第 4 章

OpenClaw 的日常维护、版本升级与安全防护

安装部署只是起点，要让 OpenClaw 长期稳定地工作，日常的维护、升级、API 配置和安全防护同样重要。本章主要内容包括日常的更新维护，即如何检查版本、备份数据、查看日志和排查故障；API 配置详解，从新手友好的内置模型一键配置，到进阶用户的自定义 API 手动编辑；版本升级指南，完整的升级流程、常见坑点和回滚方案；安全防护，安全模型、已知安全事件和 Skills 安全的全面梳理。建议按需阅读，如果你刚装好只想赶紧用起来，则可以先跳到 API 配置部分把模型接好；等用过一段时间再回来看维护、升级和安全的相关内容。

版本检查、数据备份、日志查看和常见故障排查并不复杂，但养成良好的维护习惯能让你在遇到问题时从容应对。

4.1 日常维护与版本更新

OpenClaw 是一个活跃迭代的开源项目，社区几乎每周都有新的提交和发布。定期更新可以获得新功能、性能优化和安全修复，但更新前一定要做好备份，因为这是所有运维工作基本的原则。本节介绍日常维护中常用的操作：版本检查与更新、数据备份、日志 / 监控和故障排查。

1. 版本检查与更新

在做任何维护操作之前，都要先确认当前自己使用的 OpenClaw 版本，以及 OpenClaw 的最新版本是什么，代码如下：

```
# 检查当前版本
openclaw --version

# 检查最新版本
curl -s https://api.github.com/repos/openclaw/openclaw/releases/latest |
grep tag_name
```

确认有新版本后，可以选择以下方式进行更新，代码如下：

```
# 方式一：使用安装脚本
curl -fsSL https://openclaw.ai/install.sh | bash

# 方式二：手动更新
```

```
cd ~/openclaw
git pull origin main
pnpm install
pnpm build
```

2. 数据备份

OpenClaw 的核心数据包括配置文件、对话历史、模型缓存、日志及 IM 平台接入信息，定期备份可有效避免数据丢失。下面分别介绍本地安装和 Docker 部署的备份方法。

本地安装备份，代码如下：

```
# 备份配置和数据
tar -czf openclaw-backup-$(date +%Y%m%d).tar.gz ~/.openclaw

# 恢复数据
tar -xzf openclaw-backup-20260210.tar.gz -C ~/
```

Docker 备份，代码如下：

```
# 备份数据卷
docker run --rm \
-v ~/.openclaw:/data \
-v $(pwd):/backup \
alpine tar czf /backup/openclaw-backup-$(date +%Y%m%d).tar.gz /data

# 恢复数据
docker run --rm \
-v ~/.openclaw:/data \
-v $(pwd):/backup \
alpine tar xzf /backup/openclaw-backup-20260210.tar.gz -C /
```

3. 日志和监控

日志是排查故障、监控服务状态的核心依据，下面分别介绍查看日志和监控服务指标的方法。

查看日志，代码如下：

```
# 本地安装
tail -f ~/.openclaw/logs/gateway.log

# Docker
docker logs -f openclaw
```

监控指标，代码如下：

```
# 查看系统状态
openclaw gateway status

# 查看资源使用
```

```
openclaw stats

# 查看 API 消耗
openclaw stats api
```

4. 故障排查

本节针对日常使用中常见的三类故障，提供详细的排查步骤和解决方案，帮助用户快速定位并解决问题。

Gateway 无法启动的排查步骤及解决方案，代码如下：

```
# 查看日志
openclaw logs

# 检查端口占用
lsof -i :18789

# 重启 Gateway
openclaw gateway restart
```

API 连接失败的排查步骤及解决方案，代码如下：

```
# 测试 API 连接
openclaw test api

# 检查 API Key
openclaw config get models.providers
```

性能问题的排查步骤及解决方案，代码如下：

```
# 清理缓存
openclaw cache clear

# 重启服务
openclaw gateway restart
```

5. 卸载

若需彻底卸载 OpenClaw，可根据部署方式选择对应命令。

本地安装卸载代码如下：

```
# 停止服务
openclaw gateway stop

# 删除文件
rm -rf ~/.openclaw
rm -rf ~/openclaw
```

```
# 删除命令
npm uninstall -g openclaw
```

Docker 卸载代码如下：

```
# 停止并删除容器
docker stop openclaw
docker rm openclaw

# 删除镜像
docker rmi openclaw/openclaw

# 删除数据
rm -rf ~/.openclaw
```

