第1部分

基础入门：初识OpenClaw（深入核心，零门槛上手）

OpenClaw是一个本地优先、开源的AI智能体执行框架，核心是让大模型动手操作计算机、执行任务，形成“感知、认知、决策、行动、记忆、反思”的完整闭环。

“千里之行，始于足下”，在学习OpenClaw应用之前，我们先夯实基础，帮助零门槛的朋友快速上手。下面概括一下OpenClaw的核心原理。

一、核心架构（5大模块）

OpenClaw采用分层模块化设计，所有模块由本地Gateway（网关）统一调度。

1. Gateway 核心层（总调度）

- 本地常驻进程（默认本地Gateway地址ws://127.0.0.1:18789）：这个地址并非绝对的，它是否有效取决于 OpenClaw 的版本、部署方式和配置规则。
- 负责：会话管理、路由转发、工具编排、权限校验、本地数据存储。
- 所有模块的交互都通过它中转，确保安全与可控。

2. Channel 交互层（输入/输出）

- 对接微信、飞书、Telegram、Discord 等通信渠道。
- 接收用户自然语言指令，返回执行结果。

3. 大语言模型（Large Language Model，LLM）决策层（大脑）

- 接入 GPT-4o、Claude、DeepSeek、Ollama 等模型
- 核心工作如下。

  意图理解：解析模糊指令，提取目标与约束。

  任务拆解：把复杂的任务拆成原子化步骤。

  工具决策：判断调用哪些工具/技能。

  异常处理：自动追问、重试、调整方案。

4. Tools 执行层（手脚）

- 基于机器人流程自动化（Robotic Process Automation，RPA）技术，直接操作系统、软件、API。
- 包含：文件操作、终端命令、浏览器自动化、定时任务、Webhook（推送）等。
- 支持沙箱隔离与权限管控，避免误操作。

5. Memory 记忆层（记忆库）

- 所有配置、记忆、历史以纯文本/Markdown（极简的排版语法）的形式在本地存储。
- 双层结构：短期上下文 + 长期偏好/任务记录。
- 支持版本控制、迁移与备份等功能。

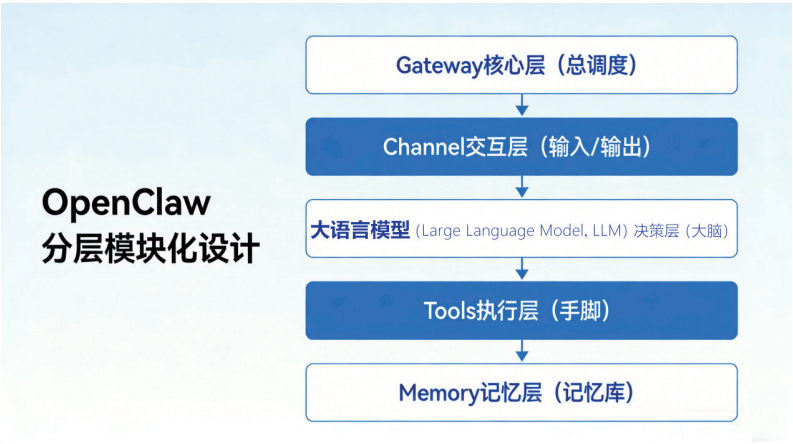
二、标准执行流程

（1）用户发指令：通过通信渠道发送自然语言（比如整理上周报告并生成周报）。
（2）Gateway接收：加载记忆与上下文，路由到对应智能体。
（3）LLM规划：理解意图→拆解子任务→生成工具调用方案。
（4）执行操作：Gateway调用Tools执行本地/云端操作。
（5）结果回传：执行结果返回LLM，整理成人类可读反馈。
（6）返回+记忆：结果发给用户，同时更新记忆，支持多轮/跨会话任务。
复杂的任务会自动循环多轮，直到完成或终止。

三、关键技术机制

- 本地优先：数据全在本地，不上传云端，隐私可控。
- 模型无关：自由切换云模型/本地模型，不被单一厂商绑定。
- 微内核+插件：高度可扩展，自定义Skills（技能）/Tools（工具）。
- 心跳调度：定时自动执行预设任务（如每日邮件汇总）。
- 沙箱隔离：执行环境隔离，防止系统风险。

总之，OpenClaw是用Gateway调度、用LLM决策、用Tools执行、用Memory记忆，把只会聊天的AI变成会干活的数字员工。OpenClaw的5大核心架构如下图所示。



第1章　OpenClaw定义与学习基础

OpenClaw是一款典型的、开源的、本地优先的AI Agent（人工智能体），而且是当前最主流的执行型Agent框架之一，让AI从“能说”变成“会做”。OpenClaw也是用于开发、部署或支持AI Agent运行的工具、框架或平台，为AI Agent的实现提供技术支持。

1.1　AI Agent的发展与应用场景为什么需要OpenClaw?

传统的AI助手（如ChatGPT）能够生成高质量的文本内容，但它们只能给出建议或答案，无法直接帮你执行操作。比如当你请求“帮我整理桌面文件”时，传统AI会建议你“可以创建文件夹，然后分类整理”，但它无法真正帮你创建文件夹、移动文件。而AI Agent的出现彻底改变了这一局面，它能够理解你的意图，调用各种工具，真正帮你把事情做成。

AI Agent主要经历了规则驱动阶段、数据驱动阶段和自主智能阶段，具体阐述如下。

规则驱动阶段（早期）：依赖预定义的逻辑与脚本，功能局限于特定场景，如简单的AI客服机器人。

数据驱动阶段（中期）：基于机器学习模型（如深度学习），可处理复杂的数据（如推荐系统、语音识别），但缺乏自主决策能力。

自主智能阶段（当前）：融合LLM（大语言模型）与多模态能力，具备环境感知、任务规划、工具调用及持续学习能力（如AutoGPT、Meta AI Agent），可自主完成复杂的目标。

而OpenClaw作为AI Agent中的一员，是2026年开源社区最活跃的AI Agent项目之一。它不仅是一个聊天机器人，而且是一个真正能替你执行操作的AI助理。与传统AI的本质区别在于：传统AI告诉你怎么做，而OpenClaw帮你直接完成，这就是需要OpenClaw的主要原因。

当前AI Agent虽然具备自主决策、感知环境的基础能力，但在落地执行、隐私安全、场景适配等方面还存在明显瓶颈，而OpenClaw作为AI Agent的“执行引擎”，恰好可以弥补这些短板，成为AI Agent从“能思考”到“能实干”的关键支撑。其核心原因可概括为以下4点。

解决AI Agent只说不做的核心痛点：多数AI Agent本质上是“语言交互工具”，擅长理解指令、输出执行步骤，但无法直接操作系统、软件或处理本地任务。比如，AI Agent能告知你如何整理客户数据，却不能实际打开计算机文件夹进行操作，而OpenClaw通过Tools（执行层）的RPA技术，赋予了AI Agent动手能力，可直接操作文件、终端、浏览器等，实现“自然语言指令→自动执行→结果反馈”的完整闭环，真正把AI Agent的想法转化为实际行动。

破解AI Agent的隐私与安全困境：普通AI Agent多依赖云端部署，用户需将数据上传云端，存在隐私泄露风险；同时部分AI Agent缺乏权限管控，执行操作时易出现误删文件、修改数据等问题。而OpenClaw坚持本地优先原则，所有数据、记忆均存储在本地，无须上传云端，从根源上保障隐私安全；同时通过Gateway权限校验、沙箱隔离机制，对执行操作进行严格管控，降低误操作风险，解决AI Agent在安全治理上的短板。

弥补AI Agent的能力局限，提升可落地性：AI Agent普遍存在长链条任务规划不足、记忆碎片化、多场景协作困难等问题。比如当任务步骤超过5层时，AI Agent决策准确率骤降，且易遗忘关键信息，无法实现跨会话协作。OpenClaw通过分层模块化架构，让LLM决策层负责复杂任务的拆解与异常处理，Memory（记忆层）实现短期上下文与长期偏好的系统化管理，Gateway负责多模块协同调度，有效解决AI Agent的规划、记忆与协作瓶颈，让复杂任务的落地成为可能。

打破AI Agent的厂商绑定与场景局限：多数AI Agent依赖特定大模型，无法自由切换，且适配场景有限；而OpenClaw具备“模型无关”特性，可自由接入GPT-4o、Claude、DeepSeek、Ollama等

各类云模型与本地模型，不被单一厂商绑定；同时支持微内核+插件扩展，可自定义技能，对接微信、飞书等各类通信与办公工具，适配个人办公、企业自动化等多场景，让AI Agent的应用范围大幅拓展，降低落地门槛。

简言之，AI Agent是能思考的大脑，而OpenClaw是“能执行的躯体+能协同的中枢”，二者结合可让AI Agent真正摆脱纸上谈兵的局限，成为可落地、可信赖、高适配的AI数字员工，这也是有了AI Agent后，仍需要OpenClaw的原因所在。

1.2 OpenClaw的核心定位：开源自托管的AI助手网关

OpenClaw采用分层式微服务架构，核心分为4个层级，各层级职责清晰、耦合度低，便于扩展与维护，如表1-1所示。

表 1-1 OpenClaw 架构层级和核心功能

架构层级	核 心 功 能
接入层	负责接收用户端请求（Web界面、API、CLI等），同时对接各类AI助手的API接口，实现请求的双向转发与格式转换
处理层	作为网关的核心业务逻辑层，实现智能请求路由、负载均衡、权限校验、日志审计、成本统计等功能。处理层内置的路由引擎可基于规则模型与机器学习算法，自动选择最优AI助手处理用户请求，同时支持自定义路由策略
存储层	负责存储网关配置信息、用户交互日志、本地AI模型数据等，支持本地磁盘、NAS、私有云存储等多种存储方式。存储层采用加密存储机制，保障数据的完整性与保密性，同时支持日志审计与追溯功能
扩展层	通过Skills插件化架构支持功能扩展，开发者可开发AI助手对接插件、自定义处理逻辑插件、交互界面插件等，快速丰富网关功能。扩展层支持热插拔机制，无须重启网关即可完成插件的安装与卸载

