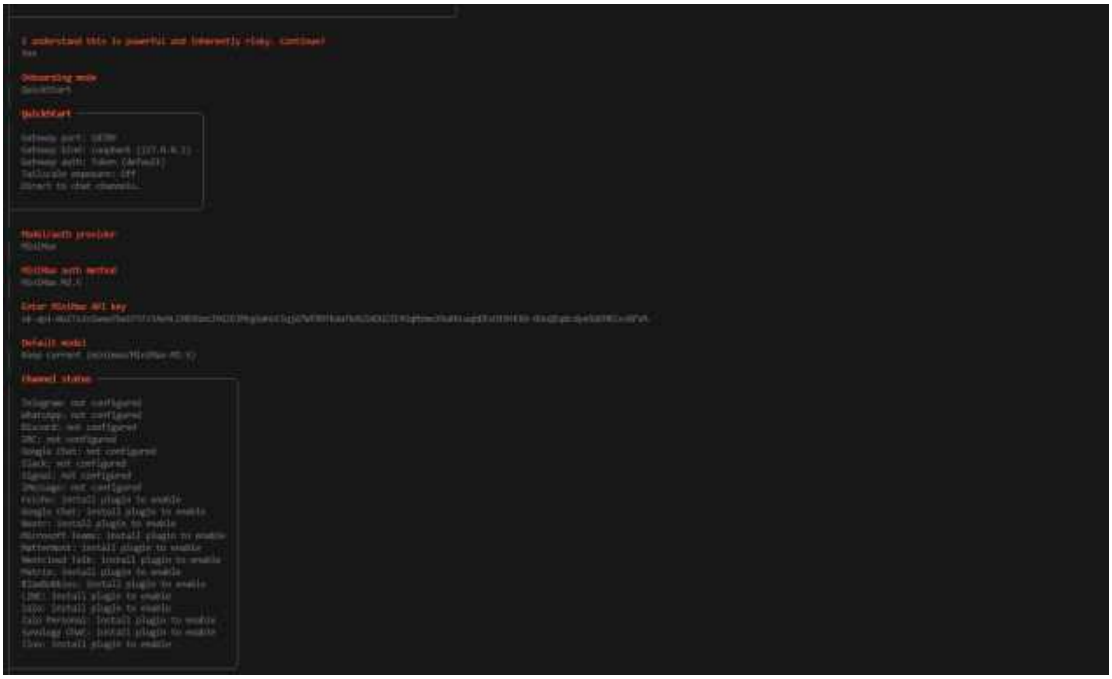
**怎么选？**

| 选项 | 适合谁 | 结果 |
| --- | --- | --- |
| QuickStart | 新手，想快速跑起来 | 自动配置推荐设置 |
| Manual | 想完全掌控配置 | 逐个配置每个选项 |

**我的建议**：第一次选 **QuickStart**，后面可以随时改配置。

（3）第三步：选模型提供商（Provider）

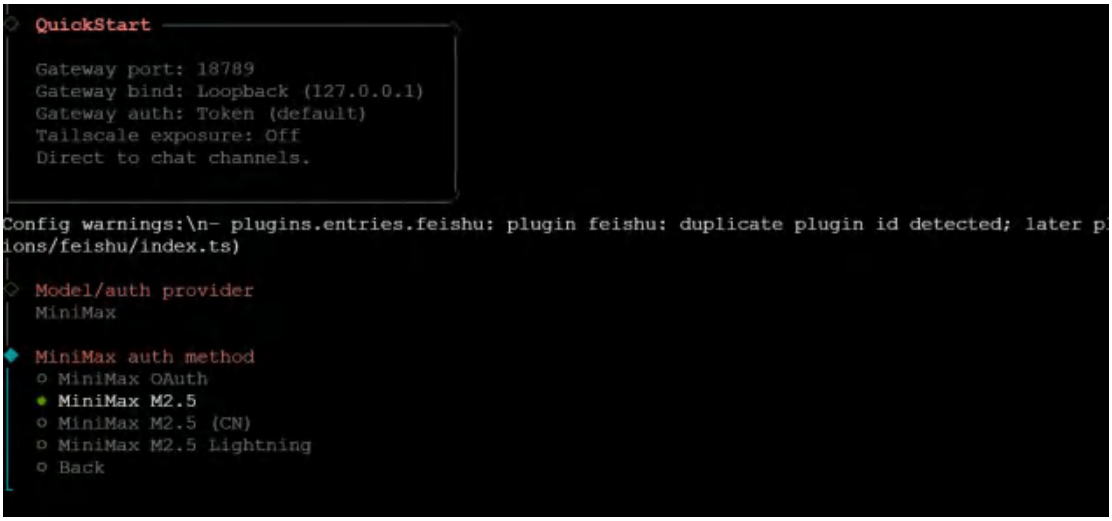
向导会提示你选择模型提供商，这里选官方推荐的 MiniMax2.5，提前到 https://www.minimaxi.com/充值并获取 API Key。



（4）第四步：再选鉴权方式（Auth method，容易漏）

在你选完厂商后，通常不会立刻要 Key，而是先进入该厂商的鉴权方式选择。

常见会看到类似：




**国内读者建议：初次配置可以选 API，这种是之前最常用的，如果长期大量使用，可以在后边选 Coding Plan 对应项（降低成本）。**

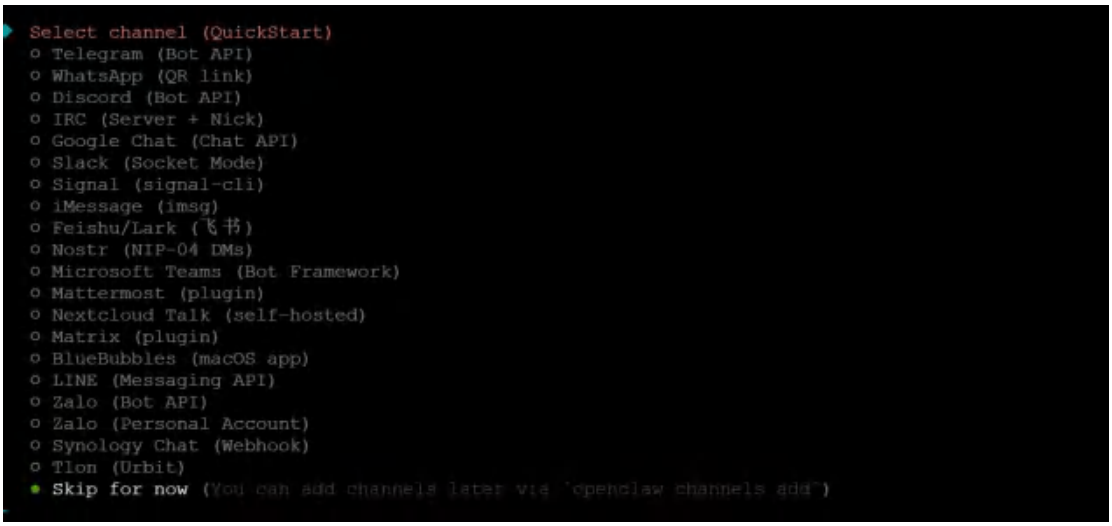
（5）第五步：输入 API Key 并选模型

这里，我们选择 MiniMax，并输入 API key。


选择 Keep current。

（6）第六步：配置 Channel（先 Skip！）

聊天工具先不选，直接 Skip for now，后边可以随时运行 openclaw onboard 再回来配置，我们将在后边的章节再进行说明。




（7）向导第七步：配置 Skills（建议开）

`Channel` 之后，向导会进入 Skills 检查与可选安装，提示是不是要配置 Skill：

```
Plain Text
Skills status
Eligible: ...
Missing requirements: ...
...

? Configure skills now? (recommended)
❯  Yes
  No
```

**可以选择需要的 skill，也可以不选，选了之后需要按提示配置一下，很多国外的大模型，不能科学上网的话配了也没啥用，所以这里我们选不配置，选择 NO。**

（8）第八步：配置 Hooks（建议最小开启）

安装 skills 后会进入 Hooks 配置：


通过敲击空格键选中需要的 hook，再点回车键，总共没几个，我这里全选上了。

补充介绍一下 Hooks：

OpenClaw 中的 Hooks（钩子），本质是系统在关键执行节点预留的“自定义扩展入口”——就像你给自家汽车装了可自定义的“改装接口”，既不破坏原车核心结构，又能在启动、换挡、刹车等关键环节，插入自己的“改装逻辑”，让 OpenClaw 按你的需求定制化运行。默认就只有三四个 Hook，直接选上即可。

OpenClaw 本身是一套标准化的 AI Agent 执行框架，Hooks 是它的“灵活扩展机制”，核心价值：

1. **不修改核心代码**：无需改动 OpenClaw 源码，就能定制流程；
2. **精准干预节点**：只在你关心的环节（如接收消息、调用工具、返回结果）插入逻辑；
3. **适配个性化需求**：比如给消息加过滤、给工具调用加权限校验、给返回结果加格式转换。

简单说，Hooks 让你“既用现成框架，又能按自己的规则改流程”。

OpenClaw 的 Hooks 配置极其简单，无需写复杂代码，核心是在 **Agent 工作区** 新建 `hooks` 文件夹，按“钩子名称”创建脚本文件即可。

（9）第九步：选择 Hatch （孵化，访问）方式（关键）

在收尾阶段，向导会给你一个启动入口选择：

```
Plain Text
? How do you want to hatch your bot?
  Hatch in TUI (recommended)
❯  Open the Web UI
  Do this later
```

**选项说明：**

- `Hatch in TUI (recommended)`：这个是通过终端访问（默认选这个）
- `Open the Web UI`：通过访问 127.0.0.1:18789 网页，对于在 CCI 中部署的 OpenClaw，由于 CCI 支持外网 IP 映射，可以通过外网访问 OpenClaw。这里我们就选 Open the Web UI。

修改一下配置文件：/root/.openclaw/openclaw.json

```JSON
{
  "meta": {
    "lastTouchedVersion": "2026.3.2",
    "lastTouchedAt": "2026-03-04T08:18:57.470Z"
  },
  "wizard": {
    "lastRunAt": "2026-03-04T08:18:57.433Z",
    "lastRunVersion": "2026.3.2",
    "lastRunCommand": "configure",
    "lastRunMode": "local"
  },
  "auth": {
    "profiles": {
      "minimax:default": {
        "provider": "minimax",
        "mode": "api_key"
      },
      "minimax-portal:default": {
        "provider": "minimax-portal",
        "mode": "oauth"
      }
    }
  },
  "models": {
    "mode": "merge",
    "providers": {
      "minimax": {
        "baseUrl": "https://api.minimaxi.com/v1",
        "apiKey": "sk-api-
v9eUSK2HeCiZDPXhGUvYGX7bngWjAIoZeiN78RRbLMV3IJS0BcxvdE8dVyN0aAkQ6V
DSMrozdv165lcU58EjPDEpHBjhoDRJ5SxnVwydkRylEuVmWNwU111",
        "api": "openai-completions",
        "models": [
          {
            "id": "MiniMax-M2.5",
            "name": "MiniMax-M2.5",
            "reasoning": true,
            "input": [
              "text"
            ],
            "cost": {
```

```
            "input": 0.3,
            "output": 1.2,
            "cacheRead": 0.03,
            "cacheWrite": 0.12
          },
          "contextWindow": 200000,
          "maxTokens": 8192
        }
      ]
    },
    "minimax-portal": {
      "baseUrl": "https://api.minimaxi.com/anthropic",
      "apiKey": "minimax-oauth",
      "api": "anthropic-messages",
      "models": [
        {
          "id": "MiniMax-M2.5",
          "name": "MiniMax M2.5",
          "reasoning": false,
          "input": [
            "text"
          ],
          "cost": {
            "input": 0,
            "output": 0,
            "cacheRead": 0,
            "cacheWrite": 0
          },
          "contextWindow": 200000,
          "maxTokens": 8192
        },
        {
          "id": "MiniMax-M2.5-highspeed",
          "name": "MiniMax M2.5 Highspeed",
          "reasoning": true,
          "input": [
            "text"
          ],
          "cost": {
            "input": 0,
            "output": 0,
            "cacheRead": 0,
            "cacheWrite": 0
          },
```

```
            "contextWindow": 200000,
            "maxTokens": 8192
          },
          {
            "id": "MiniMax-M2.5-Lightning",
            "name": "MiniMax M2.5 Lightning",
            "reasoning": true,
            "input": [
              "text"
            ],
            "cost": {
              "input": 0,
              "output": 0,
              "cacheRead": 0,
              "cacheWrite": 0
            },
            "contextWindow": 200000,
            "maxTokens": 8192
          }
        ]
      },
      "custom-api-minimaxi-com": {
        "baseUrl": "https://api.minimaxi.com/v1",
        "apiKey": "sk-api-
v9eUSK2HeCiZDPXhGUvYGX7bngWjAIoZeiN78RRbLMV3IJS0BcxvdE8dVyN0aAkQ6V
DSMrozdv165lcU58EjPDEpHBjhoDRJ5SxnVwydkRylEuVmWNwU111",
        "api": "openai-completions",
        "models": [
          {
            "id": "MiniMax-M2.5",
            "name": "MiniMax-M2.5 (Custom Provider)",
            "reasoning": false,
            "input": [
              "text"
            ],
            "cost": {
              "input": 0,
              "output": 0,
              "cacheRead": 0,
              "cacheWrite": 0
            },
            "contextWindow": 16000,
            "maxTokens": 4096
          }
```

```
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "custom-api-minimaxi-com/MiniMax-M2.5",
        "fallbacks": [
          "minimax/MiniMax-M2.5",
          "minimax-portal/MiniMax-M2.5-highspeed",
          "minimax-portal/MiniMax-M2.5-Lightning"
        ]
      },
      "models": {
        "minimax/MiniMax-M2.5": {
          "alias": "Minimax"
        },
        "minimax-portal/MiniMax-M2.5": {
          "alias": "minimax-m2.5"
        },
        "minimax-portal/MiniMax-M2.5-highspeed": {
          "alias": "minimax-m2.5-highspeed"
        },
        "minimax-portal/MiniMax-M2.5-Lightning": {
          "alias": "minimax-m2.5-lightning"
        },
        "custom-api-minimaxi-com/MiniMax-M2.5": {
          "alias": "minimaxi-2.5"
        }
      },
      "workspace": "/root/.openclaw/workspace"
    }
  },
  "tools": {
    "profile": "messaging"
  },
  "commands": {
    "native": "auto",
    "nativeSkills": "auto",
    "restart": true,
    "ownerDisplay": "raw"
  },
  "session": {
```

```
    "dmScope": "per-channel-peer"
  },
  "hooks": {
    "internal": {
      "enabled": true,
      "entries": {
        "boot-md": {
          "enabled": true
        },
        "bootstrap-extra-files": {
          "enabled": true
        },
        "command-logger": {
          "enabled": true
        },
        "session-memory": {
          "enabled": true
        }
      }
    }
  },
  "gateway": {
    "port": 18789,
    "mode": "local",
    "bind": "lan",
    "controlUi": {
      "allowedOrigins": [
        "http://localhost:18789",
        "http://127.0.0.1:18789",
        "http://60.171.65.125:30047"
      ],
      "allowInsecureAuth": true
    },
    "auth": {
      "mode": "token",
      "token": "853e151b82baf44e693f13ee21150faa117f035c12b78811"
    },
    "tailscale": {
      "mode": "off",
      "resetOnExit": false
    },
    "nodes": {
      "denyCommands": [
        "camera.snap",
```

```
        "camera.clip",
        "screen.record",
        "calendar.add",
        "contacts.add",
        "reminders.add"
      ]
    }
  },
  "skills": {
    "install": {
      "nodeManager": "npm"
    }
  },
  "plugins": {
    "entries": {
      "minimax-portal-auth": {
        "enabled": true
      }
    }
  }
}
```


