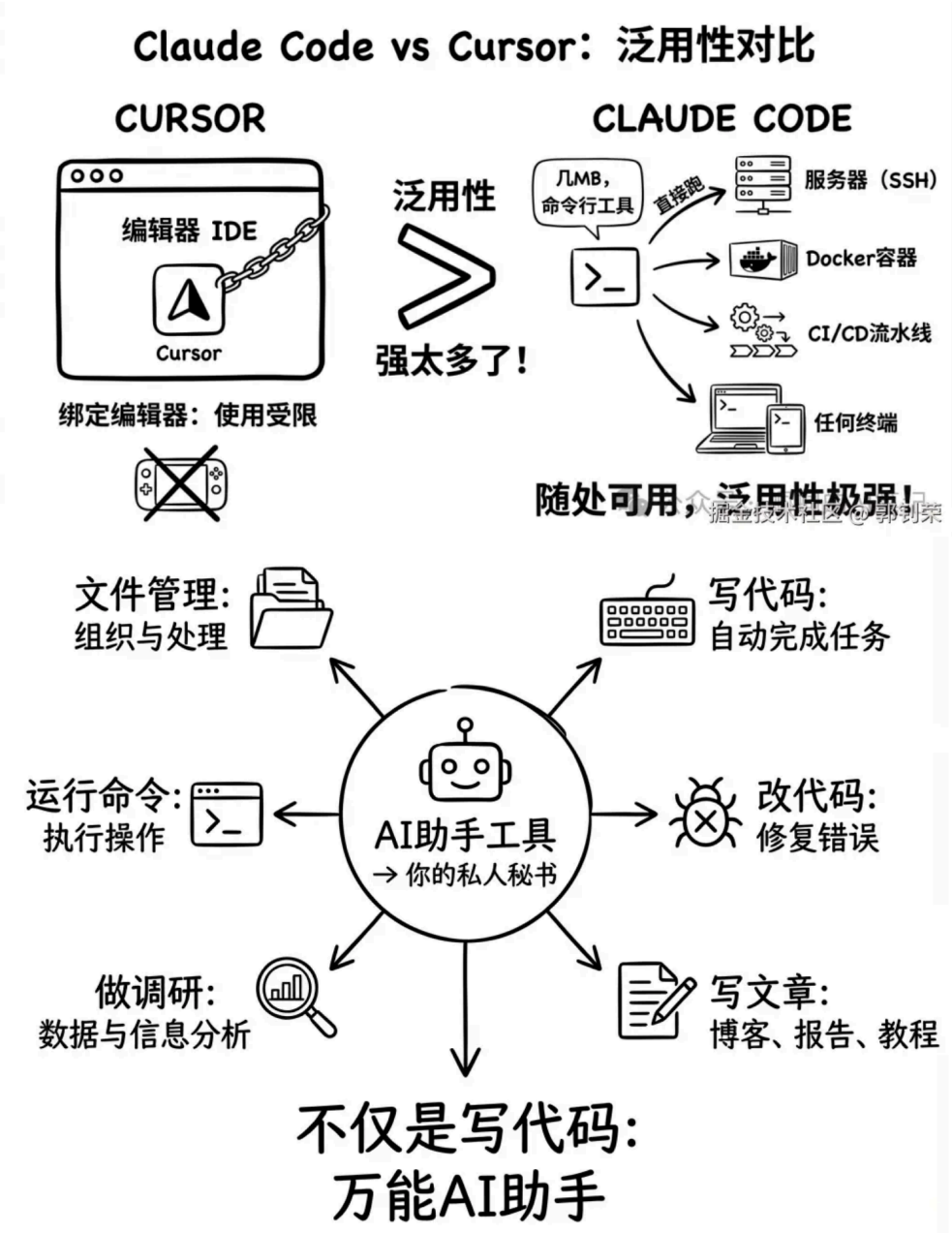
🚀 Claude Code最佳实践（内部分享）

工欲善其事，必先利其器。

> 本文档参考：
>
> https://code.claude.com/docs/zh-CN/best-practices （官方）
> https://docs.pig4cloud.com/ai#/doc/code/intro （实践）
> https://zhuanlan.zhihu.com/p/2009744974980331332 （第三方整理）

> **玩转Claude Code视频教程：** https://www.bilibili.com/video/BV1mzFVzPEB6/?spm_id_from=333.337.search-card.all.click&vd_source=0065b394de2c13fd91598f7009d80c24
> （Bilibili也有很多类似视频教程）

一、 写在最前：为什么选择 Claude Code?



Claude Code不是代码补全工具，也不是简单的AI对话助手。**它是一个基于文件系统作为上下文，终端为交互方式，操作系统为执行环境的通用Agent产品。**

Claude Code + openclaw（大龙虾）组合技：

> Claude Code可以用于辅助安装和诊断修复openclaw，只要放权足够前提下，就可以做：
>
> - 个人通用全能助手：日常问答、文档写作、代码生成、知识查询、思路梳理，日报周报总结，全场景覆盖自动化；
> - AI Ops运维助手：帮你远程操控k8s集群，自动诊断问题告警和自动修复；智能监控链路，生成修复日志等；
> - AI自动化流程（平替coze）：数据抓取，自动化产出公众号文章，营销视频等；
> - AI自动代码评审：代码质量审查符合公司标准规范；
> - AI自动化安全漏洞定期扫描巡检；
> - ……

logo	开源claw家族	语言	特点	场景
OPENCLAW	OpenClaw	TypeScript/Node	个人助手始祖、代码繁重、安全漏洞、容易自杀、最多通道与技能	自托管的全能个人 AI 助手
nanobot	Nanobot	Python	结构透明清晰、适合学习	教育教学；AI Agent入门、快速原型验证
PicoClaw	PicoClaw	Go	单二进制部署、95%代码由AI生成、适配各种老设备运行	老旧设备改造、轻量级边缘计算、免依赖快速部署
ZeroClaw	ZeroClaw	Rust	极致低资源、内存峰值<5MB	单片机、树莓派
IronClaw	IronClaw	Rust	WASM沙箱运行、权限最小化、深度防御	高安全性环境、零信任架构、运行不可信插件
TinyClaw	TinyClaw	TypeScript	多人协作、任务编排、实时仪表盘	企业团队协作、复杂业务流编排、运营调度中心

国产套壳：JVSClaw/CoClaw、ArkClaw、EasyClaw、QClaw、AutoClaw、MaxClaw、KimiClaw、miClaw

其实openclaw能做的事情Claude Code基本都能做，**openclaw是一个具有里程碑意义的过渡性产品，普通人目前消耗不起，没有好的模型也根本发挥不出作用。**企业级实践使用建议搭建本地算力平台，通过私有化部署确保数据安全。

其他组合技（了解）

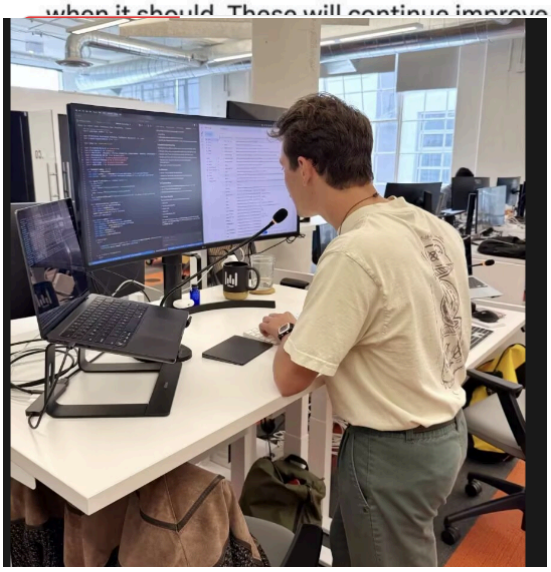
- **Claude Code + NotebookLM**：*将 NotebookLM 作为私有知识库，Claude Code 能够基于海量文档和研究资料进行精准的上下文编程和 RAG 知识库检索。*
- **Claude Code + Perplexity**：*结合 Perplexity 的实时联网搜索能力，提供最新的技术文档和在线解决方案以及相关市场调研。*
- **Claude Code + MindsDB**：*通过 MindsDB 连接企业数据库，使 Claude Code 能够直接对实时数据进行查询、分析和预测建模。*
- **Claude Code + Dify**：*利用 Claude Code 开发和调试 Dify 平台上的 AI 工作流逻辑，实现复杂 Agent 应用的快速迭代。*
- **Claude Code + NanoBanana2**：*结合 NanoBanana2 的极速图像生成能力， 自动化生成极具视觉冲击力的 UI 原型或广告素材。*
- **Claude Code + Figma/Pencil**：*将 Figma/Pencil 的设计稿直接转化为高质量的前端代码，大幅缩短从 UI 设计到功能实现的过程。*
- **Claude Code + Excel/Word/PPT**：*通过 Claude Code 编写自动化脚本，实现对 Excel 数据处理、Word 文档生成及 PPT 演示文稿的批量自动化操作。*
- **Claude Code + Firecrawl** ：*用 Claude Code 调用 Firecrawl 批量抓取并结构化任意网页内容，自动完成从网络数据采集到分析处理的全链路爬虫工作流。一行命令就能把整个网站变成 AI 可读的数据。*
- ……

1. Vibe Coding 大势不可挡

AI 辅助编程已不再是“尝鲜”，而是“生存技能”。在实际体验中，当前的 AI 编程工具已经可以完全替代 1-5 年经验的初中级开发者。如果你还在用传统方式一行一行敲代码，可能正在被时代抛弃。

Anthropic 的 Claude Code 负责人 Boris Cherny 宣布，他已经两个多月没有编写任何代码了；

国外甚至都开始语音口喷式进行编程，接一个麦克风给AI下命令，沟通即高效；



2. 放弃垃圾模型与工具

> 模型能力排行榜：
>
> https://www.superclueai.com/homepage
> https://artificialanalysis.ai/models

