创始人行动手册：打造 AI 原生创业公司

The Founder's Playbook: Building an AI-Native Startup 中文版

Claude

目录

第 1 章：为 2026 重新启动的创业生命周期 3

第 2 章：创始人的含义正在改变 5

第 3 章：构想阶段 8

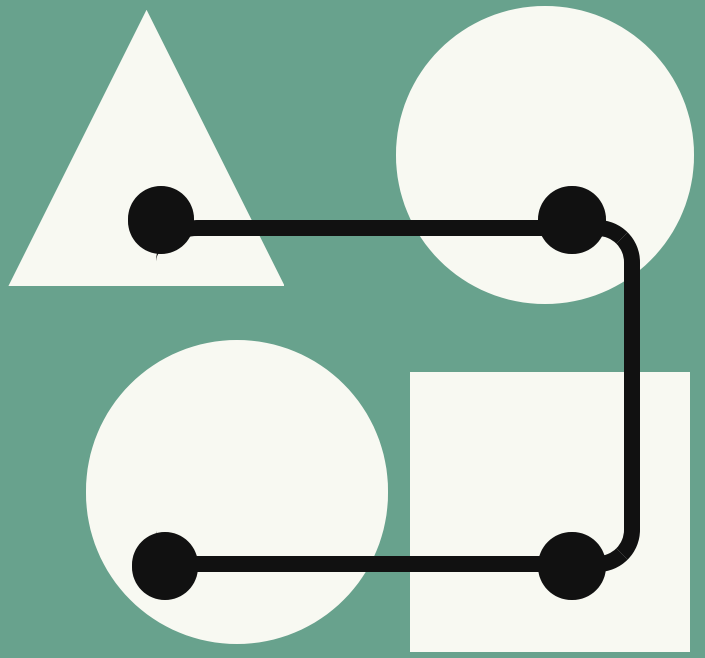
第 4 章：MVP 阶段 15

第 5 章：发布阶段 20

第 6 章：规模化阶段 24

第 7 章：同一份工作，新的规则 29

资源 31



Chapter 1

第 1 章：
为 2026 重新启动的创业生命周期

Chapter 1

第 1 章：为 2026 重新启动的创业生命周期

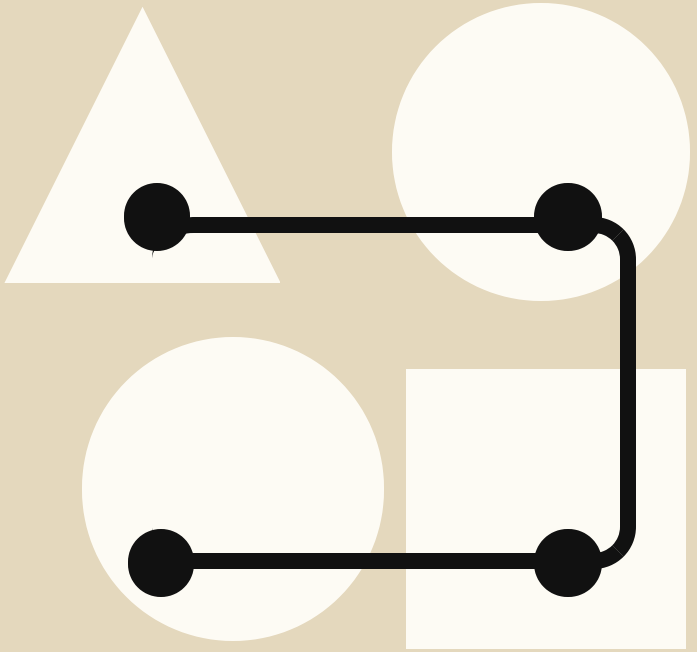
AI 正在重塑创业公司的构建方式。今天，从未写过一行代码的创始人也能发布生产级应用；而“10 人独角兽”也已经从草根逆袭故事，变成了一种可以主动规划的行动方案。

到 2026 年，AI 可以编写生产代码、开展市场研究、综合竞争格局、起草投资人材料，并自动化运营工作流。曾经，即使是经验丰富的技术创始人，也要面对陡峭的学习曲线，才能把实现想法所需的工具、平台和系统整合起来；而 AI 最大的改变，正是抹平了“谁能创办公司、谁能打造产品”这件事上的门槛。

在 2026 年，一个好想法能把创始人带得比以往更远。智能体式编码把过去需要一支工程团队完成的工作，压缩成了创始人自己就能交付的成果。

传统的创业成长路径假设，从想法走向规模化的过程是：验证 → 融资 → 招人 → 构建 → 再融资 → 增长 → 再招人 → 循环往复。现在，AI 已经打破了这样一种预期：创业生命周期中每进入一个新阶段，就必须有更大的团队、不同的技能组合，以及新一轮融资。

这本行动手册会根据这些新现实，重新绘制创业旅程的四个核心阶段：构想、MVP、发布和规模化。我们会考察，当 AI 成为技术开发和组织建设的核心时，每个阶段会是什么样；每个阶段应该使用哪些工具；以及正在使用这些工具的创始人如何压缩时间线。如果你已经准备好找到从想法到退出之间的最短路径，请继续读下去。



Chapter 2

第 2 章：
创始人的含义正在改变

Chapter 2

第 2 章：创始人的含义正在改变

过去，创始人往往由“自己能做什么”来定义：技术创始人写代码，非技术创始人负责业务运营和成交。但到 2026 年，创始人可以使用的模型、系统和 AI 智能体，已经拆掉了“能构建的人”和“有值得构建的想法的人”之间的墙。

AI 原生创业公司正在从根本上改变“创始人”这个角色的含义。现在，没有工程背景的人也能构建生产级软件，把自己的想法变成现实；而一个技术能力很强、但商业知识有限的创始人，也可以轻松产出上市策略、财务模型和高度 polished 的融资演示文稿。

历史上，创始人大部分时间都处于执行模式：写代码、管人、处理日常运营工作。在 AI 原生创业公司里，创始人的角色会更少像一个个人贡献者，而更像智能体的编排者。这些智能体是专门化的 AI 助手，可以读取文件、运行命令、执行代码，甚至浏览网页。创始人的注意力会沿着技术栈向上移动，转向更高阶的工作：生成想法，并指挥执行这些想法的系统，包括 AI 智能体、工具，以及任何已经存在的小团队。

不过，当 AI 成为核心基础设施时，最具革命性的结果，是让拥有领域专业知识的非技术创始人被彻底解放出来。当创始人群体不再局限于工程背景的人时，就会出现由生活经验截然不同的人创办的公司，去解决那些传统技术创始人流水线从未优先处理，甚至可能根本没有注意到的真实问题。