云容器实例可以允许公网访问，需要配置其 IP 与端口映射，如上图所示。我们通过 https://60.171.65.125:30047 即可访问安装后的 openclaw，也可以直接把服务器上的地址 ssh 映射到本地：

```
PowerShell
ssh -N -L 18789:127.0.0.1:18789 root@60.171.65.125 -p 30084
```

本地可以通过

http://127.0.0.1:18789/#token=853e151b82baf44e693f13ee21150faa117f035c12b78877 访问，到配置文件 openclaw.json 中找到这个 token 后边的字符串。


（10)第十步：记录 Dashboard 链接与 Gateway 状态

在你完成 Hatch 选择后，向导会输出控制台访问信息与网关状态（如 `Web UI`、`Gateway WS`、`Gateway: reachable`）。

如果你选了 `Open the Web UI`，一般会直接给出带 token 的 Dashboard 链接并尝试自动打开浏览器。

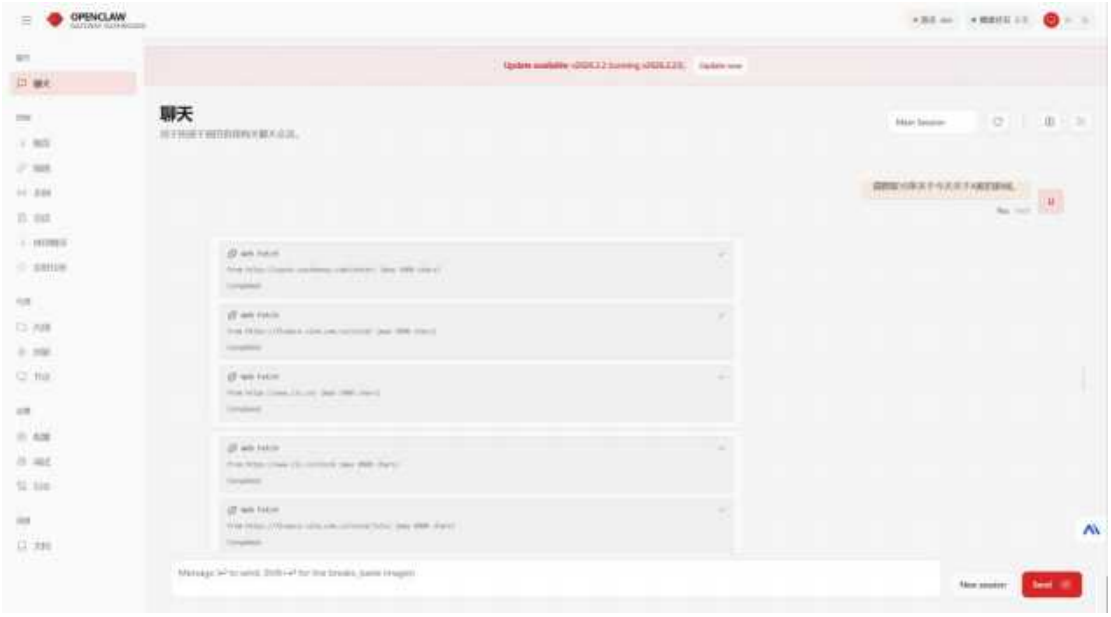
如果你选了 `Do this later`，后续可用：

```
Plain Text
openclaw dashboard --no-open
```

再次获取控制台入口。

## 3.4 开始对话

```
Plain Text
请爬取 10 条关于今天关于 A 股的新闻。
```





如果你想测“查询能力”，可再补一条：

```
Plain Text
请告诉我北京今天的天气，并给出穿衣建议（1 句话）。
```


按回车发送。

**期望的回复**：

它应该直接给出结构化结果（清单或天气建议），而不是继续做泛泛自我介绍。

**如果看到回复，恭喜你！安装成功！**

## 3.5 如果出错了怎么办

### 3.5.1 问题一：Gateway 启动失败

**症状**：向导提示 `Gateway failed to start`

**可能原因**：

1. 端口 18789 被占用
2. 权限不足
3. 配置文件错误

**解决方法**：

```
Plain Text
# 查看端口占用
lsof -i :18789

# 或者换端口启动
openclaw gateway start --port 18790
```

### 3.5.2 问题二：发送消息无回复

**症状**：消息发送后，一直显示"正在输入"但没有回复

**可能原因**：

1. API Key 错误
2. 网络不通
3. 模型服务异常

**解决方法**:

```
Plain Text
# 检查配置
openclaw config get

# 检查模型连接
openclaw doctor

# 查看日志
openclaw logs
```

### 3.5.3 问题三：Web UI 打不开

**症状**：浏览器访问 `127.0.0.1:18789` 显示无法连接

**可能原因**:

1. Gateway 没启动
2. 防火墙阻挡
3. 地址输错

**解决方法**:

```
Plain Text
# 确认 Gateway 在运行
openclaw status

# 如果未运行，手动启动
openclaw gateway

# 停止服务
openclaw gateway stop
```

# 第四部分：飞书接入 Openclaw

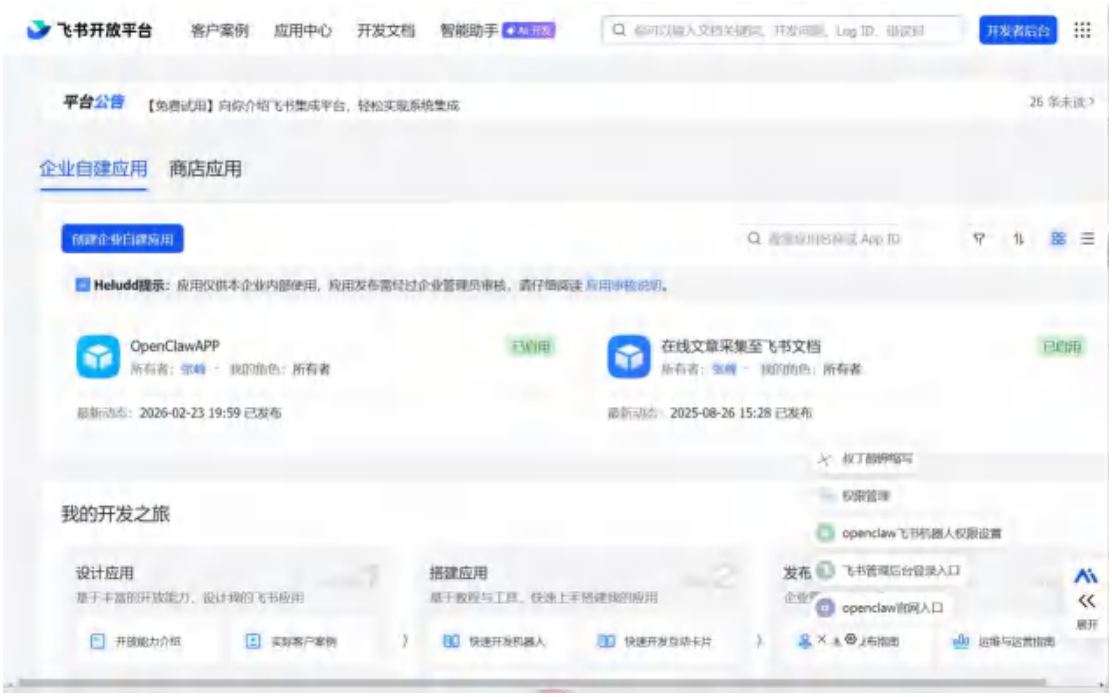
> ***本章目标***：*学完这章，你能在飞书里@AI 机器人，让它帮你办事，也可以直接通过微信与 Openclaw 对话。*

飞书接入的整体流程

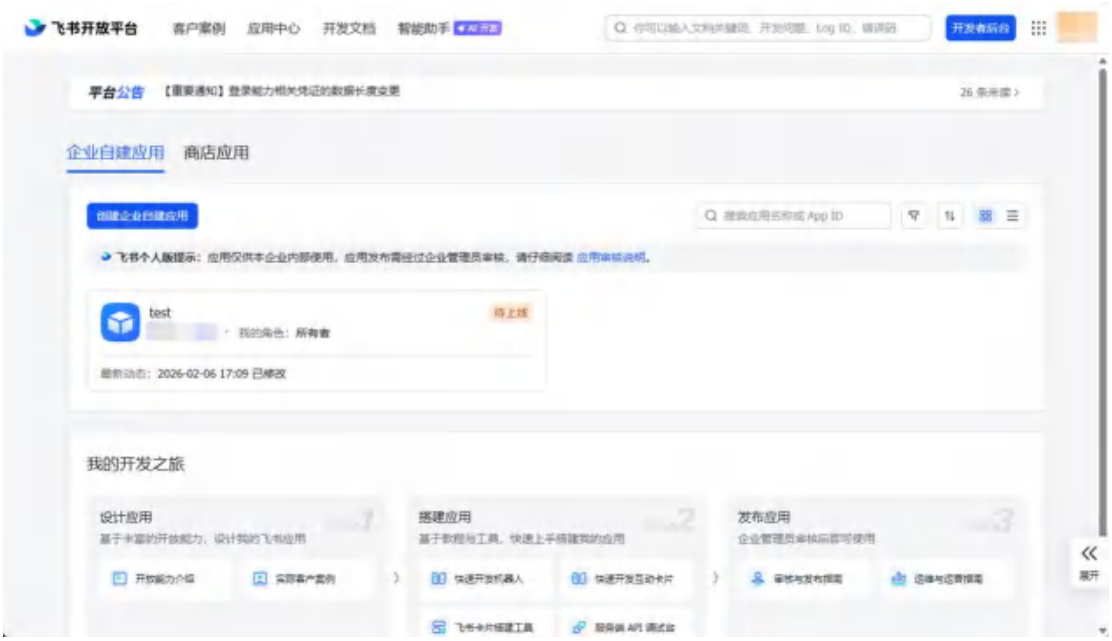
1. **平台侧建应用**（拿到凭证）
2. **OpenClaw 侧配置渠道**（`openclaw channels add`）
3. **启动 Gateway 并验证收发**
4. **配对/白名单放行**

## 4.1 Step 1：在飞书开放平台创建应用

进入 https://open.feishu.cn/开发者后台“创建企业自建应用”：




- 填写**应用名称**、**应用描述**，并选择**应用图标**，完成后点击**创建**。

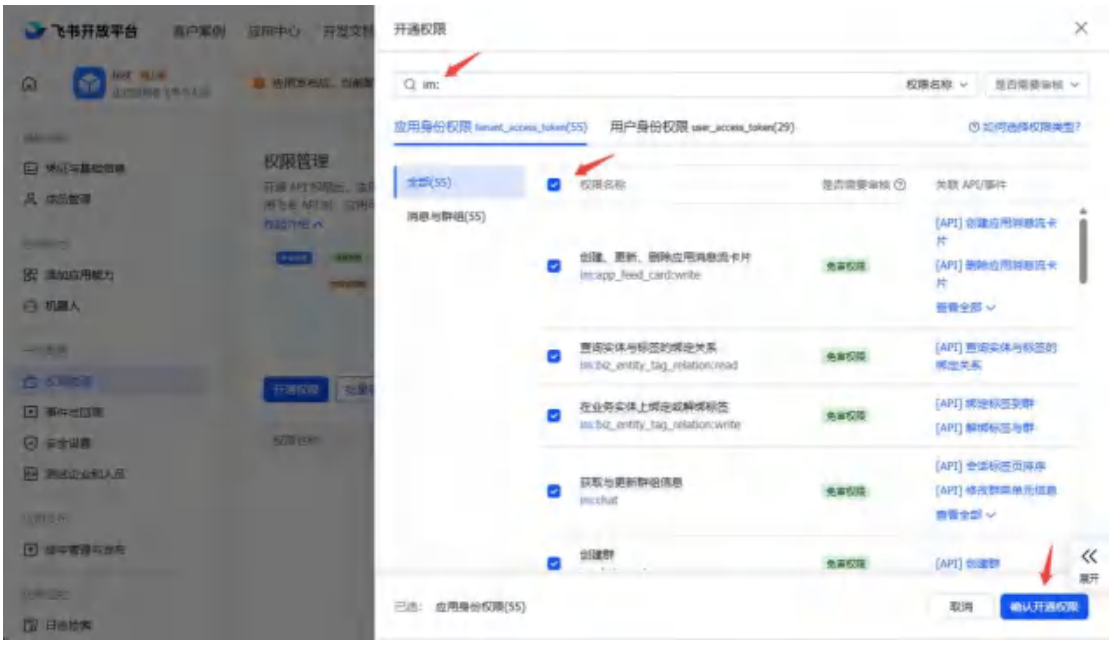



- **获取应用凭证**。创建成功后，在应用详情的 **凭证与基础信息** 页面，记录下 `App ID` 和 `App Secret`。这是核心凭证，后续在 OpenClaw 配置中必须用到，请妥善保管。


