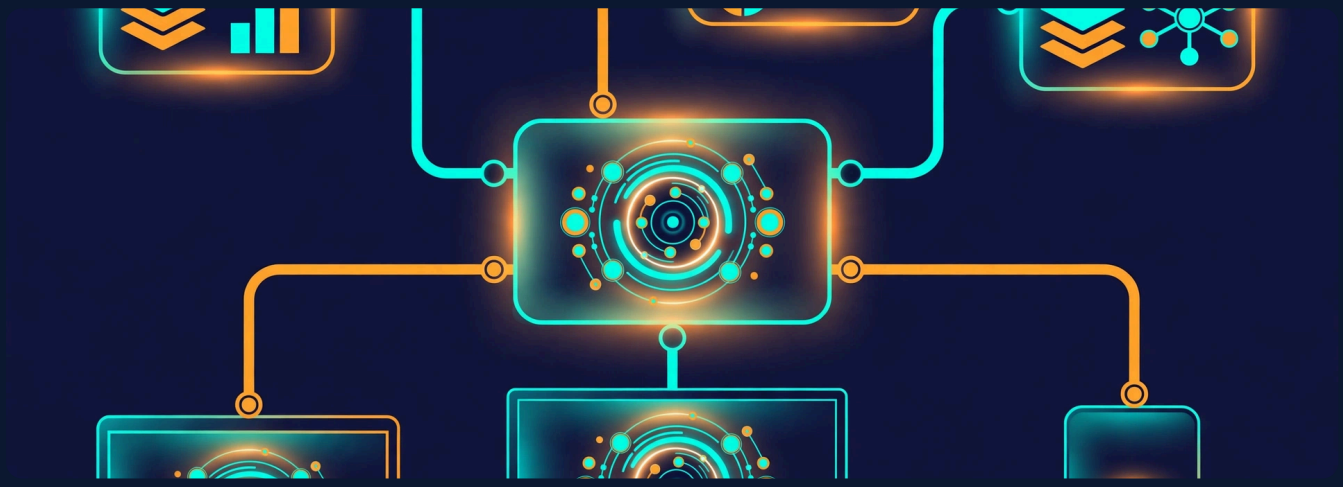
T3 CODE · BEGINNER MANUAL · 2026-11 版

T3 Code 小白操作手册

一个界面，管住你所有的 AI 编程助手。
从安装到并行干活、手机远程，全程照着做就行。

零基础友好 | 全流程实操 | 提示词模板可直接抄 | 排错大全



适合谁：刚开始用 Claude Code、Codex、Cursor 等 AI 编程工具，被多个终端窗口搞晕的人；想在手机上盯任务、批批准的人；想让两个 Agent 分工干活、互不踩代码的人。

10 分钟
装好并接上第一个 Agent

30 分钟
跑通第一条任务并提交

7 天
并行、远程、分工全部上手

版本基准：以 T3 Code 0.1.9（2026-11-11 发布）为准；0.1.10 已于 11 月 17 日发布，细节以你实际版本为准。上面的“10 分钟装好 / 30 分钟跑通 / 7 天上手”是作者为你设定的学习目标，不是官方承诺，做不到也不用焦虑。本手册只讲操作，不卖课、不卖订阅。

CONTENTS

目录：按干活的顺序读，不要按字典查

这本手册的顺序，就是你真实使用 T3 Code 的顺序：先搞懂它是什么，再装好、先把安全设对，然后跑通一条任务，最后才学并行、远程和高手技巧。每一章结尾都有一个能当场验收的小成果，做不到就不要往下跳。

第 0 章　30 秒看懂：T3 Code 到底是什么	3
第 1 章　装之前：先备齐三样东西	4
第 2 章　安装：三条路线，选一条就行	5
第 3 章　五个概念讲透，后面全通	6
第 4 章　安全第一课：权限四模式（必改默认）	7
第 5 章　第一条 Thread：八步跑通全流程	8
第 6 章　界面地图：每个区域管什么	9
第 7 章　提示词写法：六要素 + 模板库	10
第 8 章　六大实操场景（上）：修 Bug / 做功能 / 补测试	11
第 9 章　六大实操场景（下）：审查 / 并行 / 手机远程	12
第 10 章　Worktree、分支与一键 PR	13
第 11 章　多代理分工：让强的干它擅长的	14
第 12 章　远程与手机：三条路怎么选	15
第 13 章　费用与 Usage：钱花在哪，看哪三个数	16
第 14 章　15 条高手技巧	17
第 15 章　排错大全 + 7 天上手计划 + 终检清单	18
附录　命令速查 / 术语表 / 官方入口	20

怎么用这本手册最省时间：第一次只读第 0、1、2、4、5 章，先把一条任务完整跑通；第二天读第 7、8 章练提示词；等你开始同时开两条任务，再读第 10、11 章。排错时直接翻第 15 章的表，不要从头找。

先记住全书最重要的一句话：T3 Code 新任务的默认权限是 Full access（不问你就直接改文件、跑命令）。装好后的第一件事，是把默认权限改成 Supervised。细节在第 4 章，先改再用。

CHAPTER 0

第 0 章　30 秒看懂：T3 Code 不是模型，是总控制台

一句话：T3 Code 是一个免费、开源的控制台，它把你电脑上已经装好、已经登录的多个 AI 编程助手，装进同一个界面里管理。它自己不写代码，写代码的还是 Claude Code、Codex 这些助手；它负责派活、盯进度、批批准、看改动、管分支。

用餐厅的比喻，一次讲清三层关系

大模型 = 主厨
真正有手艺的人，比如 Claude、GPT 这类模型。只会做菜，不会自己开店。

Harness = 厨房
Claude Code、Codex 就是厨房：给主厨配齐刀具、灶台和传菜口，让模型能读文件、跑命令、改代码。行话叫 harness，本手册后面统一叫「运行框架」。