OpenClaw的开源生态非常活跃，官方维护的Clawhub平台已汇聚数千个Skills插件，用户可以根据自己的需求自由组合，打造定制化的AI助手。

OpenClaw被称为开源自托管的AI助手网关，核心原因在于其在开源协议、自托管部署、AI网关定位三个核心维度完全契合这一定位，三者相互支撑，构成了其区别于其他AI工具的核心身份与核心价值。

（1）开源特性保障了其透明性与可定制性，用户无须依赖单一厂商，可根据需求自由优化功能。

（2）自托管特性保障了隐私安全与数据主权，解决了用户对敏感数据泄露的担忧，适配隐私要求较高的个人与企业场景。

（3）网关核心定位实现了AI能力的统一接入与高效执行，打破了模型、渠道、工具之间的壁垒，让AI真正落地到实际操作场景中，成为可执行、可依赖的个人/企业AI助手中枢。

1.3 OpenClaw与其他AI Agent工具的区别

市场上的AI Agent产品众多，OpenClaw与其他主流产品相比有着独特的定位和优势。

OpenClaw的核心优势在于以下几点。

多平台即时通信集成：通过Gateway，可以同时接入WhatsApp、Telegram、Discord、微信、钉钉、飞书等多个平台

自我进化能力：可以基于运营数据自主分析、调整策略、更新Playbook（标准化操作手册/流程指南），实现越用越聪明。

完全开源可控：代码完全开放，用户可以自行部署，数据完全可控。

丰富的Skills生态：Clawhub（https://clawhub.ai/）提供数千种Skills插件，覆盖各行各业的使用场景。

表1-2是OpenClaw与其他AI Agent的区别。

表 1-2　OpenClaw 与其他 AI Agent 的区别

特 性	OpenClaw	ChatGPT Plugins	AutoGPT
部署方式	本地/云端灵活部署	仅云端服务	本地运行
平台集成	多平台即时通信集成	有限第三方集成	需自行配置
自我进化	支持定时复盘和策略更新	不支持	基础支持
开源程度	完全开源	闭源	部分开源
技能扩展	Clawhub生态丰富	官方插件有限	需手动开发

1.4　学习OpenClaw的必备基础

OpenClaw是基于Node.js/TypeScript开发的工具软件，核心依赖多语言工具链。要学习好OpenClaw，需要具备一定的基础知识，但不需要成为技术专家。另外，还需要满足一定的系统和硬件要求。

1. 基础知识要求

基础知识要求主要包括：基础命令行操作能力（能够执行简单的终端命令）、了解基本的编程概念（变量、函数、文件路径等）、对AI和Agent概念有初步了解。

2. 技术环境要求

操作系统：Windows 10/11、macOS、Linux（Ubuntu 22.04 LTS）；Node.js 22+；PNPM（Node.js包管理器）；Git版本控制工具；Docker等。

另外，在环境部署方面需要具备一定的技术核心能力，具体如下。

本地部署：会安装Node.js 22+、配置环境变量、解决依赖冲突。

容器化部署：了解Docker基础命令，能通过容器快速部署OpenClaw。

云部署：熟悉阿里云/腾讯云基础操作，能配置服务器安全组、开放端口。

3. 硬件建议

硬件建议：个人测试，内存4GB以上；团队使用，内存8GB以上；最好是MAC硬件。

注意：OpenClaw对系统环境有明确要求，Linux/macOS为推荐环境，Windows需使用WSL2避免路径与权限问题。

4. AI 与大模型基础

OpenClaw是大模型智能体框架，需掌握基础的AI概念与实战能力。

基础原理：理解大语言模型（LLM）的生成逻辑、上下文窗口（Context Window）、Token限制等。

主流模型：熟悉OpenAI（GPT-3.5/GPT-4o）、Claude、DeepSeek、Kimi、Ollama本地模型的特点与适用场景。

API调用：会申请API Key、理解请求参数（模型名称、精准度、最大Token）、处理响应结果。

5. OpenClaw 的二次开发与企业级应用（可选）

要对OpenClaw进行二次开发与企业级应用，需要掌握以下技术。

1）网络与API开发

核心协议：掌握HTTP/HTTPS、RESTful API设计规范、Webhook回调机制。

实战能力：能开发第三方API对接技能（如调用汽车行业数据接口、新能源电池参数接口等），处理接口认证、错误重试。

2）自动化与脚本开发

工作流编排：编写脚本实现多任务自动化（如定时同步知识库、批量处理文件等）。

网页自动化：使用Playwright/Puppeteer开发网页爬取、表单提交技能。

3）问题排查与性能优化

日志分析：读懂OpenClaw运行日志，定位API错误、技能调用失败问题。

性能优化：调整大模型参数、优化技能执行逻辑，提升任务处理速度。

1.5 OpenClaw的学习计划

结合新手学习规律，建议大家制订分阶段学习计划，确保高效掌握核心能力。

入门阶段：1～2周

目标：完成基础环境搭建，能正常运行OpenClaw控制台。

（1）安装Node.js 22+、PNPM、Git、VS Code。

（2）克隆OpenClaw仓库，完成依赖安装与API Key配置。

（3）熟悉控制台指令，完成基础任务测试（如写代码、查资料）。

（4）掌握核心命令行操作，解决基础部署问题。

进阶阶段：2～4周

目标：理解核心架构，开发简单的自定义技能。

（1）学习TypeScript/JavaScript基础语法，读懂OpenClaw核心配置。

（2）理解五大核心模块与工作流，熟悉Memory存储机制。

（3）开发简单的技能（如本地文件读写、天气查询等）。

（4）配置多通信渠道（如Telegram机器人）。

精通阶段：1～3个月

目标：实现二次开发，构建企业级智能体。

（1）阅读OpenClaw源代码，理解核心逻辑与模块交互。

（2）开发复杂的技能（如RAG知识库、自动驾驶数据解析等）。

（3）实现Docker容器化部署与云服务器运维。

（4）贡献社区代码，编写技术教程或插件文档。

1.6 本书学习路线与实战规划

本书采用“理论+实战”的学习路径，帮助读者从零基础入门到能够独立部署和优化OpenClaw。建议读者按照下面的思路学习本书。

第一阶段：认识与部署

（1）理解AI Agent的核心概念。

（2）掌握OpenClaw的架构设计。

（3）完成开发环境搭建和安装部署。

第二阶段：配置与连接

（1）初始化配置和Onboarding（初始化引导）。

（2）绑定AI模型（Claude、GPT、Ollama等）。

（3）接入各种通信平台（Telegram、Discord等）。

（4）掌握命令行工具的使用。

第三阶段：功能扩展

（1）安装和使用Skills扩展能力。

（2）配置记忆系统让Agent记住重要信息。

（3）设置定时任务实现自动化运行。

（4）掌握多Agent协作方法。

（5）配置安全权限。

第四阶段：实战应用

9大领域的真实应用场景实战，共100个可以落地的实战案例。

第五阶段：优化与进阶

（1）OpenClaw框架二次开发。

（2）性能优化技巧。

（3）常见问题排查。

建议读者按照章节顺序学习，每章的实战内容都要动手操作。随着实战经验的积累，将能够独立完成复杂的AI Agent系统搭建，真正实现“一个人运营多个平台”的高效工作模式。

第2章　OpenClaw核心概念详解：读懂底层逻辑

OpenClaw的设计理念贯穿整个框架的开发与使用过程，决定了框架的核心特性与使用体验，其主要围绕“隐私、灵活、易用”三大核心展开。

OpenClaw的核心概念可概括为“五大核心模块+一套工作流+四大设计理念”，具体如下。

（1）五大核心模块：Gateway（网关）、Agent（智能体）、Skills（技能）、Memory（存储器）、Channels（渠道）构成基础架构，各模块协同工作，实现智能任务执行。

（2）一套工作流：Think（思考）→Act（行动）→Observe（观察）→Reflect（反思）闭环工作流，是智能体自主完成任务的核心逻辑，确保任务执行的准确性与高效性。

（3）四大设计理念：本地优先、模型无关、插件化扩展、多渠道接入，决定了框架的隐私性、灵活性与易用性。

理解以上核心概念，可快速掌握OpenClaw的框架逻辑与使用方法，为后续的环境部署、技能开发、二次开发奠定坚实的基础。

2.1　五大核心模块详解

OpenClaw的核心架构设计遵循“松耦合、强内聚”的原则，各组件之间职责明确又相互协作。理解这一架构是深入掌握OpenClaw的基础。5大核心模块包括：Gateway、Agent、Skills、Memory、Channels。

2.1.1　Gateway（网关）

Gateway是OpenClaw体系的统一入口与流量中枢，是外部请求接入、协议转换、路由分发与流量管控的核心枢纽，是连接外部系统与内部智能执行单元的关键桥梁，确保所有请求能够高效、安全地流转至对应处理模块。

Gateway是整个系统的核心，它承担着以下关键职责。

（1）作为会话、路由和渠道连接的唯一事实来源。

（2）监听来自各聊天平台的消息。

（3）将消息路由到对应的Agent处理。

（4）管理会话状态和用户权限。

（5）协调多个Agent之间的通信。

Gateway采用单进程设计，通过事件驱动的方式处理各种请求。这种设计既保证了性能，又便于统一管理。外部请求通过Channels组件传入后，首先进入Gateway；Gateway先对请求进行协议转换与格式标准化，接着通过内置的路由规则引擎匹配目标Agent，最后根据流量管控策略将请求转发至目标组件；处理完成后，Gateway将响应结果转换为对应外部渠道的格式，回传至发起请求的终端。统一的入口设计简化了外部系统的对接流程，降低了系统集成成本；流量管控机制保障了系统的稳定性与安全性；路由分发能力实现了请求的精细化调度，提升了整体系统的响应速度与资源利用率。

1. 工作逻辑

Gateway的功能围绕“通信对接、消息处理、控制管理”三大维度展开，覆盖从渠道接入到会话运维的全流程，具体如下。

（1）多渠道对接与管理：作为OpenClaw与外部交互的接口，支持对接各类主流通信渠道，包括Telegram、飞书、WhatsApp、微信、企业微信、钉钉、QQ、Discord及本地控制台等，可实现多渠道同时接入、并行使用，无须修改框架核心代码，仅需配置对应渠道参数即可完成接入与启用，适配个