精益创业公司的 AI 工具能力

传统创业模型假设，你需要雇工程师来构建产品，雇销售来卖产品，雇运营人员来运转业务。员工数量被视为组织动能和产品成熟度的象征。

2026 年的早期创业公司已经完全不同。它们在设计上就极其精益，常常只有创始人一个人，或者是由少数几个人组成的团队。通过把 AI 作为基础设施来支撑技术开发和组织建设，它们可以在扩张团队之前，就达到产品验证、早期收入，甚至盈利。

AI 尤其能在三个方面帮助创业公司像更大的组织一样运转：研究、智能体式编码，以及关键业务运营工作流的自动化。

对话式智能与研究

可以这样理解：每个领域都有一位随时待命的专家。

想想创始人在第一年需要知道多少自己几乎不可能一开始就知道的事情：我该如何设置工资发放？如何规划产品开发冲刺？如何起草一份紧凑有力的投资人备忘录？

过去，这类早期创业问题的答案通常都是同一个：找一个懂行的人。对于自筹资金或种子前阶段的创始人来说，这可能意味着把本应用来构建的时间花在知识搜集上，或者烧掉一部分早期资本去请顾问。现在，他们拥有了一个横跨几乎所有可想象领域的随叫随到专家：AI。

- 深度研究：竞争分析、市场规模估算、财务建模。
- 文档起草：融资演示文稿、案例研究、投资人备忘录、产品需求文档。
- 战略思考伙伴：反方分析、事前验尸、情景规划、路线图优化。

智能体式编码

可以这样理解：一位永远在线、永远不会被卡住的工程师。

过去，构建软件需要一个技术联合创始人、一家外包开发公司，或者一条足够长的资金跑道，让你在写出第一行生产代码之前先雇一支工程团队。

现在，智能体式编码工具让每一个有志于创业的人，都可以用自然语言描述自己想构建什么，并指挥 AI 生成、测试、调试和重构一个生产级代码库，其速度和规模接近完整工程团队。“我有一个想法”到“我有一个产品”之间的时间线被压缩了。创始人的角色也因此转向决定“构建什么”和“为什么构建”，而实际构造可供真实用户使用的真实基础设施，则由 AI 完成。

工作流自动化

可以这样理解：一支按需调用、自动运转的运营团队。

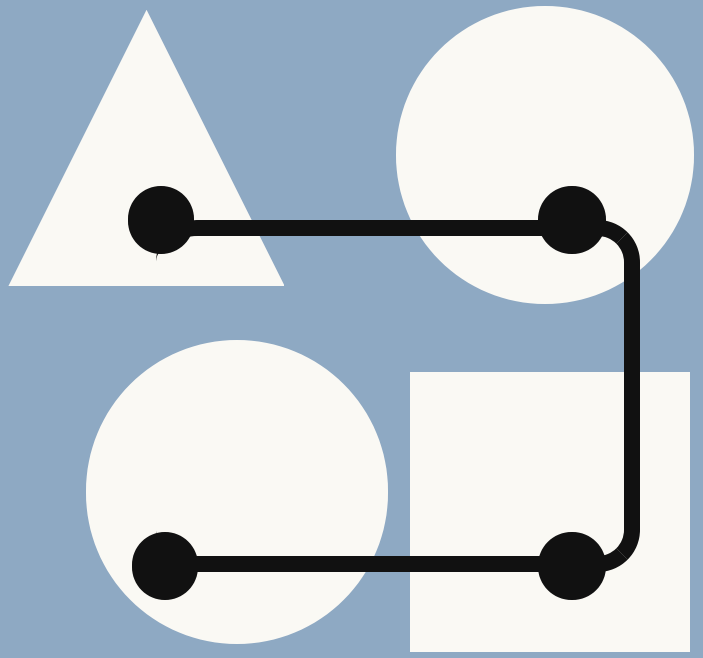
即使一个创始人能像顾问一样做研究、像工程团队一样构建产品，仍然还有一整类既不是战略规划也不是产品开发的工作必须完成。排期、更新 CRM、拉取周报、维护文档、发布内容、跟踪合规要求、管理公司所依赖的工具和系统之间的连接组织，这些也都必须发生。在精益创业公司里，这些负担主要落在创始人身上，并且会显著消耗本该用于高阶决策的时间和注意力。

AI 工具驱动的工作流自动化可以卸下这笔成本。重复性运营任务可以被配置为自动发生：当交易推进时 CRM 自动更新，周报自动汇总，产品文档与产品变更同步更新。关键在于，Claude Cowork 可以连接一家创业公司所运行的互相关联系统，包括项目管理工具、沟通栈、数据源，而不需要有人专门构建和维护这些集成。在 Day Zero 创业公司里，这个“有人”几乎总是创始人本人。

时机与编排就是一切

能够有效利用 AI 的研究、自动化和智能体式编码能力的创始人，可以构建一家杠杆远高于员工人数的公司。他们也能把大部分时间和带宽投入到真正重要的工作上。

这些工作不会自动完成。编排这些 AI 工具的创始人，必须知道如何使用它们，以及在什么时候使用。接下来的内容会探索创始人沿着 AI 原生创业路径前进时会遇到的目标和挑战，并说明如何在旅程的每个阶段有效应用 AI 工具。



Chapter 3

第 3 章：构想阶段

Chapter 3

第 3 章：构想阶段

每一位创业公司创始人都从同一个地方开始：一个让自己无法停止思考的问题。这是想法与现实相遇的阶段。2026 年的创业成功，要求创始人具备一种纪律：在证据足以支撑之前，不要急着构建。

这一阶段的工作包括研究、客户发现、竞争分析，以及诚实评估反证。所有这些都应该发生在你要求 Claude Code 生成第一行生产代码之前。

构想阶段目标

在构想阶段，创始人的主要目标是以研究为导向的验证：在投入资源开始构建之前，收集扎实证据，证明一个真实问题确实存在，并且你提出的解决方案能有效处理它。

具体来说，构想阶段就是创始人大致按以下顺序回答一组问题：

- 这个问题是否真实、具体，并且发生频率足以围绕它构建产品？
- 到底是谁遇到了这个问题？这些人是否构成一个市场？
- 有没有其他人在解决它？如果有，他们如何解决，解决得怎么样？
- 一个解决方案到底需要做到什么，才能真正解决这个问题？我的想法是否做到了？

这些问题的答案最终会汇聚成一个终极问题：这件事值得构建吗？

这意味着，在开始行动之前先变得具体。“人们在费用报销上有困难”只是一个观察。“中型市场公司的财务经理每周花 4 小时以上核对报销提交，因为他们当前的工具无法与会计软件集成”才是一个可测试的假设。