T3 Code = 餐厅
一家允许你自带任何厨房的餐厅：前台、包间、传菜系统都现成，你带哪个厨房进来都能开席。

这个比喻是 T3 Code 官方常见问题里的原话。它直接决定了三件事：第一，没有厨房，餐厅开不了张——你必须先装好至少一个编程助手并登录，T3 Code 才能工作。第二，菜好不好吃取决于主厨和厨房，不取决于餐厅——同一个任务写得好不好，看你选的助手和模型，不看 T3 Code。第三，餐厅不卖食材——T3 Code 不转售额度，你原来给哪家交的订阅费，还交哪家，T3 Code 本体不收费。

它能管哪些助手

截至 2026 年 11 月，官方列出的内置助手包括 Codex、Claude Code、Cursor、Grok Build、OpenCode、Google Antigravity、Pi、Muse Code、fx（Vercel）、Kiro（自 v0.1.7 起为 Early Access / beta）等，还能接遵循通用协议的其他助手。小白不用全装，**先装一个，用熟再加第二个**。它不支持本地模型，官方的理由很直接：目前能在本地跑的模型，还做不了像样的工程活。

它最值钱的四个本事

1. **一处管全部：**每个任务一条对话（Thread），可以各选不同的助手和模型，不用来回切终端窗口。
2. **任务互不踩踏：**每条任务可以有自己独立的工作目录和 Git 分支，改完一键提交、推送、开合并请求（PR）。
3. **手机也能管：**电脑在家跑任务，你在外面用手机或浏览器看进度、点批准，连接断了任务也不停。
4. **改了什么看得见：**每一轮对话前后都有存档点，这轮到底改了哪些文件、哪几行，一眼能查，还能退回去。

> **本章验收：**你能用自己的话说出「T3 Code 不提供模型、必须先有一个已登录的助手才能用」。说不出，就再看一遍餐厅比喻。

CHAPTER 1

第 1 章　装之前：先备齐三样东西，少走 90% 弯路

新手装 T3 Code 最常见的翻车，不是 T3 Code 装不上，而是装好之后发现里面一个能用的助手都没有。顺序反过来就顺了：先备齐三样东西，再装本体。

1. **一个已登录、能单独跑通的编程助手。**二选一起步即可：Claude Code 装好后在终端运行 `claude auth login` 登录；Codex 装好后运行 `codex login` 登录，或者装好 T3 Code 后在设置里直接连接 ChatGPT 账号。登录必须发生在「将来运行 T3 Code 的那台电脑」上，装在 A 电脑、登录在 B 电脑是不算的。

2. **Git 和一个真实项目文件夹。**T3 Code 的分支、存档点、合并请求全建立在 Git 上。先确认项目已是 Git 仓库（文件夹里有隐藏的 .git 目录），没有就先初始化并提交一次。不要拿你唯一一份重要代码当第一个试验场。

3. **（走命令行安装才需要）Node.js。**用 `npx` 临时试用或脚本安装时需要 Node.js，版本要求：22.16 以上（22 版线）、23.11 以上（23 版线）或 24.10 以上。这组数字来自服务端 engines 声明与旧版 README，现行 install.md 正文已不再列出具体版本号，装之前以你实际拿到的版本说明为准。只装桌面版的人可以跳过这一项。

装之前先做一次「单独跑通」测试

在终端里分别试一句话，能正常回话，才说明助手本身是通的：

```
claude -p "Reply with exactly: pong"
codex exec --skip-git-repo-check "Reply with exactly: pong"
```

两个都返回 pong，就放心往下走。有一个不通，先修它——**终端里单独跑不通的助手，进了 T3 Code 一样跑不通**，在 T3 Code 里排错只会更绕。这条经验来自真实实测：先分开验证，再合起来用，是全书反复出现的排错总原则。

没有官方订阅，能用第三方接口吗

能。T3 Code 不自己管密钥，它直接沿用每个助手自己的配置文件：Claude Code 读它自己的全局设置文件，Codex 读它自己的配置文件。你把第三方接口的地址和密钥按各助手自己的规矩填好，终端里能跑通，T3 Code 就能调用。想让订阅和接口密钥并存，可给同一助手建多个「实例」，各用各的配置目录。密钥要标记为敏感信息，保存后不再明文显示。

> **别把配对码当普通链接：**后面远程连接会用到配对链接和授权码，官方反复提醒：把它们当密码对待，不要放进截图、日志或问题反馈里。

> **本章验收：**终端里至少一个助手能回 pong；手边有一个 Git 项目文件夹。两项都打勾，再翻下一章。

CHAPTER 2

第 2 章　安装：三条路线，小白走桌面版

路线	怎么做	适合谁
A. 桌面版（推荐）	苹果电脑：`brew install --cask t3-code`；Windows：`winget install T3Tools.T3Code`；也可以去官网下载安装包	绝大多数小白。装好自动带命令行工具，双击就能用
B. 先试一次	终端运行 `npx t3@latest`，不安装直接跑	先花 5 分钟体验的人
C. 脚本安装	苹果/Linux：`curl -fsSL https://t3.codes/install.sh | sh`；Windows PowerShell：`irm https://t3.codes/install.ps1 | iex`	跑在服务器、常开电脑上的人

装好后，先认识这几条命令

命令	干什么
`t3`	启动服务并打开网页界面，日常最常用
`t3 serve`	只启动服务、不开浏览器，服务器场景用
`t3 up`	开放远程：让手机和其他设备连进来（第 12 章）
`t3 update`	升级。官方事实：更新会中断正在跑的 agent，但 Continue threads after restarts 默认开启、可断点续跑；先收尾、提交推送是作者建议，非官方要求
`t3 doctor`	回收数据库空间，长期用、磁盘变大时跑
`t3 thread start / send / show`	纯命令行开任务、发消息、看结果