人使用、团队协作等不同场景需求，这也是OpenClaw多平台使用优势的核心支撑。

（2）消息统一处理与转发：这是Gateway最核心的基础功能。具体分为两个核心环节：一是接收用户从各渠道发送的指令消息，自动对消息格式进行标准化处理（统一编码、去除冗余信息），避免因渠道差异导致指令传递错误；二是将标准化后的指令精准转发至Agent模块，同时记录消息来源渠道、发送时间等关键信息，便于后续结果反馈与会话追溯。反之，接收Agent模块返回的任务执行结果，根据记录的消息来源，精准转发至对应通信渠道，反馈给用户，形成“用户发送指令、Gateway转发、框架处理、Gateway反馈”的完整通信闭环。

（3）会话与上下文管理：负责管理所有用户的交互会话，记录每个会话的上下文信息（用户指令、框架回复、执行步骤），并同步至Memory模块进行存储，确保智能体能够实现连贯交互，避免出现“上下文断裂”的问题。同时，支持会话的查询、清理、归档等操作，用户可通过Web控制面板或命令行查看活跃会话、历史会话，便于运维与复盘。

（4）Web控制面板：内置可视化Web控制面板，启动Gateway后可通过默认地址访问（默认：http://127.0.0.1:18789），提供实时日志查看、会话管理、配置在线编辑、渠道状态监控、性能监控等功能，无须手动编辑配置文件或操作命令行，降低了运维与管理门槛，适合不熟悉命令行的用户使用。

（5）配置管理与热更新：支持通过命令行或Web控制面板修改Gateway相关配置（如端口、绑定模式、认证方式等），且具备热更新能力，无须重启Gateway进程即可应用新配置，提升运维效率。同时，所有配置信息统一存储在配置文件（~/.openclaw/openclaw.json）中，便于统一管理与备份。

（6）安全认证与访问控制：为保障Gateway不被未授权访问，提供Token认证与密码认证两种方式，默认采用Token认证（Gateway启动时自动生成随机Token，重启后自动更换，安全性更高），也可手动配置固定密码。同时，通过绑定模式控制访问范围，避免因服务暴露导致的安全风险。

2. 与其他模块的协同关系

Gateway作为OpenClaw的通信中枢，仅与Agent模块直接交互，不直接参与任务推理、记忆存储或技能执行，协同逻辑如下。

（1）与Agent模块：Gateway接收用户指令并转发给Agent，接收Agent返回的执行结果并反馈给用户，是Agent与用户之间唯一的通信桥梁；Agent依赖Gateway传递的指令开展任务规划，依赖Gateway反馈结果给用户。

（2）与Memory模块：Gateway不直接操作Memory，但会将会话上下文信息同步给Agent，由Agent写入Memory，确保会话记忆的连贯性；同时，Gateway可从Memory中读取会话历史，用于消息追溯与上下文衔接。

（3）与LLM模型、Skills模块：Gateway不与这两个模块直接交互，所有指令与结果均通过Agent模块中转，确保核心业务逻辑的独立性与模块化解耦，便于后续扩展与维护。

2.1.2　Agent（智能体）

Agent的功能围绕指令处理、任务调度、模块协同、结果管控四大维度展开，覆盖从指令接收至任务完成的全流程，是OpenClaw智能任务执行的核心逻辑载体。

1. 工作逻辑

Agent工作逻辑具体如下。

（1）指令解析与意图识别：接收Gateway转发的标准化用户指令，结合Memory中的会话上下文信息，完成指令的深度解析与意图识别，明确任务目标、执行条件、预期结果及潜在需求。例如，用户发送“帮我读取本地文件并分析数据”，Agent会解析出“读取文件”“数据处理”两个核心子任务，识别出需要调用“文件读写Skills”和“数据处理Skills”，同时关联Memory中用户过往的文件路径偏好，提升执行效率。

（2）任务规划与步骤拆解：调用配置的LLM模型进行推理决策，将复杂的任务拆解为可执行的有序

步骤，形成详细的执行计划，明确每个步骤的执行目标、调用模块、依赖条件及执行优先级。例如，将“生成月度数据报告”拆解为“读取数据源文件→清洗数据→分析数据→生成报告→保存报告”5个步骤，每个步骤对应调用不同的Skills，同时规划步骤间的衔接逻辑（如数据清洗完成后再执行分析）。

（3）模块协同与调度：作为全局调度中心，负责协调LLM与Memory、Skills等模块协同工作，根据任务规划自动调用对应模块，确保任务有序推进。核心调度逻辑：调用配置的LLM完成推理与步骤规划，调用Skills执行具体操作，读取/写入Memory存储上下文与执行记录，同时接收各模块的反馈信息，动态调整执行节奏。

（4）执行过程监控与异常处理：实时监控任务执行全过程，跟踪每个步骤的执行状态（成功、失败、执行中），接收Skills模块返回的执行结果，判断是否符合预期。如果出现异常（如Skills调用失败、数据读取错误），自动调用LLM分析异常原因并生成解决方案（如重试调用、更换替代Skills、提示用户补充信息），以确保任务尽可能完成，避免执行中断。

（5）结果整理与反馈：任务执行完毕后，整理各步骤的执行结果，结合用户指令的预期需求，生成清晰、规范的反馈内容（如文本、文件、数据报表等），通过Gateway模块反馈给用户。同时，将任务执行过程、结果及关键信息写入Memory模块，形成会话记忆，为后续执行同类任务提供参考。

（6）自适应学习与优化：具备基础的自适应能力，能够根据历史任务执行记录（存储在Memory中），优化任务规划逻辑与模块调度策略。例如，多次执行“文件读取+数据分析”任务后，Agent会记忆常用的文件路径、数据处理规则，后续执行同类任务时可直接调用，提升执行效率；若某类Skills频繁调用失败，会自动优先选择替代Skills。

2. 与其他模块的协同关系

Agent作为OpenClaw的全局调度中心，与其他四大模块均有直接交互，也是连接各模块的核心枢纽，确保整个框架高效运转，具体如下。

（1）与Gateway模块：Agent接收Gateway转发的标准化用户指令，任务执行完毕后，将结果反馈给Gateway，由Gateway转发给用户；Gateway负责管理Agent的任务队列，若Agent忙碌，将用户指令暂存，待Agent空闲后传递，同时监控Agent的运行状态，出现异常时反馈给用户。

（2）与Memory模块：Agent是Memory的主要读写者。在任务执行前，读取Memory中的会话上下文、用户偏好、历史任务记录，辅助任务规划；在任务执行过程中，将执行步骤、关键信息写入Memory；任务完成后，将最终结果、执行总结写入Memory，形成长期记忆，为后续任务提供参考。

（3）与Skills模块：Agent负责调度Skills执行具体操作。根据任务规划，向Skills传递执行参数（如文件路径、API地址等），接收Skills的执行结果，判断执行是否成功；若Skills调用失败，Agent会根据配置进行重试，或调用其他替代Skills，确保任务推进。Skills仅负责执行具体操作，不参与任务规划与调度，完全受Agent控制。

Agent是OpenClaw体系的智能执行核心，负责接收Gateway分发的请求，完成任务拆解、技能调度、流程编排与状态管理等核心工作，是连接用户需求与具体技能实现的关键枢纽。

Agent智能体是具有特定职责和能力的执行单元。每个Agent是一个完整的“小型AI系统”，包含以下核心要素。

SOUL.md：Agent的人设定义，描述Agent的角色、性格、语言风格等。

AGENTS.md：Agent的职责范围，定义Agent应该做什么、不应该做什么。

TOOLS.md：Agent可使用的工具列表，指定Agent能够调用哪些Skills。

MEMORY.md：Agent的长期记忆，存储重要的经验和规则。

USER.md：Agent的服务对象定义，明确Agent为谁服务、可满足什么需求。

这种文件化的配置设计使Agent的定义和管理变得简单直观，用户只需编辑相应的Markdown文件（一种使用简单标记语法编写的纯文本文件，后缀为.md），即可定制Agent的行为。

2.1.3　Skills（技能）

Skills的功能围绕“插件管理、操作执行、扩展适配”三大维度展开，覆盖从安装、调用、执行到维护的全流程，为Agent提供可落地的操作能力，同时支持灵活扩展。

1. 工作逻辑

Skills的工作逻辑具体如下。

（1）插件化管理与生命周期管控：支持Skills插件的全生命周期管理，包括安装、卸载、升级、启用、禁用等操作，所有操作可通过命令行或Web控制面板完成，无须修改框架核心代码。Skills插件被独立存储在指定目录，与框架核心解耦，单个技能的异常或升级不会影响其他技能及框架的正常运行，降低了维护成本。例如，可单独安装“数据可视化”技能，卸载“网页搜索”技能，不影响文件读写、API调用等其他技能的使用。

（2）标准化操作执行：所有技能均遵循统一的接口规范开发，确保Agent能够快速调用、统一接收执行结果。技能接收Agent传递的执行参数（如文件路径、API地址、查询关键词等），执行具体操作后，返回标准化的结果（成功/失败状态、执行输出、异常信息等），便于Agent判断执行效果并推进后续步骤。例如，Agent调用“文件读写”技能时，传递文件路径、读取模式等参数，技能执行读取操作后，返回文件内容及执行状态，Agent可根据返回结果判断是否继续执行数据处理步骤。

（3）内置技能全覆盖：框架内置常用核心技能，覆盖日常办公、数据处理、网络交互等高频场景，无须额外开发即可满足基础使用需求，降低用户上手门槛。核心内置技能分类及示例如下。

① 文件操作类：文件读写、文件夹创建/删除/遍历、文件格式转换（如TXT转CSV、Excel读取）、文件压缩/解压等，适用于本地文件管理相关任务。

② 网络交互类：网页搜索、API调用、HTTP请求发送/接收、网络资源下载等，适用于需要获取网络数据、对接第三方服务的任务。

③ 数据处理类：数据清洗、数据筛选、数据统计、数据可视化（生成折线图、柱状图等）、RAG检索（关联本地文档进行语义检索）等，适用于数据分析、文档处理等相关任务。

④ 系统操作类：系统命令执行、进程管理、环境变量查询/修改、定时任务设置等，适用于本地系统管理相关任务（需授予对应权限）。

⑤ 基础工具类：时间日期转换、字符串处理、加密解密、二维码生成/识别等，适用于各类基础辅助操作。

（4）自定义技能开发支持：提供标准化的技能开发模板、接口规范与开发文档，降低开发门槛，新手可快速上手开发自定义技能，满足特定业务场景需求。开发的自定义技能可直接集成到框架中，与内置技能一样被Agent调度，实现框架功能的个性化扩展。例如，企业可开发适配自身业务系统的“内部数据查询”技能，个人可开发“自定义报表生成”技能，无须修改OpenClaw核心代码。

