[OPS-MAN-043] 从 0 到高手：新手成长路线与 pip 安装后清单

> 目的：把"刚 `pip install` 完、完全 0 基础"的你，沿一条**线性阶梯**带到"khy 使用高手"。 每一级只回答三件事：**这一步要达到什么 → 抄哪条命令 → 你由此学会了什么、下一步去哪**。 本文是**导航阶梯**，不重复各专题文档的细节——每级末尾给出"深入读哪篇"。 适用版本：khy-os ≥ 0.1.139（命令以仓库实测为准，非杜撰）。
>
> 📖 **美观阅读版**：本文另有同目录同名的 `.html`（屏幕阅读：浮动目录 + 排版美化）与 `.pdf`（打印 / 分享）版本； 还有把本文与 [OPS-MAN-044] 合在一起的**合订本** `[OPS-MAN-043+044] 从使用入门到开发精通-合订本.{html,pdf}`。 重新生成： `npm run docs:pdf:onboarding`（详见 [OPS-MAN-044] D6）。

0. 一句话路线图

最短可用路径（3 条命令就能开聊）：

```
khy preflight          # 体检：Node/依赖/PATH 是否就绪
khy gateway config     # 配一个 AI 服务商（填你自己的 key）
khy                    # 进入 agentic 终端，开始对话
```

阶段 0 · 装完第一件事：先体检，别急着用

刚 `pip install khy-os`（或 `npm install -g @khy/os`）之后，**第一条命令永远是体检**， 而不是直接开聊。原因：khy 后端是 Node 写的（需 Node ≥ 20），首次运行还会自动初始化运行目录，

先体检能把"找不到命令""缺依赖""PATH 没配好"这类问题一次性照出来。

你要达到	抄这条命令	它做什么
确认环境就绪	`khy preflight`	集中体检四层启动依赖（Node、Python、依赖包、PATH），**给出可直接粘贴的修复指令**。别名：`khy precheck`。
知道装到哪了	`khy where`	打印可执行文件位置、运行目录、依赖是否安装。别名：`khy which` / `khy whereami`。
能修就顺手修	`khy doctor`	不只体检、还**尝试自动修复**（端口、依赖等）。`--check` 等价于只体检不动手。别名：`khy heal`。

> **preflight 与 doctor 的区别：** `preflight` 只体检 + 给你"自己粘贴执行"的命令；`doctor` 会替你动手修。 拿不准时先 `preflight`，看明白了再 `doctor`。

首次运行会发生什么：第一次跑 `khy` 时会自动创建运行目录与默认配置（auto-init），属正常行为，无需手动 init。

✅ **这一级你学会了**：khy 的"自检三件套" `preflight / where / doctor` ——以后任何"用不起来"先跑它们。 ➡️ **深入读**：[OPS-MAN-022] pip-安装布局参考（装到哪、目录长什么样）、[OPS-MAN-028] 环境要求。

阶段 1 · 配一个 AI：让 khy 真正会说话

khy 自带一个**多服务商 AI 网关**——你**自带 key**，数据留在本地。装完默认还没有任何 key， 所以这一步是"从能跑到能聊"的关键一跳。

你要达到	抄这条命令	它做什么
配置一个服务商	`khy gateway config`	交互式填写一个 AI 服务商（如 Claude / Qwen / Ollama 等）的 key 与地址。
看谁在线	`khy gateway status`	列出已配置的各后端是否可用、当前优先级。
自动探测本地模型	`khy gateway detect`	探测本机可用的本地后端（如 Ollama）。

> 配好至少一个服务商后，网关会在某个通道挂掉时**自动级联切换**到下一个可用通道——这是 khy 的稳定性底座， 你平时无感。

✅ **这一级你学会了**：khy 的 AI 不是写死一家，而是**一个网关管多家、自带 key、自动容错**。 ➡️ **深入读**：[OPS-MAN-024] pip安装后-按需配置体验、[OPS-MAN-002] ai-管理-新api对齐、[OPS-MAN-032] 网关-自定义provider配置-agnes。

阶段 2 · 日常使用：对话、一次性问、写代码

配好 AI，就能进入 khy 的"日常三形态"。三者用同一个网关，区别只是**交互方式**。

场景	抄这条命令	说明
沉浸式对话 / 干活	`khy`	进入 agentic 终端（Ink TUI）：流式输出、可折叠的工具过程、真实上下文占用条。**这是主力入口**。
问一句就走	`khy ai "总结一下这个仓库"`	一次性提问，不进 REPL，适合脚本里调用。
Claude Code 风格编程	`khy claude`	以 Claude-Code 的工具循环跑编程任务（工具调用、权限门控、子代理）。
选/切模型	`khy model`	查看与切换当前使用的模型。

> 在 `khy` 终端里，输入 `/` 可以看到斜杠命令（如切模型、看上下文等）。不必背，用到再看。

✅ **这一级你学会了**：`khy`（泡在里面）/ `khy ai`（问完即走）/ `khy claude`（编程专精）三种姿势按需切换。 ➡️ **深入读**：[OPS-MAN-027] 快速开始（最全的逐步上手）、[OPS-MAN-001] ai-快速通道、[OPS-MAN-004] claude-code-代理配置。