构想阶段退出标准

构想阶段的退出条件，是找到问题-解决方案匹配。你已经主要通过真实的人际对话，建立了定性证据，证明你正在为真实的人解决真实的问题，然后再开始构建那个解决问题的东西。

当你能对以下三个问题全部回答“是”时，就可以离开构想阶段：

1. 问题是否真实而具体？要肯定回答这个问题，你必须能准确说出谁会遇到这个问题、他们多久遇到一次、它对他们的影响有多严重，以及他们目前如何处理。
2. 你的解决方案是否解决了真正的问题？不是你一开始假设的问题，而是验证过程揭示出来的问题。有时候两者相同，但并不总是如此。
3. 你是否有足够信号来证明值得构建？这个阶段你永远不会拥有确定性，等待确定性本身也是一种失败模式；但你需要足够的定性证据，让投入 MVP 成为一个经过推理的决定，而不是一次信仰跳跃。

构想阶段挑战

构想阶段是整个创业旅程中最重要的工作发生之处，因为最具后果的错误也发生在这里：现在错了，很快就会把一个刚发芽的创业项目带离轨道。

大多数构想阶段的挑战，都是因为行动速度超过了理解深度。因此，那些以思考和审慎推进的创始人，会经历稳定的进展。

把构建误认为验证

挑战：当技术障碍被移除时，充满激情的创始人可能会跳过创业旅程中最重要的工作：验证自己的想法是否真的是人们需要并会使用的解决方案。

即使在当前智能体式编码时代之前，也有 42% 的创业公司因为构建了没人想要的东西而失败。如今，Claude Code 这类智能体式编码方案大幅缩短了“我有一个想法”和“我有一个产品”之间的距离，这个失败率只会继续上升。

对于拥有一个足够令人兴奋好想法的创始人来说，从未有过比现在更好的时代。但能够快速、轻松地搭建出一个看起来像产品的原型，反而给 AI 原生创业公司带来了一种真正危险的生存风险。

直到不久前，构建仍然需要真实的开发时间和预算。即使只是做出一个基本原型，通常也需要几个月。现在，技术开发门槛基本消失，AI 让创始人太容易直接跳进构建，而没有验证它在真实世界中的效用。

达成问题-解决方案匹配，要求先验证假设，再开始构建。但许多第一次创业者，甚至一些有经验的创始人，都会错误地相信 AI 可以绕过这个要求，把流程变成：有想法 → 立刻构建原型 → 把原型存在本身当作验证。原型变成了相信自己假设一直正确的理由，但这个假设从未真正被测试过。

一个能运行的原型很容易被误认为“我正在解决真实问题”的具体证据，但它不是。原型更适合作为与潜在用户对话时的压力测试道具。对话本身才是真正的证据。

过早扩张

挑战：当构建变得轻松而即时，你可能会让执行规模远远跑在业务需求前面。

过早扩张意味着，在你真正验证某条产品路径值得投入之前，就承诺走上这条路径。

这一直是创业公司的杀手，但 AI 让创始人更容易在没有察觉的情况下掉进过早扩张陷阱。智能体式编码助手非常强大，以至于你很容易在没有有意识偏离路线的情况下，就让执行远远超过问题-解决方案匹配的验证。

它会围绕一个根本有缺陷的前提生成、测试、调试和重构代码库，并且热情程度和面对绝佳想法时完全一样。系统里的智慧来自你。这个阶段的首要指令，是让你的理解速度始终领先于构建速度，尤其是在构建如此快速、如此轻松的时候。

失去客观性

挑战：如果你要求 AI 工具寻找支持你既有信念的证据，它会找到。确认偏误现在配上了一台研究引擎。

确认偏误一直是创业中的职业风险：创始人天生对自己的想法充满热情。现在，AI 工具显著增强了确认偏误。要求 AI 验证你的创业想法，它会找到支持证据；要求它估算潜在市场，它会找到一个让你的 TAM 看起来足够可融资的数字。

AI 会遵循你的方向，这意味着，一个不提尖锐问题的创始人，现在可以比以往更快地为一个坏想法构建出复杂、看似研究充分的论证，并且自信地相信自己正在做尽职调查。解药仍然是同一个工具，只是方向相反：AI 可以像验证一个想法一样彻底地压力测试它。

当研究和结构化对抗性思考浮现出“你的想法需要修正”的证据时，这就是转向的信号。

Claude 如何帮助构想阶段的创始人

推动一个 AI 原生创业概念走过构想阶段，可能感觉永远也做不完。你是创始人，你就是想开始构建。但这个至关重要的开端阶段，本质上是研究和验证练习。这意味着，在全力写代码之前，应先使用能帮助你更严谨思考的工具。下面是在构想阶段尽快推进，同时完成必要尽职调查的 Claude 使用方式，涵盖 Chat、Claude Cowork 和 Claude Code 等产品界面。

Chat、Claude Cowork 或 Claude Code：选择正确的 Claude 使用界面

AI 让创业公司创始人更容易更快发布、自动化繁琐工作流，并以规模化方式运营。但你使用的界面很重要。下面说明如何根据任务选择 Chat、Claude Cowork 或 Claude Code。

Chat 适合在不离开当前应用的情况下完成快速交流。可以用它处理经营公司时不断出现的小任务：从一份厚重的投资人备忘录里提炼一句话结论，在董事会会议前检查一个说法是否站得住脚，或者理解团队中一条很长的 Slack 线程。

Claude Cowork 适合真正耗时的知识工作：从多个来源提取信息、理解信息，并产出一个完成品，比如文档、演示文稿或电子表格。可以想象这样一些任务：把一整个文件夹的客户访谈转录稿转化成下一次产品评审用的主题化发现文档；在融资前从十几个供应商网站构建竞争格局；或者设定一个周一早上的固定任务，从连接的工具中拉取指标，并把每周 KPI 简报放进共享文件夹。

Claude Code 是团队工程师使用的智能体式编码环境：可以直接访问代码库，支持 Plan Mode、Git 集成，以及本地、IDE 或沙盒云环境。精益团队会在这里跨不断增长的代码库发布功能，迁移 MVP 时期的遗留代码，并在不等待更多人手的情况下从原型推进到生产。

如果任务是	选择	原因
一个问题、一次改写、快速头脑风暴	Chat	快速、对话式、无需设置
基于你的文件和系统完成研究、分析或最终文档	Claude Cowork	文件夹访问、连接器、技能、定时运行
编写、测试或发布软件	Claude Code	代码库访问、diff、Git、开发环境