启动成功后会显示服务就绪提示和本地地址（默认端口 3773），并可能打印配对二维码。

第一次打开，先做三件事

1. **确认助手状态：**打开 Settings → Providers，看助手是不是「已就绪、已认证」。不是就补登录或指定程序路径（第 15 章有解法）。
2. **改默认权限：**去 Settings → General → New threads → Permissions 把新任务默认权限从 Full access 改成 Supervised。为什么必改，第 4 章讲透。
3. **添加第一个项目：**把备好的 Git 项目加进来，或在项目目录里运行 `t3 project add .`。

> **本章验收：**界面能打开、至少一个助手已就绪、默认权限不是 Full access、项目列表里有你的项目。四项齐了，就往下走。

CHAPTER 3

第 3 章　五个概念讲透：理解它们，界面再无生按钮

T3 Code 的界面按钮不少，但全部只围绕五个词转。把这五个词的关系记住，后面所有操作都是它们的组合。

概念	人话解释	你要记住的点
Environment **运行环境**	一台正在运行 T3 Code 服务的电脑	活儿永远在这台电脑上干；手机、浏览器只是遥控器。一句话：界面负责看，环境负责干
Project **项目**	环境里登记的一个代码文件夹	同一仓库在两台电脑上登记，是两个项目，因为它们属于两个环境
Workspace **工作目录**	一条任务实际改文件的那个文件夹	两种选法：直接用项目当前文件夹，或新建一个隔离副本（worktree），第 10 章细讲
Thread **任务对话**	一件活儿的完整对话与历史	绑定一个助手、一个工作目录；关掉界面、重启服务都还在，能跨天接着干
Turn **一轮**	你发一条消息，到助手干完为止	每轮前后自动存档，所以能回答「这句话到底改了什么」，也能整轮退回

把五者串成一句话

你在**手机界面**上，选中**家里电脑这个环境**里的**某个项目**，开一条**任务对话**，让它在**独立工作目录**里干活；你每发一句话就是**一轮**，每轮的改动都被单独存档。

新手最容易混的三组词

- **助手（Provider）不等于模型：**助手是干活的工具（比如 Claude Code），模型是它脑子里可换的引擎。一条任务先选助手，再选模型。分清两件事：同一助手（provider）内换模型，一般继续原会话、不丢失，放心换；有损的是中途换助手（provider）——新助手只拿到按预算挑选的部分消息，推理过程、工具调用和附件都带不过去（第 11 章细讲）。
- **配对不等于登录：**手机和电脑「配对」，只是让手机能连上你的 T3 Code 服务；它不会替你登录任何编程助手。助手登录永远在运行服务的那台电脑上完成。
- **存档点不等于备份：**每轮的存档点能救回一次改错，不等于代码备份。重要工作照样要提交 Git、推到远程仓库。

> **本章验收：**合上书，你能说清「我的任务跑在哪台电脑、哪个文件夹、用的哪个助手」。能说清，下一章的安全课就听得懂了。

第 4 章　安全第一课：权限四模式，第一天就设对

权限模式决定一件事：助手改文件、跑命令之前，要不要先问你。T3 Code 一共四档。最坑的事实先说：新任务的出厂默认是最高档 Full access——什么都不问，直接干。官方文档、多篇实测都把「先改默认」列为头号提醒。

模式	改文件	跑命令等动作	什么时候用
Supervised 全程盯着	先问你	先问你	新手默认档；重要仓库、不熟悉的任务一律用它
Auto-accept edits 改文件放行	自动通过	需要时问你	日常改代码熟了之后；文件随便改，命令还归你管
Auto 自动审查	例行动作放行	高风险动作问你	环境可控、任务可信时提速用
Full access 完全放行	不问	不问	只在可丢弃、可隔离的环境里用，比如练手副本、一次性试验

怎么改默认（只改一次，终身受益）

路径：Settings → General → New threads → Permissions（设置 → 通用 → 新任务 → 权限）。注意：Settings → Threads 是另一组设置（自动归档等），不要去那里找权限，把默认改成 **Supervised**。单个项目还可以再覆盖一层自己的默认。开新任务时，在输入框附近也能临时切换当前任务的权限，快捷键是 `mod+shift+a`。团队电脑上，管理员还能用统一策略把最高可选档位锁死。

四个必须知道的细节

- **四档是统一外壳，实际拦截能力看助手：**比如 Auto 档在不同助手上走的是各自的自动审查；个别助手只有两档可用。不要以为切了档就万事大吉，重要动作还是要看。
- **子任务的批准会冒到主对话里：**助手派出去的小助手要权限时，会带着名字出现在主对话里，你不用切窗口就能批。
- **助手反问你问题，不受权限档位限制：**它拿不准需求时提问，是正常协作，不是报错。
- **隔离副本挡不住整台电脑：**独立工作目录只能防几条任务互相踩文件，防不了助手访问这台电脑的其他资源。用 Full access 之前，先想清楚这台机器上有什么。

红线清单（背下来）：真实仓库不用 Full access 起步；不认识的命令不点批准；批准前先看它要动哪个文件、跑什么命令；重要代码先提交 Git 再让助手动手；配对链接和授权码不进截图、不外发。

本章验收：默认权限已改为 Supervised；你能说出四档的区别。下一章正式开干时，你会发现「它每步都问我」不是麻烦，而是新手最好的老师。

第 5 章　第一条 Thread：八步跑通，今晚就能交活

下面这八步，就是你以后每件活儿的标准动作。第一次请拿一个无关痛痒的小任务练，比如「给计算器程序修一个加法 bug 并补测试」——真实实测里，同类任务两个主流助手都在约半分钟内完成并通过测试，节奏就是这么快。