阶段 3 · 进阶能力：学习模式、工作流、智能体

会聊天之后，khy 的"广度"才开始展开。这一级把你从"会用 AI"带到"会指挥 khy 干成体系的活"。

能力	抄这条命令	它给你什么
内置学习模式	`khy learn`	交互式学习路线；`khy learn roadmap` 看路线、`khy learn progress` 看进度、`khy learn next` 进入下一节。
可视化/可执行工作流	`khy workflow list` / `khy wf run <名>`	把多步编排成工作流并用原生引擎执行；支持导入。
子代理 / 智能体	（在 `khy` 终端里派发）	boss 把任务派发给 worker 子代理并行干活。

> **学习模式**是 0 基础最该用的一档：它把"该学什么、学到哪了"结构化，而不是漫无目的地问。

✅ **这一级你学会了**：khy 不止是聊天框——它能**学（learn）、能编排（workflow）、能分身（agent）**。 ➡️ **深入读**：[OPS-MAN-011] khy-os-学习指南、[OPS-MAN-017] khy-智能体-五步实施、[OPS-MAN-041] 通过KHY学习模式-从0到1面试大厂Agent岗-路线图。

阶段 4 · 成为高手：内核、远程、还原、量化

到这一级，你开始碰 khy "深度"的部分——它底下是一台**真的手写操作系统**，外面还有远程、源码还原、量化等专业能力。

高手能力	抄这条命令	说明
跑真内核	`khy os`	在 QEMU 里启动 khy 手写内核。需主机装有 `qemu-system-x86_64`（缺了会给出真实的安装指引，**不会**虚假承诺"自动下载"）。
构建内核 ISO	`khy os build`	从内核源码构建可启动 ISO。别名：`khy os rebuild`。
内核环境体检	`khy os doctor`	体检内核运行/构建所需的工具链与 QEMU。
移动端 / 远程	`khy mobile`	把 khy 接到移动端 / 远程使用。
源码一字不差还原	`khy restore`	把 pip/npm 装进来的加密源码快照解密、校验 sha256 后还原成可读源码树（默认无需密码）。
量化（内置应用）	见 khyquant 文档	khy 自带的默认应用：行情、回测、自选股等。

> **源码还原**是 khy 的一个硬承诺：你 `pip install` 得到的不是黑箱——`khy restore` 能把**与发布时工作树字节一致**的源码还原出来供你研读。

✅ **这一级你学会了**：khy 是"从内核到智能体"的一整套，你可以一路深到操作系统、远程与专业应用。 ➡️ **深入读**：[OPS-MAN-037] pip安装后-完整还原与全功能开启指南、[OPS-MAN-030] 移动端远程指南、[OPS-MAN-019] khy-远程ssh-实施清单、[OPS-MAN-036] khyos跨平台构建-Windows支持方案。

随时 · 出问题怎么自救（按顺序试）

任何阶段卡住，**不要先怀疑"网络不好"**——khy 的设计是把真实原因告诉你。按这个顺序自救：

```
khy preflight      # 1) 启动依赖体检（Node/依赖/PATH），给可粘贴的修复命令
khy where          # 2) 确认装在哪、依赖是否就绪
khy doctor         # 3) 让它尝试自动修复
khy os doctor      # 4) 只在跑内核出问题时：体检 QEMU/工具链
```

症状	多半是	去哪
`khy` 命令找不到	PATH 没配好	先 `khy where`（若连它都找不到，看 [OPS-MAN-027] 第 A/B 部分的 PATH 段）
启动报缺 Node / 版本低	Node < 20	装 Node ≥ 20（[OPS-MAN-028] 环境要求）
AI 不回话 / 报错	没配 key 或通道挂了	`khy gateway status` 看在线情况，重配 `khy gateway config`
跑内核报缺 QEMU	主机没装 QEMU	按提示装 `qemu-system-x86_64` 并加 PATH（提示里会给真实步骤）

一页速查（打印贴墙）

我想……	命令
装完先体检	`khy preflight`
看装哪了	`khy where`
自动修复	`khy doctor`
配 AI	`khy gateway config`
看 AI 在线	`khy gateway status`
开聊 / 干活	`khy`
问一句就走	`khy ai "……"`
写代码	`khy claude`
学习模式	`khy learn`
工作流	`khy workflow list`
跑内核	`khy os`
构建内核	`khy os build`
还原源码	`khy restore`

下一条阶梯：从"会用"到"会开发"

走完本文，你已是 khy 的**使用高手**。如果你想更进一步——**改 khy 本身、给它加能力（接新模型/写工具）、 甚至改内核、参与维护与发布**——请接续 **[OPS-MAN-044] 从使用入门到开发精通：khy 开发者成长路线**： 拿到可读源码 → 搭 editable 环境 → 走通主链路 → 第一次改 CLI 命令 → 扩展网关/前端 → 深入内核 → 可维护性与发布纪律。

关联

- 上手最全的逐步文档：[OPS-MAN-027] 快速开始。
- 装完功能全景：[OPS-MAN-023] pip安装后-完整功能清单。
- 完整还原与全功能开启：[OPS-MAN-037]。
- **开发者成长路线（本文的开发续篇）**：[OPS-MAN-044] 从使用入门到开发精通。
- 文档总入口：`docs/00_INDEX_文档索引.md`。