- **为应用添加"机器人"能力**。在 **应用能力** 的 **添加应用能力** 列表中，找到并启用 **机器人** 能力。赋予您的应用接收和发送消息的能力。

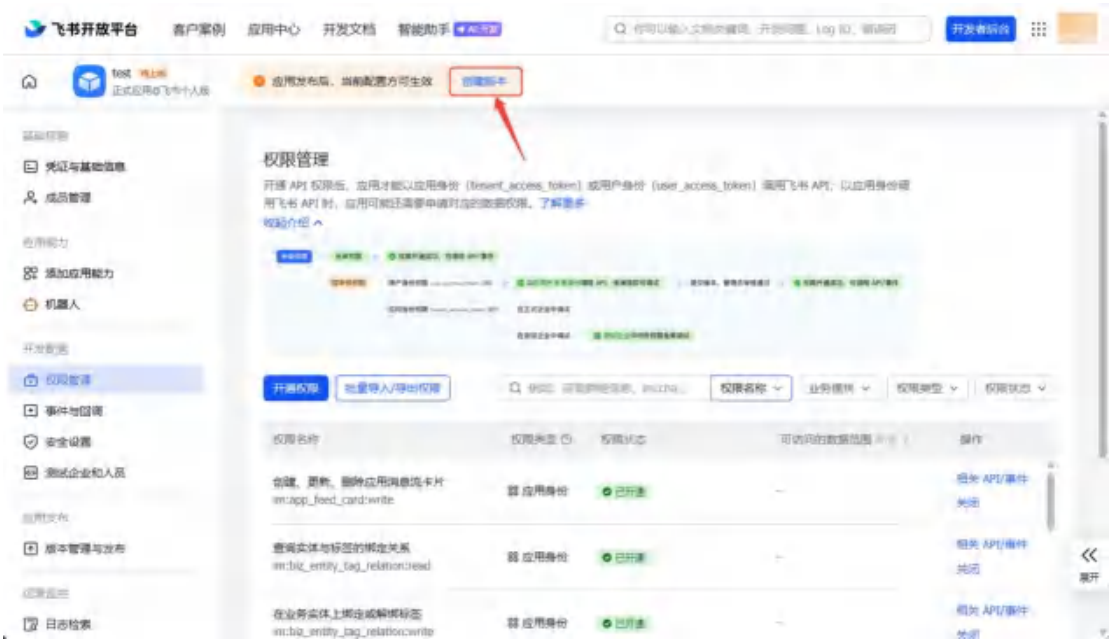

- 添加机器人成功后，左侧边栏出现 **机器人** 板块。


- **配置关键权限**。进入 **权限管理** 页面，搜索并添加以下与即时通讯相关的核心权限。搜索 im:，让机器人有权限在会话中正常工作。注意：冒号是英文的

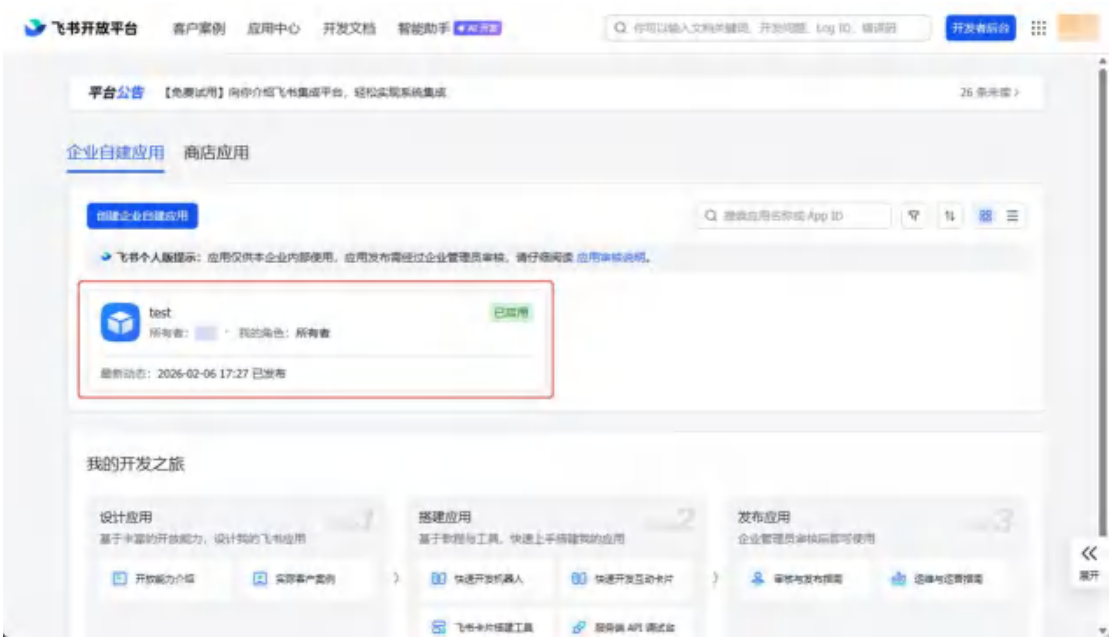




- **发布应用版本**。进入 **版本管理与发布** 页面，填写版本和更新说明，下拉点击 **保存** 即可。


- 点击 **创建版本**。




- 应用已发布成功。



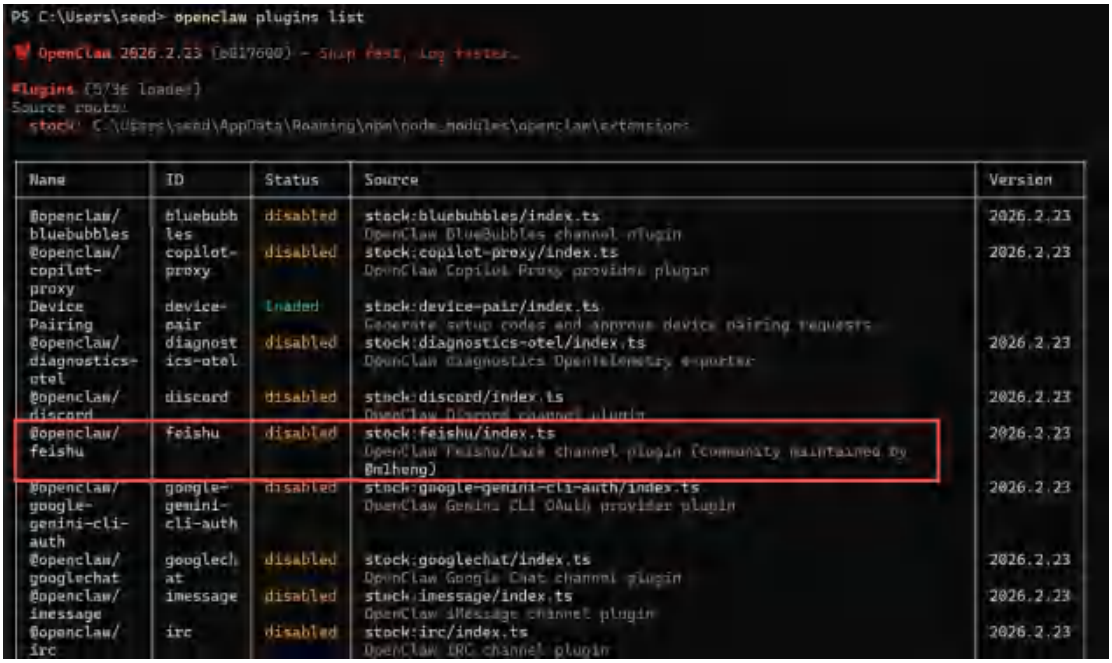

如发布后需要审核，可登录飞书客户端进行审核并批准该应用上线。


## 4.2 Step 2：在 OpenClaw 配置飞书

### 4.2.1 启用飞书插件

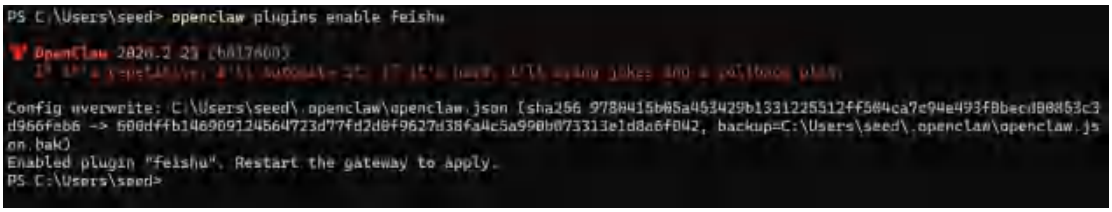
先查看插件列表：

```
Plain Text
openclaw plugins list
```




如果存在 `feishu` 且状态是 `disabled`，启用它：

```
Plain Text
openclaw plugins enable feishu
```




*提示：官方文档也给出 `openclaw plugins install @openclaw/feishu`，但用这种方法我在 Windows 下操作报告，建议还是按以下操作说明进行操作。*

我们优先启用内置的 plugin，Windows 下直接安装的 openclaw 在安装飞书插件时有点问题。需要重启网关才生效。

### 4.2.2 交互式添加 Channel

运行命令：

```
Plain Text
openclaw channels add
```

按提示完成配置：

**问题 1：选择渠道类型**

```
Plain Text
```

```
? Select channel type:
❯  Feishu/Lark (飞书)
  Telegram
  WebChat
  ...
```

选择 `Feishu/Lark (飞书)`

**问题 2：输入 App ID**

```
Plain Text
? Enter Feishu App ID: cli_xxxxxxxxxxxxxxx
```

粘贴你在获取的 App ID

**问题 3：输入 App Secret**

```
Plain Text
? Enter Feishu App Secret: [粘贴 Secret]
```

粘贴你获取的 App Secret（粘贴时不显示字符，这是正常的）

**问题 4：选择飞书域名**

```
Plain Text
? Which Feishu domain?
❯  feishu.cn (国内版)
  larksuite.com (国际版)
```

国内用户选 `feishu.cn`

**问题 5：群聊策略**

```
Plain Text
? Group chat policy:
❯  disabled (先不通群聊)
  enabled
```

**先选 `disabled`**，等私聊通了再开群聊。

**问题 6：需要 mention 才回复?**

```
Plain Text
? Require mention in group chats?
❯  yes (群里需要@才回复)
```

```
no
```

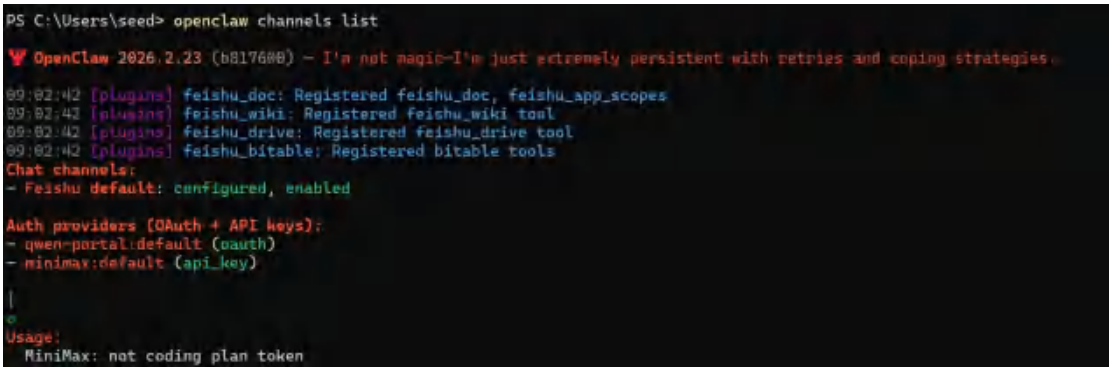
选 `yes`，避免机器人在群里乱说话。

### 4.2.3 验证配置

配置完成后，查看 Channel 列表：

```
Plain Text
openclaw channels list
```

应该显示：




## 4.3 Step 3：回到飞书开放平台开启事件订阅（长连接）

### 4.3.1 重点：顺序不能搞错

**正确的时序是**：

1. ✓ 飞书侧：创建应用 → 配置权限 → **发布应用**
2. ✓ OpenClaw 侧：`channels add` 配置渠道
3. OpenClaw 侧：**启动 Gateway**
4. 飞书侧：**开启事件订阅（长连接）**
5. 飞书侧：**配置事件订阅地址**

**如果顺序错了**，长连接会订阅失败，表现为"消息发出去，机器人没反应"。

我们已经完成前两步的配置，接下来继续操作后面三步。

### 4.3.2 启动 Gateway

```
Plain Text
openclaw gateway start
```

确认输出：

```
Plain Text
✓ Gateway started on http://127.0.0.1:18789
```

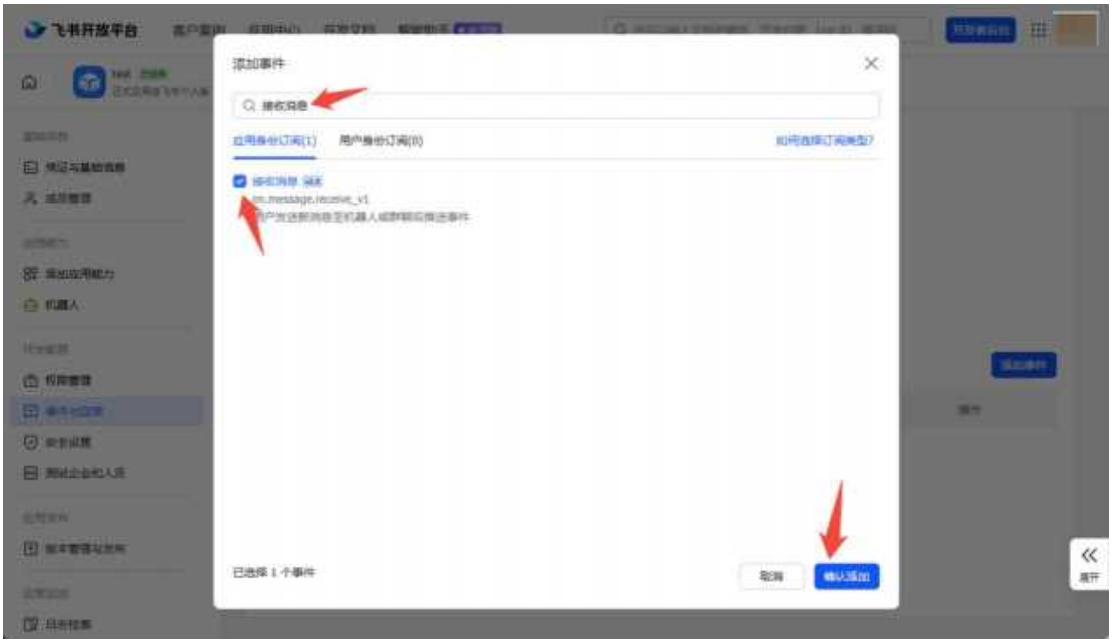
### 4.3.3 在飞书平台开启事件订阅

1. 回到飞书开放平台
2. 点击左侧"事件与回调"
3. 在"事件订阅方式"中，选择"长连接"