1. **选项目。**在左侧项目列表点你的项目。没有就先添加项目文件夹。
2. **开新任务。**点 New thread（新任务）。
3. **选助手和模型。**新手就用你第 1 章跑通的那个助手，模型用默认推荐档，不要第一天就纠结选型。
4. **确认权限是 Supervised。**不是就当场切过来。
5. **选工作目录。**第一次练手选「当前目录」就行；只要你打算同时开第二条任务，就选「新建 worktree（隔离副本）」。强烈建议（不是官方硬性规则）：一个目录同一时间只让一条任务写入——两条任务同写一个目录会互相覆盖；要并行写，就给第二条开 worktree。
6. **写第一句话。**别写「帮我修一下 bug」。照这个句式写：*「calc.py 里的 add 函数有 bug，请修复它，并新增 test_calc.py 用 unittest 测试，最后运行测试并报告结果。」*——对象、动作、验收，一句齐全。第 7 章会把这套写法拆成模板。
7. **盯批准、看改动。**它要改文件、跑命令时会停下来等你批准，逐个看清再点。干完后打开改动对比（diff），重点看「本轮改动」：删了什么、加了什么，和你的要求对不对得上。不要只看它最后的文字总结——**以你亲手重跑测试的结果为准**，实测中出现过助手写测试文件时缩进出错、自己重试才通过的情况，它会如实汇报，但你仍要复核。
8. **提交收尾。**确认没问题，在界面里提交（commit）、推送（push），需要的话一键开 PR。任务干完可以收起归档（settle），长期没动静的任务也会自动归档。

补充：输入框上限是 120,000 个字符（characters，不是 12 万个汉字），整份需求文档、整段报错日志都能直接贴；粘贴内容超过 32 KiB 会自动转成文本附件；图片可以直接附上。输入斜杠 `/` 唤出命令菜单；想在同一条任务里重开一个干净的助手会话，用 `/clear`。

> **本章验收：**一条任务从创建到提交完整走完，且你亲手重跑过一次测试。走通这一遍，后面所有章节都是在这八步上加buff。

CHAPTER 6

第 6 章　界面地图：四个区域，各管一摊

左栏 · 任务总管
项目和任务列表都在这里。每条任务实时显示状态（在跑、等批准、已完成）和耗时。重要的任务可以钉在最上面。在新建任务草稿里按 Cmd+Enter，会在后台启动这条任务、并立即再打开一个新草稿，方便连开多条；在已有对话里，它是排队发送与立即干预之间的反向操作（与普通发送键互换），别把它理解成“把当前任务丢到后台”。

中栏 · 对话主战场
你和助手的对话、它的计划、工具调用、批准请求、子任务卡片都在这里。子任务卡片会显示用的什么模型、花了多久、用了多少额度。想临时问个旁支问题又不打断主线，用 `/btw` 开一个新的侧聊（裸 `/side` 是重开最近一个侧聊），聊完把结论带回主线。

右栏 · 审查台
看改动对比、浏览文件、看任务血缘（Lineage：这条任务派出了哪些子任务）。改动对比有三种看法：本轮改动、整条分支相对主干的改动、还没提交的改动，分别回答「刚才那句改了啥」「这件活儿总共改了啥」「还剩啥没提交」。

下方 · 终端抽屉
内置终端就在对话下方，自己跑测试、看日志、复现助手跑过的命令，不用离开界面。注意：终端跑在运行服务的那台电脑上，你在手机上打开终端，命令也是在电脑上执行，不是在手机上。

审查的正确姿势（高手和新手的分水岭）

1. 先看**本轮改动**，确认这一句话的改动范围没超纲——让它改 A 文件，它却动了 B、C，先停下来问为什么。
2. 在改动对比里**选中几行代码直接写评论**，评论会进入你的下一句话，等于指着代码跟助手说话，比打字描述位置准得多。文件面板里选行附评论也是同一招。
3. 改错了不要慌：找到出错的那条历史消息，用「从这里编辑（Edit from here）」，可以选**连文件一起退回**，或**只退对话、保留文件改动**。个别助手不支持对话回退，界面会说明。注意：如果同一目录还有别的任务在干活，文件退回会被拒绝，这是保护机制。

消息排队：想到什么先扔进去

助手干活时你突然想到补充要求，不用等：直接发出去，可以选择**插进当前这轮**，或**排队等下一轮**。队列存在服务器上，重启也不丢，还能编辑和调整顺序。用量到顶的任务也不用守着：是否在限额恢复后自动接着跑，取决于你的设置——它提供手动、定时、自动恢复等选项，先去设置里确认选的是哪一种，不是无条件自动续。

> **本章验收：**你能不看手册指出：任务状态在哪看、本轮改动在哪看、终端在哪开、改错了从哪退回。四处都找到，界面这一关就毕业了。

第 7 章　提示词写法：六要素齐全，返工砍一半

助手干得好不好，一大半取决于你那句话写得清不清楚。记住六要素：目标、范围、约束、验收、禁止、输出。写全了，它一次做对；写漏了，它只能猜，猜错就返工。

要素	写什么	例句
目标	要解决什么问题、做成什么样	修复 add 函数算错的问题
范围	只动哪些文件 / 模块	只改 calc.py 和新增的测试文件
约束	必须遵守的规矩	不改函数签名，不引入新依赖
验收	怎么算做完	运行全部测试并贴出结果
禁止	明确不要做的事	不要顺手重构其他函数
输出	最后汇报的格式	用三行说明：改了什么、测试结果、剩余风险