4.2 API 配置详解

OpenClaw 本身不包含 AI 模型，因为它是一个连接器和调度器，通过 API 调用外部的大语言模型来完成对话和任务。因此，正确配置 API 是使用 OpenClaw 的前提。本节将从零开始，详细讲解两种配置方式：适合新手的内置模型一键配置，以及面向进阶用户的自定义 API 手动配置。不管选择哪种方式，读完本小节内容都能顺利地把模型接好。

1. 模型供应商总览

OpenClaw 支持十余家模型供应商，从国际顶尖到国产平价再到完全免费的本地模型，覆盖所有预算和场景。通过 ~/.openclaw/openclaw.json 配置文件，可以灵活切换主力模型、设置 Fallback 备选链，甚至让不同任务走不同模型。

模型供应商总览如表 4–1 所示。

表 4–1　模型供应商总览

供应商	代表模型	输入价格 /1M tokens	输出价格 /1M tokens	接入方式	推荐场景
Anthropic	Claude Sonnet 4.6	$3.00	$15.00	内置 Provider	Agent 任务效果最佳
OpenAI	GPT-5.4	$2.50	$15.00	内置 Provider	通用能力强
Google	Gemini 3.1 Pro	$2.00	$18.00	内置 Provider	多模态、超长上下文
DeepSeek	DeepSeek-V3.2	$0.14	$0.28	自定义 Provider	极致低价、代码任务
智谱 GLM	GLM-5	$0.80	$2.56	内置（zai）	国产最强代码能力
通义千问	Qwen3.5 Max	$1.20	$6.00	插件（OAuth）	中文 NLP、代码生成

续表

供应商	代表模型	输入价格 /1M tokens	输出价格 /1M tokens	接入方式	推荐场景
豆包	Seed 2.0 Pro	$0.47	$2.37	自定义 Provider	批量处理、低成本
百度文心	文心 5.0	~$0.58	~$1.16	自定义（需适配）	百度云生态用户
Kimi	Kimi K2.5	$0.60	$3.00	自定义 Provider	中文 Agent、长上下文
MiniMax	MiniMax M2.5	$0.50	$2.00	自定义 Provider	SWE-bench 高分、性价比
Ollama	Qwen3.5-Coder:32B	免费	免费	自动发现	隐私敏感、零成本
LM Studio	Devstral-24B	免费	免费	自定义 Provider	本地 GUI、模型测试

注：以上价格为 2026 年 3 月的参考价格，各供应商可能随时调整。Google Gemini 3.1 Pro 于 2026 年 2 月发布，为当前最新旗舰模型。DeepSeek V4 预计即将发布，届时价格和性能可能有较大变化。

2. 配置核心概念

理解三个关键概念，就能掌握 OpenClaw 的模型配置：内置 Provider（Anthropic、OpenAI、Google、智谱等）无须额外配置，设置 API Key 即可使用；自定义 Provider（DeepSeek、豆包、Kimi 等）需要在 models.providers 中手动添加；Fallback 机制——主模型不可用时自动切换到备选，这是最核心的省钱策略。

angents 是核心执行单元配置块，defaults 表示全局默认的 agent 配置。

```
{
  env: { "API_KEY_NAME": "sk-xxx" },
  agents: {
    defaults: {
      model: {
        primary: "provider/model-name",  // 主力模型
        fallbacks: ["provider/model-b"]  // 备选
      }
    }
  },
  models: {
    mode: "merge",   // 保留内置provider，叠加自定义
    providers: { /* 自定义provider配置 */ }
  }
}
```

3. 核心建议

设置 models.mode: "merge" 非常重要，这样可以保留所有内置 Provider 的同时叠加自定义配置。如果不设置，自定义配置会覆盖内置 Provider。

4. API 模型分类

OpenClaw 支持两种 API 模型配置方式，面向不同技术水平的用户，适配不同使用场景，具体分类如下。

（1）内置 API 模型：预集成主流模型厂商的连接参数，配置极简，无须手动编写配置文件，推荐新手使用。

（2）自定义 API：需手动编辑配置文件，可接入非内置模型、第三方代理或自建模型服务，灵活性极高，适合进阶用户。

4.2.1 内置 API 模型

1. 什么是内置 API 模型

所谓“内置 API 模型”，是指 OpenClaw 已经在代码中预先写好了各大主流模型的连接参数（API 地址、协议格式、默认参数等）。用户不需要知道这些技术细节，只要做三件事：到模型供应商的官网注册账号并获取 API Key、在 OpenClaw 的配置向导中选择对应的供应商、把 API Key 粘贴进去，就可以开始使用了。