核心观点： 如果你还在用 Cursor、Trae、Qoder，或者任何 GC 大模型做编程，建议停下来。在代码理解、架构设计、复杂重构上，Claude + Claude Code 目前具有代际优势，生态良好，**没有替代品。**

- 国产模型平替：**Claude Code + GLM-5**（目前公司用的） / MiniMax-2.5 / Kimi-2.5
- 有条件的可以自己上国外模型：gpt-5.4 / opus-4.6 / gemini-3.1（token昂贵、翻墙、安全问题）
    - 第三方中转：openrouter | 闲鱼（基于newapi二开）

在选择中转站点前，可通过以下工具查看各站点的稳定性和延迟数据：https://relaypulse.top/?period=30d&category=commercial&service=cc
    - 自己搭反向代理：需要有较强技术背景且希望完全掌控 API 访问整条链路

个人建议：Claude Code作为主力工具，cursor/trae/idea等作为辅助工具，chatgpt/claude/gemini/manus/千问/豆包等作为辅助聊天

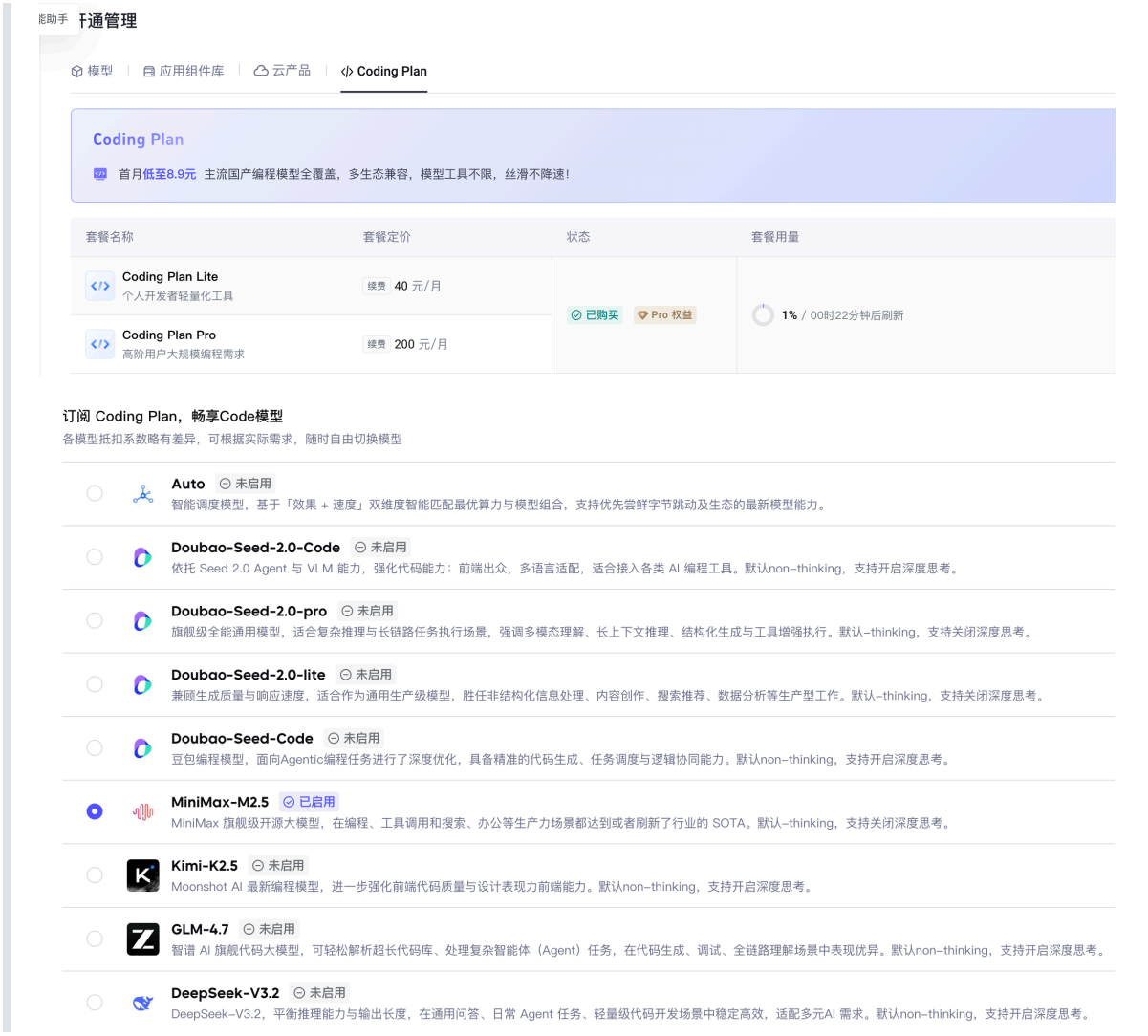
OpenAI Codex替代方案：在企业级开发中，除了 Claude Code，Codex + GPT5.4 也是另一种最优解，CodeX 编程工具的优势：

- 国内友好
    - 订阅费用相对较低（ChatGPT Plus约 $20/月，同时包含：聊天对话 + Codex编码）
    - 账号封禁风险低，适合国内团队长期使用
- 复杂问题分析
    - GPT 5.4 模型提供了桌面自主交互能力以及更强的代码能力
    - 适合解决复杂的技术难题和性能瓶颈
    - 虽然响应较慢，但结果质量更高

> **何时从Claude Code切换到Codex?**
>
> 1、反复失败的细节：当你多次要求 Claude Code 实现某个功能，但某个特定的点一直实现不好时：
>
> ```
> # 场景：Claude Code 多次尝试后仍无法正确处理
> # 问题：登录接口在特殊字符密码下验证失败
>
> # Claude Code 尝试 3 次后仍未解决
> claude "修复密码包含特殊字符时的验证问题"
> # 依然存在问题...
>
> # 切换到 Codex
> codex "分析密码验证逻辑，处理特殊字符（@#$%&）的转义问题"
> # Codex 会深入分析字符编码、正则表达式、URL 编码等细节
> ```
>
> 2、顽固的bug：Bug 修复尝试多次仍未彻底解决：
>
> ```
> # 问题：分页查询在某些条件下返回重复数据
>
> # Claude Code 修复后仍有问题
> claude "修复用户列表分页重复问题"
> # 部分场景修复，但特定条件下依然重复
>
> # 使用 Codex 深度分析
> codex "分析分页查询的 SQL 逻辑，检查 ORDER BY、DISTINCT、子查询"
> # Codex 会检查排序字段唯一性、索引使用、并发写入等深层问题
> ```
>
> 不要在 Claude Code 第一次失败后立即切换。先尝试 2-3 次优化提示词。如果同一个细节问题反复出现，这是切换到 Codex/Antigravity 或者其他模型的明确信号。

个人学习娱乐用token套餐分享：阿里云百炼Coding Plan、豆包火山方舟Coding Plan等；

> 支持多种国产模型切换，但是这种套餐基本都会做阉割，性能不如官方套餐，适合个人学习和测试用（5小时刷新一次token套餐总量），例如自己玩玩openclaw



御三家模型能力对比

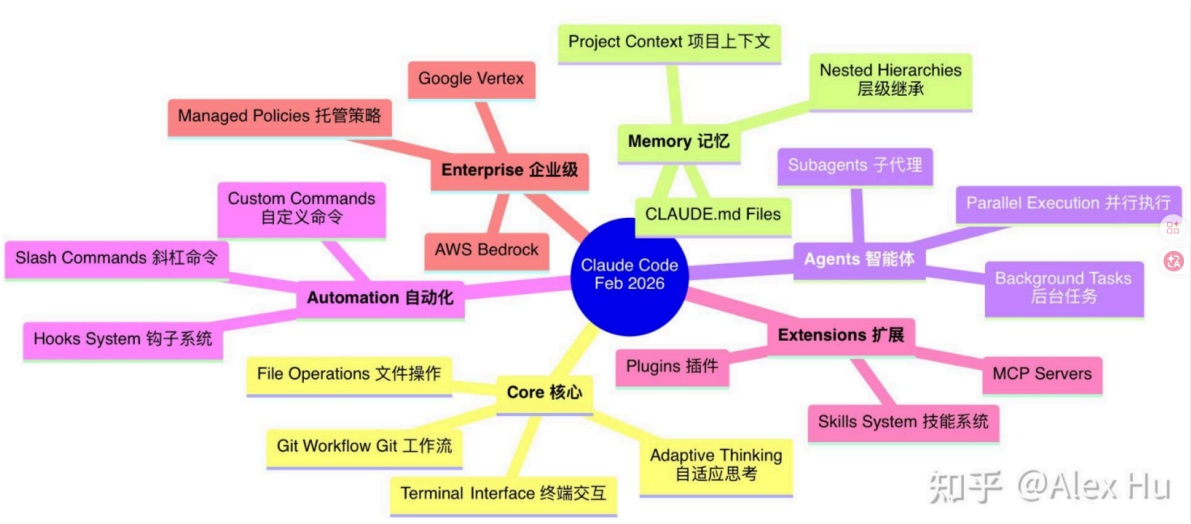
	GPT-5.4	Claude Opus 4.6	Gemini 3.1 Pro
发布时间	3月5日	2月5日	2月19日
知识工作	83.0% GDPval	78.0%	落后
计算机操控	75.0% OSWorld	72.7%	未披露
编程	77.2% SWE-Bench Verified	80.8%	80.6%
科学推理	92.8% GPQA Diamond	91.3%	94.3%
上下文窗口	105万token	20万/100万（扩展）	100万
API输入价格	$2.5/M token	$5/M token	$2/M token
API输出价格	$15/M token	$25/M token	$12/M token