（5）技能依赖管理：自动管理技能所需的依赖环境（如Python库、系统工具等），安装技能时自动检测并安装缺失的依赖，卸载技能时自动清理相关依赖（可选），避免依赖冲突，确保技能能够正常运行。例如，安装“数据可视化”技能时，自动安装Matplotlib、Seaborn等依赖库，无须用户手动安装。

（6）技能权限控制：支持对技能进行权限分级管理，根据技能的功能风险（如系统操作、文件修改），设置不同的调用权限，避免恶意技能或误操作导致系统安全风险。例如，“系统命令执行”技能需授予管理员权限才能调用，普通用户仅可调用文件读取、网页搜索等低风险技能。

2. 与其他模块的协同关系

Skills作为OpenClaw的操作执行载体，仅与Agent模块直接交互，接受Agent的调度，为其提供具体的操作能力，协同逻辑清晰，与其他模块的协同风格保持一致，具体如下。

（1）与Agent模块：Agent是Skills模块的唯一调度者，也是协同最紧密的模块。Agent根据任务规划，确定需要调用的技能，传递正确的执行参数（如文件路径、API密钥等，这些参数大多来自Memory

模块），调用对应技能执行具体操作；Skills执行完毕后，将标准化的执行结果（成功/失败、输出内容、异常信息）反馈给Agent，Agent根据反馈结果判断是否继续执行后续步骤，或调用LLM调整执行策略。Skills完全受Agent控制，不具备自主决策与调度能力。

（2）与Memory模块：Skills与Memory无直接交互，Skills执行操作所需的参数（如用户常用的文件路径、API密钥等），由Agent从Memory中读取并传递；Skills执行的结果（如文件读取内容、数据处理结果等），由Agent写入Memory，供后续任务调用、追溯与复盘。例如，Skills读取文件后，Agent将文件内容写入Short记忆，后续步骤可直接调用该内容，无须重复读取文件。

（3）与Gateway模块：Skills与Gateway无直接交互，Gateway仅负责传递用户指令与最终执行结果，不参与技能的调用与执行；Skills的执行状态与结果，需通过Agent、Gateway反馈给用户，确保用户能够实时了解任务执行进度。

（4）与LLM：Skills与LLM无直接交互，LLM负责任务规划与异常分析，确定需要调用的技能及执行步骤，通过Agent传递给Skills；Skills执行过程中出现异常时，Agent将异常信息反馈给LLM，LLM分析异常原因，调整执行策略（如更换替代Skills、重试调用），再通过Agent调度Skills重新执行，形成某个功能的“规划→执行→反馈→调整”闭环。

2.1.4 Memory（存储器）

Memory的功能围绕“数据存储、分层管理、读写调度、数据安全”四大维度展开，覆盖Memory从生成、存储、调用到清理的全流程，为OpenClaw各模块提供稳定、安全的记忆数据支撑。

1. 工作逻辑

Memory的工作逻辑如下。

（1）分层记忆存储与管理：采用四类分层记忆设计，分别对应不同的应用场景与数据类型，相互独立又协同工作，确保记忆数据的有序存储与高效调用，这是Memory模块的核心特色。四类分层记忆明确区分功能、用途与使用场景。为便于理解与配置，详细讲解如下。

Soul记忆：核心是智能体的“角色设定库”，存储智能体的固定属性信息，决定智能体的交互调性与行为模式，一旦配置后可长期复用数据，无须频繁修改。具体包括：智能体的性格（如严谨、活泼、专业）、角色身份（如Python开发专家、客服助手、技术文档工程师）、回复风格（如简洁明了、详细全面、口语化）、固定规则（如禁止泄露隐私、优先使用中文回复）等。例如，将智能体角色设定为“技术支持专家”，Soul记忆会存储“专业、严谨、优先解决技术问题、用技术术语回复”等固定规则，确保所有交互均贴合该角色。

Short记忆：核心是“会话上下文缓存”，存储当前用户会话的实时交互数据，确保智能体能够理解用户的连续指令，实现连贯交互，避免出现“上下文断裂”。具体包括：用户当前会话发送的指令、智能体的回复、任务执行的中间步骤、临时参数等。数据仅在当前会话有效，会话结束后可手动清理或自动过期（默认会话超时时间为30分钟）。例如，用户先发送“读取本地文件”，再发送“分析文件中的数据”，Short记忆会存储“读取文件”的指令与执行结果，Agent可直接调用该记忆，无须用户重复说明文件路径。

Long记忆：核心是“长期关键数据存储”，存储用户偏好、历史任务结果、常用配置等需要长期复用的数据，数据长期有效，不会随会话结束而过期，可被所有会话调用。具体包括：用户常用的文件路径、API密钥（加密存储）、常用技能配置、历史任务的最终结果、用户偏好设置（如喜欢的反馈格式、常用模型）等。例如，用户多次使用“数据可视化”技能，Long记忆会存储用户偏好的图表类型（如折线图、柱状图等），后续在执行同类任务时，Agent可直接调用该偏好，无须用户重复指定。

Episodic记忆：核心是“场景化记忆归档”，存储特定时间、特定场景下的完整交互记录与任务执行过程，用于后续追溯、复盘与自适应学习，数据按时间戳归档，长期有效。具体包括：任务执行的完整步骤、各步骤的执行结果、异常信息、用户与智能体的完整对话、时间戳等。例如，用户上周执行了

“生成月度数据报告”任务，Episodic记忆会存储该任务的执行时间、步骤、生成的报告路径、异常处理过程等，后续用户查询该任务时，Agent可快速调取相关记录。

（2）记忆读写与调度：由Agent模块统一调用，实现记忆数据的精准读写与调度，确保各模块能够高效获取所需记忆。核心逻辑：在执行任务前，Agent根据任务需求读取对应类型的记忆（如执行任务规划时读取Short记忆与Long记忆，调用角色设定时读取Soul记忆）；在任务执行过程中，实时写入中间步骤、临时数据到Short记忆Episodic记忆；任务完成后，将最终结果写入Long记忆与Episodic记忆，形成完整的记忆闭环。支持记忆优先级调度，默认按“Short记忆> Episodic记忆> Long记忆> Soul记忆”的优先级读取，可通过配置进行调整。

（3）数据安全与加密存储：遵循OpenClaw本地优先与数据隐私的设计理念，所有记忆数据均存储在本地文件中，不上传至任何第三方服务器。对于敏感数据（如API密钥、用户隐私信息），采用AES加密存储，确保数据不被泄露；同时支持记忆数据的备份与恢复，可手动导出备份记忆文件，避免数据丢失。

（4）记忆清理与过期管理：支持手动清理与自动过期两种方式，避免记忆数据过多导致存储占用过高、调用效率下降。具体包括：Short记忆默认会话超时后自动清理；Long记忆与Episodic记忆可手动清理指定数据或按时间范围清理；支持设置记忆存储上限，超出上限后自动清理最早的记忆数据（优先清理Episodic记忆，保留Soul记忆与核心Long记忆）。

（5）记忆导出与导入：支持将所有记忆数据导出为Markdown或YAML格式文件，便于备份、迁移与编辑；同时支持导入已备份的记忆文件，快速恢复记忆数据，适用于设备更换、框架重装等场景。例如，重装OpenClaw后，可导入之前导出的记忆文件，快速恢复智能体的角色设定、用户偏好与历史任务记录，无须重新配置。

2. 与其他模块的协同关系

Memory作为OpenClaw的记忆中枢，主要与Agent、LLM直接交互，为其提供核心记忆数据支撑，协同逻辑清晰，与Gateway、Agent的协同风格保持一致，具体如下。

（1）与Agent模块：Agent是Memory模块唯一的读写调用者，也是协同最紧密的模块。在执行任务前，Agent根据任务需求读取对应类型的记忆（如解析指令时读取Short记忆，规划任务时读取Long记忆，调用角色时读取Soul记忆）；在任务执行过程中，实时写入中间数据、执行步骤到Short记忆与Episodic记忆；任务完成后，将最终结果写入Long记忆与Episodic记忆。同时，Agent负责记忆的优先级调度与清理触发，确保记忆数据的高效利用。

（2）与LLM：LLM依赖Memory中的记忆数据完成推理决策，无直接读写权限，需通过Agent调用记忆。具体逻辑：LLM进行意图识别、任务规划时，Agent会将Short记忆（上下文）、Long记忆（用户偏好）、Soul记忆（角色设定）传递给LLM，LLM结合这些数据进行推理，生成更贴合用户需求、符合智能体角色的执行计划；在反思阶段，LLM结合Episodic记忆（历史执行记录）分析异常原因，优化执行策略。

（3）与Gateway模块：Memory与Gateway无直接交互，Gateway传递的会话上下文信息，由Agent写入Short记忆；Gateway无须读取记忆数据，仅负责传递用户指令与执行结果，确保通信链路与记忆存储的解耦，提升框架的稳定性与可扩展性。

（4）与Skills模块：Memory与Skills无直接交互，Skills执行具体操作时所需的参数（如文件路径、API密钥），由Agent从Memory中读取并传递给Skills；Skills执行的结果，由Agent写入Memory中，Skills不直接操作记忆数据，以确保记忆存储的安全性。

2.1.5　Channels（渠道）

Channels是OpenClaw体系的用户交互入口，负责对接各类外部交互渠道，实现用户请求的接收与响应结果的推送，是连接用户与系统的直接桥梁。Channels是OpenClaw连接各种通信平台的接口。通过