2. 支持的内置模型

截至 2026 年年初，内置支持以下主流模型（列表会随 OpenClaw 版本更新而扩充，具体以实际安装版本为准），推荐国内用户优先选择国内模型，其价格便宜、访问速度快、中文能力强。

- Moonshot AI（Kimi）：长文本处理专家，支持 200 万字超长上下文，处理论文、报告等长文档的首选。
- DeepSeek（深度求索）：价格极低的性价比之王，推理和编程能力出色。
- 智谱 GLM（已更新至 GLM-5）：中文理解优秀，支持图像等多模态输入。
- 通义千问（Qwen，已更新至 Qwen 3.5）：阿里出品，生态完善、稳定可靠。
- MiniMax：对话风格自然流畅，创意写作能力突出。
- 百度文心：中文语料训练最丰富。
- 字节豆包：性价比高，响应速度快。
- OpenAI（GPT-4o/o1/o3 系列）：综合能力领先的老牌选手，生态最完善。
- Anthropic（Claude 4.5/4.6 系列）：推理深度和代码能力出色，安全性行业领先。
- Google（Gemini）：多模态能力强（图片、视频理解），免费额度慷慨。
- Groq：基于自研 LPU 芯片，推理速度极快，适合对延迟敏感的场景。

内置模型的优势在于配置极简，不需要手动编写 JSON 配置文件，通过向导就可完成；参数已调优，上下文窗口大小、最大输出长度等参数都已针对 OpenClaw 的使用场景优化过；自动跟随更新，当 OpenClaw 升级时，内置模型列表也会同步更新；学习成本低，对于不想折腾配置文件的用户，这是最省心的方式。

内置模型适合以下用户：刚接触 OpenClaw 想快速跑通的新手、使用主流大模型且不需

要特殊定制的普通用户、不想深入了解 JSON 配置文件格式的非技术用户。

4.2.2 自定义 API

1. 什么是自定义 API

以下几种情况需要手动编辑配置文件来添加自定义 API：想使用 OpenClaw 尚未内置的新模型或小众模型、通过第三方 API 代理服务（如 OpenRouter、API2D）来中转访问国外模型、连接自己搭建的私有模型服务（如本地部署的开源模型）或者对接企业内部的模型网关接口。

这些场景都需要手动告诉 OpenClaw：API 的地址在哪里、用什么密钥认证、用什么协议通信、有哪些模型可用。

2. 配置方式

具体操作是编辑 ~/.openclaw/openclaw.json 配置文件（Windows 系统路径为 C:\Users\ 用户名 \.openclaw\openclaw.json），在其中添加供应商信息。每个供应商需要指定四个关键参数：baseUrl（API 服务的地址，如 https://api.deepseek.com）、apiKey（认证密钥，在供应商官网获取）、api（通信协议类型，不同供应商使用不同的协议），以及 models（该供应商可用的模型列表，包括每个模型的上下文窗口大小和最大输出长度等参数）。

3. 自定义配置的优势与劣势

自定义配置的优势是灵活性极高：理论上任何兼容 OpenAI Chat API 格式的模型服务都可以接入；可以精确控制每个模型的参数（如调大上下文窗口、修改默认温度等）；同时支持配置多个供应商并设置主备切换策略。代价是需要了解 JSON 格式和 API 基础知识，配置出错时排查也相对复杂。

自定义配置的劣势：JSON 格式对语法要求严格，少一个逗号或引号都会导致解析失败；配置参数需要手动维护，供应商接口变更时需要自己跟进调整；填错参数（如 API 地址拼写错误、协议类型不匹配）可能导致 OpenClaw 完全无法调用模型，且报错信息有时不够直观。

适合以下用户：有一定技术基础的进阶用户、需要接入非主流模型或自建模型的开发者、企业环境下需要对接内部 API 网关的运维人员，以及需要精细调控模型参数的 AI 应用开发者。