万能模板（直接抄，填空即用）

```
任务：[一句话说清要做什么]
范围：只处理 [文件/模块]，其他文件先不要动。
要求：[约束1]；[约束2]。
禁止：不要 [最怕它自作主张做的事]。
验收：完成后运行 [测试/命令]，把真实输出贴给我。
汇报：改了哪些文件、测试结果如何、还有什么风险没覆盖。
```

五个立竿见影的技巧

- **先让它出计划，再让它动手：**复杂任务第一句只说「先给计划，不要动文件」，计划靠谱了再说「按计划执行」。计划阶段选行评论、改方向，成本最低。
- **指着代码说话：**在文件或改动里选中具体几行附评论，比「那个函数有点问题」精确十倍。
- **一次只派一件活：**一句话塞三件不相干的事，它会平均用力、验收混乱。拆成三条任务，还能并行。
- **把报错原文喂回去：**测试红了，把完整报错贴回对话让它修，不要自己翻译概括——你概括的那句，往往正好漏掉关键行。
- **长任务分段验收：**大活儿拆成「先做第一步，做完停下来等我确认」，每段都有存档点，错了只退一段。

> **反面教材：**「帮我优化一下这个项目」——没有范围、没有验收、没有禁止，助手只能大开大合乱改一气。把它改成模板句式，哪怕只填三行，都比一句空话强。

第 8 章　实操场景（上）：修 Bug、做功能、补测试

场景一：修 Bug —— 先复现，再修，最后留证据

适用：程序报错、算错、页面不对。**权限：**Supervised。**目录：**当前目录即可。

1. 第一句先让它复现：「运行 [命令/步骤] 复现这个问题，把报错原文贴出来，先不要改代码。」
2. 它定位到原因后，再放行修改。批准前看一眼：要动的文件是不是出问题的地方。
3. 修完让它重跑复现步骤，并跑一遍相关测试。
4. 你亲手再跑一次测试，看改动对比里有没有「顺手」改的无关文件，有就让它撤掉。

```
calc.py 的 add(2,3) 返回了 -1。先运行复现并贴出结果，
定位原因后只修复 add 函数，不要动其他函数。
修完新增回归测试并运行全部测试，把真实输出贴给我。
```

场景二：做一个小功能 —— 先计划，后执行，分两轮

适用：加页面、加接口、加导出按钮这类边界清楚的新功能。**权限：**Supervised 起步，计划确认后可切 Auto-accept edits 提速。**目录：**功能稍大就开隔离副本。

1. 第一轮只要计划：列出要新建/修改的文件、每处改什么、怎么验收。
2. 你在计划上删繁就简（新手最常见的坑是让它一次做太多），确认后说「按计划执行，每完成一步汇报一次」。
3. 完成后在下方终端自己把功能点一遍，或让它启动项目脚本给你预览。
4. 验收句式：「对照计划逐条核对，告诉我哪条没做到。」——让它自查，比你逐行翻代码快。

场景三：补测试 —— 性价比最高的新手任务

适用：老项目没测试、改代码心里没底。**权限：**Auto-accept edits 即可（只新增测试文件时风险低）。

- 提示词要点：指定测试框架、覆盖哪几个函数、必须包含边界情况（空值、超大值、异常输入）。
- 验收只认一件事：测试真的能跑，且把被测函数改错时测试会变红。你可以故意让它演示一次「改错 → 变红 → 改回」，确认测试不是摆设。
- 补测试和改功能要分开两条任务：先补测试锁住现状，再改功能，红了就是改坏了，责任清清楚楚。

> **三场景共用心法：**每件活都以「能重跑的验收」结尾——测试输出、复现步骤、计划核对表，三选一。没有验收的任务，等于没做完。

第 9 章　实操场景（下）：审查、并行、手机远程

场景四：让一个助手审查另一个助手的活

打法：实现任务干完后，另开一条只读型任务（权限 Supervised、工作目录选当前目录即可），让另一个助手只看改动对比、只提意见、不动文件。提示词：「只审查这条分支相对主干的改动，按严重程度列问题，每条指出文件和行号，不要改任何文件。」审查意见你认可哪条，再回实现任务里让它改哪条。**写的人和查的人分开**，是小白最容易学会的质量保险。

场景五：两件活并行 —— 一人成团的秘诀

1. 两条任务各开一个隔离副本（worktree），各有自己的分支和文件夹，谁也踩不到谁。
2. 左栏盯状态：哪条在跑、哪条在等批准，一目了然；等批准的那条先处理，别让它干等。
3. 两条都完成后分别看改动、分别提交、分别开 PR，合并顺序你说了算。
4. 注意共用资源打架：两个副本可能抢同一个端口、同一个本地数据库。启动第二个项目服务前先改端口，或让任务在创建副本时自动分配——项目配置 t3.json 的准备脚本就能干这事（第 10 章）。

场景六：出门在外，手机批批准



手机只是遥控器：活在家里电脑上干，你在哪都能批。

1. 出门前在电脑上运行 `t3 up` 开启远程（或在桌面版设置里打开共享），任务先派出去。
2. 手机装官方应用，登录同一账号或扫码配对，连回家里电脑这个环境。
3. 助手等批准时手机会收到推送，点开看清要跑什么命令、改什么文件，再点批准或拒绝。拿不准就拒绝，回去在电脑上细看——拒绝不会毁掉任务。
4. 路上想到新要求，直接在手机上给任务发消息排队；语音输入转文字会先进草稿框，检查无误再发送。

第 10 章　Worktree、分支与一键 PR：并行不踩踏的底层规矩

Worktree（工作树）这个词听着唬人，其实就是：同一个仓库，另开一个文件夹副本，自带独立分支。几条任务各住一间房，公共的仓库历史是同一本账。

选法	什么意思	什么时候选
当前目录	助手直接改你眼前的项目文件夹	只有一条任务在写；看文档、做审查这类只读任务
新建 worktree	给这条任务单独开副本 + 新分支	两条以上任务同时写；大改动先隔离试；怕改坏主目录时