Channels，OpenClaw可以同时接入多个聊天平台，每个平台都可以独立配置访问权限、群组规则等。

常见的渠道类型包括以下3种。

文本交互渠道：包括微信、QQ、短信、网页聊天框等，支持用户通过文本输入发送请求，系统通过文本返回响应，是最常见的交互渠道。

语音交互渠道：包括智能音箱、语音助手、电话客服系统等，支持用户通过语音输入发送请求，系统通过语音返回响应，适用于不方便进行文本输入的场景。

API对接渠道：包括第三方系统的API、物联网设备的通信接口等，支持系统与外部系统的自动化交互，例如与航空公司的API对接实现机票的自动预订。

多渠道适配能力扩大了系统的服务覆盖范围，满足了用户在不同场景下的交互需求；消息格式转换实现了渠道与系统内部的无缝对接；渠道状态监控保障了交互的稳定性与可靠性。

OpenClaw的Channels是Agent与外界沟通的“耳朵和嘴巴”，负责接收用户指令并返回执行结果。它支持超过28个主流平台，分为原生内置与插件扩展两大类，覆盖即时通信、办公协作、Web终端等场景。

OpenClaw五大核心模块通过紧密协同，形成完整的智能服务闭环：用户通过Channels组件发送请求，请求经格式转换后传入Gateway；Gateway完成协议转换与路由分发，将请求转发至匹配的Agent；Agent从Memory获取用户记忆数据，辅助任务理解与拆解，随后调用Skills模块库中的对应技能执行子任务；Skills执行完成后将结果返回给Agent，Agent整合结果并更新用户记忆至Memory；最后，Agent将响应结果回传给Gateway，Gateway转换为对应渠道格式后，通过Channels推送至用户。这种协同机制保障了系统的智能化、高效性与可扩展性，为用户提供了连贯、个性化的智能服务体验。

2.2 OpenClaw的工作流程

OpenClaw是一套本地优先、可自主执行任务的AI智能体框架，其工作流程是从接收自然语言指令到完成本地到云端操作的完整闭环。

1. Think（思考阶段）

该阶段由LLM主导，Agent协同配合，核心是明确任务目标、规划执行步骤。具体流程：Agent将用户指令传递给配置的LLM，LLM结合Memory中的记忆数据，进行自然语言理解与推理决策，明确任务的核心目标、执行条件与具体步骤，形成详细的执行计划，反馈给Agent。

2. Act（执行阶段）

该阶段由Agent主导，核心是按照思考阶段规划的步骤，调用对应技能完成具体操作。Agent根据LLM反馈的执行计划，调度Skills模块中的对应技能，执行具体操作（如调用天气API查询天气、读取本地文件获取数据等），同时将执行过程中的关键信息写入Memory模块，便于后续追溯。

3. Observe（观察阶段）

该阶段由Agent负责，核心是获取技能执行结果，判断任务执行是否成功。Agent接收Skills模块返回的执行结果，结合任务目标判断执行是否达到预期效果。若执行成功，则记录执行结果；若执行失败，则记录失败的原因（如API调用超时、文件不存在等），并将执行结果反馈给LLM。

4. Reflect（反思阶段）

该阶段由LLM主导，核心是根据观察结果调整执行策略，形成任务闭环。LLM接收Agent反馈的观察结果，若任务执行成功，则整理执行结果，通过Gateway反馈给用户；若任务执行失败，则分析失败的原因，重新规划执行步骤，反馈给Agent，由Agent重新调度技能执行，直至任务完成或确认无法完成。

核心工作流程如下。

（1）指令接入：用户通过IM工具（微信、飞书等）发送自然语言指令。

（2）网关分发：Gateway接收、标准化消息，路由到对应Agent。

（3）意图解析与任务规划：Agent调用LLM理解意图，拆解为可执行步骤，生成DAG任务链。

（4）技能/工具调用：按规划调用Skills（文件、浏览器、命令等），执行前做权限校验、参数校验、安全过滤等工作。

（5）执行与结果处理：在本地/指定环境执行操作，标准化结果并记录日志。

（6）结果反馈：将执行结果返回给用户，并更新Short记忆与Long记忆。

（7）记忆迭代：记忆系统保存上下文与偏好，优化后续任务。

理解OpenClaw的工作流程有助于更好地使用和调试系统。一个典型的请求处理流程如下。

步骤1：消息接收。

步骤2：用户通过某个通信渠道比如飞书发送消息，Gateway监听并接收这条消息。

步骤3：权限验证。

步骤4：Gateway检查发送者的权限，确认是否有权访问系统。这一步根据渠道配置中的allowFrom、allowedUsernames等规则进行过滤。

步骤5：会话匹配。

步骤6：Gateway根据发送者ID或会话ID找到对应的Agent。如果配置了“按发送者创建独立会话”，则每个新发送者都会有一个独立的会话上下文。

步骤7：上下文构建。

步骤8：Agent从Memory中加载相关的历史信息和长期记忆，结合当前消息，构建完整的上下文。

步骤9：意图分析。

步骤10：Agent分析用户消息，理解用户的意图。这一步由绑定的AI大模型完成，模型会根据上下文决定需要执行什么操作。

步骤11：工具规划。

步骤12：如果需要执行实际操作，Agent会规划需要调用的技能（Skills），确定调用顺序和参数。

步骤13：工具执行。

步骤14：Agent按计划调用各个Skills，执行具体的操作。每个Skills的执行结果都会返回给Agent。

步骤15：响应生成。

步骤16：Agent根据工具执行结果，生成最终的响应消息。

步骤17：结果返回。

步骤18：Gateway将响应消息通过原来的渠道比如飞书、微信等返回给用户。

步骤19：记忆更新。

步骤20：Agent将本次交互的重要信息更新到Memory中，供后续使用。

以上内容就是OpenClaw从指令输入到任务执行的完整链路。

2.3　关键术语解析

OpenClaw围绕模型绑定、技能调用、记忆持久化、渠道对接四大核心，构建了可扩展、可定制的AI Agent运行机制。

2.3.1　模型绑定（Model Binding）

模型绑定是OpenClaw必要的操作，指将LLM推理引擎与调度策略关联，实现“多模型热插拔、按场景路由、自动容错”的统一推理入口。它屏蔽了不同LLM的API差异，让Agent无须关心底层模型，专注于任务的执行。

1. 核心机制

模型注册：通过配置或命令注册LLM提供商（OpenAI、Anthropic、Kimi、Ollama等）及模型版本，生成唯一模型标识。

路由策略：按渠道、用户、任务类型自动分配模型（如高敏感任务用GPT-4o，低成本任务用Claude 3 Haiku）。

容错降级：当主模型发生故障时自动切换至备用模型，保障服务可用。

参数隔离：不同的模型可独立配置随机度、最大Token、上下文窗口等推理参数，适配自身特性。

2. 关键命令

查看模型列表：openclaw models list。

切换默认模型：openclaw models set default openai/gpt-4o。

测试模型连通性：openclaw models test --model openai/gpt-4o。

3. 核心价值

解耦：Agent逻辑与模型解耦，换模型无须更改代码。

降本：低成本任务用轻量模型，高成本任务用高端模型，平衡成本与效果。

高可用：多模型容错避免单点故障，保障服务稳定。

2.3.2 技能调用（Skills Invocation）

技能调用是指通过声明式Skills规范，让Agent自主调用工具、执行任务的机制。每个Skills是一个包含Skills.md的文件夹，详细描述了工具的用途、输入参数、执行逻辑，以实现零代码扩展Agent的能力。

1. 核心机制

Skills加载：按优先级加载（工作区技能>本地技能>捆绑技能），自动扫描Skills.md注册工具元数据。

意图匹配：Agent解析用户指令，匹配Skills描述，决定是否调用及调用顺序。

参数校验：根据Skills元数据校验输入参数，若缺失则向用户询问。

执行与回调：调用工具后获取结果，解析并整合到回复中，支持异步执行与流式回调。

技能门控：通过metadata限制技能依赖（如二进制、环境变量等），保障安全。

标准Skills结构：~/.openclaw/Skills/my-Skills/Skills.md。

2. 配置与管理

安装社区技能：clawhub install weather-query。

查看技能详情：openclaw Skills info my-Skills。

3. 核心价值

快速扩展：无须编写代码，用自然语言描述即可创建技能；接入第三方工具（如浏览器、API、数据库等）。

可复用：Skills可在多个Agent间共享，降低重复开发成本。

安全可控：依赖校验避免恶意调用，保障执行环境安全。

2.3.3 记忆持久化（Memory Persistence）

记忆持久化是指将Agent的会话上下文、用户偏好、长期事实存储为本地文件，实现跨会话记忆恢复。它解决了LLM“无原生持久记忆”的痛点，让Agent记住“人、事、规则”。

1. 核心机制

自动写入：用户说“记住”或Agent决策时，自动将信息写入MEMORY.md（Long记忆）或memory/YYYY-MM-DD.md（每日日志）。

向量检索：对记忆文件生成向量索引，支持语义搜索（混合向量相似度+ BM25关键词），精准匹配相关记忆。

自动压缩：上下文接近窗口上限时，自动压缩旧对话为摘要，保留核心信息。

可编辑审计：记忆文件为纯Markdown，可手动修改、删除，支持Obsidian等工具管理。

2. 关键配置与命令

语义搜索记忆：openclaw memory search“用户常用办公软件”。

手动压缩：/compact（会话内输入）。

3. 核心价值

不失忆：跨会话保留用户偏好、历史对话，长会话不混乱。

可解释：记忆文件可读可改，透明可控，避免“黑盒”。

降本：压缩记忆减少Token消耗，降低推理成本。

2.3.4　渠道对接（Channel Integration）

渠道对接是OpenClaw接入层的核心，指通过统一Gateway，将Agent接入各类通信平台的机制。它支持超过28个主流平台，分为原生内置与插件扩展两类，实现“一处配置、多端交互”。