4. 配置方式对比

配置方式对比如表 4–2 所示，清晰呈现两种方式的核心差异，方便用户根据自身情况选择。

表 4–2　配置方式比较

特性	内置 API 模型	自定义 API
配置难度	简单	复杂
适用人群	新手	进阶用户
模型选择	主流模型	任意模型
配置方式	向导选择	手动编辑
维护成本	低	高
灵活性	中	高

5. 推荐配置路径

如果不确定该用哪种方式，参考以下路径。新手推荐路径：

第一步：使用内置 API 模型快速跑通（5 分钟搞定）。

第二步：优先选择国产模型（如 DeepSeek、Kimi），价格便宜，国内访问速度快。

第三步：通过 openclaw onboard 配置向导完成设置，全程交互式操作无需编辑文件。

第四步：先用起来体验一段时间，等到确实有需求时再考虑自定义 API 配置。

进阶用户推荐路径：

第一步：先用内置模型把 OpenClaw 的基本功能跑熟。

第二步：阅读本节后面的配置文件结构说明，理解 JSON 配置的基本格式。

第三步：根据实际需求（如接入企业内部模型、使用第三方代理等）手动添加自定义 API。

第四步：修改配置后务必重启 Gateway 并测试，确认能正常对话后再投入日常使用。

4.2.3 内置 API 模型配置

Kimi（推荐）是月之暗面公司推出的大语言模型，最大的特点是支持 200 万字的超长上下文窗口，这意味着把一整篇论文、一份完整的项目文档甚至一本书扔给它处理，它都能“记住”全部内容并给出回答。中文理解能力在国产模型中名列前茅，特别适合论文分析、报告总结、长文档处理等中文场景。如果使用量较大，建议购买官方套餐，比按量付费更划算，配置步骤如下。

■ 第一步：访问 Kimi Code 平台。

访问：https://www.kimi.com/code，使用手机号或邮箱注册并登录账号。

■ 第二步：购买套餐（可选）。

使用提示：OpenClaw 在每次对话中会发送较长的系统提示词（包括人设配置、工具描述等），导致实际 token 消耗比普通聊天高很多。因此，如果日常使用（每天 10 次以上对话），购买 Kimi 的按月套餐通常比按量付费节省 30% ~ 50% 的费用。

■ 第三步：创建 API Key。

登录后，单击页面右上角“控制台”，进入控制台页面，找到“API Keys”选项，单击“创建 API Key”按钮，名称可随意填写，单击“创建”按钮。

■ 第四步：保存 API Key。

提醒：API Key 只在创建时完整显示一次，单击“完成”按钮后就无法再查看了。请务必立即复制并保存到安全的地方（如密码管理器、加密文档）。如果不小心丢失了，只能删除这个 Key 再重新创建。

■ 第五步：配置到 OpenClaw，代码如下：

```
# 运行配置向导
openclaw onboard
```

```
# 配置流程:
# 1. 选择 QuickStart
# 2. 选择模型供应商:Moonshot AI
# 3. 粘贴刚才复制的API Key
# 4. 选择默认模型:kimi-code/kimi-for-codi
# 5. 完成其他配置
```

Kimi 使用成本估算（基于 2026 年年初的价格，模型和套餐可能随时调整）：轻度使用（每天 3 ~ 5 次对话）为 10 ~ 20 元 / 月；中度使用（每天 10 ~ 20 次对话）为 30 ~ 50 元 / 月；重度使用（每天 50 次以上）强烈建议购买月度套餐以控制成本。

4.2.4 自定义 API 配置

以下内容面向有一定技术基础的进阶用户。如果是新手，建议先用内置 API 模型跑通 OpenClaw，等熟练之后再来看这部分内容。

1. 什么时候需要自定义 API

通常以下几种场景需要手动编辑配置文件。

（1）使用非内置模型。

- 想使用的模型 OpenClaw 还没有内置（比如刚发布的新模型、小众的垂直领域模型）。
- 模型供应商更新了新版本但 OpenClaw 还没跟进适配（通常会有几天到几周的延迟）。
- 区域限定的模型（仅特定地区可访问，需通过自定义配置适配）。

（2）使用第三方代理。

- 使用第三方 API 代理服务（如 OpenRouter、API2D 等），它们提供统一的中转接口来访问多个模型供应商。
- 对接企业内部的 API 网关（企业环境下，模型服务部署在内部网络，需自定义配置访问地址）。
- 自建的模型服务（如本地部署的 Llama、Qwen 等开源模型，需配置本地 API 地址）。