三者底层使用的是同一个 Claude；变化的是它周围的工作空间。

定义并压力测试问题假设

你的领域专业知识和前期研究，已经生成了一个假设。第一项工作是把它打磨到真正可测试。Claude 在这里尤其有用，因为它会迫使你变得具体：到底谁有这个问题？多久发生一次？严重程度如何？他们目前怎么处理？一个无法精确回答这些问题的问题陈述，还没准备好被验证。

- 练习：和 Claude 一起打磨你的问题陈述，直到它成为一个可测试假设。比如，“合同审查太慢”并没有实际可测试性。但“中型市场公司的内部法务团队每个合同审查周期花费 3 天以上，因为修订意见分散在邮件线程里，而不是集中在一个带版本控制的文档中”，就非常可测试。

下一步，是要求 Claude 反驳你的想法，并寻找能够推翻你假设的反证。这可以浮现负面市场信号、失败的竞争对手、客户行为模式，以及支持性综合分析可能悄悄降权的结构性障碍。

目标是，在进入客户发现之前，已经用最强的反方论证压力测试过你的假设。这样，信息性用户访谈才会真正开放，而不是变成寻找确认的过程。

提示：把 Claude 用作结构化的反方辩手，是 AI 创业生命周期每个阶段的核心用法。

市场研究与竞争格局绘制

评估竞争对手

创业公司里有一种特殊现象叫竞争对手忽视：创始人过于专注自己的愿景和执行，以至于系统性低估了同一领域里其他人正在做的事。幸运的是，AI 提供了解药：要求 Claude 提出最有说服力的论证，说明为什么这个解决方案空间里的某个竞争者会成功，而你不会。

Claude 可以分析为什么他们的方法实际上更好，为什么客户会选择他们，以及为什么你认为的潜在差异化可能没有想象中那么牢固。

- 练习：要求 Claude 按层级绘制竞争格局：直接竞争者、间接竞争者、潜在收购方，以及可能进入你所在空间的相邻玩家。然后要求它论证每一层为什么都会对你的成功构成真实威胁，而不是只讨论那些最容易被你驳倒的威胁版本。

市场研究

Claude Code 可以综合公开可得的客户反馈，找出反复出现的抱怨和未满足需求。额外好处是：这样做本质上是在免费对竞争对手的客户进行定性研究。

- 练习：指示 Claude Cowork 综合你关键来源里的竞争对手评价，并识别现有解决方案尚未解决的主要抱怨。如果你的假设处理了其中一个或多个抱怨，那就是问题-解决方案匹配的强信号。如果没有，也同样值得知道。

Claude Cowork 也可以从密集的行业报告、分析师文件和市场研究文档中提取相关信息和数据。接下来，这些经过清理和综合的输入，会成为 Claude 进行分析工作的理想上下文。

- 练习：基于公开数据构建 TAM/SAM/SOM 模型，并压力测试背后的假设。识别市场是在扩张、整合还是已经成熟；这个背景会影响你对时机和差异化的判断。绘制买方格局：谁掌握预算，谁影响决策，以及这两者是否是同一个人。

趋势分析

最后，使用 Claude 监听早期指标，判断你是否在正确时间进入市场。跟踪已经围绕你的问题展开对话的 subreddit 和 LinkedIn 群组，记录用户描述问题时使用的具体语言。要求 Claude 识别已经解决过类似问题的相似市场，并提取其中有效和无效的做法。浮现可能加速或威胁机会的监管、技术或人口结构趋势。

- 练习：要求 Claude 识别三个可能在未来两年显著影响你市场的外部趋势，可以是监管、技术或人口结构趋势，并评估每个趋势对你的具体假设是顺风还是逆风。

提示：本节中的市场研究和竞争地图不是一次性练习。你会在 MVP 和发布阶段继续产生发现并演化自己的思考，因此每当你的假设发生变化，都应重复这些练习。

规划并设计客户发现

你通过与潜在用户对话所学到的东西，其质量取决于两点：你问的问题质量，以及你是否把这些问题问给了正确的人。Claude 在客户发现中尤其有用，包括判断该和谁聊、该问什么，以及如何理解你听到的内容。

应该和谁聊

一个精确的目标画像，比一长串联系人名单有价值得多。这个画像应该包括最有可能深切体验该问题的具体职位、公司类型、团队结构和资历层级。然后，识别这些人实际上在哪里可以被触达：他们聚集的社区、活动、LinkedIn 群组和 Slack 工作区。最后，基于他们离问题有多近，建立一个优先联系谁的框架。

该问什么

定义目标对象之后，用 Claude 构建访谈框架本身：正确的问题、正确的顺序，并以能浮现人们真实行为，而不是他们以为自己未来会做什么的方式来组织。新手创始人的常见错误，是询问一个泛泛的、面向未来的开放问题，比如“你会用这样的东西吗？”；而不是具体追问相关的过去，比如“告诉我你上一次处理这个问题时发生了什么。”

Claude 还可以标记你草拟的问题哪里在引导受访者、哪里太宽泛，或哪里更可能产生噪声而不是信号。Claude 也能帮助你设计追问，用来探查回避回答，或深入挖掘对重要问题的模糊回答。

如果你的假设涉及多个用户画像，Claude 也可以为每个画像设计不同的问题集。财务经理和 CFO 与同一个问题的关系不同，单一访谈框架会抹平这种差异。

- 练习：先手写你的访谈问题，再请 Claude 审核。明确要求它标记任何具有引导性、面向未来、过于宽泛，或可能得到社会期许式回答而非诚实回答的问题。然后要求它为访谈中最可能出现回避的两三个时刻建议追问。

访谈后分析

每次对话之后，用 Claude 复盘：把你的笔记交给它，并要求它识别哪些内容确认了你的假设，哪些内容挑战了假设，哪些内容真正令人意外。收集一批访谈之后，通过 Claude Cowork 跑完整访谈笔记集，浮现反复出现的主题、矛盾，以及正反两个方向上最强的信号。然后把这个综合输出带回 Claude，要求它指出你对数据的解读在哪里可能是在匹配自己想听见的模式，而不是数据真正呈现的内容。

- 练习：每完成五次访谈，就指示 Claude Cowork 综合你的笔记，并产出两个清单：支持你假设的证据，以及挑战你假设的证据。如果第一份清单明显长于第二份，要求 Claude 判断这种不对称是来自数据本身，还是来自你希望看到的东西。