强烈建议（官方并无此硬性规则）：一个当前目录，同一时间只让一条任务写入。两条任务同写一个目录会互相覆盖，并行写任务请开 worktree；文件退回在共用目录时会被官方保护机制拒绝。审查、查资料这类只读任务可以共用当前目录。多开副本的代价也要知道：每个副本占一份磁盘，还可能各装一遍依赖；设置里可以开自动清理，磁盘紧了跑一次 `t3 doctor`。

从任务到 PR 的标准收尾

1. 在任务里看「整条分支的改动」，确认这件活的全部改动都在预期内。
2. 点提交：写清提交说明（界面能自动生成标题和正文，生成后自己改一遍，别直接用机器味的原文）。
3. 推送到远程，再一键开 PR；修改上次提交也支持。注意范围：草稿 PR、堆叠 PR 以及 Merge stack / Rebase stack 都是 GitHub 场景，官方 source-control 文档只对 GitHub 的 stacks 有详述，不要推广到其他平台；GitLab 上的对应物叫 merge request（合并请求）。
4. PR 合并后，这条任务会自动归档收起，列表保持干净。

项目配置 t3.json：把重复动作写成一键

项目根目录放一个 t3.json，可以预先写好常用脚本（上限 50 个）：一键启动开发服务、每个新副本创建后自动装依赖、自动分配端口等。关键一步别漏：t3.json 不是放进目录就自动生效，必须先在 Actions 菜单里导入才生效；而且 runOnWorktreeCreate（新建副本时自动跑的脚本）仅第一个生效，写多个不会依次全跑。脚本运行时能拿到两个现成变量：项目根目录和当前副本目录，写脚本时用它们区分「共用的东西放哪、隔离的东西放哪」。把你每周都要手敲三遍的命令写进去，这是从新手到熟手最实在的一步。

> **本章验收：**同时开两条任务、各自独立副本、各自提交、各自开出 PR，且两个 PR 的改动互不串味。做到这一步，你已经具备「一人多线程」的完整工作流了。

第 11 章　多代理分工：让强的干它擅长的，别中途硬换人

不同助手有不同的长处：有的擅长先想清楚，有的擅长闷头实现，有的审查更挑剔。T3 Code 0.1 版本的编排能力（Orchestrator V2）就是为分工而生：一条主任务里的助手，可以把子活儿派给任意助手和模型，并等它交活。

黄金分工：规划、实现、审查三段式

环节	派给谁	交付物
规划	擅长理需求、拆步骤的助手	一份可核对的执行计划
实现	写代码快、工具稳的助手	改动 + 测试结果
审查	另一个助手（必须换人）	按严重程度排的问题清单

派活的动作叫 delegate_task（委派子任务）。子任务的批准请求会带着名字回到主对话，你在一个界面里全批完；子任务卡片显示模型、耗时和用量，主任务的血缘面板能看到整条委派链。你的提示词可以写得很直白：「先自己出计划；实现环节委派给 [另一个助手] 做；做完再委派给 [第三个助手] 只审查不改代码；最后汇总三方结果给我。」

换模型可以，换助手（provider）要当心

T3 Code 允许一条任务中途换模型、也能换助手（provider），界面会标记切换点。两者完全不同：同一 provider 内换模型，一般继续原会话、不丢失，尽管换；换 provider 才是有损交接——新 provider 只拿到按预算挑选的部分消息，前任的推理过程、工具调用记录和附件都不会带过去。结论：**借另一个助手的长处，用委派；想原地换人，先让前任写一份交接总结**（做了什么、改了哪些文件、还剩什么、踩过什么坑），新助手读完总结再接手。

选助手的土办法（够用，不玄学）

- 拿不准就用「双跑对比」：同一件小活开两条任务、两个助手各干一遍，比改动质量和测试结果，用数据选主力。
- 主力只定一个，另一个当审查和替补。频繁换主力，每次都要重新磨合提示词习惯。
- 用量是硬约束：谁的额度先紧，谁就干短活；长活留给额度宽裕的助手（第 13 章教你看额度）。

第 12 章　远程与手机：三条路，按信任程度选

路线	怎么连	适合
T3 Connect 官方穿墙	电脑上 `t3 up` 选它，或桌面版设置里开启；手机/浏览器登录同一账号即可连回	最省事，推荐小白首选。免费账号可连 3 台运行 T3 Code 的机器（手机和浏览器不计数），更多机器需要付费档
局域网配对	同一网络下用 `t3 pair` 生成二维码或配对链接，扫码即连	在家、在办公室临时用手机看一眼，不想走任何外部服务
Tailscale	设备都加入同一个私有网络后直连	已有自己的私有网络、想把连接完全攥在自己手里的人

三条路的共同点：活儿始终在运行服务的那台电脑上干，远程的只是画面和操作。电脑关机或休眠，任务就停——想全天候跑活，就把 T3 Code 装在一台常开的电脑或服务器上。后台常驻也有现成办法：命令行可用服务方式安装；不同系统开机自启的行为不同，常开场景更推荐用 Linux 主机。

远程安全四条

- 新设备接入时开启「逐台批准」，陌生设备进不来；批准请求有有效期，过期自动作废。
- 配对链接、授权码按密码管理：不截图外发、不贴进公开的问题反馈。
- 远程批批准时更要看清命令再点——你不在电脑前，撤回成本更高。
- 公用电脑上用完网页端记得退出登录。

桌面版的隐藏本事：接管远程电脑

桌面版还能通过 SSH 接管另一台电脑上的 T3 Code 服务：代码和构建环境本来就在服务器上的团队，不用换工作习惯，在本地一个界面里就能管多台机器的任务。多台环境之间还能跨环境搜索任务，点开就跳到对应机器的那条对话。