（3）精细控制参数。

- 自定义模型参数（如温度、采样率，调整对话的随机性和逻辑性）。
- 调整上下文窗口大小（根据需求扩大或缩小，平衡性能和体验）。
- 修改默认配置（如默认模型、最大输出长度等）。

2. 配置文件位置

配置文件路径及编辑配置文件如下：

```
# 配置文件路径
~/.openclaw/openclaw.json

# 编辑配置文件
```

```
nano ~/.openclaw/openclaw.json
```

3. 配置文件结构

配置文件为 JSON 格式，核心结构包括 providers 数组（存储所有供应商配置），每个供应商包含 name、baseUrl、apiKey、api、models 五个核心字段，配置文件结构代码如下：

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "你的供应商名称": {
        "baseUrl": "API服务地址",
        "apiKey": "你的API密钥",
        "auth": "认证方式",
        "api": "API协议类型",
        "models": [
          {
            "id": "模型ID",
            "name": "模型显示名称",
            "contextWindow": 上下文窗口大小,
            "maxTokens": 最大输出tokens
          }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "供应商名称/模型ID"
      }
    }
  }
}
```

4. 示例 1：配置第三方 API 代理

如果使用 API 代理服务（如 OpenRouter），配置如下：

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "openrouter": {
        "baseUrl": "https://openrouter.ai/api/v1",
        "apiKey": "sk-or-v1-你的密钥",
```

```
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "anthropic/claude-4.6-sonnet",
            "name": "Claude 4.6 Sonnet",
            "contextWindow": 200000,
            "maxTokens": 8192
          },
          {
            "id": "openai/gpt-4",
            "name": "GPT-4",
            "contextWindow": 128000,
            "maxTokens": 4096
          }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "openrouter/anthropic/claude-4.6-sonnet"
      }
    }
  }
}
```

5. 示例 2：配置多个模型供应商

在实际使用中，同时配置多个供应商是非常实用的策略，以下示例展示了如何在一个配置文件中同时配置多个供应商：

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "deepseek": {
        "baseUrl": "https://api.deepseek.com",
        "apiKey": "sk-你的DeepSeek密钥",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "deepseek-chat",
            "name": "DeepSeek Chat",
```

```
            "contextWindow": 64000,
            "maxTokens": 4096
          }
        ]
      },
      "moonshot": {
        "baseUrl": "https://api.moonshot.cn/v1",
        "apiKey": "sk-你的Kimi密钥",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "moonshot-v1-128k",
            "name": "Kimi 128K",
            "contextWindow": 128000,
            "maxTokens": 4096
          }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "deepseek/deepseek-chat",
        "fallback": "moonshot/moonshot-v1-128k"
      }
    }
  }
}
```

6. 配置参数说明

配置文件中各核心参数的详细说明如表 4–3 所示，可帮助用户理解每个参数的作用，避免配置错误。

表 4–3　参数说明和示例

参数	说明	示例
baseUrl	API 服务地址	https://api.deepseek.com
apiKey	API 密钥	sk-xxx
auth	认证方式	api-key 或 bearer
api	API 协议	openai-chat、anthropic-messages
id	模型 ID	deepseek-chat

续表

参数	说明	示例
name	显示名称	DeepSeek Chat
contextWindow	上下文窗口	64000
maxTokens	最大输出	4096

7. 常见 API 协议类型

不同模型供应商对应不同的通信协议，配置时需确保协议类型与供应商一致，否则会导致 API 调用失败。常见 API 协议类型如下。

- openai-chat：OpenAI 兼容接口（最常用，适用于 Kimi、通义千问等大部分国产模型及 OpenAI、OpenRouter 代理）。
- anthropic-messages：Anthropic Claude 接口（仅适用于 Claude 系列模型，不兼容 OpenAI 格式）。
- google-generative-ai：Google Gemini 接口（仅适用于 Gemini 系列模型）。
- azure-openai：Azure OpenAI 接口（适用于 Azure 部署的 OpenAI 模型）。

8. 配置后重启服务

自定义 API 配置完成后，必须重启 OpenClaw 网关服务，配置才能生效。重启服务代码如下：