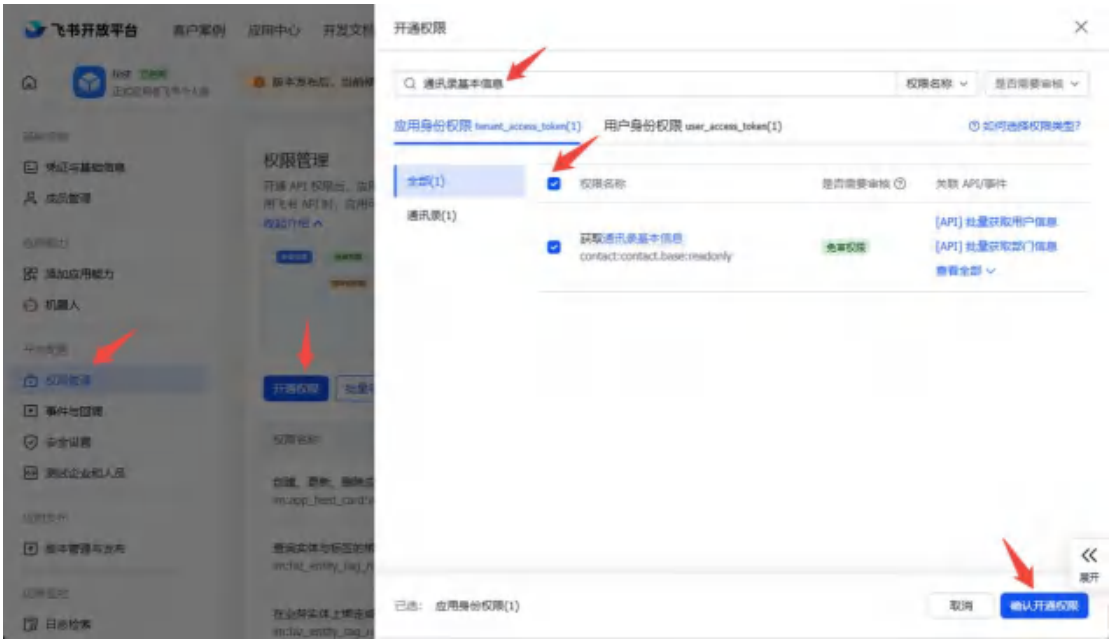
- 回到 飞书开放平台 的开发者后台，进入刚才自建成功的应用。
- 找到 **事件与回调** 模块，在 **事件配置** 中选择 **使用长连接接收事件**，点击**保存**。


- 此时 **添加事件** 按钮会从灰色不可点击变为可操作状态，点击 **添加事件** 。


- 点击 **添加事件**，搜索并勾选 **接收消息事件**（核心事件，确保机器人能接收飞书消息）。




- 进入「**权限管理**」页面，点击 **开通权限**，搜索并开通 **获通讯录基本信息** 权限。





- 所有配置完成后，进入 **版本管理与发布** 页面，按此前流程重新发布应用，点击 **保存**。

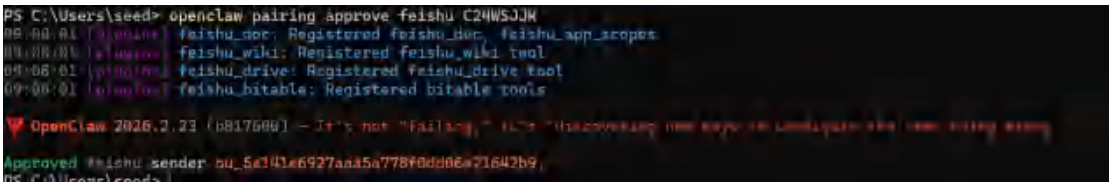
- 沿用此前发布流程，直接发布为在线应用即可。

## 4.4 Step 4：与机器人对话

- 打开飞书电脑客户端或手机 APP， 搜索之前创建的机器人。


- 应用发布成功后，返回飞书客户端（电脑端或手机 APP），找到之前创建的机器人，这时还不能直接对话。需要进行配对与放行。首次交互后机器人会回复含一段配对码（Pairing code）的内容。就是下图中的 C24WSJJK。需要在命令行执行一下：

```PowerShell
openclaw pairing approve feishu C24WSJJK
```


接下来就可以正常与飞书对话了：

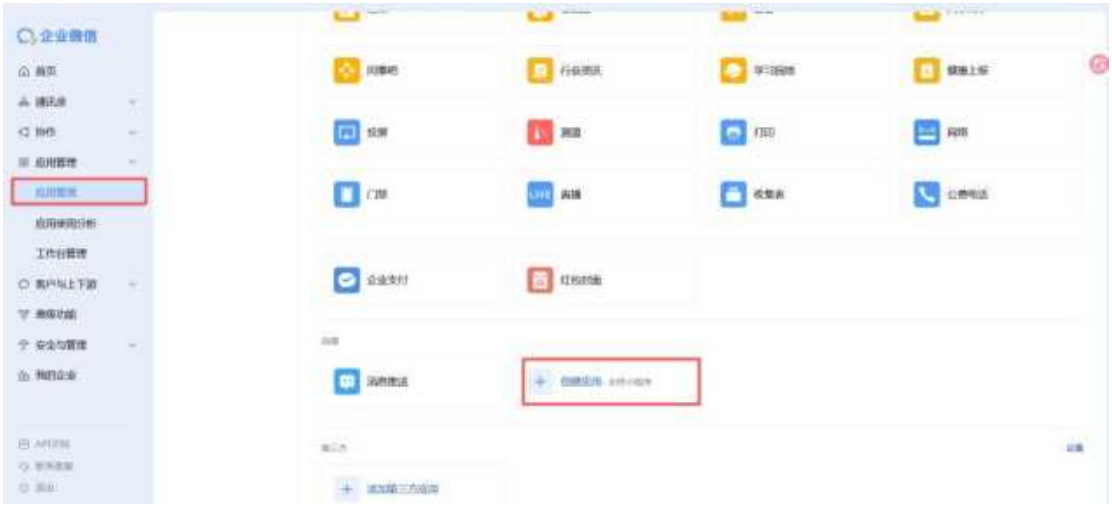

## 第五部分：微信接入 OpenClaw

> ***本章目标***：*学完这章，你能掌握微信如何接入 OpenClaw*

以前还得开网页？现在直接在微信对话框就能开聊！看这响应速度，先看看效果：


1、你先需要有一个 OpenClaw 的，可以本地安装（需要折腾一会），也可以到这里申请：https://www.alayanew.com/product/openClaw

2、你需要申请一个企业微信（不是你当前所有企业，需要注册一个新的，你是老板），任何人都能申请， work.weixin.qq.com 不需要企业认证 。注册过程很简单，按提示填写即可，过程中用微信扫下码。3、用网页登录（一定网页登录，不然看不到后台管理）登录地址和注册是同一个地址。进入应用管理->自建->创建应用



3.1 给应用取个名字 比如叫"疯聊 AI 客服机器人"，随便传个 Logo：

3.2 可见范围选你自己就行


3.3 创建完成点击应用记下 2 个参数 AgentId 和 Secret


其中 Secret 点击查看按钮，按提示在企业微信中查看。

4、创建 API 接收地址

在企业微信当前网页下方配置接收消息，通过设置 API 接收。

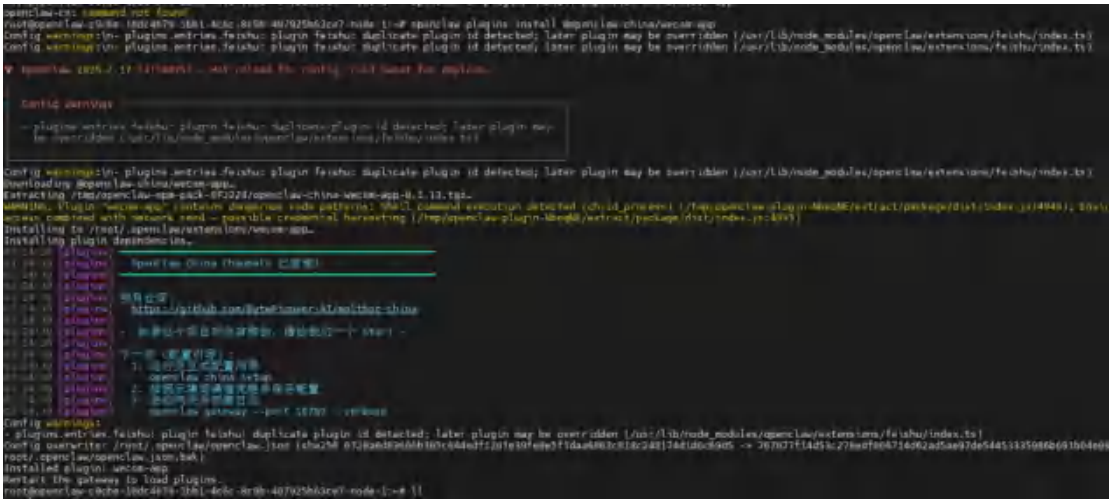
这里需要配置 3 个信息（有个门槛是需要一个公网 IP，所以本地部署的 Openclaw 就不适用了）

| 配置项 | 填写内容 | 说明 |
| --- | --- | --- |
| URL | http://服务器公网IP:端口/wecom-app | 把"服务器公网IP:端口"换成公网IP+端口 |
| Token | 点"随机获取" | 复制保存下来 |
| EncodingAESKey | 点"随机获取" | 复制保存下来 |

这个时候点击保存是保存不了的，因为这个地址是不能访问，需要在 OpenClaw 安装插件才行。

5、安装插件，在安装了 Openclaw 的机器上运行以下命令：

```
PowerShell
openclaw plugins install @openclaw-china/wecom-app
```



6、配置企业微信 API

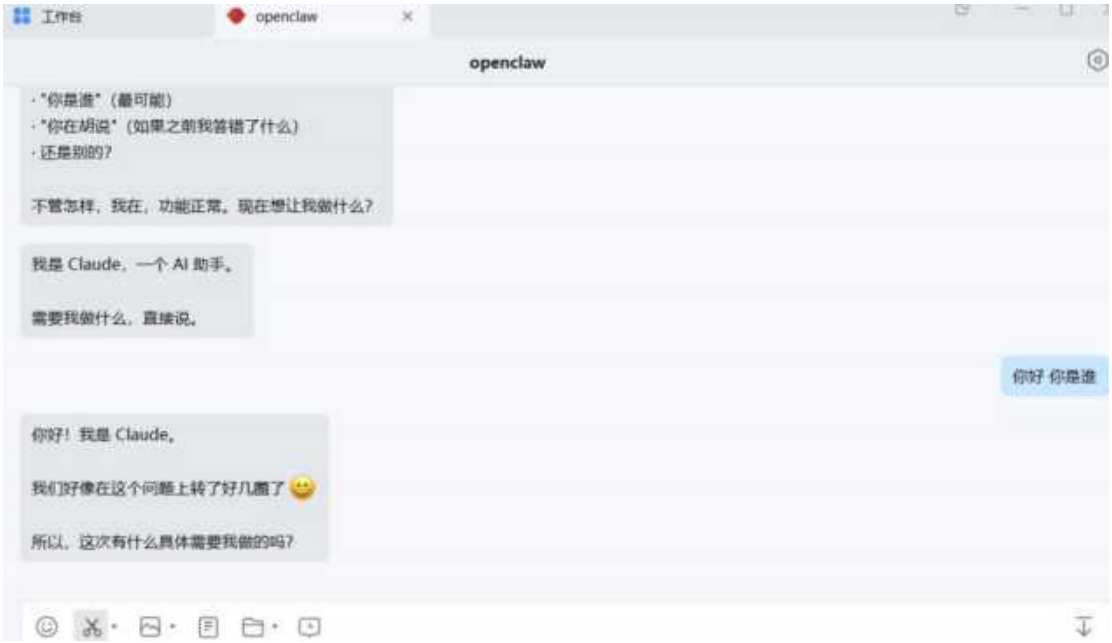
```
Bash
vim /root/.openclaw/openclaw.json

 "channels": {
    "wecom-app": {
      "enabled": true,
      "corpId": "你的企业 ID-在网页版-我的企业能找到",
      "corpSecret": "你的应用 Secret",
      "agentId": 你的应用 agentId,
      "token": "上面随机获取的 Token",
      "encodingAESKey": "上面随机获取的 EncodingAESKey",
      "dmPolicy": "open",
      "allowFrom": [
        "*"
      ]
    }
  },
```


7、重启 OpenClaw

```
Plain Text
openclaw gateway restart
```

8、将第 4 步的信息重新配置点击保存，现在就能保存成功了。9、配置企业可信 IP,这个 IP 就是 openclaw 所在的 ip 地址。ip 如果配置错误可能发消息不会有反应，大家如果使用我们平台部署，这里出现问题的话可以文末扫码小编本人来协助一起解决。


10、 打开企业微信测试 ->工作台->点击应用




10、最重要的一步来了，启动微信插件点击我的企业->微信插件->扫码关注


关注成功后，你的个人微信里就多了一个企业微信的消息入口。


现在就能和它聊天了。补充说明一下，如果是用了云环境有外网独立 IP 的才可以满足此功能，如果本地部署的 OpenClaw，即使可以用花生壳这类软件生成对外开放访问（低带宽）的域名是不能接入微信的，厂商对这块做了限制，因此建议大家使用以下现成产品：https://www.alayanew.com/product/openClaw

# 第六部分：Skill 技术原理与规范

> ***本章目标***：*学完这章，你能更解 skill 的原理与规范*

## 6.1 什么是 Skill?

Skill（技能）是一个包含指令的文件夹，用于教 Claude 如何处理特定任务或工作流程。它是定制 Claude 以满足特定需求的最强大方式之一。

**核心价值：**

- **一次教学，永久受益**：无需在每次对话中重复解释偏好、流程和专业知识
- **标准化工作流**：确保团队成员使用一致的方法完成任务
- **降低使用门槛**：新用户可以快速上手复杂的 Claude 功能

## 6.2 skill 的组成部

一个完整的 skill 文件夹包含以下部分：

```
Plain Text
your-skill-name/            # skill 文件夹（必需）
├── SKILL.md               # 核心文件（必需）：包含 YAML 前置元数据和 Markdown 指令
├── scripts/               # 可选：可执行代码（Python、Bash 等）
│   ├── process_data.py
│   └── validate.sh
├── references/            # 可选：按需加载的参考文档
│   ├── api-guide.md
│   └── examples/
└── assets/                # 可选：模板、字体、图标等资源
    └── report-template.md
```

## 6.3 Skill vs MCP：厨房比喻

| 组件 | 类比 | 功能 |
| --- | --- | --- |
| **MCP** | 专业厨房 | 提供工具、食材和设备（连接外部服务） |
| **Skill** | 食谱 | 提供步骤指导和最佳实践（如何使用工具） |

**协同工作方式：**

- **MCP** 提供连接性：将 Claude 连接到您的服务（Notion、Asana、Linear 等）
- **Skill** 提供知识层：教授 Claude 如何有效使用这些服务，嵌入工作流程和最佳实践

## 6.4 为什么需要 Skill?

**没有 Skill 时的问题：**

- 用户连接 MCP 后不知道下一步该做什么
- 每次对话都从零开始，结果不一致
- 用户因提示方式不同而获得不同质量的结果
- 支持工单激增："如何用您的集成做 X?