1. 核心架构

Gateway：统一接收所有渠道消息，当转发至Agent运行时，进行会话隔离、权限控制、消息队列等工作。

适配器模式：每个渠道对应独立适配器，屏蔽平台差异，支持自定义扩展。

会话隔离：给每一个“渠道 + 用户”分配独立会话（独立记忆、独立上下文），避免串扰（如per-channel-peer策略）。

2. 渠道分类与配置

（1）原生内置渠道（开箱即用）。

（2）插件扩展渠道（需安装）。

3. 核心命令

查看渠道状态：openclaw channels status。

启用/禁用渠道：openclaw channels enable telegram / disable telegram。

查看日志：openclaw channels logs --channel telegram。

4. 核心价值

全平台覆盖：接入即时通信、办公协作、本地终端等场景，适配不同用户的习惯。

统一管理：通过Gateway统一管理所有渠道，简化配置与运维。

安全可控：会话隔离、权限策略避免信息泄露，保障多用户场景安全。

OpenClaw的四大核心术语并非孤立的，而是层层递进、协同工作的闭环。

（1）渠道对接接收用户请求，传递至Agent。

（2）模型绑定选择最优LLM进行推理。

（3）技能调用让Agent执行工具任务（如查天气、写文档等）。

（4）记忆持久化存储用户偏好与执行结果，支撑下一轮交互的个性化与连贯性。

这一闭环不仅让OpenClaw可灵活接入各类平台，又能通过多模型与技能扩展能力，同时还可通过持久记忆保持长期价值，真正实现“全场景、高可用、可定制”的AI Agent。

2.4　开源特性与隐私优势（本地部署、数据可控的核心价值）

在AI智能体技术快速普及的当下，数据隐私泄露、厂商依赖、流程黑盒等问题，已成为个人与企业应用AI工具的核心顾虑。OpenClaw作为开源、本地优先的AI智能体框架，以“数据不出域、权限全可控”为核心定位，通过开源架构与本地部署能力，打破传统云端AI智能体的隐私壁垒，让AI自动化能力真正服务于用户自身，而非第三方平台。

本节将详细阐述OpenClaw的开源特性，以及本地部署带来的隐私安全优势，解析其“能力开放、

数据可控”的核心价值。

OpenClaw核心定位：开源赋能，本地优先。

OpenClaw是一套面向个人、企业及涉密场景设计的开源AI智能体框架，其核心定位是“完全私有化部署、数据全程可控、能力可定制扩展”。与传统云端AI智能体不同，OpenClaw不依赖第三方云服务，默认将所有任务执行、数据处理放在用户本地设备或内网环境中，从架构根源上解决数据泄露风险。同时通过开源特性，让用户实现对AI智能体的完全掌控，真正做到我的数据我做主。

其核心价值可概括为：无须将敏感数据上传至云端，即可在本地完成从指令解析、任务规划到工具执行的全流程AI自动化，兼顾效率与隐私安全，适配个人隐私保护、企业内部自动化、涉密场景应用等多种需求。

1. OpenClaw的开源特性

OpenClaw的开源特性是其实现“数据可控、能力可扩展”的基础。区别于闭源AI智能体的黑盒操作，其所有核心逻辑均开源可审计、可二次开发，具体特性如下。

1）完全开源，流程透明可审计

OpenClaw的核心代码（包括Agent智能调度、Gateway网关分发、Skills技能调用、权限校验、记忆系统等）全部开源，无任何隐藏逻辑与黑盒模块。用户可直接查看、审计每一行代码，明确数据流转路径、任务执行逻辑，杜绝“数据被偷偷收集、指令被恶意篡改”的风险。同时，开源特性支持用户根据自身需求进行二次开发，可内嵌到自有产品中，或用于商业方案落地，无须担心版权限制与厂商绑定。

2）模块化架构，可插拔易扩展

OpenClaw各模块独立运行、可自由插拔。其中，Agent、Gateway、Skills、Memory、Channels均可单独替换或升级。例如，用户可替换核心模型、更换接入渠道（微信、飞书等）、增减工具能力，无须修改整体框架，极大地降低了定制化成本，适配不同场景的个性化需求。

3）多模型兼容，本地与云端灵活切换

OpenClaw不绑定任何模型厂商，支持本地模型与云端模型混合调度。用户可根据任务敏感性灵活选择，如敏感任务（如隐私文件处理、内部数据统计等）可调用本地模型（如Ollama部署的Llama系列、Qwen系列模型等），全程在本地执行，不产生任何云端数据交互；通用任务（如普通信息查询、格式转换等）可调用云端模型（如GPT-4o、Claude 3），兼顾效率与成本。这种混合模式既保证了敏感数据的安全性，又提升了任务执行的灵活性。

4）可扩展技能体系，适配多场景需求

OpenClaw内置十多种常用工具技能（如文件操作、浏览器自动化、系统命令执行、API调用、邮件发送、文档生成等），用户可根据需求自由开关，避免不必要的权限开放。同时，框架支持自定义Skills开发，用户可根据自身业务场景（如企业内部流程自动化、科研数据处理、个人事务管理等）开发专属工具技能，形成个性化的AI自动化能力，让智能体真正贴合自身需求。

5）轻量化部署，跨平台易落地

OpenClaw具备轻量化特性，不需要超高配置的硬件设备，普通PC、家用服务器即可完成本地部署，降低了使用门槛。同时，支持Windows、Linux、macOS跨平台运行，无论是个人计算机、企业内网服务器，还是涉密隔离设备，均可顺利部署使用，无须担心系统兼容性问题。

OpenClaw的核心隐私价值源于其“本地优先”的部署模式——所有数据默认在用户本地设备或内网环境中流转、处理，不主动上传至任何第三方云端服务器，从根源上杜绝了数据泄露的风险。相较于传统云端AI智能体，其隐私安全优势主要体现在以下五个方面。

2. OpenClaw隐私安全优势

1）数据不出本地，从根源防范泄露

数据不出本地，从根源防范泄露是OpenClaw最核心的优势。用户的所有指令、处理的文件内容、

任务执行记录、个人偏好等数据，全程留在用户自己的设备或内网中，默认不与任何第三方云服务交互。无论是个人的隐私文件、聊天记录，还是企业的商业机密、内部流程数据、财务信息，都不会被收集、被用于模型训练、被第三方平台获取，从架构上彻底解决了云端AI智能体“数据裸奔”的痛点。

2）权限完全可控，筑牢安全边界

OpenClaw支持精细化权限管控，用户可根据自身需求，设置操作白名单与黑名单。例如，限制智能体只能访问指定文件夹、禁止执行高危系统命令、关闭网络访问权限等。所有工具调用、任务执行都需经过权限校验，确保智能体的操作在用户预设的安全边界内。同时，所有操作均有详细日志记录，可审计、可回溯，一旦出现异常操作，可快速定位问题、追溯源头，进一步提升安全性。

3）离线可运行，摆脱外部依赖

OpenClaw支持纯离线环境运行，无须依赖外部网络与第三方服务。在无外网访问权限的涉密环境、内网隔离机器中，用户依然可以部署并使用OpenClaw完成本地任务（如文件整理、本地数据统计、离线文档处理等）。这种特性使其能够适配政府、医疗、法律、军工等对网络隔离、数据保密有严格要求的场景，彻底摆脱对云端服务的依赖。

4）无厂商锁定，自主掌控全流程

由于OpenClaw完全开源且支持本地部署，用户无须依赖任何厂商的API服务、服务器资源，彻底摆脱厂商锁定。不会出现“API收费涨价、服务下架、地域限制”等问题，也无须担心因厂商停止维护导致工具无法使用。模型选择、数据存储、流程设置、权限管控等全部由用户自主决定，真正实现对AI智能体的完全掌控。

5）适配高度敏感场景，兼顾效率与安全

基于“数据不出本地、权限可控”的核心优势，OpenClaw特别适配各类高度敏感的场景：对于个人用户，可用于隐私文件整理、个人数据备份、私密信息分析，保护个人隐私不泄露；对于企业用户，可用于内部流程自动化、敏感数据处理、研发文档管理，防范商业机密泄露；对于政府、医疗、法律等行业，可用于涉密数据处理、隐私信息保护等方面，满足行业合规要求，实现“效率与安全兼顾”。

OpenClaw的核心价值在于通过“开源特性+本地部署”的组合，打破传统AI智能体“能力封闭、数据失控”的困境。开源让其能力开放、流程透明、可定制扩展，本地部署让其实现数据不出域、权限全可控。它不再是“将任务交给云端AI”的被动模式，而是“让AI成为本地设备上可信任、可审计、可完全掌控的自动化工具”的主动模式。

对于个人用户，OpenClaw是保护隐私的智能助手，让AI服务不侵犯个人隐私；对于企业与涉密场景，OpenClaw是安全可控的自动化解决方案，让AI能力服务于业务发展的同时，守住数据安全底线。在隐私保护日益重要的今天，OpenClaw以“开源赋能、本地可控”的核心优势，为AI智能体的安全应用提供了全新的可能。

OpenClaw支持完全本地部署，数据全部存储在用户自己的设备上，这是其核心价值之一。

本地部署的优势如下。

数据隐私可控：所有对话数据、配置信息都存储在本地，不上传到第三方服务器。

无服务中断风险：不依赖外部服务，Internet断开再恢复后仍可继续使用。

定制化自由：可以任意修改代码和配置，满足个性化需求。

零运营成本：除硬件和电费外，无须支付任何服务费用。

云端部署的优势如下。

7×24小时稳定运行：不关机、不断网，适合长期运营。

多设备远程访问：随时随地通过浏览器访问管理界面。

算力弹性扩展：可以根据需要升级服务器配置。

对于个人用户和中小企业，建议初期选择本地汉化部署进行测试和轻量使用，待熟悉后再考虑云端部署实现大规模运营。

第3章　环境准备：从零搭建OpenClaw运行环境

从零搭建OpenClaw运行环境是一套标准化、低门槛的本地部署流程，分为“环境准备→核心组件安装→基础配置→验证启动”四步。

先安装Python 3.10或以上版本、Node.js 22或以上版本及Git等基础依赖（Windows/macOS/Linux均适配），通过npm install -g openclaw@latest或源代码克隆GitHub仓库完成核心包安装。

再通过openclaw onboard生成默认配置文件（openclaw.json），按需配置LLM密钥（如OpenAI/Kimi/Ollama）、启用的Channels渠道（如CLI/WebChat/飞书）及记忆存储路径。

最后执行openclaw gateway start命令启动网关服务，通过openclaw tui终端交互或访问http://127.0.0.1:18789控制台验证连接，全程不需要复杂的编译。