3. 核心理念：不需要花里胡哨

保持默认配置就好，工具会持续优化。不要过度自定义，过多干预只会产生干扰。**少折腾配置，多实战输出。**

4. Claude Code功能全景图

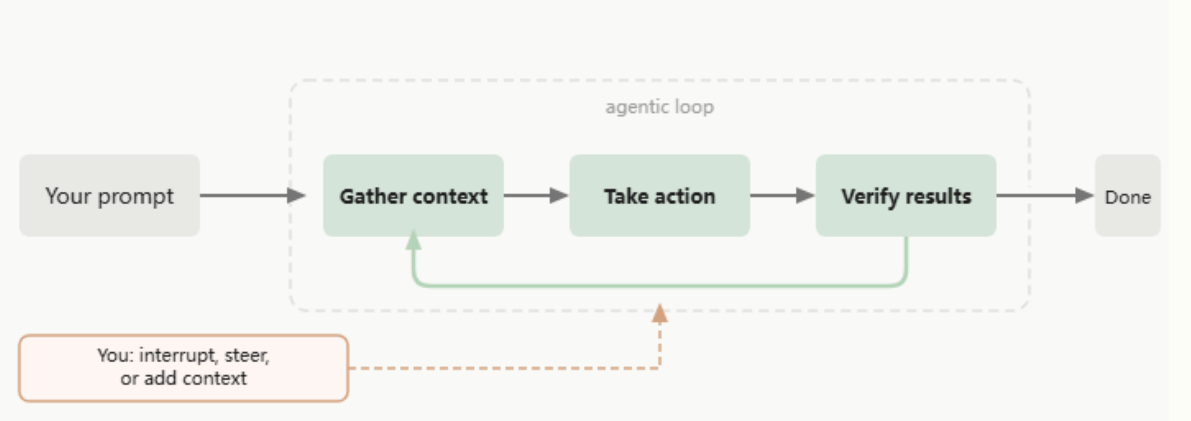
经过一年的高速迭代（2025.02 → 2026.02），Claude Code 已从一个终端工具演进为完整的 AI 编程平台。下图展示了截至 2026 年 2 月的六大功能支柱



5. Claude Code原理

设计哲学：Bash is all you need

> Unix：一切皆文件，一切皆可管道



Tools工具：

工具是赋予 Claude Code 智能体自主能力（agentic）的核心所在。如果没有工具，Claude 仅仅只能输出文本回复。而有了工具，Claude 就能真正采取行动：读取你的代码、编辑文件、执行命令、进行网络搜索，以及与外部服务进行交互。每一次调用工具所返回的信息，都会重新输入到执行闭环中，从而为 Claude 的下一步决策提供依据。

内置工具通常分为五类，每一类代表不同类型的代理能力。

类别	Claude 可以做什么
文件操作	读取文件、编辑代码、创建新文件、重命名和重新组织
搜索	按模式查找文件、使用正则表达式搜索内容、探索代码库
执行	运行 shell 命令、启动服务器、运行测试、使用 git
网络	搜索网络、获取文档、查找错误消息
代码智能	编辑后查看类型错误和警告、跳转到定义、查找引用（需要**代码智能插件**）

与Codex设计对比：

Claude Code是用的批量代码替换； Codex用的是更符合git原始训练集的git unified patch

二、 安装与基础配置

1. 正确的安装方式

强烈建议在 Mac 或原生 Unix 环境下使用。

- **macOS / Linux (推荐官方脚本)**:

```
curl -fsSL https://claude.ai/install.sh | bash
```

- **Windows (PowerShell)**:

```
irm https://claude.ai/install.ps1 | iex
```

(注：不再推荐使用 npm 全局安装，可能存在兼容性问题)

Claude Code 可以在多种环境中运行：

环境	适用场景	特点
终端 CLI	日常开发主力	功能最完整，直接操作文件系统
VS Code 扩展	偏好 IDE 内操作	内联 Diff、@引用、计划审查
JetBrains 插件	IntelliJ/PyCharm 用户	交互式 Diff、选择上下文
Desktop 应用	多会话并行	可视化 Diff 审查，多窗口管理
Web 版	无需本地安装	云端运行，支持长任务
Chrome 扩展	UI 测试调试	直接操作浏览器验证 UI

1.1 定期更新升级

定期更新升级，新版本会做很多优化

```
claude update
```

1.2 使用CLI

CLI 是 AI 的“感官”，告诉 Claude Code 在与外部服务交互时使用 CLI 工具，是AI工程师的终端基础配置环境，如 `lazygit`、`sshpass`、`glow`、`gh`、`aws`、`gcloud`、`sentry-cli`、`jq`、`ripgrep`、`llmfit`、`taproom`、`mactop`、`chafa`、`playwright`、`cli-anything`等等，例如：

- sshpass：自动操作远程服务器环境（人用ssh/xshell/iTerm2，AI用sshpass）
- glow：方便阅读markdown文件
- csvlens：查看阅读csv文件
- ripgrep：高速代码搜索
- playwright：和浏览器交互自动化测试
- cli-anything：将已有项目转为 CLI 给 Agent 操控

Claude 也能有效地学习它不知道的 CLI 工具。尝试像 `Use 'foo-cli-tool --help' to learn about foo tool, then use it to solve A, B, C.` 这样的提示。

> **Windows环境**：可以安装WSL2（Windows Subsystem for Linux），在其中运行Claude Code，并将项目文件存储在WSL文件系统中以获得更好的性能；Windows原生CMD/PowerShell环境，性能和兼容性都较差。
>
> ```
> # CLAUDE.md
> 环境声明：
> "当前运行环境为 Windows + PowerShell/WSL2。交互时优先使用已安装的 CLI 工具（如 ripgrep, jq, lazygit）。如果遇到路径问题，请优先使用正斜杠 / 或确保路径转义正确。"
> ```

1.3 项目目录结构全景

一个完整的 Claude Code 项目配置结构：

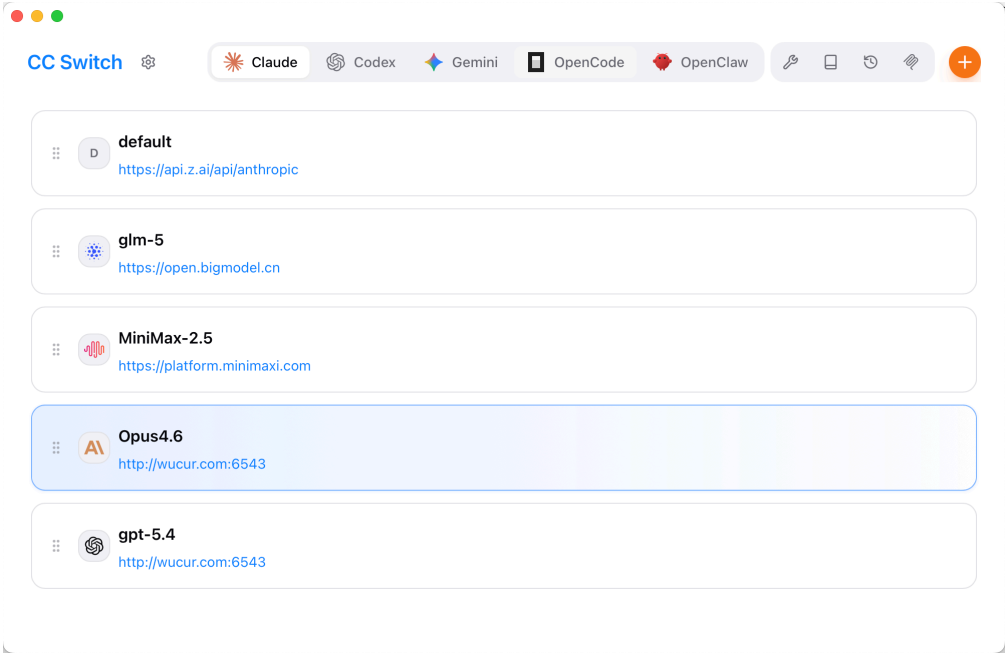
```
your-project/
├── CLAUDE.md                    # 📋 项目级指令（团队共享，提交到 Git）
├── CLAUDE.local.md              # 👤 个人项目偏好（自动 gitignore）
├── .claude/                     # 需要提交git，团队统一
│   ├── settings.json            # ⚙️ 项目设置（团队共享）
│   ├── settings.local.json      # 👤 个人项目设置（gitignore）
│   ├── CLAUDE.md                # 📋 等效于根目录 CLAUDE.md
│   ├── rules/                   # 📏 模块化规则文件
│   │   ├── code-style.md        #    代码风格
│   │   ├── testing.md           #    测试规范
│   │   └── security.md          #    安全要求
│   ├── agents/                  # 🤖 自定义子代理
│   │   ├── code-reviewer.md
│   │   └── debugger.md
│   ├── skills/                  # ⚡ 自定义技能
│   │   └── fix-issue/
│   │       └── SKILL.md
│   └── worktrees/               # 🌳 Git Worktree 目录（加入 .gitignore）
├── .mcp.json                    # 🔌 项目级 MCP 服务器配置
└── .github/
    └── workflows/
        └── claude.yml           # 🔄 Claude Code GitHub Actions
```