**有了 Skill 后的优势：**

- 预构建的工作流在需要时自动激活
- 一致、可靠的工具使用
- 每次交互都嵌入最佳实践
- 降低集成的学习曲线

## 6.5 Skill 的核心设计原则

（1）渐进式披露（Progressive Disclosure）

Skill 采用三级系统来优化 Token 使用：

| 级别 | 内容 | 加载时机 | 目的 |
| --- | --- | --- | --- |
| **第一级** | YAML 前置元数据 | 始终加载到系统提示中 | 让 Claude 知道何时使用该 Skill |
| **第二级** | SKILL.md 正文 | 当 Claude 判断 Skill 与当前任务相关时 | 包含完整指令和指导 |
| **第三级** | 链接文件 | 仅在需要时导航和发现 | 详细文档和参考材料 |

优势：最小化 Token 消耗，同时保持专业化能力。

（2）可组合性（Composability）

Claude 可以同时加载多个 Skill。您的 Skill 应该能够与其他 Skill 良好协作，而不是假设它是唯一的能力来源。

**最佳实践：**

- 避免过于宽泛的触发条件
- 明确 Skill 的边界和适用范围
- 设计时考虑可能与其他 Skill 一起使用的情况

（3）可移植性（Portability）

Skill 在 Claude.ai、Claude Code、openclaw 以及国内外各个 vibe coding 工具、API 中工作方式完全相同。创建一次，即可在所有平台上使用，无需修改（前提是环境支持 Skill 的任何依赖项）。

## 6.6 常见的 Skill 用例类别

根据 Anthropic 的观察，常见的 Skill 用例分为三类：

（1）类别 1：文档与资源创建

用途：创建一致、高质量的输出，包括文档、演示文稿、应用、设计、代码等。

**真实示例**：frontend-design Skill

```
Plain Text
"创建具有高品质设计的独特、生产级前端界面。用于构建 Web 组件、页面、工件、海报或应用。"
```

**关键技术：**

- 嵌入风格指南和品牌标准
- 模板结构确保一致输出
- 最终确定前的质量检查清单
- 无需外部工具——使用 Claude 的内置功能

（2）：工作流自动化

用途：受益于一致方法论的多步骤流程，包括跨多个 MCP 的协调。

**真实示例**：skill-creator Skill

```
Plain Text
"创建新 Skill 的交互式指南。引导用户完成用例定义、前置元数据生成、指令编写和测试。"
```

**关键技术：**

- 带验证门的分步工作流
- 常见结构的模板
- 内置审查和改进建议
- 迭代优化循环

（3）MCP 增强

用途：增强 MCP 服务器提供工具访问的工作流指导。

**真实示例**：sentry-code-review Skill（来自 Sentry）

```
Plain Text
"使用 Sentry 通过其 MCP 服务器提供的错误监控数据，自动分析和修复 GitHub
```

```
Pull Request 中检测到的错误。"
```

**关键技术：**

- 按顺序协调多个 MCP 调用
- 嵌入领域专业知识
- 提供用户原本需要指定的上下文
- 常见 MCP 问题的错误处理

（3）定义成功标准

定量指标（目标值）：

| 指标 | 目标 | 测量方法 |
| --- | --- | --- |
| **触发准确率** | 90%的相关查询触发 Skill | 运行 10-20 个应触发 Skill 的测试查询，追踪自动加载 vs 需要显式调用的次数 |
| **工具调用效率** | 在 X 次工具调用内完成工作流 | 比较启用 Skill 前后的同一任务，计算工具调用次数和总 Token 消耗 |
| **API 失败率** | 每个工作流 0 次失败 API 调用 | 监控测试运行期间的 MCP 服务器日志，追踪重试率和错误恢复 |

定性指标:

- **用户无需提示下一步**：测试期间记录需要重定向或澄清的频率，询问 Beta 用户反馈
- **工作流无需用户纠正完成**：同一请求运行 3-5 次，比较输出的结构一致性和质量
- **跨会话结果一致**：新用户能否在最少指导下首次完成任务?

## 6.7 技术规范与文件结构

（1）Skill 文件结构规范

```
Plain Text
your-skill-name/                  # 文件夹命名规范
├── SKILL.md                     # 必需：主 Skill 文件
├── scripts/                     # 可选：可执行代码
│   ├── process_data.py
│   └── validate.sh
```

```
├── references/              # 可选：文档
│   ├── api-guide.md
│   └── examples/
└── assets/                  # 可选：模板等
    └── report-template.md
```

（2）关键命名规则

SKILL.md 命名：

- **必须**完全匹配 `SKILL.md`（区分大小写）
- 不接受任何变体（如 `skill.md`、`Skill.md`）

Skill 文件夹命名：

- ✅ **使用短横线连接（kebab-case）**：`notion-project-setup`
- ❌ **不要有空格**：`Notion Project Setup`
- ❌ **不要有下划线**：`notion_project_setup`
- ❌ **不要有大写字母**：`NotionProjectSetup`

禁止事项：

- 不要在 Skill 文件夹内包含 `README.md`
- 所有文档应放在 `SKILL.md` 或 `references/` 中
- 注意：通过 GitHub 分发时，您仍然需要仓库级别的 README 供人类用户阅读

YAML 前置元数据：最重要的部分

YAML 前置元数据是 Claude 决定是否加载您的 Skill 的方式。**务必正确编写**。

最小必需格式：

```yaml
---
name: your-skill-name
description: 它的功能。当用户要求[具体触发短语]时使用！
---
```

字段要求：

**name（必需）：**

- 仅使用短横线连接（kebab-case）
- 无空格或大写字母
- 应与文件夹名称匹配

**description（必需）：**

- **必须**包含两部分：
  - 它的功能（What）
  - 何时使用它（触发条件 When）
- 少于 1024 个字符
- 无 XML 标签（< 或 >）
- 包含用户可能说的具体任务
- 如有相关，提及文件类型

**license（可选）：**

- 如果开源 Skill 则使用
- 常见：MIT、Apache-2.0

**compatibility（可选）：**

- 1-500 个字符
- 指示环境要求：例如目标产品、需要的系统包、网络访问需求等

**metadata（可选）：**

- 任何自定义键值对
- 建议：author、version、mcp-server

```YAML
metadata:
  author: ProjectHub
  version: 1.0.0
  mcp-server: projecthub
```

# 第七部分：Skill 实战

> ***本章目标***：*学完这章，你能搜索、安装、创建并使用 Skills*

## 7.1 最简单的 Claude skill 示例

我们知道 skill 是 claude 的首创，claude 定义了 skill 规范已然成为行业的通用规范。我们先来开发了一个最简单的 claude skill 以便于理解 skill 原理。简单来讲，Claude 例通过目录及文件的组织及 SKILL.md 文件内容格式来规范 Skills。

我们在任意目录下再创建以下子目录，其中.claude\skills 为固定路径，字幕转 markdown 是当前 skill 名称:

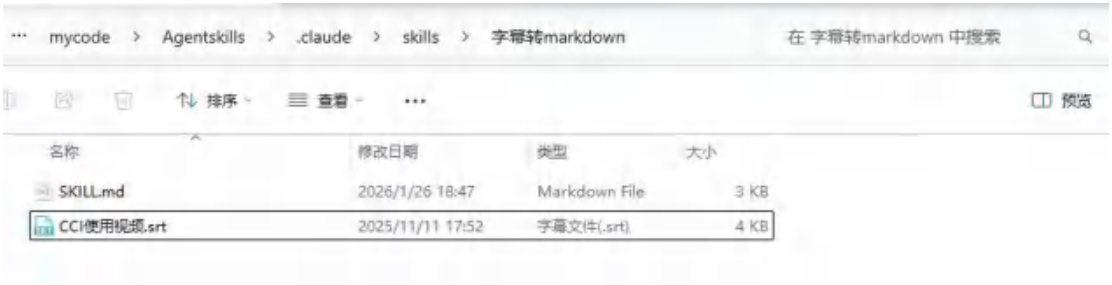
```JSON
.claude\skills\字幕转 markdown
```

在这个路径下创建 SKILL.md（注意大写 SKILL），打开 SKILL.md，输入以下 MarkDown 格式信息(两个"---"之间的内容为固定写法，name,description 两个单词不能改，下边的提示词可以按实际情况修改):

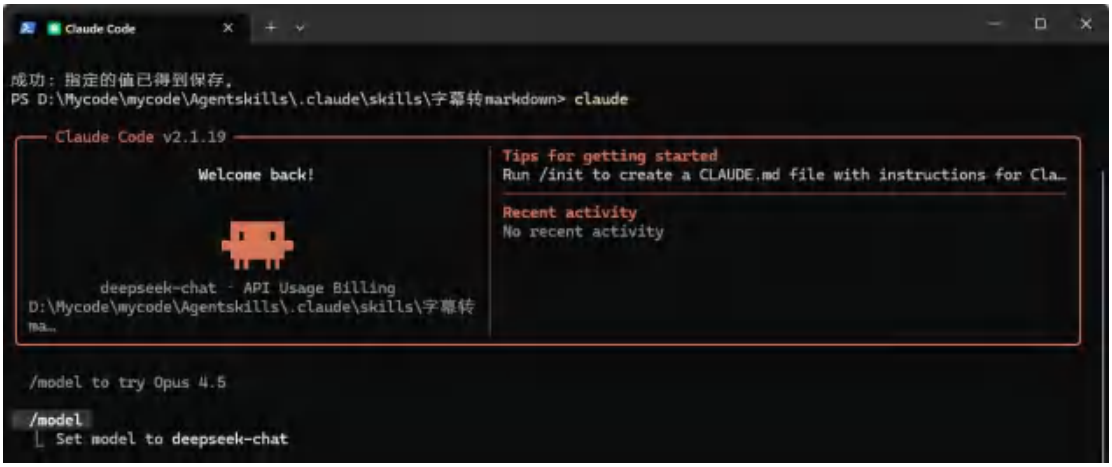
```Markdown
---
name: srt 字幕转 markdown 笔记
description: 把 srt 转成 markdown
---
# SRT 字幕转 Markdown 笔记提示词（专业版）
你是专业的字幕文本处理助手，核心任务是将 SRT 字幕文件完整转换为规范 Markdown 笔记，并保存到项目目录下的.md 文件中，严格遵循以下所有规则，不擅自修改、拓展或简化要求：

## 一、核心原则：内容完整性
1. 必须**逐字保留 SRT 字幕所有文字**，禁止任何形式的删减、总结、省略、改写或意译，包括语气词、重复表述、口语化表达等。
2. 保留必要的英文专有名词（如技术术语、人名、地名、品牌名、代码、网址等），其余内容使用中文规范书写，不擅自翻译英文专有名词。

## 二、段落结构规范
1. 整体仅划分 **2-3 个段落**，结构简洁清晰，不拆分过多细碎段落。
2. 第一段为引言（对应字幕开篇内容），**不添加任何标题**，直接以正文呈现。
3. 后续段落（1-2 个）各配一个标题，标题统一使用 `##` 二级标题格式（仅一个层级，禁止使用一级、三级及以上标题），标题需贴合对应段落核心内容，简洁凝练。

## 三、标点与排版要求
1. 为每句话补充合适的中文标点（句号、逗号、问号、感叹号等），修正 SRT 字幕中缺失、错误的标点，使语句通顺连贯。
2. 适当划分自然段，根据语义逻辑拆分长句、整合短句，避免段落过于冗长，同时不破坏原句语义关联。
3. 排版整洁，仅保留必要换行（段落间空一行，段内不随意换行），无多余空格、符号或格式标记。
```

```
## 四、输出要求
最终输出仅为转换后的 Markdown 笔记内容，无额外说明、注释或规则重申，直接交付可直接使用的 Markdown 文件。
```