进阶场景可补充安装技能插件、配置本地Ollama模型或扩展飞书/钉钉等渠道适配器，实现从“能运行”到“适配个性化场景”的快速落地。

3.1　系统要求与前置依赖（Windows、macOS、Linux通用配置）

OpenClaw基于Node.js/TypeScript开发，支持三大主流操作系统，跨平台兼容性优秀。在开始安装前，需要确保系统满足以下要求。

1）通用硬件要求

CPU：Intel i5 / AMD Ryzen 5或更高；内存：至少4GB（个人使用），建议8GB（团队使用）；磁盘：至少20GB可用空间（推荐SSD）；网络：稳定的互联网连接。

2）操作系统具体要求

通用软件依赖：Node.js 22或以上版本、pnpm（Node.js包管理器）、Git、Docker（可选，用于Docker部署）。具体要求见表3-1。

表 3-1　操作系统具体要求

操作系统	版本要求	安装方式
Windows	Windows 10/11 64位	PowerShell脚本 / 手动安装
macOS	10.15 (Catalina) 或更高	终端 / Homebrew
Linux	Ubuntu 22.04 LTS / Debian 11+	终端 / Docker

3.2　四种安装方式实操

OpenClaw安装方式分为一键脚本安装（推荐个人用户）、npm/pnpm安装（推荐开发者用户）、Docker安装（推荐云端部署）、阿里云一键部署（推荐长期运营）四类，适配不同的场景。

3.2.1　一键脚本安装

以管理员身份打开PowerShell，执行以下命令。

```
#一键安装OpenClaw
iwr -useb https://openclaw.ai/install.ps1 | iex
#验证安装成功
openclaw --version#一键安装OpenClaw
iwr -useb https://openclaw.ai/install.ps1 | iex
```

以管理员身份打开PowerShell可以采用以下四种方法。

方法一：Windows+X快捷菜单（最快）

按快捷键Windows+X（或右击“开始”按钮）；Windows 10选择Windows PowerShell（管理员），Windows 11选择终端（管理员）（默认就是PowerShell环境）；弹出“用户账户控制”时，单击“是”按钮确认。

方法二：搜索框打开（最直观）

单击任务栏上的搜索框，输入PowerShell；在搜索结果上右击，选择“以管理员身份运行”命令，单击“是”按钮确认。

方法三：运行窗口（快捷键）

按快捷键Windows+R打开“运行”对话框；输入powershell，按住快捷键Ctrl+Shift不放，再按Enter键，单击“是”按钮确认。

方法四：任务管理器

按快捷键Ctrl+Shift+Esc打开任务管理器；选择“文件”→“运行新任务”命令，输入powershell；勾选“以系统管理员权限创建此任务”复选框，单击“确定”按钮；窗口标题栏显示“管理员：Windows PowerShell”，即已获得管理员权限。在安装过程中，脚本会自动配置WSL2环境、安装所需依赖。整个过程需5～10分钟。

3.2.2　npm/pnpm安装

OpenClaw基于npm/pnpm管理，必须先安装Node.js（自带npm），步骤如下。

下载Node.js：访问https://nodejs.org/zh-cn/download，选择LTS长期支持版（推荐22.x），根据系统（Windows/macOS/Linux）下载安装包，如图3-1所示。

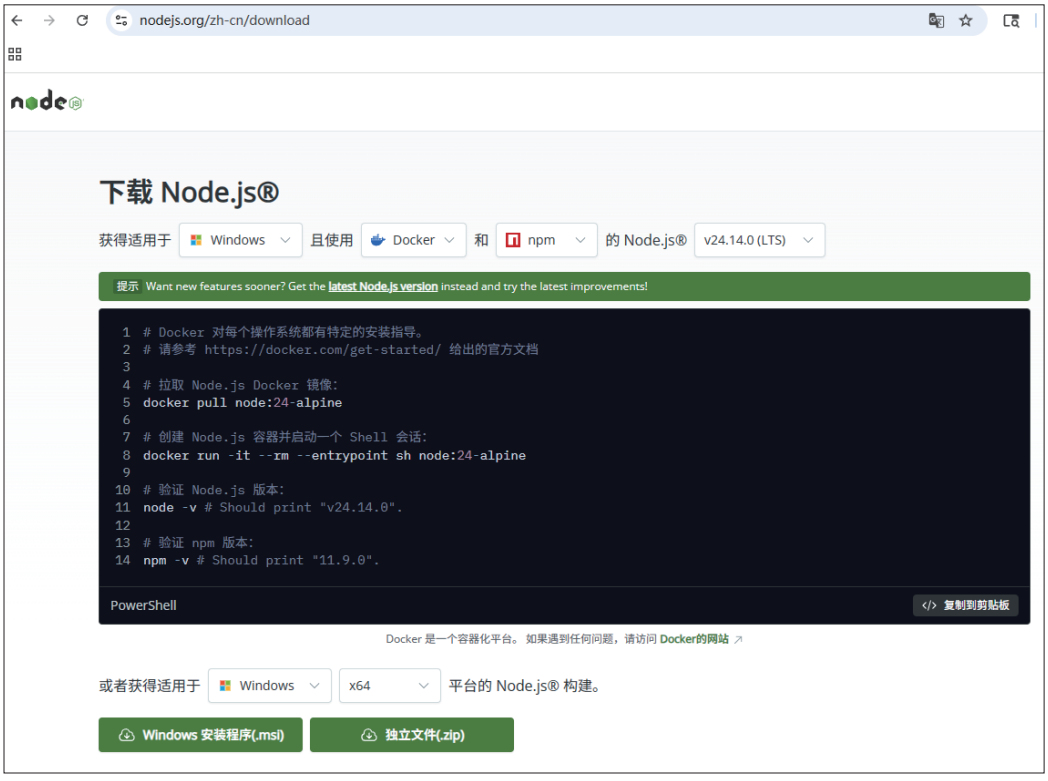


图 3-1　Node.js 安装

执行下面的命令即可。

```
#全局安装OpenClaw
npm install -g openclaw
#或使用pnpm（推荐）
pnpm add -g openclaw
#验证安装
openclaw --versionopenclaw onboard --install-daemon#全局安装OpenClaw
npm install -g openclaw
```

3.2.3 Docker安装

OpenClaw的Docker安装是面向企业级/规模化部署的标准化方式，适配跨系统环境且不需要手动配置依赖。首先确保本地已安装Docker与Docker Compose，通过docker pull openclaw/openclaw:latest拉取官方镜像（国内可使用阿里云/网易镜像加速），接着可直接通过docker run -d -p 18789:18789 --name openclaw openclaw/openclaw快速启动单容器实例（映射18789网关端口），或编写docker-compose.yml配置文件，整合Gateway服务、LLM推理层、记忆存储卷（如-v ~/.openclaw:/root/.openclaw挂载本地目录）与环境变量（如模型密钥、渠道配置），执行docker-compose up -d命令实现多组件一键编排。安装完成后通过docker logs openclaw查看启动日志，访问http://127.0.0.1:18789验证服务。该方式隔离性强、部署与升级便捷，支持集群扩展与私有化部署，无须关心底层系统依赖，是企业级场景的首选安装方式。

前置准备：安装并配置Docker。

Windows系统需要下载Docker Desktop for Windows。

安装时勾选Use WSL2 instead of Hyper-V复选框。

启动Docker Desktop，显示Docker is running即表示成功。

macOS系统需要下载Docker Desktop for Mac。

拖拽到应用程序，启动后状态栏出现Docker图标即可。

OpenClaw Docker部署基于Node环境镜像，其核心步骤为编写Dockerfile →构建镜像→运行容器→验证安装，同时支持npm /pnpm双包管理器，支持Docker Compose编排。

```
#拉取官方镜像
docker pull openclaw/openclaw:latest
#创建数据目录
mkdir -p ~/.openclaw/{config,data,logs}
#启动容器
docker run -d \
--name openclaw \
--restart always \
-p 18789:18789 \
-v ~/.openclaw/config:/app/config \
-v ~/.openclaw/data:/app/data \
-v ~/.openclaw/logs:/app/logs \
openclaw/openclaw:latestogs:/app/logs \
openclaw/openclaw:latest#拉取官方镜像
docker pull openclaw/openclaw:latest
docker run -d \
--name openclaw \
--restart always \
```

```
-p 18789:18789 \
-v ~/.openclaw/config:/app/config \
-v ~/.openclaw/data:/app/data \
-v ~/.openclaw/logs:/app/logs \
openclaw/openclaw:latest          #拉取官方镜像
docker pull openclaw/openclaw:latest
#创建数据目录
mkdir -p ~/.openclaw/{config,data,logs}
#启动容器
docker run -d \
--name openclaw \
--restart always \
-p 18789:18789 \
-v ~/.openclaw/config:/app/config \
-v ~/.openclaw/data:/app/data \
-v ~/.openclaw/l
```

3.2.4　阿里云一键部署

阿里云一键部署OpenClaw是面向新手与个人/小团队的零代码、可视化云端部署方案，全程不需要命令行与手动配置依赖。方案支持新购服务器部署与现有服务器重置部署两种模式，适配Alibaba Cloud Linux系统，具备开箱即用、稳定可靠、数据私有可控的特点，是云端快速落地OpenClaw的首选方式。

（1）注册并完成阿里云账号实名认证（必需，否则无法创建云服务器实例）。

（2）准备阿里云百炼大模型API-Key（后续配置核心参数，无API-Key会导致OpenClaw核心功能无法使用）。

部署方式：轻量应用服务器一键部署（推荐，新手首选）。

步骤1：进入一键部署入口。

访问阿里云OpenClaw专属部署页面，单击“一键购买并部署”按钮，阅读并勾选“OpenClaw服务协议”复选框（必须勾选，否则无法继续）。

步骤2：配置轻量应用服务器。

表3-2为推荐配置，大家可根据自身需求（测试/正式使用）进行调整，最低配置不低于2核2GB。

表 3-2　轻量应用服务器配置表

配置项	推 荐 参 数	详 细 说 明
实例规格	2核2GB（最低）/2核4GB（推荐）	低于2核2GB内存，易导致OpenClaw服务启动失败或频繁崩溃
镜像	应用镜像 → 搜索“OpenClaw（原Moltbot/Clawdbot）”	官方预制镜像，已预装 Node.js、Python 及 OpenClaw 所有依赖，无须手动安装
购买时长	月付/季付/年付/按量付费	短期测试（1～7天）选按量付费更划算，长期使用选月付/年付更优惠
登录密码	大小写字母+数字+特殊符号（如：OpenClaw@2026）	用于远程登录服务器，建议设置复杂的密码，避免泄露