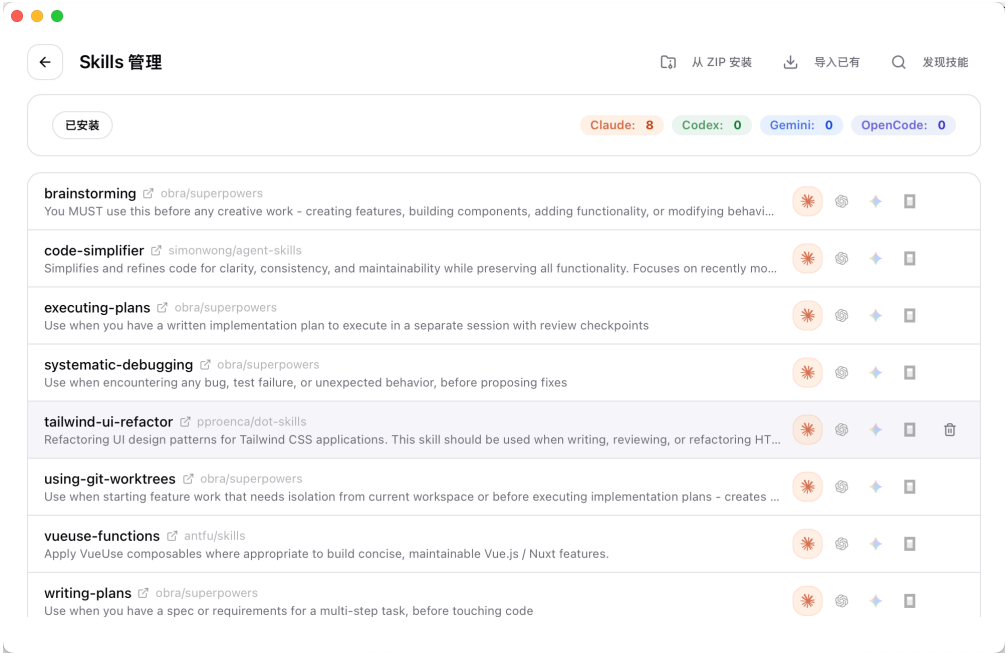
2. CC-Switch 工具 (国内网络环境必备)

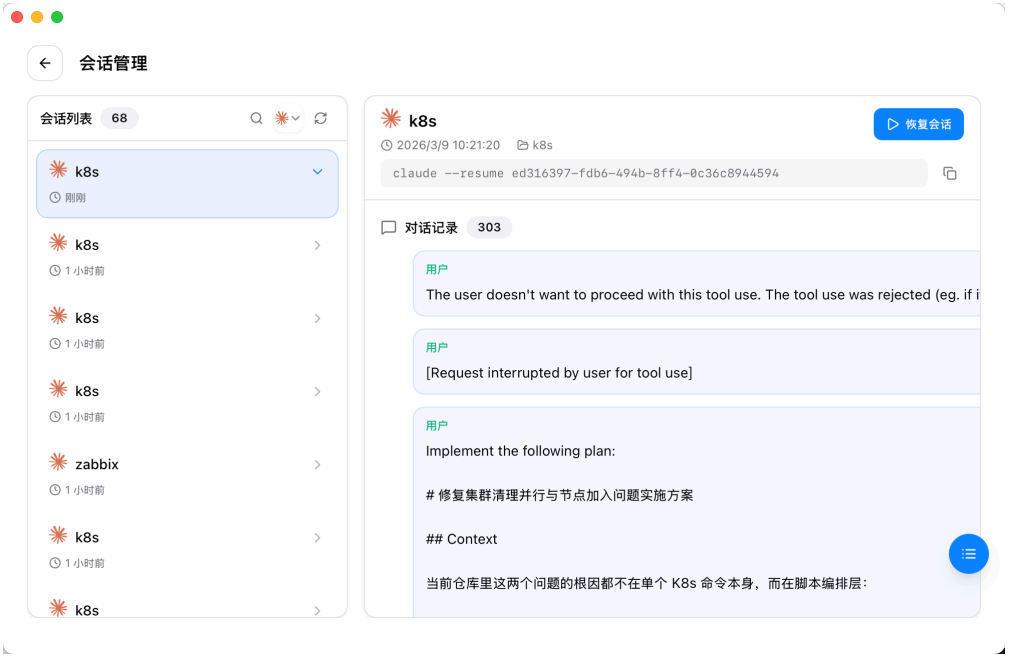
从 GitHub Releases 下载对应安装包：https://github.com/farion1231/cc-switch/releases

由于区域限制，国内直接订阅可能面临封号风险。推荐使用 **CC-Switch** 辅助工具配合 API 中转服务使用。

- **主要功能**：一键切换 API 配置、统一管理 MCP 配置、自动发现和安装 Skills、多预设系统提示词快速切换、历史session会话管理。

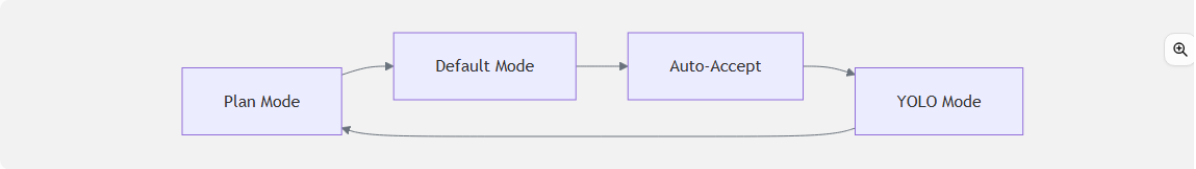






三、 核心开发模式

Claude Code 提供了多种工作模式，可通过 `Shift + Tab` / `Alt + m` 快速切换，最佳实践（先探索，再规划，再编码），将研究和规划与实现分开，以避免解决错误的问题。



1. **Plan Mode (计划模式)**
   - **特点**：研究与执行分离，只分析和规划，不会做实际代码修改。
   - **场景**：复杂的多文件重构、不熟悉的代码库探索、需要团队审批的架构变更、高风险操作前的预演。
2. **Default Mode（默认模式）**
   - **特点**：默认模式下，Claude 每次执行文件编辑或命令前都会请求确认。
   - **场景：**日常开发任务、学习 Claude Code 的新用户、需要逐步审查每个变更的场景。
3. **Auto-Accept Mode (自动接受模式)**
   - **特点**：自动执行文件编辑，减少确认中断（网络请求和 Bash 命令仍需确认）。
   - **场景**：日常开发信任度高的重复性任务、快速迭代开发、已通过 Plan Mode 审批的执行阶段。
4. **YOLO Mode (Bypass Permissions 超级权限模式)**

   `claude --dangerously-skip-permissions`
   - **特点**：完全跳过所有权限确认，自主执行任何操作。
   - **场景**：完全信任的环境、沙盒测试（注意破坏性命令风险）。
5. **Interview Mode (交互模式)**
   - **特点**：贯穿所有模式的交互机制，Claude 会在执行中主动提问，获取关键信息或澄清需求，最后由用户确认提交。
   - **场景**：需求不明确的任务、多方案选择场景、需要用户偏好输入的设计决策。

四、 核心工作流与最佳实践

1. CLAUDE.md：项目的核心大脑

> 强烈建议每个项目都准备一份CLAUDE.md，复杂项目需要每个模块准备一份CLAUDE.md，或者复杂内容文件夹内也准备一份CLAUDE.md。例如：安然商城项目接手后先准备好CLAUDE.md，**根目录 -> 各个模块 -> 具体文件夹**

`CLAUDE.md` 是唯一默认加载到每次对话的文件，内容越精简，执行效果越好。

- **编写原则**：少即是多（控制在 300 行内）、具体明确、不写 Linter（机械的静态规则） 能做的事、遵循WHY-WHAT-HOW 框架、采用渐进式披露：将详细文档拆分到独立文件，CLAUDE.md 只做索引

```
## 详细文档
- 构建指南：@docs/building.md
- 测试规范：@docs/testing.md
- 部署流程：@docs/deployment.md
```

- **企业级多模块项目**：
  - **根目录 CLAUDE.md**：全局视角，描述模块间关系、整体架构、跨模块开发规范。
  - **子模块 CLAUDE.md**：局部视角，描述该模块的技术栈、代码规范。

整体思想：由外到里，由浅入深，由架构到细节，按需精准投喂上下文。

> **为什么选择md（markdown）文件格式?**
>
> 1、结构清晰，AI更容易理解，Markdown 是一种 **轻量级标记语言**，结构简单、规则明确、清晰的结构化导航；
> 2、纯文本格式（最适合训练模型），非常适合代码和技术内容，训练数据来自：GitHub、维基百科、README文件；
> 3、人类可读 + 机器可读，Token效率更高，极高的信噪比，AI生态已经默认Markdown；

2. 自定义命令与 Hook (钩子)

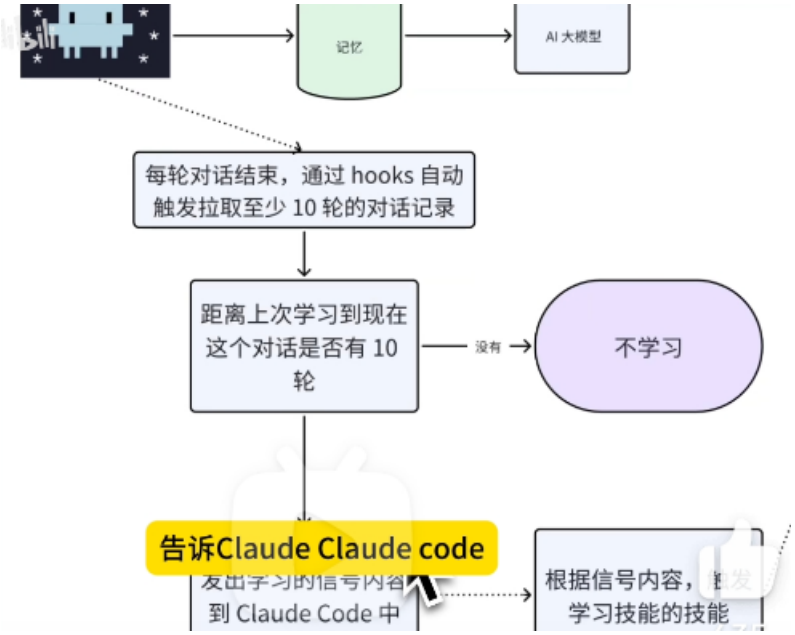
> **Hook 的本质**：Claude Code 运行时会经历「会话启动→用户输入→工具调用→任务结束」等生命周期阶段，Hook 就是绑定在这些阶段的「自动触发脚本」