> **本章验收：**关掉电脑屏幕，用手机连回家里环境，批准一条正在等的请求，并给任务排队发一条新消息。三步都顺，远程这一关毕业。

第 13 章　费用与 Usage：T3 Code 免费，钱花在订阅上

先把账算清：T3 Code 本体免费、开源（MIT 许可），官方明确目前没有任何付费使用 T3 Code 本体的入口。你要付的钱，只有各家助手自己的订阅或接口费用，一分不多、一分不少。唯一的例外是远程机器数：T3 Connect 免费连 3 台主机，超过才需要更高档位；不想付费，改走局域网或 Tailscale 即可，功能不受影响。

Usage 面板：三个数字必须分开看

数字	是什么	拿来干什么
Token 用量	助手实际消耗的文字量记录，含缓存节省	看哪条任务最费、哪种活最烧额度
订阅额度	各家给你的限额窗口（比如几小时一档的额度）	以这个为准安排长活，别把额度打爆在半路
API 等效成本	按接口牌价折算的「如果按量付费要花多少」	只用来估算和对比，不是你的账单，别被它吓到

面板会把多家助手的用量汇总到一处，还能按助手和模型筛选。没有公开牌价的模型可以手动填单价。历史保留有明确口径：服务器默认保留扫描到的用量历史 92 天，用量视图最多显示 90 天，超出这个范围的旧记录就看不到了。

省额度的四个习惯

- **小活不用最贵模型：**改文案、补测试这类活，用中档模型足够；难题才上旗舰。
- **提示词写清，减少来回轮次：**每一轮往返都烧额度，第 7 章的六要素就是省钱工具。
- **长任务排队等额度恢复：**撞上限额后是否自动续跑要看设置——提供手动、定时、自动恢复等选项；先确认已选自动恢复，任务才会自己接着跑，不是无条件自动续。
- **多账号分开记账：**个人和工作订阅并存时，用多实例分开配置、界面里切换，历史各归各的目录，不串账。

> **Claude 5 小时额度消耗的已知个案（别当普遍定论）：**曾有人 A/B 实测：同一任务交互版约消耗 7%，经 T3 Code 约 13%-16%，约 1.7 倍——这是单人实测，不是普遍结论。进展要分两边看：T3 侧追踪 issue #24649 已于 11 月 2 日左右关闭（改用 t3-code 入口后，单人复测显示与交互版同速；官方 FAQ 称自 v0.1.7 起同速）；但 Anthropic 侧 issue #97074 截至 2026-11-19 仍开放，没有 Anthropic 确认的修复。旧版用户仍可能受影响：先升级，再对照官方额度页面核对。

第 14 章　15 条高手技巧：从「能用」到「顺手」

1. **改默认权限只做一次，管一辈子：**Supervised 设为默认，特殊任务再临时放宽，而不是反过来。
2. **每轮都是存档点，胆子可以大一点：**敢让它尝试，是因为退得回去——但退回前先确认没有别的任务在共用同一目录。
3. **/btw 侧聊问旁支：**主线干活时想查个概念、问个方案，用 /btw 开新侧聊（裸 /side 是重开最近一个侧聊），不污染主线上下文，结论带回即可。
4. **消息先排队，灵感不丢失：**它在跑、你想到补充，先发出去排队，可编辑可重排，比记在备忘录里强。
5. **定时任务交给排程：**固定重复或单次定时的活（如每天跑一遍检查）可以排程；撞额度后能否自动续跑要看设置里的恢复选项（手动 / 定时 / 自动恢复），先设好再指望它自己续。
6. **任务链接分享，先分清两种：**线程链接（Copy thread）只有已连接、且运行着该线程的那台机器的人才能打开，发给没连接的同事点开也看不到。访客链接是另一回事：仍是 beta，需先开 Labs + T3 Connect，24 小时过期、只能在一个浏览器里打开、看不到 diffs（改动对比），访客发的消息还要经你批准才会执行。
7. **钉住重要任务：**并行多了，左栏只盯钉住的和等批准的，其余让状态自己跑。
8. **选行评论胜过千言：**审查时选中代码行写意见，位置零歧义，这是全书最值得养成的肌肉记忆。
9. **t3.json 写一次，省一百次手敲：**启动服务、装依赖、分配端口，全部脚本化（见第 10 章）。
10. **同一助手建双实例：**订阅一个、接口密钥一个，界面切换，互不串账（见第 1 章）。
11. **找不到程序就填路径：**用版本管理工具装的助手可能不在默认路径里，在供应方设置里填 Binary path（程序路径）即可认出。
12. **升级纪律（作者建议，不是官方要求）：**先收尾任务、提交推送，再 `t3 update`。官方事实是：更新会中断正在跑的 agent，但 Continue threads after restarts 默认开启，重启后可断点续跑，所以偶尔被打断也不必慌。
13. **版本一起升：**版本不匹配是双向拒绝——app 比 server 新会被拦，server 比 app 新也会拒绝连接；桌面、手机、各台机器的服务尽量同批更新到同一版本。
14. **遥测可关，但分清开关：**官方说明会发送不含提示词、代码和密钥的产品使用统计；服务端可用环境变量 `T3CODE_TELEMETRY_ENABLED=false` 关闭。注意：浏览器端没有这个开关；手机端另有独立的崩溃报告开关，要单独关；关遥测也不等于关掉通知用的 Publish agent activity（发布 agent 活动），那是另一项。
15. **命令行是后备全能键：**界面不在手边时，`t3 thread` 系列命令能开任务、发消息、看结果，服务器上照样干活。