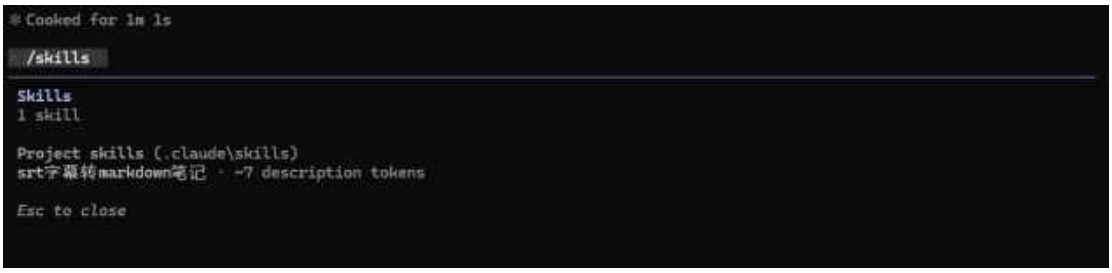
准备需要的其它文件，我们将一个 srt 文件也放到 SKILL.md 文件的同级目录下：




启动 claude，输入/model 选择 deepseek-chat 模型：




通过/skills 进入 skills，可以查看有哪些 skills。

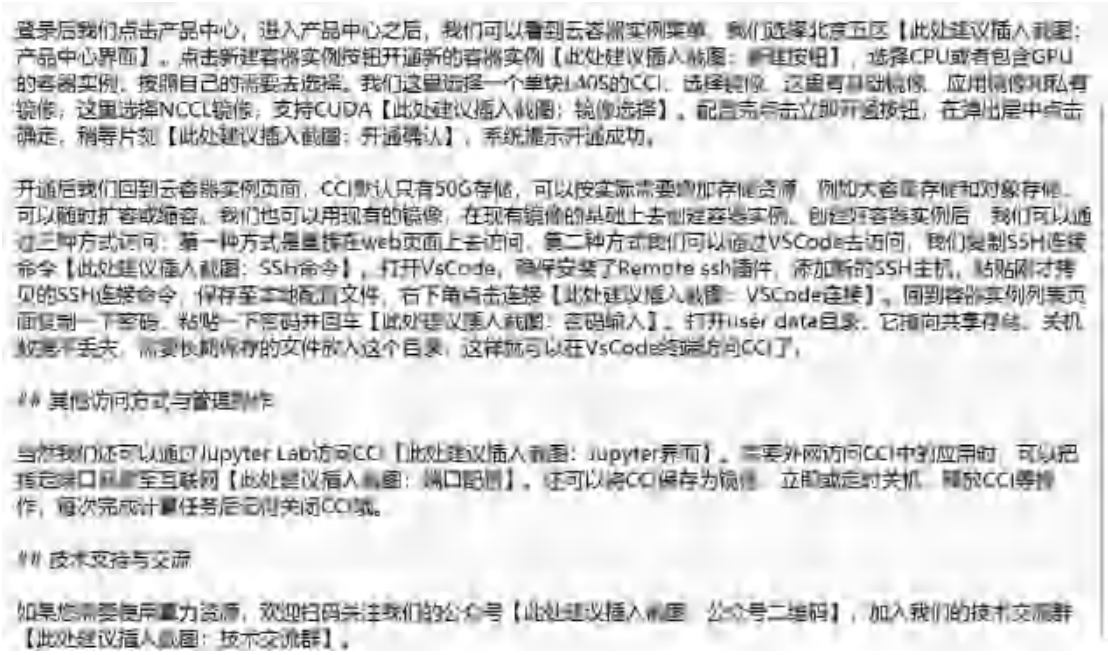



这就把 srt 内容读取出来了，还没完呢，我们不是要转成 MarkDown 吗？

通过/srt 字幕转 markdown 笔记，进入到这个 skill。先读取，后存为 MarkDown 文件。


我们可以看到同级目录下生成了 CCI 使用视频.md 文件。


打开 CCI 使用视频.md 看一眼：


登录后我们点击产品中心，进入产品中心之后，我们可以看到云容器实例菜单，我们选择北京五区【此处建议插入截图：产品中心界面】，点击新建容器实例按钮开通新的容器实例【此处建议插入截图：新建按钮】，选择CPU或者包含GPU的容器实例，按照自己的需要去选择。我们这里选择一个单块L40S的CCI，选择镜像，这里有基础镜像，应用镜像和私有镜像，这里选择NCCL镜像，支持CUDA【此处建议插入截图：镜像选择】，配置完点击立即开通按钮，在弹出层中点击确定，稍等片刻【此处建议插入截图：开通确认】，系统提示开通成功。

开通后我们回到云容器实例页面，CCI默认只有50G存储，可以按实际需要增加存储资源，例如大容量存储和对象存储。可以随时扩容或缩容。我们也可以用现有的镜像，在现有镜像的基础上去创建容器实例。创建好容器实例后，我们可以通过三种方式访问：第一种方式是直接在web页面上去访问，第二种方式我们可以通过VSCode去访问，我们复制SSH连接命令【此处建议插入截图：SSH命令】，打开VsCode，确保安装了Remote ssh插件，添加新的SSH主机，粘贴刚才拷贝的SSH连接命令，保存至本地配置文件，右下角点击连接【此处建议插入截图：VSCode连接】，回到容器实例列表页面复制一下密码，粘贴一下密码并回车【此处建议插入截图：密码输入】，打开user data目录，它指向共享存储，关机数据不丢失，需要长期保存的文件放入这个目录，这样就可以在VsCode终端访问CCI了，

## 其他访问方式与管理操作

当然我们还可以通过Jupyter Lab访问CCI【此处建议插入截图：Jupyter界面】，需要外网访问CCI中的应用时，可以把指定端口暴露至互联网【此处建议插入截图：端口配置】，还可以将CCI保存为镜像，立即或定时关机，释放CCI等操作，每次完成计算任务后记得关闭CCI哦。

## 技术支持与交流

如果您需要更多资源，欢迎扫码关注我们的公众号【此处建议插入截图：公众号二维码】，加入我们的技术交流群【此处建议插入截图：技术交流群】。

至此，我们用 Skill 成功生成了 MarkDown 格式文件。

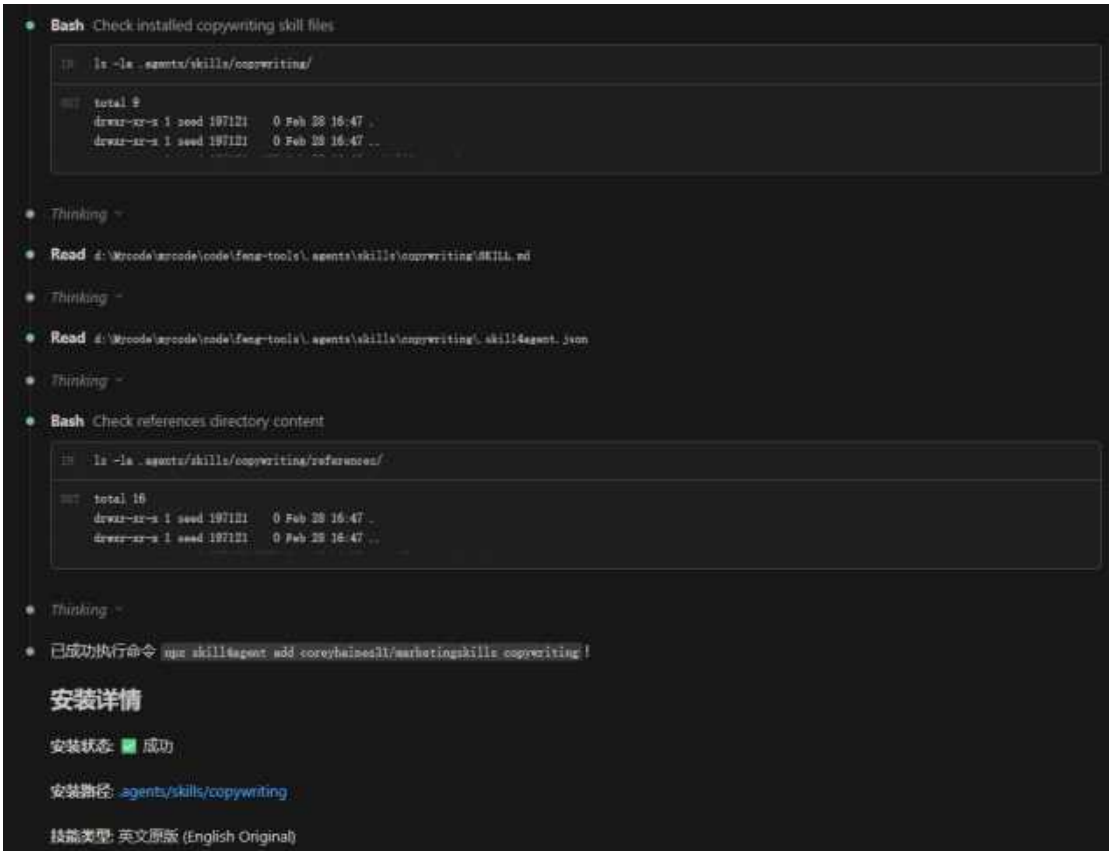
也可以直接用/plugin markplace add anthropics/skills 后，用自然语言来让 claude code 来创建 skill。

## 7.2 从资源站点安装现有 skill

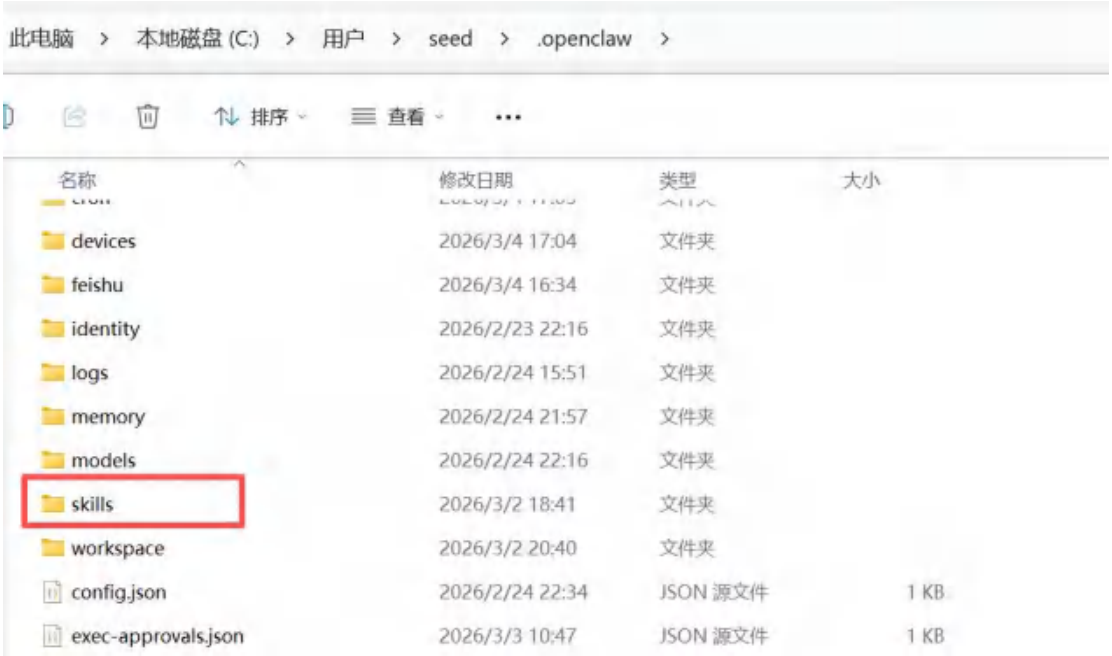
现在市面上有大量的 skill，可以直接安装，以供 claude code、openclaw 等平台使用。以下是一些常用的 skill 资源站点，每个 skill 说明文字里都有安装命令，一般是通过 nodejs 相关命令安装。以下是几个常用的 skill 资源站点。

- https://skill4agent.com/zh
- https://skills.sh/
- https://github.com/VoltAgent/awesome-openclaw-skills
- https://clawhub.com

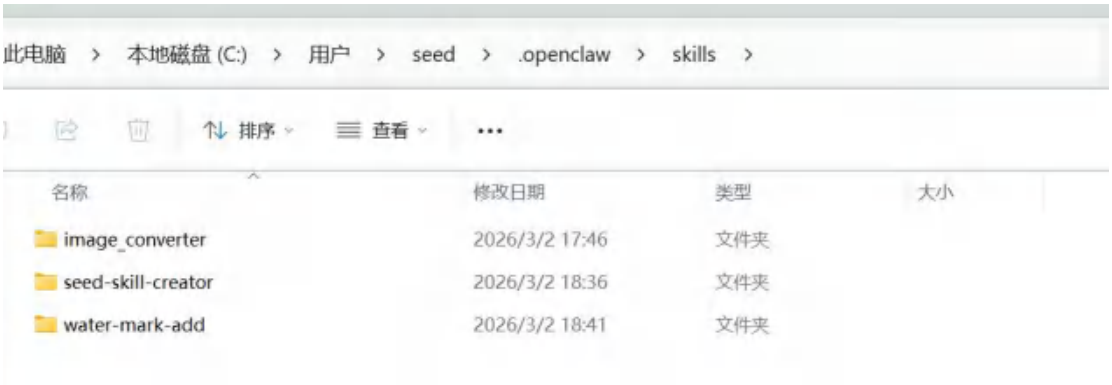
```PowerShell
npx skill4agent add coreyhaines31/marketingskills copywriting
```




如果没有安装命令的，可直接将 skill 文件包下载下来，放到.openclaw\skills 目录下:



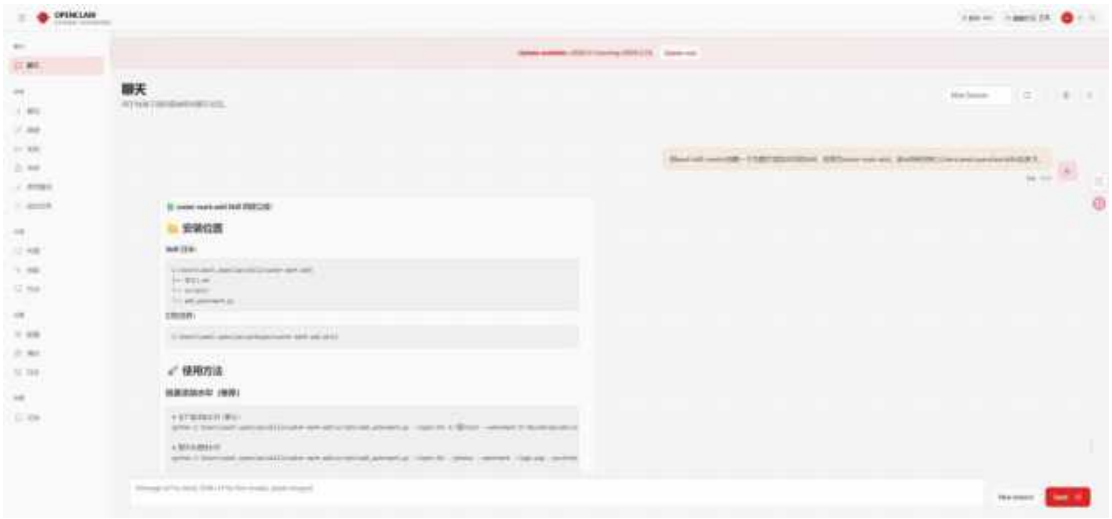

打开 skills 目录，我们可以看到已经安装的各个 skill 子目录，一个子目录对应一个 skill。




三、用 skill 创建 skill

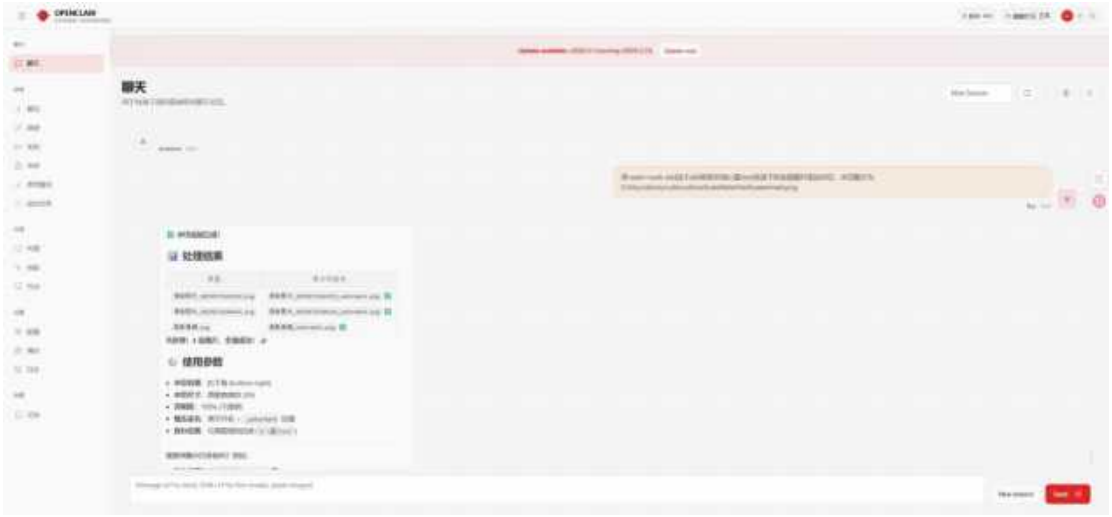
skill-creator 这个 skill 本身已经学会了 skill 的创建方法，可以安装这个 skill 后，通过自然语言让它来创建其它的 skill。可直接在 openclaw chat 窗口聊，让它创建 skill，例如，我们输入：

“用 seed-skill-creator 建一个为图片添加水印的 skill，名称为 water-mark-add，该 skill 保存到 C:\Users\seed\.openclaw\skills 目录下”

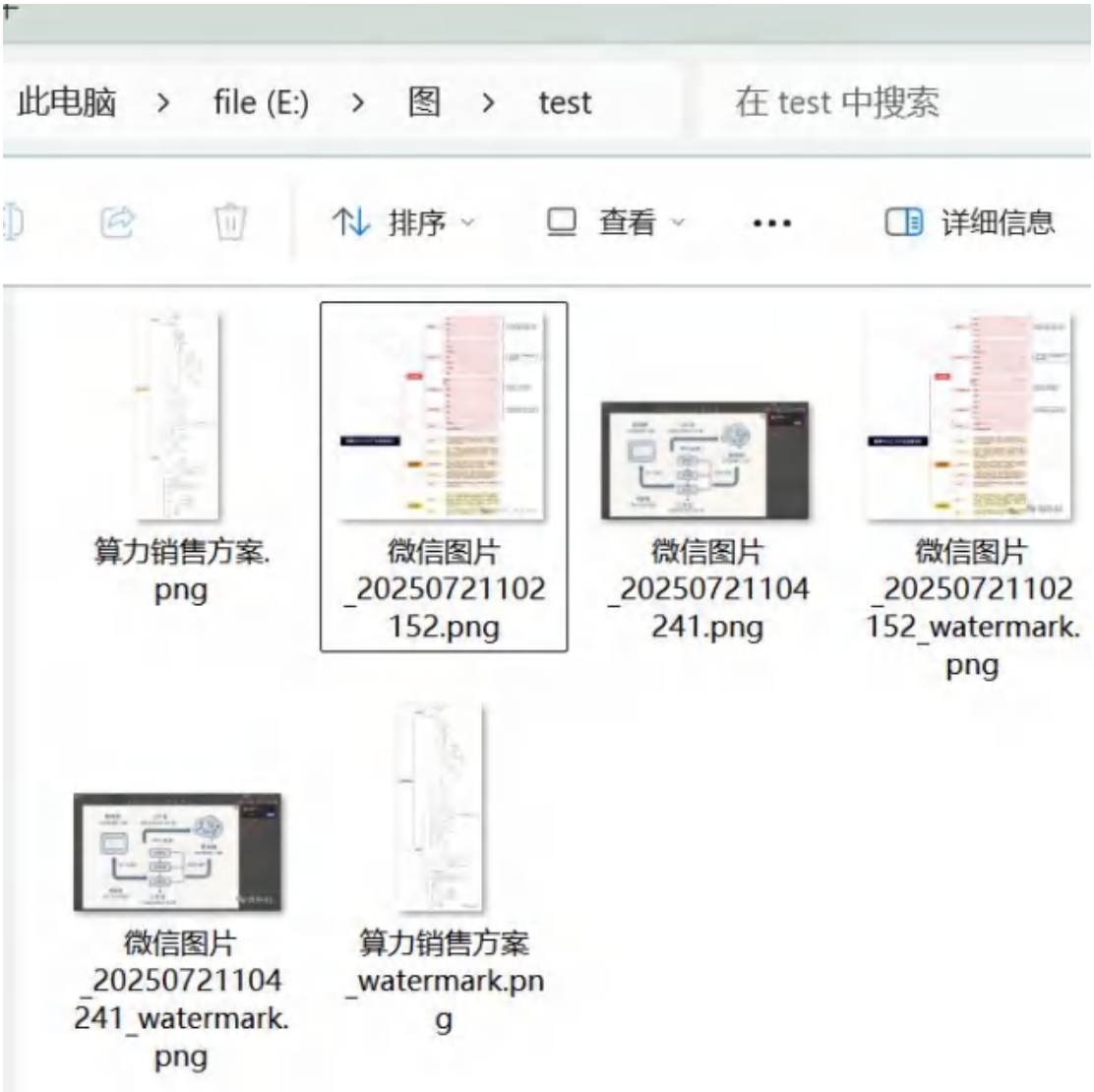


✓ **water-mark-add Skill 创建完成！**

使用 Skill：



效果：成功批量添加水印。


此电脑 > file (E:) > 图 > test
在 test 中搜索
排序
查看
详细信息
算力销售方案.png
微信图片_20250721102152.png
微信图片_20250721104241.png
微信图片_20250721102152_watermark.png
微信图片_20250721104241_watermark.png
算力销售方案_watermark.png

## 7.3 检查 Skills 是否可用

```
Plain Text
# 检查所有 Skills 的可用性
openclaw skills list --eligible

# 输出示例：
# NAME               STATUS      REASON
# ----               ------      ------
# daily-report       ✓ eligible  All requirements met
# github-pr-helper   ✗ missing   Missing tool: github_api
```

**eligible** 表示：

- • Skill 已正确安装
- • 所有依赖的 Tools 都已启用
- • 可以正常使用

7.4 查看 Skill 详情

```
Plain Text
openclaw skills info daily-report
```

**如果存在这个 skill，则输出如下示例**：

```
Plain Text
Name: daily-report
Version: 1.2.0
Path: ~/.openclaw/workspace/skills/daily-report
Requirements:
  - read_file ✓
  - write_file ✓
  - fetch_url ✓
Inputs:
  - date: 日期（可选，默认今天）
  - sources: 数据源（如：email,calendar）
```

> *否则提示：*
>
> ```
> PowerShell
> Skill "daily-report" not found. Run `openclaw skills list` to
> see available skills.
> ```

```
Tip: use `npx clawhub` to search, install, and sync skills.
```

## 7.4 openclaw skill 使用示例

一个 Skill 只能有一个基本的功能，一个任务往往需要多个 Skill 按先后顺序执行，才能达到最终的目标，类似之前的 coze,n8n 等工作流产品。

### 7.4.1 安装示例 Skill

我们以 `daily-report` 为例：

```
Plain Text
# 安装
不需要手动 install，Agent 会在需要时自动检索并拉取 daily-report

# 验证安装
openclaw skills list --eligible
```

### 7.4.2 调用 Skill

**方式一：在对话中直接调用**

```
Plain Text
用户：运行 daily-report 生成今天的日报
Agent：调用 daily-report Skill，生成日报
```

**方式二：使用命令**

```
Plain Text
openclaw agent --message "调用 daily-report 生成今天的日报"
```

**方式三：带参数调用**

```
Plain Text
openclaw agent --message "调用 daily-report，日期=2026-02-18，数据源
=email,calendar"
```

### 7.4.3 查看执行结果

Skill 执行后，会：

1. 显示执行日志

2. 返回执行结果
3. （可选）生成输出文件

```
Plain Text
✓ daily-report executed successfully
  Output: /home/user/reports/daily-2026-02-18.md
  Time: 3.2s
```

## 7.5 skill 更新与回退处理

skill 是活动，不是创建后就一成不变了，因为 skill 本身是一些目录和文件，是静态内容，它也是可以编辑的，openclaw 可以更新这些 skill 的内容。skill 平台也提供更新 skill 的命令。你也可以在 openclaw 中用自然语言跟它聊，让它更新指定 skill。

### 7.5.1 更新 Skills

```
Plain Text
# 更新指定 Skill
clawhub update daily-report

# 更新所有 Skills
clawhub update --all
```

### 7.5.2 回退或覆盖安装

```
Plain Text
# 强制覆盖当前目录中的同名 Skill
不需要手动 install，Agent 会在需要时自动检索并拉取 daily-report --
force
```

如果你要“停用”某个 Skill，建议在 OpenClaw 配置里将对应条目 `enabled: false`，而不是直接删除文件夹。

## 7.6 推荐 Skills 清单

**（1）核心必备技能（必装/基础前置）**

- **clawhub**：所有技能的搜索、安装、更新都通过它完成，是适配所有人的必装前置技能，为其他技能的使用提供基础支持。
- **skill creator/anthropics/skill-creator**：当现成技能无法满足需求时，可通过它创建符合最佳实践的新技能，也能优化现有技能；还可将个人常用流程（如日报生

成、发群、归档）封装成技能，一次沉淀、反复使用。

- **find-skills**：若需要某类技能但不想手动查找，安装后可让 AI 自动帮你筛选匹配的技能，节省查找时间。

**（2）效率工具类技能（提升办公/日常操作效率）**

- **agent-browser/agent browser**：传统 AI 仅能通过 API 获取数据，该技能让 AI 像拥有 Chrome 浏览器助手，可操作浏览器完成各类繁琐工作（dirty work），包括登录网站、抓取网课信息、填写表单、截图网页、导出网页为 PDF 等。
- **research**：安装后 OpenClaw 可直接上网查资料、搜新闻、找答案，摆脱对自身训练死数据的依赖，获取实时、最新信息。
- **shell**：让 OpenClaw 能在电脑上执行命令行操作，包括文件处理、脚本执行、命令行安装服务等，适配开发者或需要批量处理电脑文件的场景。
- **Cron Wake**：实现定时任务功能，例如每天早上推送天气、每周五下午提醒写周报等，可搭配其他技能提升时间管理效率。
- **Gmail 自动处理邮件**：支持邮件读取、归档、搜索，每天早上可自动扫描收件箱，筛选出需要回复的邮件，节省邮件处理时间。
- **Google Calendar**：谷歌日程服务，具备日程管理、提醒设置等功能，与 Cron Wake 搭配使用效果更佳，相当于拥有专属日程秘书，实时跟进日程安排。
- **pdf**：强大的文献资料处理工具，可对 PDF 文件进行增删改、拆分、合并、数据提取、翻译等操作，适配需要处理大量文献的场景。

**（3）内容创作类技能（辅助内容生产、知识管理）**

- **frontend-design**：适合需要 vibe coding 网站、制作产品网页的人群，可摆脱大模型生成内容的“AI 感”，解决网页设计千篇一律的问题。
- **notebooklm-skill**：可将日常看到的内容同步至 NotebookLM，构建个人专属知识库，在创作或查资料时，调用该技能进行内容增删改写，提升创作效率。
- **Copywriting**：根据用户需求进行内容创作，有效降低内容的“AI 味道”，让产出内容更自然、贴合需求。
- **jimliu/baoyu-skills**：AI 大神宝玉的套装技能，包含封面图制作、文案撰写、排版设计、自动发布公众号、自动发推特等十几个技能，一站式覆盖内容创作到发布全流程。
- **notion 双向同步**：自动整理 Notion 页面笔记、更新数据库，是创作者和项目经理的实用工具，实现知识高效管理。
- **obsidian**：可读写 Obsidian vault，协助整理笔记、打标签、建立知识关联，贴合 OpenClaw 本地优先理念，数据存储在本地，无需担心云端同步问题。

- **domain-name-brainstormer**：适合经常搭建网站、需要大量购买域名的人群，可生成创意域名选项，并检查域名注册可用性，帮助找到适配项目的完美域名。
- **Github**：开发者必备技能，可创建 H5、查看 PR、管理代码仓库，助力开发者高效开展开发工作。

**（4）社媒运营类技能（自动化运营、热点追踪）**

- **twitter-automation**：自动化运营推特，目前处于实验阶段，支持自动发布、点赞、删除帖子等功能，简化推特运营流程。
- **TwitterX**：具备社媒监控和发布功能，可监控热门话题、自动生成摘要并快速发帖，适合自媒体创作者追热点、积累素材。