- **常用内置命令**：/clear，/init，/plan，/compact，/rewind，/model，/context，/cost，/agents，/plugins，/resume，/exit等等
- **自定义命令 (Custom Commands)**：通过 Markdown 封装常用 Prompt 模板（如 `/commit`，`/review`），支持参数化和 Shell 上下文注入。
- **Hook**：在生命周期特定时刻自动执行的 Shell 命令，执行时不会消耗token（类似Vue Hook、Spring AOP）

场景	Hook 事件	示例
代码格式化，质量验证	PostToolUse（工具调用后触发）	Prettier, Black, gofmt
任务通知	Stop, SubagentStop（任务 / 子代理停止时触发）	企业微信、Slack、邮件

场景	Hook 事件	示例
权限控制	PreToolUse（工具调用前触发）	阻止删除生产文件
审计日志	PreToolUse, PostToolUse	记录所有操作
上下文注入	SessionStart, UserPromptSubmit（会话启动时触发）	加载项目配置
自定义通知	Notification（自定义通知触发）	替换默认通知方式

案例：会话结束后触发Hook，拉取最近10条对话记录进行学习，从而恢复记忆，减少幻觉



3. Skills vs MCP：扩展能力的双剑客

- **MCP (Model Context Protocol)**：
  - **定位**：AI连接外部系统（数据库、API接口文档、实时搜索、Chrome拓展）。
  - **痛点**：上下文开销大（每个 MCP 占用 2000-5000 tokens）。
  - **建议**：只安装高价值 MCP（如 `Context7` 查文档, `Exa` 实时搜索），按需使用。
  - **推荐 MCP**：https://mcp.so/（按需安装，不做推荐，也可以让AI自己安装需要的mcp）
- **Skills**：
  - **定位**：任务流程编排、领域知识专家逻辑的封装（Prompt 工程的最佳载体）。
  - **优势**：零基础设施依赖，占用 Token 极少，跨平台通用。
  - **建议：**作为技能包可以赋能各行各业：人力行政、电商运营、财务审计、项目管理等等。
  - **推荐 Skills**：https://skills.sh/（按需安装，也不做推荐）

```
# 后端  （superpowers平替spec规范驱动开发）
npx skills add https://github.com/anthropics/skills --skill skill-creator
npx skills add https://github.com/obra/superpowers --skill brainstorming

npx skills add https://github.com/obra/superpowers --skill writing-plans
npx skills add https://github.com/obra/superpowers --skill executing-plans
```

```
npx skills add https://github.com/obra/superpowers --skill systematic-debugging
npx skills add https://github.com/simonwong/agent-skills --skill code-simplifier
# 前端
npx skills add https://github.com/pproenca/dot-skills --skill tailwind-ui-refactor
npx skills add https://github.com/antfu/skills --skill vueuse-functions
npx skills add https://github.com/vercel-labs/agent-skills --skill web-design-guidelines
npx skills add https://github.com/anthropics/skills --skill frontend-design
# 合作开发
npx skills add https://github.com/obra/superpowers --skill using-git-worktrees
# 其他
npx skills add https://github.com/charon-fan/agent-playbook --skill self-improving-agent
npx skills add https://github.com/vercel-labs/skills --skill find-skills
npx skills add https://github.com/vercel-labs/agent-browser --skill agent-browser
```

  - **自定义开发Skills**：将公司基础设施能力等赋能到Skills中，例如：开发规范、技术栈、脚手架、UI风格、CICD、通用能力等，可以上传到公司skills仓库：http://gitlab.anran.com/ai/skills（昊哥），按需下载使用，嵌入到自己项目中或者本地环境中。

- **Plugins：**
  - **定位：**Claude Code 的**能力打包与分发机制**，将 Slash Commands、Subagents、MCP Servers、Hooks、Skills 打包成一个可一键安装的单元，方便团队共享和跨项目复用。
  - **与 Skills/MCP 的关系：**Plugin 是"容器"，Skills 和 MCP 是"内容"——一个 Plugin 可以同时包含 Skills（流程编排）、MCP（外部连接）、Hooks（行为干预）和自定义指令。
  - 核心组成要素：

组件	作用
Slash Commands	自定义斜杠快捷指令，如 `/code-review`、`/feature-dev`
Subagents	专项任务子智能体，如安全审计 Agent、PR 审查 Agent
MCP Servers	内嵌外部服务连接，如 GitHub、Linear、Vercel
Hooks	在 Claude Code 工作流关键节点注入自定义行为
Skills	任务流程知识封装（见上节）

  - 安装和使用

```
# 从官方 marketplace 安装
/plugin install ralph-loop@claude-plugins-official
/plugin install pr-review-toolkit@claude-plugins-official
/plugin install feature-dev@claude-plugins-official

# 从自定义 marketplace 安装（团队私有源）
# 第一步：添加市场
/plugin marketplace add your-org/claude-plugins
# 第二步：安装插件
```

```
/plugin install your-plugin@your-org-plugins

# 比如：想安装黑客松冠军大神的插件包
/plugin marketplace add affaan-m/everything-claude-code
/plugin install everything-claude-code@everything-claude-code

# 本地开发调试
claude --plugin-dir ./my-plugin
```

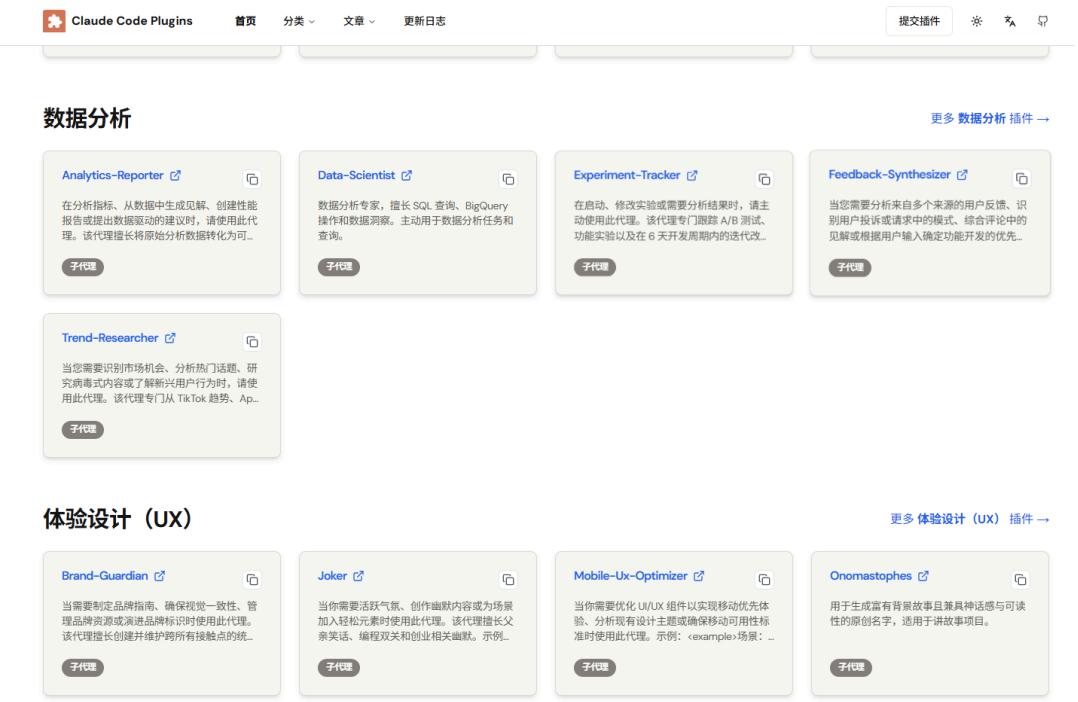
* 推荐官方 Plugins（来自 claude.com/plugins）：

Plugin	说明
`superpowers`	教授结构化的软件开发方法论。它提供了用于测试驱动开发（TDD）、系统化调试、头脑风暴、内置代码评审的子代理驱动开发以及创建新技能的能力
`feature-dev`	功能开发三段式 Agent：探索 → 架构设计 → 代码审查
`claude-md-management`	根据历史对话，优化claude.md
`frontend-design`	前端优化避免AI味
`pr-review-toolkit`	5 个并行 Sonnet Agent 多维度 PR 审查（Bug/测试/类型/质量）
`ralph-loop`	Ralph Wiggum 技术：Claude 循环执行任务直至完成，适合复杂迭代场景
`github`	官方 GitHub MCP，管理 Issue、PR、代码搜索
`playwright`	浏览器自动化与 E2E 测试，Claude 可直接操作页面
`plugin-dev`	开发 Plugin 的 Plugin，7 个专家 Skills 辅助构建 hooks/MCP/agents

* **自定义 Plugin**：团队可将开发规范、脚手架、CICD 流程、内部工具连接封装成 Plugin，统一上传到公司 Plugin 仓库，实现**一条命令对齐全团队 AI 辅助行为**。Plugin 结构示意：

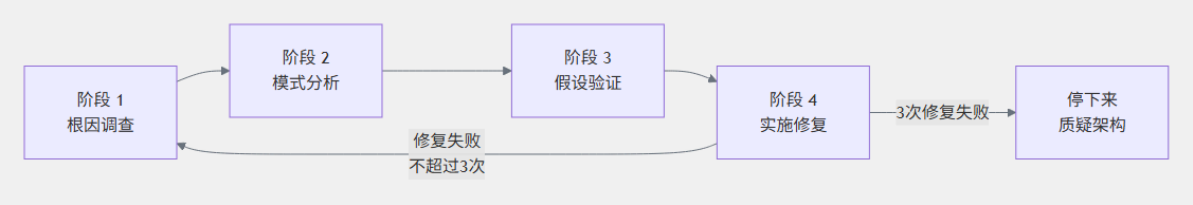
```
my-company-plugin/
├── plugin.json          # 插件清单
├── commands/            # 自定义 slash 命令
│   └── deploy.md
├── agents/              # 专项子 Agent
│   └── code-reviewer.md
├── hooks/               # 工作流钩子
│   └── pre-commit.json
├── skills/              # 领域知识/Skills
│   └── our-stack.md
└── mcp/                 # 内部 MCP 服务配置
    └── internal-api.json
```