> **挑选顺序：**新手先练第 5、6、8 条（侧聊、排队、选行评论），一周后加第 9、10 条，并行跑顺后再碰排程和多实例。一次全上，只会乱。

第 15 章　排错大全：九成问题，回到三处查

排错总原则只有一句：先分开验证，再合起来看。助手单独跑不通、程序路径找不到、项目没添加——这三处占了绝大多数报错。

现象	多半是	解法
设置里看不到我装的助手	助手没装在运行服务的那台电脑上，或不在路径里	在那台电脑上装好并登录；在供应方设置里填 Binary path 指定程序位置
任务里报 401 / 认证失败	密钥错误、过期或登录失效	先在终端单独跑第 1 章的 pong 测试，逐项查密钥、接口地址、登录状态；终端通了再回 T3 Code
报 429 / 被限流	触发用量窗口或并发太高	等窗口恢复或降低并行数；长任务可排队，限额恢复后是否自动续跑取决于设置里的恢复选项（手动 / 定时 / 自动恢复）
提示没有匹配的项目	当前文件夹还没添加为项目	先添加项目（界面添加或 `t3 project add .`）
命令行选助手报「实例被禁用」	填的名字不是实例编号	去 Settings → Providers 抄准确的实例编号，并确认该供应方已启用
提示已有服务在运行	同一数据目录只能跑一个服务	直接连正在跑的那个；确实要另开一个，换一个数据目录
手机在外面连不上	远程链路没通或配对过期	先在家用同一网络验证，再查 T3 Connect 状态；仍不行改走 Tailscale；新设备批准请求约 10 分钟过期，过期重新发起
改错了想退回却被拒绝	同一目录有其他任务在工作	等其他任务停下再退；以后并行任务一律开独立副本
升级后应用连不上服务	版本不匹配，双向都会拒绝：app 比 server 新会被拦，server 比 app 新也会拒绝	把桌面、手机和各机服务同批升级到同一版本
额度掉得异常快	可能是旧版的已知计量个案（单人实测约 1.7 倍，非普遍定论；T3 侧 #24649 已关闭、称 v0.1.7 起同速，Anthropic 侧 #97074 截至 2026-11-19 仍开放），也可能只是任务太费	先升级到最新版；在 Usage 里按模型筛出最费的任务，换中档模型重跑对比

第 15 章（续） 7 天上手计划：每天一个成果，不贪多

照着这张表走，每天只完成一行，验收不过不进入下一天。一周后，你会自然用上前面所有章节。

天	任务	验收标准
第 1 天	装好 + 接通一个助手 + 改默认权限	第 2 章四项验收全勾
第 2 天	跑通第一条修 Bug 任务	亲手重跑测试通过并提交
第 3 天	用模板写提示词，补一轮测试	测试改错能变红
第 4 天	两条任务并行，各开独立副本	两个 PR 改动互不串味
第 5 天	换人审查：第二个助手只审查	拿到一份分级问题清单
第 6 天	开通远程，手机批一次批准	第 12 章三步验收完成
第 7 天	写好 t3.json + 试一次委派分工	一键启动项目；完成一次三段式分工

☐ 默认权限不是 Full access ☐ 至少一个助手终端单独跑通 ☐ 重要代码已推远程仓库 ☐ 配对码从未外发

APPENDIX

附录　速查卡：贴在手边，用时不翻书

命令速查

命令	用途
`t3` / `t3 serve`	启动服务（后者不开浏览器）
`npx t3@latest`	不安装，先试一次
`t3 up` / `t3 pair`	开启远程 / 生成局域网配对码
`t3 project add .`	把当前文件夹添加为项目
`t3 thread start / send / show`	命令行开任务 / 发消息 / 看结果
`t3 update` / `t3 doctor`	升级 / 回收数据库空间
`claude auth login` / `codex login`	在运行服务的电脑上登录助手

安装速查

平台	命令
苹果电脑（桌面版）	`brew install --cask t3-code`
Windows（桌面版）	`winget install T3Tools.T3Code`
苹果 / Linux（脚本）	`curl -fsSL https://t3.codes/install.sh | sh`
Windows（脚本）	`irm https://t3.codes/install.ps1 | iex`

附录（续） 术语表与官方入口

术语表

词	一句话
Thread（任务对话）	一件活儿的完整对话，关掉界面也不丢
Turn（一轮）	你发一句话到它干完，前后都有存档点
Worktree（隔离副本）	同仓库另开的文件夹 + 独立分支，并行不踩踏
Diff（改动对比）	到底改了哪几行，审查就看它
Provider（供应方/助手）	真正干活的编程工具，如 Claude Code、Codex
PR（合并请求）	把一条分支的改动正式申请合进主干
delegate_task（委派）	主任务把子活儿派给其他助手并等结果

官方入口（以官网为准）

T3 Code 官网：https://t3.codes
网页版入口：https://app.t3.codes
开源仓库：在 GitHub 搜索 pingdotgg/t3code（团队 pingdotgg 的 T3 Code 仓库，安装、权限、远程、供应方文档都在仓库内）

资料基准说明：本手册依据 2026 年 11 月公开可查的 T3 Code 官方文档、GitHub 仓库说明及多篇同期实测文章交叉核对编写，基准版本为 0.1.9（2026-11-11）；0.1.10 已于 2026-11-17 发布。T3 Code 仍处快速迭代期（官方自述仍很早期、会有 bug），功能与界面以你安装时的实际版本和官方最新文档为准。文中涉及的额度计量个案为 2026 年 10-11 月的公开记录，不代表当前版本表现。

> **最后一句话：**工具再强，也只放大你已经有的工作习惯。先把「写清要求、盯住批准、亲手验收」这三件事练成本能，T3 Code 才会从一个新玩具，变成你每天离不开的总控制台。