```
# 方式1：重启Gateway
openclaw gateway restart

# 方式2：停止后重新启动
openclaw gateway stop
openclaw gateway start

# 方式3：查看Gateway状态
openclaw gateway status
```

9. 版本升级指南

OpenClaw 的版本升级需要比日常更新更加谨慎——涉及可执行文件的替换、配置文件的迁移和后台服务的重启。升级前务必备份，升级后务必验证。下面会给出完整的升级流程、每一步的注意事项、常见问题的排查方法，以及出问题时的回滚方案。

截至 2026 年 3 月，推荐使用 2026.3.7 版本。2026.2.12 版本存在一个已确认的严重 bug（GitHub Issue #15141），会导致 Session 路径验证失败，进而影响 heartbeat 心跳功能和飞书 / Telegram 等平台的消息处理。如果读者还在使用旧版本，建议直接升级到 2026.3.7 版本。

4.2.5 OpenClaw 的升级方式

OpenClaw 的升级方式与初始安装方式强相关，为确保升级过程顺畅、避免兼容问题，需根据自身安装方式选择对应的升级方法。安装方式及其匹配的升级方法如表 4–4 所示。

表 4–4　安装方式及其升级方法

安装方式	升级方法	难度	推荐度
云端部署	重新部署镜像	★★☆☆☆	★★★☆☆
Docker	拉取新镜像	★★☆☆☆	★★★★☆
本地安装（npm）	npm 升级	★☆☆☆☆	★★★★★
本地安装（源码）	git pull	★★★☆☆	★★★☆☆

不同安装方式的升级逻辑存在差异，核心原则为“对应匹配、先备后升”，无论选择哪种方式，升级前备份配置与数据都是首要前提，可有效规避升级失败导致的损失。

1. npm 直接升级（推荐）

该方式是经过 OpenClaw 社区长期验证、最可靠的升级方式，适用于所有通过 npm 全局安装的用户，包括通过一键脚本安装和手动执行 npm install 命令安装的场景。整个升级流程共 7 步，操作耗时约 5 ~ 10 分钟，具体时长取决于网络下载速度和系统性能。

升级核心思路为“先卸后装”——彻底卸载旧版本 OpenClaw，再安装新版本，以此避免旧版本文件残留，防止出现启动异常、功能报错、参数冲突等各类疑难问题，这也是 npm 全局包升级的通用最优实践。

第一步：备份配置。

```
# 备份整个配置目录
cp -r ~/.openclaw ~/.openclaw.backup-$(date +%Y%m%d)

# 验证备份
ls -la ~/.openclaw.backup-*
```

第二步：停止 Gateway 服务。

```
# 停止Gateway
openclaw gateway stop

# 验证已停止
openclaw gateway status
```

第三步：卸载旧版本。

```
# 卸载旧版本
npm uninstall -g openclaw

# 验证卸载
which openclaw # 应该没有输出
```

第四步：安装新版本。

```
# 安装推荐版本 2026.3.7（需要使用--force参数）
npm install -g openclaw@2026.3.7 --force
```

版本选择说明：推荐安装 2026.3.7 版本，这是当前经社区验证最稳定的版本，无已知重大 bug，适配绝大多数使用场景。

--force **参数说明**：npm 在全局安装软件时，若检测到同名可执行文件（来自旧版本卸载残留或其他场景），会触发 EEXIST 错误并拒绝安装。添加 --force 参数的作用是告诉 npm 强制覆盖已有文件，该操作是 npm 升级全局包的常规操作，不会对系统环境、其他软件及 OpenClaw 配置数据造成任何副作用，可放心使用。

第五步：修复配置。

```
# 运行doctor工具自动修复配置
openclaw doctor --fix
```

doctor 工具是 OpenClaw 内置的“全科医生”，它会自动完成以下修复工作：更新 Gateway 后台服务的入口点路径（指向新版本的可执行文件）、检查配置文件格式是否与新版本兼容、自动修复已知的配置迁移问题、在 macOS 上重新安装 LaunchAgent 开机自启服务或在 Linux 上更新 systemd 服务文件。运行这一步可以避免 90% 的升级后异常问题。

第六步：重启 Gateway。

```
# 重启Gateway
openclaw gateway restart

# 等待几秒后检查状态
sleep 5
openclaw gateway status
```

第七步：验证升级。

```
# 检查版本
openclaw --version

# 检查Gateway状态
openclaw gateway status

# 测试连接
openclaw channels status
```

成功的输出应该显示：

```
2026.3.7

Runtime: running (pid xxxxx, state active)
RPC probe: ok
Listening: *:18789
Dashboard: http://127.0.0.1:18789/
```

2. 官方脚本升级

如果不想手动执行上面的七步操作，也可以使用官方提供的一键升级脚本，它会自动处

理大部分步骤，代码如下：