- 第三方插件市场（了解）：https://claudecodeplugins.dev/zh



4. Systematic Debugging (系统化调试Skills)

告别"随机试错"，采用强制的四阶段排查流程，必须按顺序完成，不能跳过，该Skills能力可以自动debug调试问题直到全部修复成功：



1. **根因调查**：仔细阅读错误、稳定复现、检查变更、追踪数据流。
2. **模式分析**：对比正常案例，理解依赖关系。
3. **假设与验证**：提出单一假设，做最小化测试。
4. **实施修复**：确认根因后，针对性修复并验证。

5. Agent 及子代理 (Subagents)

Agent（主智能体） 就是你正在对话的 Claude Code 本体——它接收你的指令、理解上下文、规划任务、调用工具，是整个工作流的"大脑"和"调度员"。

Subagent（子智能体） 是 Agent 派生出来的**独立隔离实例**。每个 Subagent 运行在自己独立的上下文窗口中，拥有专属的系统提示、特定的工具权限。当 Claude 遇到匹配 Subagent 描述的任务时，会自动委托给它处理，每个子代理是独立的 Claude 实例，拥有独立的 **200K token** 上下文窗口，执行完毕后结果返回给主 Agent，保持主会话清洁。

- **内置类型**：
  - `Explore`：快速探索代码库（文件搜索、结构分析）。
  - `Plan`：架构设计与实现方案制定。
  - `code-reviewer`：代码审查与质量把控。
- **并行执行**：支持同时启动多个子代理从不同维度（如前端、后端、数据库）分析代码，极大提升效率。

并行子代理的典型工作流举例，当接手一个陌生项目时：

```
第一轮（并行探索，3个子代理同时跑）：
├── 子代理A：分析整体模块结构和依赖关系
├── 子代理B：找出所有数据库 Schema 和核心实体
└── 子代理C：梳理 API 接口清单和认证机制

第二轮（基于摘要，主 Agent 制定开发计划）：
└── 主 Agent：综合三份摘要，生成 CLAUDE.md 初稿

第三轮（并行开发）：
├── 子代理A：实现用户模块功能
├── 子代理B：实现订单模块功能
└── 主 Agent：等待两者完成后做集成测试
```

- **自定义agent**

```
.claude/agents/
├── code-reviewer.md      # 安全+质量审查，只读工具
├── test-writer.md        # 专职写测试，了解项目测试规范
├── api-designer.md       # REST/GraphQL API 设计专家
├── db-migration.md       # 数据库变更专家，熟悉回滚策略
└── doc-writer.md         # 文档生成，只读工具
```

6. token节省技巧

认识token：

- 在计算机相关领域中， "token" 通常是指一串**字符**，通常被视为**令牌**或**凭证**。它是一串代表特定身份、权限或会话状态的字符（如登录授权用的 Auth Token），起着“通行证”的作用。
- 在人工智能领域中， "token" 指的是处理文本时的**最小计算单元**。AI 并非以“字”或“词”为单位进行逻辑推理，而是将输入内容拆解为一系列基本元素。这些元素可以是一个完整的单词、一个词组、一个标点，甚至是长词中的某个子词（Subword）。

认识模型分类：

模型	性能与特点	典型场景
Opus	最高智能，顶尖推理能力	复杂的系统设计、屎山项目重构、前沿研发
Sonnet	性能与成本的完美平衡	日常开发、代码生成、大部分调试任务
Haiku	速度最快，成本最低	简单的脚本编写、格式化、终端命令

速查表：15 个省 Token 技巧

#	技巧	难度	预估节省	优先级
1	配置 .claudeignore	低	30-50%	必做
2	创建 CLAUDE.md	低	20-40%	必做
3	精准描述需求（文件名+行号）	低	50-85%	必做
4	及时 /compact 或 /clear	低	50-88%	必做
5	用 /model 切换合适的模型	低	60-95%	必做
6	使用 Plan Mode 先规划后执行	中	40-78%	强烈推荐
7	告诉 Claude “不需要做什么”	低	30-50%	强烈推荐
8	分阶段开发，每阶段清理上下文	中	50-70%	强烈推荐
9	先审查方案再执行	低	30-60%	推荐
10	直接给文件路径，少让 Claude 搜索	低	40-70%	推荐
11	自动允许只读工具权限	低	10-20%	推荐
12	–max-turns 限制交互轮数	中	20-50%	推荐
13	API 调用使用 Prompt Caching	高	60-90%	API用户必做
14	API 调用限制 max_tokens	中	20-40%	API用户推荐
15	定期查看 Token 用量并优化	低	持续优化	推荐

> **三句核心心法**
>
> 1. 减少输入： 别让 Claude 读不该读的东西——配好 .claudeignore，写好 CLAUDE.md，精准定位文件。
> 2. 减少轮次： 一次把事情说清楚——用 Plan Mode 先对齐方案，避免返工带来的 Token 加倍消耗。
> 3. 控制上下文： 对话是会膨胀的——及时 /compact 或 /clear，别让历史拖垮你的 Token 预算（建议对话 5 ~ 10次）。

7. Prompts提示词优化

> 模型和人一样，良好的沟通表达能让模型更了解你的意图，**沟通的精度决定了执行的深度**，能最大限度释放模型的 Agency。

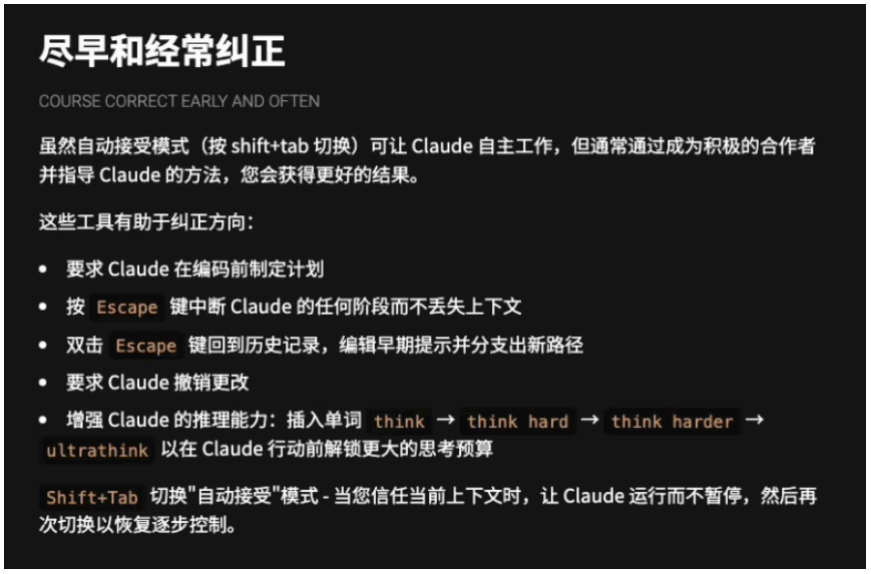
1、能用英文就用英文交流：大模型对于英文提示词天生支持良好，使用英文能更精准地触发模型的底层逻辑，减少语义转换中的损耗；
2、需求描述要具体，明确结果：描述时尽量包含**场景 + 触发条件 + 期望结果**（是就是，否就否）；面对复杂模糊的需求，需要 AI 能自主向你确认解决方案，例如输入完提示词后追加说明：`有任何问题请随时问我`，让Claude随时采访你；同时避免使用“可能、大概、尽量”等模糊词汇；
3、复杂需求需拆步执行：大需求一次性丢给 AI，容易因上下文过长而输出不全或被截断。建议拆解为独立小步骤，每步完成后即时 review/测试，再推进下一步；

4、激活深度思考模式：在 Prompt 中加入 `think` 关键词，可以触发 Claude 的扩展推理，级别越高，思考越深入，消耗 token 也越多；

关键词	思考深度	适用场景
`think`	★☆☆☆	一般逻辑问题
`think hard`	★★☆☆	中等复杂度任务

关键词	思考深度	适用场景
`think harder`	★★★☆	架构设计、性能优化
`ultrathink`	★★★★	最复杂的系统级问题

5、多模态辅助：善用图片输入，发送图片辅助表达意图，Claude Code 支持粘贴图片输入，使用拖拽或者`Ctrl + v`；
6、用自然语言操作 Git：无需记忆复杂的 Git 命令，直接用中文描述意图提交代码；
7、培养人机直觉：随着时间的推移，你会培养出没有指南也能捕捉的直觉。你会知道何时具体，何时开放，何时规划，何时探索，何时清除上下文，何时让它累积，有问题及早发现及早回退；我现在会在模型思考过程中习惯使用`Ctrl + o`来查看背后的思考执行逻辑；



8、示例驱动：利用已有示例或项目规范来定义当前项目规范，避免重复造轮子；
9、精准引用：对于命令式或者文件式交互，请直接精确限定：

```
# @ 的常见用法（精确引用文件）
> 解释 @src/auth.ts 的逻辑                  # 引用单个文件
> 对比 @old-api.ts 和 @new-api.ts           # 引用多个文件
> @src/components 的结构是什么?              # 引用目录

# ! 的常见用法（精确执行命令）
> ! git diff --stat                         # 执行命令，结果注入上下文
> ! npm test 2>&1 | tail -20                # 运行测试，截取尾部
```

10、具体化原则：Claude 能推断意图，但无法读你的心。引用具体文件、提及约束、指向示例模式

策略	模糊提示 ❌	具体提示 ✅
限定范围	"添加测试"	"为 foo.py 编写测试，覆盖用户已注销的边界情况。不要使用 mock。"
指向来源	"为什么 API 设计这么奇怪？"	"查看 ExecutionFactory 的 git 历史，总结它的 API 是怎么演变成现在这样的"
参考已有模式	"添加日历组件"	"查看首页已有 widget 的实现方式，HotDogWidget.php 是好的参考。按相同模式实现日历组件。"
描述症状	"修复登录 bug"	"用户报告会话超时后登录失败。检查 src/auth/ 中的 token 刷新。写一个能复现问题的失败测试，然后修复它。"