客户触达和排期

使用 Claude Cowork 自动化构建联系人列表、运行触达流程和安排用户访谈所需的运营工作。

Claude Cowork 可以使用你与 Claude 一起定义的目标画像，包括职位、公司类型和资历层级，研究并编制一个结构化的潜在访谈对象列表和已验证联系方式。然后，它可以大规模起草个性化触达邮件，并根据每个人的角色和上下文进行调整。

当回复到来时，它可以通过 MCP 连接 Gmail 和 Google Calendar，管理邮件线程、处理排期请求，并把访谈放进日历。这个工作流还会继续：Claude Cowork 会按定义好的节奏生成后续跟进草稿，比如第 7 天提醒尚未回复的人，并在每一步完成时更新你的追踪表，让你始终知道每个潜在访谈对象在管道中的状态。

- 练习：把已经验证过的访谈目标画像交给 Claude Cowork，要求它构建潜在对象名单、起草个性化触达序列，并建立一张追踪表，列包含触达状态、跟进节奏和访谈完成情况。然后让它负责协调，你则专注于为对话本身做准备。

设计最终解决方案概念

你已经完成验证工作：问题是真实的，你知道谁有这个问题，并且有一个由证据支撑的解决方案概念。使用 Claude 从各个角度发展并挑战你的解决方案概念：有什么缺口？有哪些替代方案？为了让这个解决方案在规模化时有效，必须满足哪些前提？

这是一个重要的现实检查点：这个设计是否真的处理了验证过程揭示的问题，而不是你一开始假设的问题？

- 练习：把你的解决方案概念呈现给 Claude，要求它识别这个设计最依赖的三个假设。然后追问：每个假设要成立，需要哪些条件为真？如果任何一个假设不成立，会有什么后果？

用 Claude Code 构建轻量原型

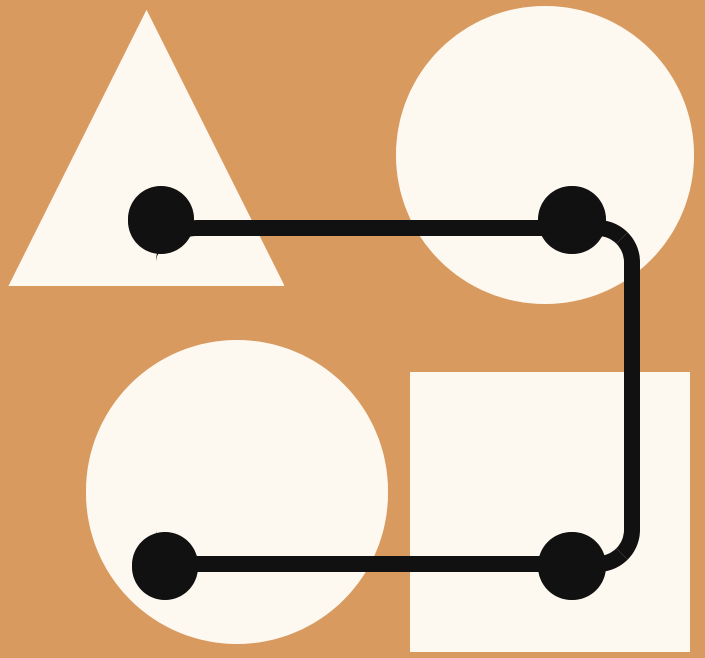
现在到了有趣的部分：有了已验证的假设和经过压力测试的解决方案概念，你终于准备好构建一些东西了。

这就是构想阶段中 Claude Code 登场的时刻。即使你之前一直在 tinkering，现在也到了生成正式轻量原型的时候：它只包含把你的想法放到真实人面前，并获得真实反应所需的最小表面积。

你还不是在构建真实世界产品；你是在构造一个想法的功能性样本，用于客户和投资人对话。真实用户对一个他们能真正触碰的东西作出的反应，会告诉你十几次问题-解决方案发现访谈都无法告诉你的东西。此前，你是在确认自己解决的问题是真实的；现在，你是在请潜在用户与拟议解决方案互动。

- 练习：定义你的解决方案所依赖的单个核心交互。指示 Claude Code 只构建这个。当你拥有它之后，把它放到来自已验证目标画像的五个人面前，请他们试用。这五次对话中学到的东西，会决定你是继续构建，还是回到绘图板。

走到构想阶段的终点，是 AI 创业竞赛中的一次巨大跃迁，因为你现在不是在押注直觉，而是在根据证据执行。接下来是 MVP 阶段，创始人的引导性问题会从“这值得构建吗？”变成“我们到底应该先构建什么？”，而 AI 的主要角色也会从研究伙伴转变为施工队。



Chapter 4

第 4 章：MVP 阶段

Chapter 4

第 4 章：MVP 阶段

许多创始人把 MVP 阶段视为一个施工阶段，但 MVP 阶段本质上仍然是证据收集练习。区别在于，现在你收集的是关于解决方案的证据，而不再是关于问题空间的证据；具体来说，是一个真实、可识别的人群是否认为它有价值，是否愿意使用、再次使用、付费，或者告诉别人。

MVP 阶段目标

作为 AI 原生创业公司的创始人，你的目标是把一个已验证的问题转化为真实用户会实际使用的可运行产品。这不是包含路线图上所有功能的完整版本，而是你的想法中最小、最聚焦的一次迭代：把真实解决方案放到真实用户面前，并生成产品-市场匹配的真实证据。

与此同时，你现在如何构建，会决定以后什么是可能的。这意味着 MVP 阶段还有第二个同等重要的目标：在快速推进的同时，不积累那种会复利式增长，并在真实用户大量到来时立刻缠上你的技术债。

最后，从第一天起就投资持久上下文，是让 AI 成为倍增器而不是熵源的关键。在 AI 原生创业公司里，代码库是你一轮又一轮与 AI 协作的对象，因此可读性是基础。那些跳过规格说明、架构决策和上下文文件（如 CLAUDE.md）的创始人，会撞上可预测的墙：每一次新会话都需要重新解释代码库，而 AI 生成的变更会逐渐偏离最初愿景。

MVP 阶段退出标准

MVP 阶段的退出条件，是关于产品-市场匹配的真实证据：证明某个具体、可识别的用户群体已经认为产品足够有价值，愿意回访（留存）、付费（收入），或告诉别人（推荐）。

MVP 阶段挑战

在 MVP 阶段，创始人的首要指令是速度和判断。这里的挑战集中在：你能否足够快地、用正确方式构建正确的东西，并且不走那些以后会让你付出代价的捷径。

智能体式技术债

挑战：AI 基本移除了过去控制“什么能进入生产环境”的所有自然瓶颈，因此速度已经得到保证。但如果创始人在 MVP 构建中唯一考虑的变量就是速度，就会有积累难以偿还技术债的风险。