```
# 运行升级脚本
curl -fsSL https://openclaw.ai/install.sh | bash
```

该方式的优点是一键完成、自动处理依赖关系；缺点是在某些环境下可能遇到 npm 权限错误或网络超时，可靠性不如手动的“先卸后装”方式。

3. Docker 升级

Docker 用户的升级体验最省心，不需要卸载任何东西，只需三条命令：停止旧容器、拉取新镜像、启动新容器，代码如下：

```
# 停止并删除旧容器
docker compose down

# 拉取最新镜像
docker compose pull

# 启动新容器
docker compose up -d openclaw-cn-gateway

# 查看日志
docker compose logs -f openclaw-cn-gateway
```

4. 云端部署升级

本节以腾讯云 Lighthouse（轻量应用服务器）部署为例，讲解云端 OpenClaw 的升级方法，该方式操作简单，无须手动处理依赖，升级后配置会自动保留，具体步骤如下：

（1）登录腾讯云官网，进入“轻量应用服务器”控制台。

（2）在服务器列表中，找到已部署 OpenClaw 的实例，单击实例名称进入详情页。

（3）在详情页左侧导航栏，找到“应用管理”，单击“重置应用”。

（4）在弹出的窗口中，选择最新版本的 OpenClaw 官方镜像，确认重置。

（5）等待重置完成，重置过程中服务器会自动重启。

（6）重置完成后，登录 OpenClaw Web 管理界面，重新配置 API 相关参数。

注意：其他云端部署（如阿里云 ECS、华为云弹性云服务器）的升级方法，可参考对应云厂商的应用重置教程，核心逻辑与腾讯云 Lighthouse 一致，均为通过官方镜像重置实现升级。

4.2.6 OpenClaw 升级故障排查

1. 问题 1：npm install 报错 EEXIST

症状代码如下：

```
npm error code EEXIST
npm error path /usr/local/bin/openclaw
npm error EEXIST: file already exists
```

解决方案代码如下：

```
# 使用--force参数强制覆盖
npm install -g openclaw@2026.3.7 --force
```

2. 问题 2：Gateway 启动失败

症状代码如下：

```
Gateway not running
```

解决方案代码如下：

```
# 运行doctor修复
openclaw doctor --fix

# 重启Gateway
openclaw gateway restart

# 查看详细日志
tail -f ~/.openclaw/logs/gateway.log
```

3. 问题 3：配置文件版本不匹配

症状代码如下：

```
Config was last written by a newer OpenClaw (2026.2.6-3);
current version is 2026.2.1-zh.3
```

解决方案代码如下：

```
# 升级到推荐版本 2026.3.7
npm install -g openclaw@2026.3.7 --force
openclaw doctor --fix
```

4. 问题 4：插件加载失败

症状代码如下：

```
plugin not found: xxx
```

解决方案代码如下：

```
# 重新安装插件
openclaw plugins install <plugin-name>

# 或禁用有问题的插件
openclaw config set plugins.allow []
```

5. 问题 5：端口被占用

症状代码如下：

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

解决方案代码如下：

```
# 查找占用端口的进程
lsof -i :18789

# 停止旧的Gateway进程
kill -9 <PID>

# 或修改端口
openclaw config set gateway.port 18790
openclaw gateway restart
```

4.2.7 回滚到旧版本

如果 OpenClaw 升级后遇到问题，可以回滚到旧版本。

1. 方式一：从备份恢复

该方式适用于升级前已执行备份操作的用户，可快速恢复旧版本的配置和程序，回滚代码如下：

```
# 停止Gateway
openclaw gateway stop

# 恢复配置
cp -r ~/.openclaw.backup-20260213/* ~/.openclaw/

# 降级到旧版本（以中文版为例）
npm uninstall -g openclaw
npm install -g @qingchencloud/openclaw-zh@2026.2.1-zh.3 --force

# 重启Gateway
openclaw gateway restart
```

2. 方式二：安装指定版本

若未执行备份操作，可直接卸载新版本，安装升级前的指定旧版本，代码如下：

```
# 查看可用版本
npm view openclaw versions

# 安装指定版本
npm install -g openclaw@2026.2.8 --force

# 修复配置
openclaw doctor --fix

# 重启Gateway
openclaw gateway restart
```