8. 相关开源项目（了解）

https://github.com/2025Emma/vibe-coding-cn：氛围式编程指南，适合快速了解AI-native开发模式，**规则就是一切**；
https://github.com/shareAI-lab/learn-claude-code： 从 0 到 1 构建 nano Claude Code-like agent；
https://github.com/affaan-m/everything-claude-code：**来自 Anthropic 黑客马拉松获胜者的完整 Claude Code 配置集合**；
https://github.com/Yeachan-Heo/oh-my-claudecode：把Claude Code升级为 **多模型协作的AI Agent 系统**；
https://github.com/ComposioHQ/awesome-claude-skills：让Claude学会固定技能的Prompt模板；
https://github.com/nextlevelbuilder/ui-ux-pro-max-skill：一个让AI生成 **更专业UI设计和前端代码** 的技能包；
https://github.com/thedotmack/claude-mem：持久化记忆可用于未来会话，**无缝保留跨会话的上下文**；
https://github.com/slopus/happy：通过网页或者移动端来**远程**安全加密地操控Claude Code；
https://github.com/uditgoenka/autoresearch：一个可以**自我实验迭代进化**从而达到预期目标值的AI研究员；

9. 相关技术文章（了解）

龙虾之父怎么写代码：https://mp.weixin.qq.com/s/FqdwQ1JYVFDv-fDw15543A
得物基于Spec Coding项目实战：https://mp.weixin.qq.com/s/1Oi7YhMgcPp9gQAWPndXWA
怎么搭一个能长期跑的 AI 写代码系统：https://mp.weixin.qq.com/s/oAjz6VjDb2Yjjyxf-esvgw

五、 新特性速递

除了上述文档中的核心功能，Claude Code 近期引入了强大的新特性，进一步提升了开发体验（平均1~2天一个小版本）：

> 更新日志：https://github.com/anthropics/claude-code/blob/main/CHANGELOG.md

1. Memory (长期记忆)

v2.1.59版本后，Claude Code 现在具备了**跨会话的记忆能力 (Memory)**。它不再是一个每次都需要重新调教的"失忆助手"，而是能随着使用不断成长的专属 AI 工程师。配合CLAUDE.md辅助记忆，简直强无敌。

- **核心能力**：
  - **记住偏好**：自动记住你的编码风格（例如：偏好 `async/await`、特定的命名规范、喜欢用单引号等）。
  - **记住环境**：记住项目的特定构建命令、测试脚本路径或常用的终端操作。
  - **记住上下文**：记住之前解决过的复杂 Bug 的背景，或者某个特定模块的业务逻辑，下次提问无需从头解释。
- **工作机制**：
  - Claude Code 会在对话过程中自动提取有价值的信息并存储到记忆库中。（通常位于 `~/.claude/projects/<project>/memory`）
  - 在后续的对话中，它会自动检索相关的记忆并作为上下文注入，让回答更懂你、更贴合项目实际。
- **如何管理**：
  - 你可以通过自然语言直接告诉它："**记住**以后在这个项目中都使用 Yarn 而不是 npm；**记住** API 测试前先启动本地 redis "。
  - 也可以通过特定的命令（如 `/memory` 相关指令）来查看、编辑或清除不必要的记忆。

2. 增强的上下文管理与 UI 体验

- **更智能的 Token 优化**：在处理大型项目时，Claude Code 进一步优化了上下文的裁剪和保留策略，确保在 Token 限制内保留最核心的代码片段。
- **VS Code 插件深度融合**：UI 模式持续进化，提供更直观的 Diff 视图、更便捷的单文件/多文件 Accept/Reject 操作，让 AI 协作无缝融入日常 IDE 工作流。

3. 工具调用上下文归零

Claude Code 在执行复杂任务时会调用大量工具，例如：Shell、Git、文件读写、MCP Server、API 调用。

早期版本中，工具调用的上下文可能会不断累积，从而造成：Token 消耗过高、上下文噪声增加、推理质量下降。为了解决这个问题，Claude Code 引入了 **工具调用上下文归零机制（Tool Context Reset）**。

核心原理：

在一次工具调用完成后：

1. 保留最终结果
2. 丢弃中间推理过程
3. 清理无关上下文

带来的好处：

- 长时间运行 Agent 不会"上下文污染"
- 大型项目扫描更稳定

- 多工具链协作更可靠

简单理解就是：Claude 会像一个工程师一样 **只记住结果，不记住所有思考过程。**

4. Agent Teams

> Agent teams 是实验性功能，默认禁用。通过将 `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS` 添加到你的 settings.json 或环境变量来启用它们。Agent teams 在 已知限制 中存在关于会话恢复、任务协调和关闭行为的问题。

Agent teams 让你协调多个 Claude Code 实例一起工作。一个会话充当团队负责人，协调工作、分配任务和综合结果。队友独立工作，每个都在自己的 context window 中，并直接相互通信。与 subagents 不同，subagents 在单个会话中运行，只能向主代理报告，你也可以直接与个别队友互动，无需通过负责人。

5. 其他

- /batch：批量并行任务，专为高度并行无冲突任务设计，适合大规模迁移、重构场景
- /loop：定时任务，可调度最长3天的重复任务和工作
    - 1、部署看护

      ```
      /loop 5m 检查当前项目的部署状态，如果有异常 Pod 就分析日志找原因，把结果写到使用 github cli 的命令 gh 给项目 lltx/note 创建issue
      ```

    - 2、PR巡检

      ```
      /loop 30m 检查当前分支的 PR，如果 CI 失败了，看看报错日志，能修的就修掉并 push
      ```

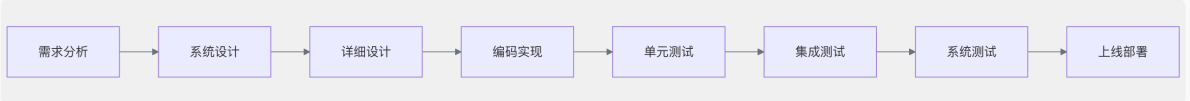
    - 3、日报总结

      ```
      /loop 24h 总结过去8小时我完成的工作内容，总结成日报，通过MCP接口/Skills能力来上传到云7
      ```

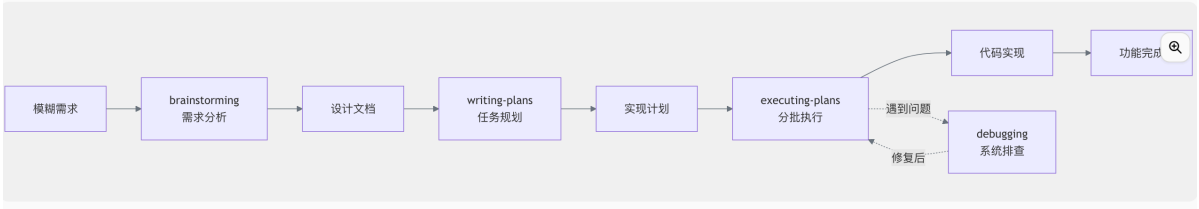
- /simplify：简化输出，任务完成后调用，让 Claude 自动精简/总结输出
- /btw：（by the way）补充说明，不打断主流程，插个话
- /voice：语音功能
- /color：实时改 prompt 输入框颜色
- claude --name < NAME >：启动时直接命名当前会话

六、 重塑软件开发流程

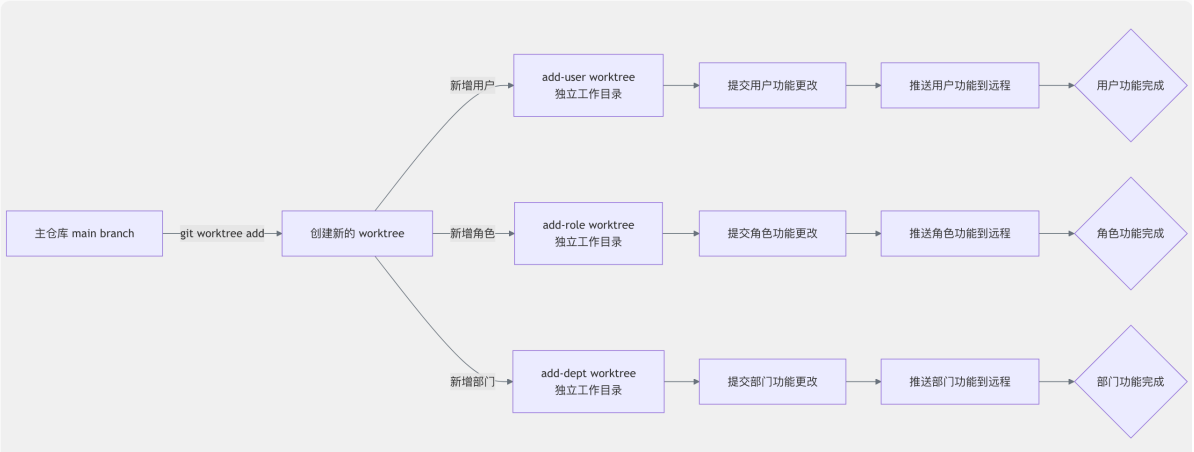
经典的软件工程瀑布开发模型包含以下阶段：



Superpowers 将瀑布模型的每个阶段映射为 AI 可执行的技能，用强制性工作流替代人工协调：



Git Worktree可以创建多个工作目录，在不同终端窗口同时运行多个 Claude Code 实例，让不同 AI 会话并行处理不同功能模块，大幅降低等待时间，显著提升整体产出效率。



在 **Agent 工程师时代**，前后端的界限正在快速消融，编程语言之间的壁垒也逐渐淡化。前端不再只看你掌握 Vue/React/Angular 中的哪一种技术栈，后端也不再单纯以 Java/Go/Python 区分能力。行业真正需要的，是**具备真实问题解决能力的产品架构复合型人才**：既能高质量完成 UI 设计与原型输出，又具备产品思维和架构能力；既能理解业务需求，又能做架构设计与技术落地。