在 MVP 阶段，某些技术债是合适的，前提是你理解它必须在规模化之前被管理。它会逐渐积累，可以随着时间或在专门的冲刺中清理。但 AI 技术债会复利增长。

如果规格和架构约束没有写在 AI 可读取的地方，每次会话都会从头重新推导基础决策，而这些决策会漂移。你最终会得到一个背后没有一致心智模型的代码库。问题不在于某个单独部分很糟糕，而在于这些部分从未被设计为相互配合。这是真正的问题，而且通常会在很晚的时候才暴露。

误把虚假的产品-市场匹配当成真实 PMF

挑战：AI 工具可以生成令人印象深刻的早期数字，但这些数字并不能保证市场真的需要你的产品。

早期动能是创始人可能经历的最强心理体验之一。经历数周或数月验证工作和谨慎、纪律化构建之后，发布产品感觉就像确认了你从一开始就是对的。

智能体式编码工具可以帮助你比以往更快来到这个时刻，但早期牵引力并不等于产品-市场匹配。发布时的热度可能来自短暂因素，比如创始人的朋友、投资人其他被投公司的潜在买家，或者 Hacker News 首页带来的流量尖峰。不幸的是，这些都不能可靠预测第 6 周或第 12 周，当初始助推消退之后会发生什么。

零摩擦的范围蔓延

挑战：当构建感觉毫不费力且几乎免费时，总会还有一个很酷的功能可以加，或还有一个边界情况可以处理。这种范围蔓延可能弊大于利。

范围蔓延一直是创业风险。现在的区别在于，过去抵抗它的强制函数是真实工程时间成本；而当增加一个功能从一个冲刺变成一个下午时，这个成本不再以同样方式存在。

这里的挑战是，每一次单独的新增都有道理。产品当然应该处理那个边界情况；用户当然会需要那个工作流。这些在当下都不像范围蔓延，因为用智能体式编码构建每一个都很轻松。但随着产品不断扩展到原始边界之外，你会冒着失去方向和动能的风险。

解药是在构建开始前写下范围定义：产品做什么，刻意不做什么，以及哪些来自真实用户的具体证据可以证明现在值得新增某个东西。这会把决策点从“我们要不要构建这个？”转变为“是否有足够多的关键用户告诉我们，没有它就无法从产品中获得价值？”

因经验不足而不安全

挑战：创始人如果使用 AI 工具匆忙把应用推向市场，却没有先理解基本安全原则，最终会让用户暴露在本可避免的风险之下。

一个残酷事实是，智能体式编码工具会生成能工作的代码，而不是天然安全的代码。功能性代码很容易判断，因为功能要么能用，要么不能用。安全漏洞在被利用之前是不可见的，这意味着没有自然反馈回路提醒第一次创业的创始人有问题。把一个实时 MVP 发布给真实用户，意味着真实数据、真实暴露面，以及一旦出错就会有真实后果。

轻视安全并不是 AI 原生项目才有的新问题。每个时代的自筹资金创业公司，都常常把安全考虑延后到构建后期，有时甚至等到接近生产发布时才处理。但在任何用户接触你的应用或解决方案之前进行安全审查，是把最小可行产品发布到世界上的最低负责门槛。

Claude 如何帮助 MVP 阶段的创始人

在构建前定义架构

在 Claude Code 写下第一行生产代码之前，使用 Claude 定义并记录会支配这一阶段所有构建工作的架构决策：要遵循的模式、要避免的依赖、正在做出的取舍以及为什么。这个输出会成为聚焦的架构上下文文档，并建立 Claude Code 运行时的护栏。

如果没有这个上下文，每次会话都会从零开始，Claude Code 被迫推断自己的结构性假设。让 Claude Code 在没有护栏的情况下构建，会产生一个功能可用但结构不连贯的代码库。迭代和扩展不连贯的代码库，最终是在浪费时间和 token。迟早会有一个点，代码不可避免地坍塌，迫使你从头重建。

- 练习：在打开 Claude Code 之前，先打开 Claude 并描述你正在构建什么：它解决的核心问题、服务的用户，以及你现实地预期未来 6 个月的规模。要求它帮助你定义应该支配 MVP 构建的架构原则、在你的约束下应该避免的依赖，以及这一阶段你有意识接受的取舍。

接下来，把这个输出保存为 CLAUDE.md Markdown 文件。这是你的架构上下文文档：构建的第一个产物，也是后续每个会话依赖的东西。CLAUDE.md 文件作为 Claude Code 的项目级说明，为 Agent SDK 在某个目录中运行时自动读取的项目提供特定上下文和指令。功能上，它们就是项目的持久“记忆”。

定义并执行 MVP 范围

无摩擦的范围蔓延，是 AI 时代 MVP 的典型失败模式之一。就像你定义并记录产品应用架构一样，你也需要在任何一个功能被构建之前定义 MVP 的范围。

Claude 可以帮助你创建范围文档，描述你的 MVP 产品做什么、刻意不做什么，以及功能修订标准：什么来自真实用户的具体证据，可以证明现在值得新增某个东西。

当新的功能想法浮现时，它们一定会浮现，你可以用 Claude 压力测试：这是真正来自用户的信号，还是披着产品思考外衣的创始人热情。

用 Claude Code 构建 MVP

一旦架构和范围已经定义，Claude Code 就成为主要的 MVP 构建工具。用它生成、测试、调试和迭代代码库，但要把每次会话视为执行你已经做出的产品决策，而不是趁机再塞入新的产品决策。

每次 Claude Code 会话开始时，先做两件事：重新查看范围文档，并提供 CLAUDE.md 架构上下文文档。每次会话结束时，用会话中浮现的任何决策更新它。目标是拥有一个你能解释其结构的代码库，而不仅仅是一个能运行的代码库。

- 练习：为你的 Claude Code 工作创建一个简单会话模板，包含架构上下文文档、本次会话的具体任务，以及需要遵守的约束或模式。每次会话结束时，向上下文文档添加一条简短日志，说明构建了什么、做出了什么决策，以及会话引入了什么假设。每次会话花 5 分钟记录，是抵御架构漂移复利成不可管理代码库的廉价保险。

在任何用户接触之前做安全审查

作为 AI 原生创业公司创始人，你有责任知道代码库里有什么，理解任何潜在暴露向量，并且不要把明显漏洞发布给信任你处理数据的真实用户。

Claude 可以对 AI 生成的代码进行有用的第一轮安全审查，并帮助识别常见漏洞。这是发布前应嵌入循环的好习惯。不过，它不能替代安全工具；在风险更高时，也不能替代人类审查者。把它当作替代品的创始人，才会出现在数据泄露故事里。