步骤3：提交订单并支付。

核对所有配置参数，确认无误后，勾选《阿里云服务条款》《OpenClaw服务协议》，单击“立即

支付”按钮，支付完成后，等待实例自动创建（3~5分钟，期间无须操作）。

步骤4：进入实例应用详情。

（1）登录阿里云控制台。

（2）在左侧导航栏找到“轻量应用服务器”，单击进入相应页面。

（3）在实例列表中，找到刚刚创建的OpenClaw实例（可通过实例名称筛选），单击实例名称进入详情页。

（4）切换至“应用详情”选项卡，查看OpenClaw应用的相关信息（如端口、配置路径等）。

步骤5：核心配置（一键操作，无须手动输入命令）。

（1）放行端口：在“应用详情”选项卡中，找到“端口放行”模块，单击“一键放行”，系统会自动开放18789端口（OpenClaw服务默认端口，不可随意修改）。

（2）配置百炼API-Key：在“应用详情”选项卡中找到“配置API-Key”模块，单击“一键配置”按钮。在弹出的输入框中，粘贴已获取的阿里云百炼API-Key（确保无多余空格、无拼写错误）。单击“执行命令”按钮，系统会自动将API-Key写入OpenClaw配置文件，无须手动修改。

（3）生成访问Token：在“应用详情”选项卡中，找到“生成访问Token”模块，单击“执行命令”按钮，系统会生成管理员访问Token（务必保存好此Token，首次登录OpenClaw控制台时需要使用）。

步骤6：访问并使用OpenClaw。

（1）访问方式：在“应用详情”选项卡中单击“打开网站”，或手动在浏览器输入：http://服务器公网IP:18789（服务器公网IP可在实例详情页查看）。

（2）首次登录：打开页面后会提示设置管理员密码，设置完成后，输入之前生成的访问Token，单击“登录”按钮，即可进入OpenClaw控制台，完成部署。

阿里官方有更详细的部署教程（可访问相应的页面）以及更多的部署方式，如ECS云服务器一键部署（适合进阶用户/自定义配置）、SAE Serverless一键部署（无服务器，弹性扩容，适合轻量使用），还包括部署的常见问题与排查方法。

3.2.5 macOS安装

本小节提供在macOS系统上完整、详细、可复现的OpenClaw部署流程，包含环境准备、依赖安装、软件部署、初始配置、功能验证及常见问题解决方案，适用于所有主流macOS版本（macOS Ventura 13及以上优先推荐）。

macOS安装Openclaw的基础要求如下。

- 操作系统：macOS Ventura 13.0 及以上版本。
- 内存要求：至少 4GB RAM（推荐 8GB 以上）。
- 存储空间：至少 2GB 可用磁盘空间。
- 网络环境：稳定的互联网连接（需访问 GitHub、Homebrew、Node.js 官方源）。
- 权限要求：当前用户需具备管理员权限。

进入官网安装，如图3-2所示。

复制到控制台并运行命令：curl -fsSL https://openclaw.ai/install.sh | bash，如图3-3所示。

如果遇到图3-3所示的提示，说明缺少依赖环境，依次安装Brew和Node。

注意：目前OpenClaw仿冒品较多，请务必通过官方网站获取安装信息，避免安装恶意软件。

更详细的OpenClaw官方安装教程和疑难解答地址如下。

1. 官方主站安装页（中文）

网址为https://openclaws.io/zh/install/，教程页面如图3-4所示。

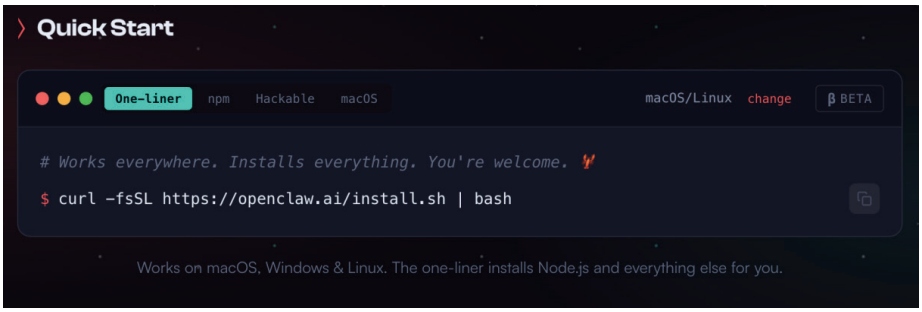


图 3-2　官网安装

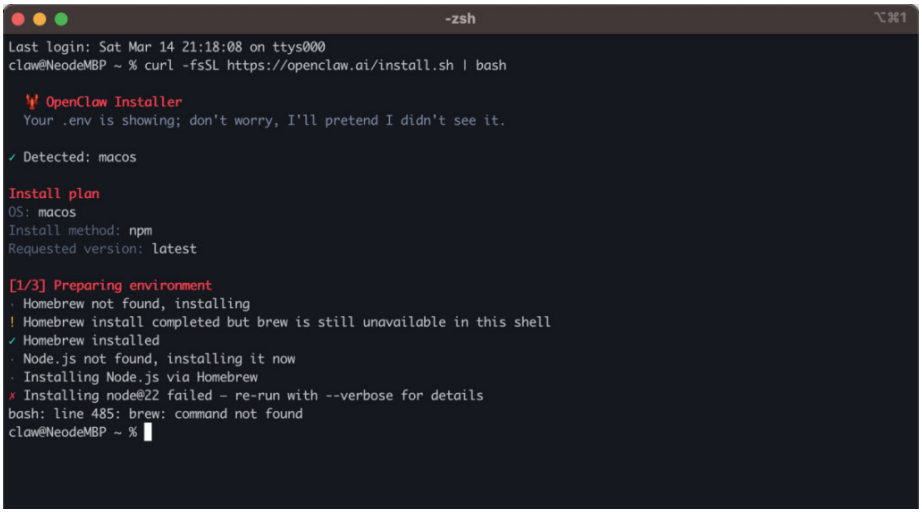


图 3-3　在控制台运行安装命令

图 3-4　安装 OpenClaw 的官方教程页面

2. 官方文档站安装指南（英文/中文）

网址为https://docs.openclaw.ai/install/。

3. GitHub文档入口

网址为https://github.com/openclaw/openclaw/blob/main/docs/install/index.md。

3.3 安装验证与版本管理（常见安装失败问题排查）

OpenClaw安装完成后，文件夹如图3-5所示。

图 3-5 安装 OpenClaw 后的文件夹

OpenClaw安装完成后，双击打开gateway.cmd文件可以输入命令。需要通过下面的命令验证安装是否成功。

```
#检查OpenClaw版本
openclaw --version
#检查服务状态
openclaw status
#查看实时日志
openclaw logs --foflow#检查OpenClaw版本
openclaw --version
```

常见安装问题及解决方案如下。

问题1：Node.js版本过低。

Error: Node.js version 18.x is not supported, minimum version is 22.0.0

解决方案：升级Node.js到22.0.0或更高版本。

```
#安装 Node.js 22 最新版本
nvm install 22
#切换使用 Node.js 22
nvm use 22
#验证是否成功
node -v
```

问题2：pnpm安装失败。

Err: EACCES permission denied

解决方案：配置npm镜像源后重试。

```
npm config set registry https://registry.npmmirror.com/
npm install -g pnpmnpm config set registry https://registry.npmmirror.com/
```

问题3：端口18789被占用。

Error: listen EADDRINUSE: address already in use :::18789

解决方案：更换端口或释放占用。

```
#查看占用进程
netstat -ano | findstr 18789
#更换端口启动
openclaw gateway --port 18790   #查看占用进程
netstat -ano | findstr 18789
```

问题4：权限不足。

Error: EACCES: permission denied

解决方案：Windows系统使用管理员PowerShell，Linux/macOS系统使用sudo。

OpenClaw的安装验证与版本管理是保障环境可用性、适配不同功能需求的核心环节，操作简捷且标准化。安装完成后，可通过终端执行openclaw -v或openclaw version快速验证安装是否成功（输出版本号则代表环境正常），若需深度校验服务可用性，可执行openclaw doctor自动诊断网关端口、模型连通性、依赖完整性等关键项并给出修复建议。在版本管理方面，升级稳定版可执行pip install --upgrade openclaw（Python版）或npm update -g openclaw（Node.js版）命令；Docker部署则通过docker pull openclaw/openclaw:latest拉取最新镜像；如需回滚至指定版本，可在安装命令后指定版本号（如pip install openclaw==2.2.0）；Docker可指定镜像标签（如openclaw/openclaw:2.2.0），同时支持通过openclaw version list查看本地已安装版本与官方最新版本。该套流程适配所有安装方式，能快速确认环境状态、灵活切换版本，避免因版本不兼容导致的功能异常。

3.4　开发工具推荐（提升实操效率的辅助工具）

为了让OpenClaw运行更加高效，建议进行以下优化配置。

1. 镜像源配置（国内用户必做）

由于国内网络访问海外npm仓库较慢，建议配置国内镜像。

```
#配置npm镜像
npm config set registry https://registry.npmmirror.com/                #配置pnpm镜像
pnpm config set registry https://registry.npmmirror.com/
```

2. Docker加速配置（可选）

如果使用Docker部署，可以加速配置Docker镜像。

```
#编辑Docker配置
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<EOF
{
    "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker#编辑Docker配置
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<EOF
{
    "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart dockerdocker/daemon.json <<EOF
{
    "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]
}
```

3. 时区配置

确保系统时区设置正确，特别是使用定时任务功能时。

```
sudo timedatectl set-timezone Asia/Beijing          #设置时区为北京时间
sudo timedatectl set-timezone Asia/Shanghai         #设置时区为上海时间
```

对于高并发场景，可以调整Node.js的内存限制。

```
#高并发场景下设置 Node.js 内存限制
export NODE_OPTIONS="--max-old-space-size=4096"
openclaw gateway
```