未来的核心竞争力，不再是 “会什么语言 / 框架”，而是**能否独立完成从需求→设计→开发→部署→运维的全链路闭环。**

借助 Claude Code：

- 后端工程师可以通过 `Tailwind CSS` + `VueUse` 配合专属 Skills，轻松输出高质量的前端代码。
- 传统的“需求->设计->编码->测试”瀑布流，正在被 `brainstorming -> plans -> executing -> debugging` 的 AI 工作流所替代。

拥抱 Claude Code，从今天开始，让 AI 成为你最强大的结对编程伙伴！人人都是AI鼓励师！

1. 关于语音输入（了解）

在 AI 编程场景中，语音输入正在成为提升效率的重要工具。相比传统键盘输入，语音输入有以下优势：

- **速度更快**：正常语速约 160 字/分钟，是键盘输入（约 40 字/分钟）的 4-6 倍
- **降低疲劳**：长时间编程时，语音输入能有效减轻手腕和手指的负担
- **思维更流畅**：说话比打字更接近自然思考，有助于快速表达复杂的编程思路

推荐工具

> **Typeless** - 跨平台智能语音输入
>
> - 官网：https://www.typeless.com
> - 支持 macOS、Windows、iOS、Android 全平台
> - 核心特性：
>   - 自动移除填充词（"um"、"uh"、"嗯"等）
>   - 智能去除重复内容，保持语言简洁
>   - 改变主意时自动编辑，只保留最终想表达的内容
>   - 自动格式化列表、步骤等结构化内容
>   - 根据不同应用调整语气（邮件 vs 聊天）
> - 支持 100+ 种语言，可混合语言输入
> - 个人词典功能，可添加专业术语
>
> **Spokenly** - Mac 平台首选
>
> - 官网：https://spokenly.app/
> - 支持 100+ 种语言，自动语言检测
> - 提供本地模式，所有语音数据仅在本地处理，保护隐私
> - 集成 GPT-4 等 AI 模型，自动优化语法和格式
> - 支持 Agent 模式，可通过语音指令控制 Mac
> - 本地模型完全免费，支持离线使用

> **值得关注**：豆包语音输入法
>
> 字节跳动推出的豆包语音输入法在移动端表现出色，具备 15 种方言识别、98.2% 的方言转标准语准确率、上下文感知预测等特性。目前仅支持移动端，**PC 版尚未发布**。考虑到字节在 AI 领域的技术积累，豆包语音输入法的桌面版值得持续关注。

2. GSD、Spec、Superpowers（了解）

目前 AI 开发主流的三种工作流框架，目的都是解决同一个问题——让 AI 写大型项目代码时更可靠、不跑偏。

2.1 GSD (Get Shit Done)

GSD 是一个轻量级且强大的元提示、上下文工程和规范驱动开发系统，专为 Claude Code、OpenCode、Gemini CLI 和 Codex 等AI编码代理设计。它旨在解决AI代理在处理复杂任务时常见的“上下文腐烂”问题，即随着上下文窗口的增大，AI代理的输出质量会下降。GSD 通过结构化的项目文件和多代理编排机制，确保AI代理始终拥有清晰、相关的上下文，从而实现长时间自主工作并保持一致的卓越表现 。

2.2 Spec Kit

Spec Kit 是一个开源工具包，旨在帮助开发者摆脱传统的“凭感觉编码”模式，专注于产品场景和可预测的开发结果。它倡导“规范驱动开发 (Spec-Driven Development)”理念，将规范视为可执行的实体，直接生成工作实现，而非仅仅作为指导。Spec Kit 提供了一系列工具和命令，支持从高层需求到详细实现计划的整个开发流程，并兼容多种AI代理 。

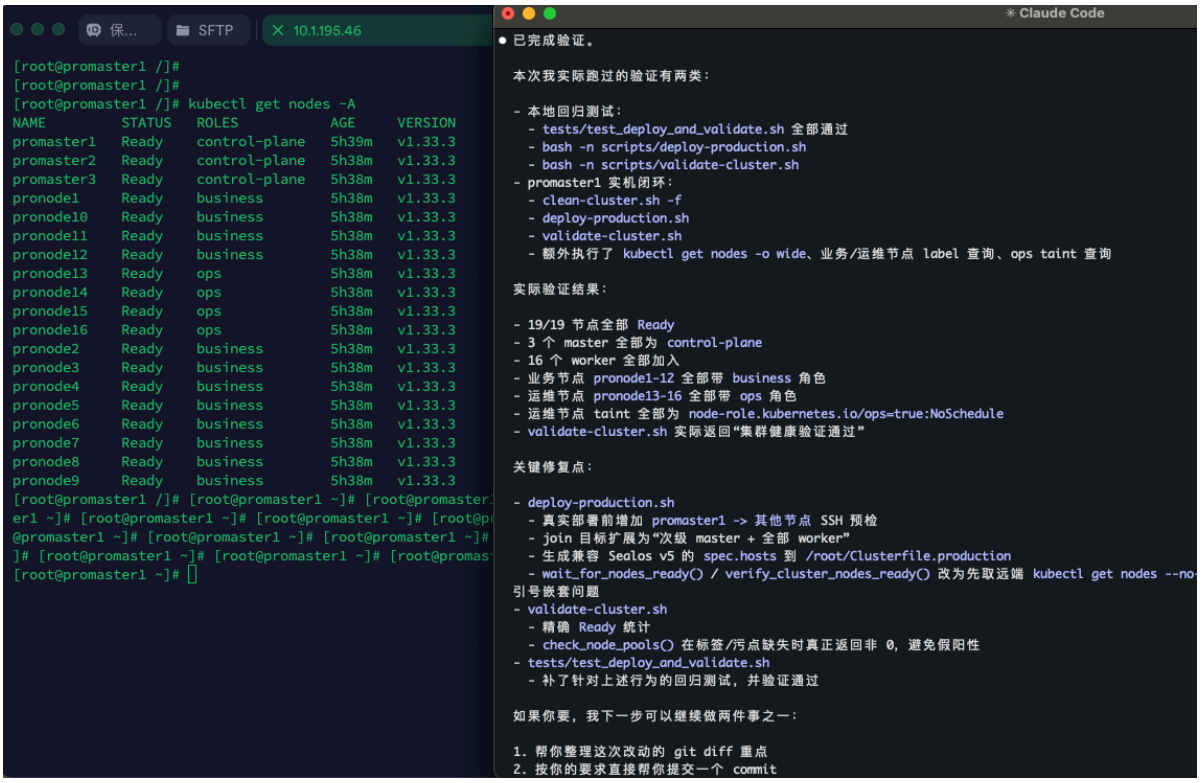
2.3 Superpowers

Superpowers 是一个为AI编码代理设计的完整软件开发工作流，它建立在一套可组合的“技能”之上，并辅以初始指令，确保AI代理能够有效利用这些技能。Superpowers 的核心在于其“子代理驱动开发 (Subagent-Driven Development)”过程，通过将复杂任务分解为更小的工程任务，并由独立的子代理执行，同时进行两阶段审查（规范符合性审查和代码质量审查），从而实现高效的自主开发 。

七、个人AI实践分享

案例一

基于Claude Code实现k8s生产集群3主16从节点自动化部署和测试，提前把AI所需所有资源都给准备好，控制好文件夹权限，放手让AI干



案例二

新开一个项目：使用前后端脚手架进行初始化和CICD打通

```
# 提示词
新建一个前后端整合项目，名称为anran-project-test，要求如下：
1、前端初始化脚手架参考：http://gitlab.anran.com/infra/anran-v3admin.git（伟哥）；后端初始化脚手架参考：http://gitlab.anran.com/infra/scaffolding.git（小徐）；
2、包命名规范：com.anran.test；数据库地址：192.168.0.10:3306/project-test anran/anran
3、出于方便考虑，前后端不分离，CICD执行完毕打包后，使用同一个pod：前端一个container、后端一个container，具体CICD前后端整合可以参考这个项目：http://gitlab.anran.com/shop/beicha-project.git
4、使用mysql mcp操作数据库；使用chrome mcp验证系统功能完整性
5、整体回归性验证测试
6、完成后用code-simplifier简化代码
7、简化完成后更新CLAUDE.md、README.md

有任何问题请随时向我提问
```

向你提问 -> 确认需求 -> 设计规划 -> 实践编码 -> 测试验证 -> 发版 -> 迭代修复 -> 回归测试 -> 迭代修复 -> 完工上线

案例三

一个典型的验证驱动型的提示词示例：

```
# 提示词
需求：
为 @src/utils/date.ts 添加 formatRelativeTime 函数。

要求：
- 输入 Date，输出 "刚刚"、"3 分钟前"、"2 小时前"、"昨天"、"3 天前" 等
- 超过 7 天返回 YYYY-MM-DD 格式
```

```
验证：
1. 先为以上每种情况编写测试用例
2. 实现函数
3. 运行 `npm test -- formatRelativeTime`
4. 运行 `npm run typecheck`
5. 所有通过后告诉我结果
```

让你的验证机制更加可靠：

- **写好 CLAUDE.md 中的测试命令**，让 Claude 知道如何运行测试
- **用 Hooks 自动验证**，例如 Stop Hook 自动运行测试套件
- **在 Skills 中嵌入验证步骤**，确保工作流最后一步总是验证
- **如果你无法验证它，就不要发布它**

八、最后

注意事项

1、保护公司隐私和安全，禁止把公网上相关明文密码以及机密隐私信息等上传到模型对话调用中；
2、AI生成的代码需要人工介入严格审阅把控；
3、严格限制AI能访问的文件夹和文件权限，Claude Code默认只授权当前目录；
4、不建议AI直接操作和访问生产级别重要的数据库、服务器和业务系统；