Claude Code Security 会更进一步：它扫描代码库中的安全漏洞，并为人类审查建议针对性补丁，浮现传统方法可能遗漏的问题。

提示：在这本电子书发布时，Claude Code Security 仍是有限 beta 版本，因此在把它纳入工作流之前，请检查当前可用性。

- 练习：在部署给任何真实用户之前，带着明确任务让 Claude 审查你的核心应用代码：审查认证和会话处理、API 响应中的数据暴露、输入验证与注入风险，以及包含已知漏洞的依赖。认真对待每个发现，并评估是否需要修复；任何涉及认证、密钥或数据处理的事项都应有人类审查。

在发布前构建度量框架

那些把早期牵引力误认为产品-市场匹配的创始人，通常也是发布之后才开始追踪数据的人，而且选择的指标往往用来评估什么在有效，而不是浮现什么无效。解药是在第一个用户出现之前建立度量框架。

使用 Claude 定义你的具体产品应该关注哪些指标、基准是什么，以及哪些数据模式构成真正的产品-市场匹配，哪些只是讨人喜欢的噪声。具体来说，在发布 MVP 之前，先设定留存基准、激活标准，以及第 7 天和第 30 天目标。

接下来，为你的具体产品定义“假阳性”是什么样子。例如，有注册但无激活，有收入但无留存，或有初始热情但没有重复使用。当数据到来时，要求 Claude 对你自己的牵引力提出对抗性论证：一个怀疑者会如何解读这些数字？

管理发现与用户反馈流程

一旦真实用户进入产品，运营层会迅速扩张。Claude Cowork 可以处理重要但繁琐的工作，比如构建和维护用户联系人列表、运行触达序列、安排反馈会、分流 bug 报告，以及跟踪迭代周期。构想阶段用于管理发现流程的 MCP 集成，在这里同样适用。

对细腻的用户反馈探索，应让人类保留在收集循环中。例如，用户说“这很棒，但我希望它也能……”，需要解释：这是核心需求还是锦上添花？它只属于这个客户，还是代表了一个细分群体？缺失功能是真正问题，还是 onboarding 上游出了问题？没有工具能替你回答这些问题。

- 练习：配置 Claude Cowork 运行你的 MVP 阶段反馈循环：为早期用户列表起草触达邮件，安排反馈会，设计结构化的 bug 报告和功能请求接收流程，并每周撰写一次来信综合。先由你自己审阅综合内容；之后，你可以要求 Claude 分析这些信息，捕捉你可能遗漏的重要点。

朝证据迭代，而不是朝完整性迭代

MVP 阶段结束于你拥有真正的产品-市场匹配证据，而不是产品感觉有多“完成”。宣布你已经实现产品-市场匹配，并准备从 MVP 阶段进入发布阶段，本质上是一项判断练习，结合了创始人直觉和收集到的证据。不过，也有一些有用的试金石：

- Sean Ellis 测试：询问活跃用户：“如果你不能再使用这个产品，你会有什么感觉？”如果超过 40% 的人回答“非常失望”，这是有意义的 PMF 指标。
- 努力测试：在产品-市场匹配之前，留存需要持续干预，包括频繁触达、激励、个人跟进，以及创始人投入大量精力保持用户参与。在产品-市场匹配之后，产品开始自己承担这项工作。当事情开始从“推”变成“拉”时，这种努力方向的转变，是某些真实变化已经发生的最清晰信号之一。

归根结底，没有任何单一数据点可以确认产品-市场匹配，因为它必须是在多个迭代周期中都能保持的模式，才能被明确称为 PMF。

当证据要求时转向

如果你投入了所有这些工作之后，仍然似乎无法达到产品-市场匹配，该怎么办？结果没有确认你最初的方向，并不是失败，而是系统正在工作：MVP 阶段的设计目的，就是在你过度投资错误答案之前浮现这些信息。

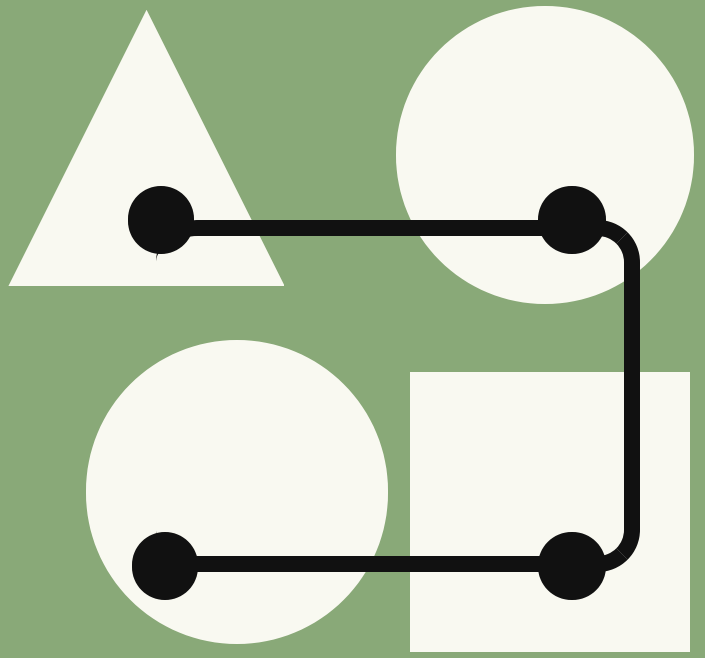
当数据不支持当前产品时，使用 Claude 梳理这些数据到底在告诉你什么：

- 探索替代客户细分。也许那些没有转化的用户从一开始就不是正确目标。正确受众往往已经在你的数据里，只是权重不足。
- 调整产品价值主张。也许你的受众是正确的，但 MVP 并没有真正打动用户。调整 onboarding、信息表达或核心功能重点，可能在不改变已构建内容的情况下修复问题。

也要保持开放：这种断裂可能深到需要更根本的改变。

- 练习：如果你已经完成三个或更多迭代周期，但朝产品-市场匹配基准没有明显进展，在决定下一步之前，使用 Claude 进行诊断。把你的留存数据、用户反馈和最初的问题假设交给它，并问三个问题：
- 这组数据里是否有某个细分群体的反应不同于其他人？
- 设计价值和体验价值之间的差距，是定位问题还是产品问题？
- 要让当前产品找到真正 PMF，必须满足哪些条件？鉴于你看到的情况，这种情景现实吗？