**（5）设备联动类技能（拓展 AI 使用场景）**

- **Nodes**：将 AI 触角从电脑延伸至手机，支持手机截图、录屏、定位、摄像头推送通知等功能，解决不在电脑前的操作断点（如让 AI 识别手机验证码截图并自动填写）。
- **Spotify**：支持语音或文字控制音乐，可实现播放、切歌、查询歌词等操作，工作时可让 OpenClaw 充当 DJ，语音指令即可切换音乐风格。
- **home assistant**：对接家中智能设备（灯光、空调、窗帘、摄像头等），让 OpenClaw 成为语音管家，一句话即可控制各类智能家居，提升生活便捷度。

## 第八部分：资源站点

| 标题 | 说明 | URL |
| --- | --- | --- |
| openclaw 官网中文教程 | 完整的 API 参考、配置指南和架构说明 | https://docs.openclaw.ai/zh-CN |
| openclaw 官网 | openclaw 官网 | https://openclaw.ai/ |
| GitHub — openclaw/openclaw | 源代码、Issue 跟踪和社区贡献指南 (150k+ ⭐) | https://github.com/openclaw/openclaw |
| ClawHub 技能市场 | 发现、安装和分享 AI 技能插件 | https://clawhub.com |
| Getting Started — 官方入门指南 | 从零到第一次对话的最快路径 | https://docs.openclaw.ai/start/getting-started |

| | | |
|---|---|---|
| Discord 社区 | 与数万开发者和用户实时交流 | https://discord.com/invite/clawd |
| OpenClaw — Wikipedia | 维基百科词条，了解 OpenClaw 的历史和影响 | https://en.wikipedia.org/wiki/OpenClaw |
| ClawHub Skills 仓库 | 所有已发布技能的源码归档 | https://github.com/openclaw/skills |
| The Verge — OpenClaw: all the news about the trending AI agent | Comprehensive news hub tracking the AI agent that "actually does things" — reminders, forms, flight check-ins, and more | https://www.theverge.com/news/872091/openclaw-moltbot-clawdbot-ai-agent-news |
| GitHub Issues — Claude Opus 4.6 Support | OpenClaw GitHub：Claude Opus 4.6 支持请求和默认上下文令牌更新 | https://github.com/openclaw/openclaw/issues/12621 |
| GitHub Issue — 2026.2.14 更新后权限错误：missing scope operator.read | OpenClaw 2026.2.14 版本更新后出现权限范围错误的解决方案和讨论 | https://github.com/openclaw/openclaw/issues/16820 |
| GitHub Issue — Copilot 提供商模型列表更新：Claude Opus 4.6-fast、GPT-5.3-codex | 请求更新 Copilot 提供商支持最新模型，保持与 Copilot CLI 和 Proxy 功能同步 | https://github.com/openclaw/openclaw/issues/15014 |
| GitHub Issue — Claude Opus 4.6 支持请求 | 2026 年 2 月 5 日发布的 Claude Opus 4.6 模型支持请求，需要 SDK 版本升级到 0.73.0+ | https://github.com/openclaw/openclaw/issues/12621 |
| TechCrunch — OpenClaw 创始人 Peter Steinberger 加入 OpenAI | 重磅新闻：OpenClaw 创始人正式加入 OpenAI，负责"下一代个人代理"开发，OpenClaw 将作为开源项目继续由 OpenAI 支持 | https://techcrunch.com/2026/02/15/openclaw-creator-peter-steinberger-joins-openai/ |
| CNBC — OpenClaw 创始人 | CNBC 确认报道：Sam Altman 发推 | https://www.cnbc.com/2026/02/15/openclaw- |

| | | |
|---|---|---|
| Peter Steinberger 加入 OpenAI，Sam Altman 确认 | 确认 OpenClaw 创始人加入 OpenAI，开源项目将由 OpenAI 基金会继续支持 | creator-peter-steinberger-joining-openai-altman-says.html |
| Peter Steinberger — OpenClaw, OpenAI and the future（创始人公告） | tl;dr: 我加入 OpenAI 负责个人 Agent，OpenClaw 将移交基金会保持开源独立。创始人第一视角全文声明 | https://steipete.me/posts/2026/openclaw |
| Reuters — OpenClaw 创始人加入 OpenAI，项目转为基金会 | Sam Altman 确认 Peter Steinberger 加入 OpenAI，OpenClaw 将"生活在基金会中"保持开源 | https://www.reuters.com/business/openclaw-founder-steinberger-joins-openai-open-source-bot-becomes-foundation-2026-02-15/ |
| CNET — The Year of the Agent: OpenAI Strikes Deal With OpenClaw Founder | Lex Fridman 播客后 Zuckerberg 与 Altman 均抛出 offer，Steinberger 最终选择 OpenAI 全过程报道 | https://www.cnet.com/tech/services-and-software/openai-strikes-deal-with-openclaw-founder/ |
| Business Insider — OpenAI Hires OpenClaw Creator: Praise, Memes & Rivalry | 科技圈的反应：Steinberger 加入 OpenAI 后的赞誉、梗图和竞争话题报道 | https://www.businessinsider.com/openais-openclaw-hire-sparks-praise-memes-rivalry-chatter-2026-2 |
| Parameter.io — OpenClaw Developer Picks OpenAI After Rejecting Meta | Meta vs OpenAI 争夺战报道，+ Moonshot AI 于同日发布 Kimi Claw（浏览器版 OpenClaw）分析 | https://parameter.io/openclaw-developer-picks-openai-after-rejecting-meta-acquisition-deal/ |
| The Register — OpenAI Grabs OpenClaw Creator Peter Steinberger | The Register 报道 Altman 确认 Steinberger 加入 OpenAI，OpenClaw 将"作为开源项目在基金会中继续生存" | https://www.theregister.com/2026/02/16/open_ai_grabs_openclaw |
| Tech.eu — | Steinberger："我加入 OpenAI 为所 | https://tech.eu/2026/02/ |

| | | |
|---|---|---|
| Austrian Creator of Viral OpenClaw Joins OpenAI | 有人带来 Agent。OpenClaw 正在成为基金会：开放、独立、刚刚起步。" | 16/austrian-creator-of-viral-openclaw-joins-openai/ |
| OpenClaw Newsletter — 2026-02-15（v2026.2.14：50+ 安全修复） | 官方周刊：v2026.2.14 情人节发布，50+ 安全修复，Telegram Polls、Slack/Discord DM 策略、Matrix 语音消息改进 | https://buttondown.com/openclaw-newsletter/archive/openclaw-newsletter-2026-02-15/ |
| gradually.ai — OpenClaw Changelog（2026 年 2 月） | v2026.2.14 ~ v2026.2.17 完整更新日志：Telegram Polls、Slack/Discord DM 策略、新权限系统等功能一览 | https://www.gradually.ai/en/changelogs/openclaw/ |
| Reddit — OpenClaw 2026.2.15 更新：重大功能与改进 | Reddit 社区 r/aicuriosity 精华帖：2026 年 2 月 16 日新版本发布，AI Agent 平台核心功能升级详解 | https://www.reddit.com/r/aicuriosity/comments/1r6536z/openclaw_2026215_update_major_features_and/ |
| Hacker News — OpenClaw (ClawdBot) joins OpenAI（讨论帖） | HN 社区讨论：OpenClaw 加入 OpenAI 事件，n8n 类比、设置摩擦、小企业自动化的真实用户视角 | https://news.ycombinator.com/item?id=47027907 |
| OpenClaw Newsletter — 2026-02-19（安全争议 + v2026.2.17 上游同步） | 官方周刊：HN 252 评论安全分析引发热议，Fork 维护者同步 55 个安全补丁，Telegram/Slack 新功能播报 | https://buttondown.com/openclaw-newsletter/archive/openclaw-newsletter-2026-02-19/ |
| GitHub Issue — Claude Sonnet 4.6 支持请求 | 2026-02-17 Anthropic 发布 Sonnet 4.6，OpenClaw 同日提 Issue 并快速在 v2026.2.17 修复 | <https://github.com/openclaw/openclaw/issues/19529 |
| NewReleases.io — OpenClaw v2026.2.19 发布说明 | v2026.2.19 新功能：设备配对卫生流程（device.pair.remove）、openclaw devices remove / clear 命令 | https://newreleases.io/project/github/openclaw/openclaw/release/v2026.2.19 |

| | | |
|---|---|---|
| Reddit r/AI_Agents — OpenAI just hired the OpenClaw creator（社区反应） | Clawdbot 创始人（以 Claude 命名）被 OpenAI 而非 Anthropic 雇用，Reddit 科技圈热议讨论 | https://www.reddit.com/r/AI_Agents/comments/1r6xksq/openai_just_hired_the_openclaw_creator/ |
| OpenClaw Newsletter — 2026-02-17（NanoClaw 安全修复 + v2026.2.17 发布） | 官方周刊：NanoClaw 安全修复重磅发布，Adversa AI 推出 SecureClaw，v2026.2.17 完整功能播报 | https://buttondown.com/openclaw-newsletter/archive/openclaw-newsletter-2026-02-17/ |
| OpenClaw Newsletter — 2026-02-20（v2026.2.19 Apple Watch + 安全加固） | 官方周刊：v2026.2.19 新增 Apple Watch 伴侣 App（收件箱 UI、通知中继、Gateway 指令）+ 史上最大安全加固补丁，含 openclaw acp --token-file/--password-file 和 exec 沙箱边界修复 | https://buttondown.com/openclaw-newsletter/archive/openclaw-newsletter-2026-02-20/ |
| MeetNeura — OpenClaw 2026.2.21: Gemini 3.1 & GLM-5 Integration | v2026.2.21 功能全解析：Gemini 3.1 和 GLM-5 正式接入，Token 计数修复、记忆管理改进、嵌套子 Agent 处理优化 | https://blog.meetneura.ai/openclaw-2026-2-21/ |
| OpenClaw v2026.2.19 深度解读：Apple Watch 功能 + 大规模安全加固 | v2026.2.19 深度解读：Apple Watch 伴侣 App 让 AI 自动化延伸到手腕，同时这是项目史上安全补丁最密集的一次发版 | https://openclawlaunch.com/news/openclaw-v2026-2-19-apple-watch-security |
| NewReleases.io — OpenClaw v2026.2.21 发布说明（Gemini 3.1 + Doubao） | v2026.2.21 完整 changelog：Gemini 3.1、GLM-5、Doubao（Volcano Engine/BytePlus）提供商接入，每账户/频道 defaultTo 路由回退机制 | https://www.gradually.ai/en/changelogs/openclaw/ |
| CyberSecurity News — | OpenClaw 215K+ Stars 项目发布 | https://cybersecuritynews.com/openclaw- |

| | | |
|---|---|---|
| OpenClaw 2026.2.23：安全加固 + Claude Opus 4.6 支持 | v2026.2.23：修复多个安全漏洞，新增 Claude Opus 4.6 支持，SSRF 策略重大变更——浏览器默认切换到 "trusted-network" 模式，升级前须用 `openclaw doctor --fix` 迁移旧配置 | 2026-2-23-released/ |
| NewReleases.io — OpenClaw v2026.2.22-beta.1 发布说明 | v2026.2.22-beta.1 细节：Gateway 重启循环边界情况修复，bootstrap 检测显式化，`openclaw.mjs -> dist/entry.js` 引导路径锁收紧，重启回归覆盖率提升 | https://newreleases.io/project/github/openclaw/openclaw/release/v2026.2.22-beta.1 |
| Releasebot — OpenClaw 发版追踪：2026 年 2 月更新日志 | v2026.2.23/2026.2.24 自动更新追踪：openclaw update --dry-run 预览命令、内置自动更新器（update.auto. 默认关闭）、多语言停止指令扩展（ZH/JP/AR 等） | https://releasebot.io/updates/openclaw |
| OpenClaw Newsletter — 2026-02-25（v2026.2.24：多语言停止指令 + 226K Stars） | 官方周刊：v2026.2.24 扩展自动停止短语（支持 ES/FR/ZH/HI/AR/JP/DE/PT/RU 多语言），新增 `do not do that` 作为停止触发词；Homebrew 30 天内 3,284 次安装，仓库突破 226,887 Stars | https://buttondown.com/openclaw-newsletter/archive/openclaw-newsletter-2026-02-25/ |
| Reddit r/myclaw — OpenClaw v2026.2.25 正式发布 | v2026.2.25 关键更新：Heartbeat DM 投递恢复、子 Agent 投递全面重构、Slack 线程会话修复、跨频道 Reaction 认证加固 | https://www.reddit.com/r/myclaw/comments/1rf4zr0/openclaw_just_launched_v2026225/ |
| PatchBot — OpenClaw 补丁说明（含 v2026.2.26 External Secrets Management） | v2026.2.26 重大新功能：External Secrets Management 完整工作流（audit/configure/apply/reload），运行时快照激活、严格 target-path 验证、安全迁移清洗 | https://patchbot.io/ai/openclaw |
| Software Mirrors | 第三方软件镜像站收录最新 | https://www.software- |

| | | |
|---|---|---|
| — OpenClaw 2026.2.26（版本信息与下载） | OpenClaw 2026.2.26，含版本历史和下载链接，便于追踪版本更新 | mirrors.com/products/openclaw-formerly-moltbot-clawd-bot |
| New OpenClaw v2026.2.26: Secrets, Browser Control, Multi-DM & Android | Reddit r/LocalLLM 社区对最新版本的详细讨论，涵盖 secrets 改进、浏览器控制优化、多 DM 支持和 Android 适配等核心变化。 | https://www.reddit.com/r/LocalLLM/comments/1rimve1/new_openclaw_release_version_2026226_way_less/ |

## 关于我们

九章云极 DataCanvas 是领先的人工智能基础设施与智算云提供商、国家级专精特新重点“小巨人”企业，构建了完整的 AIDC 技术栈、智算操作系统及产业链。旗下九章智算云（Alaya NeW Cloud）和九章智算操作系统（Alaya NeW OS），面向 AI 训练与推理提供高性能计算、云服务及 AI 软件，赋能广大开发者与企业客户。

作为普惠算力倡导者，公司牵头全球首个算力按“度”计量标准，具备万 P 级算力储备，是数字中国 AI 基建的核心力量。更多信息请访问 DataCanvas.com 与 AlayaNeW.com。