让这些答案决定你是调整、转向，还是回到构想阶段。



Chapter 5

第 5 章：发布阶段

Chapter 5

第 5 章：发布阶段

如果说 MVP 阶段是在证明你的产品值得存在，那么发布阶段就是在证明你的业务值得增长。

发布阶段目标

在发布阶段，创业公司创始人必须把早期牵引力转化为可重复、可持续的增长引擎。除了让产品准备好进入生产环境，你还必须加固它底层的基础设施，同时围绕产品构建一家真正的公司。

在构想和 MVP 阶段，创业公司天然是以创始人为中心的，因为你需要完整的情境感知和紧密反馈循环。现在，如果创始人仍然试图亲自抓住每一根线，就会变成发布阶段的瓶颈。目标不是把你自己从公司中移除，而是构建运营系统，释放你的注意力，让它用于只有创始人才能做出的决策。

发布阶段退出标准

发布阶段的退出条件包含三个要素：

1. 增长可重复，并由渠道驱动。你不只是留住用户，而是通过特定渠道可预测地获取用户，并且理解单位经济：CAC、LTV 和回本周期都是你知道并能辩护的数字。
2. 产品能够承载生产工作负载。基础设施得到加固，安全和合规到位，可靠性在真实生产条件下成立，而不仅仅是在你测试过的条件下成立。
3. 运营不再依赖创始人瓶颈。流程存在，自动化也已到位。你不再是亲自处理支持、分流、冲刺规划或报告的人。

发布阶段挑战

找到产品-市场匹配，是早期创业生命周期中最难的问题。现在，创始人的挑战变成了守住它。发布阶段是这样一个阶段：那些找到真实产品牵引力的公司，如果围绕和支持产品的组织跟不上，仍然可能解体。以下是需要警惕的失败模式。

技术债到期

挑战：为速度和验证而构建的 MVP 代码库，曾经足以证明产品有效；但生产流量、新功能和不断增长的复杂性，现在开始暴露那些捷径。

在 MVP 阶段，积累一些技术债是为了速度而做出的合理取舍。在发布阶段，这些债开始计息，而且越久不处理，修复成本越高。

解决方案包括系统性架构审计，以识别结构性弱点；针对性重构，以处理最严重的问题；以及有意义地扩展测试覆盖率，避免下一轮功能开发重新引入同样的问题。

创始人成为瓶颈

挑战：在 MVP 阶段，创始人处在每个循环里是一项资产。到了发布阶段，随着支持量增长、产品决策堆积、运营复杂性成倍增加，这种同样的本能会变成约束。

从亲自做事，转向设计能做事的系统，是创业生命周期中最难的转变之一。因为很少有一个清晰时刻宣告它已经发生，风险就在于完全错过这个时刻，继续停留在构建者模式，而组织在你周围停滞。

这正在发生的迹象包括：本应一小时完成的决策，现在要等你一周才轮得到；支持请求不断堆积，因为只有你知道答案；运营任务只有在你亲自想起时才会发生。

补救方式是对你亲自处理的一切进行全面审计，从最细小的任务到最高风险的决策，识别哪些可以系统化，哪些可以委派，哪些真的仍然值得创始人的时间和注意力。

安全与合规不再可以推迟

挑战：在 MVP 阶段保持简单的安全与合规措施是可以接受的；但现在，真实用户、真实数据，以及潜在企业合同都摆在桌面上，这会变成责任风险。

在 MVP 阶段，只有少量 beta 用户且生产环境中没有敏感数据时，安全漏洞还是理论风险。但一旦产品带着依赖它的真实用户进入生产环境，假设就会变成非常真实的暴露风险。此外，那些不适用于原型的合规要求，在你处理客户数据、处理付款或销售给受监管行业的那一刻，绝对会适用。

补救方式是在生产规模到来之前进行系统性安全和合规审查，而不是之后。把所有浮现的问题都当作下一波用户到来之前必须修复的事项，而不是建议。

在尚未准备好时扩张

挑战：新市场和融资机会看起来像增长机会。它们也可能是产品-市场匹配消亡的地方。

你已经建立的初始牵引力是真实的，但它也特定于你的早期受众。太早扩张到一个与你原始市场有意义差异的新市场，会引入新的用户行为、合规要求、支付基础设施和基线期待，而你的产品并不是围绕它们设计的。突然之间，变量过多，你失去了清晰解释自己数据的能力。你还会冒着忽视原始用户群、转而追逐一个未经证明的新受众的风险。

Claude 如何帮助发布阶段的创始人

在发布阶段，三种 Claude 形态都会被充分使用，并且相互支持：每个工具的输出都会成为另外两个工具的输入。结果会自然复合。一个同时使用这三个工具的创始人，得到的会超过各部分之和。

这就是超精益创业模型在结构上成为可能的原因。当 Claude Code 构建产品，Claude Cowork 围绕产品构建公司，而 Claude 帮助把产品和组织知识运营化时，一个小团队可以像大自己许多倍的公司一样运行。

在技术债复利之前修复它

你的 MVP 代码库能工作，但它也需要一次系统性的修复检查，以寻找任何可能成为结构性责任的技术债。

首先，使用 Claude Code 运行完整架构审计：识别代码库哪里脆弱，哪些捷径会变得昂贵难维护，以及哪里测试覆盖太薄，以至于下一轮功能工作会重新引入同样的问题。

把 Claude Code 的审计发现反馈给 Claude，用来分流并排序修复工作：哪些必须在下一次发布前修复，哪些可以等一个冲刺，哪些在当前阶段属于可接受的持续债务。

这也是记录你在 MVP 阶段做出的架构决策的时刻。这些决策之前可能都在你脑子里，因为没有时间写下来。现在把它们写进 CLAUDE.md，可以确保未来每一次 Claude Code 会话都从共享理解出发：系统是如何设计的，以及为什么这样设计。

- 练习：指示 Claude Code 审计你的 MVP 代码库，并产出一个按优先级排列的清单，列出结构性弱点、测试覆盖缺口和重构候选项。然后把这份清单交给 Claude，要求它把修复工作拆到多个冲刺中：哪些重大问题必须先处理，哪些可以与功能开发并行，哪些可以等待。

构建能替代创始人注意力的系统

要构建释放你注意力的运营系统，让你能处理只有创始人才能承担的职责，必须先知道你的注意力到底花在哪里。使用 Claude Cowork 对当前运营负载做结构化审计，记录每个重复任务、每个落到你桌上的决策，

以及每个只因为你亲自想起才会发生的工作流。然后让 Claude Cowork 把这份清单分类为：可以完全自动化的、需要人但不一定需要你的，以及确实需要创始人判断的。

审计完成后，使用 Claude Cowork 为自动化候选项设计工作流逻辑：每个工作流由什么触发，决策规则是什么，输出长什么样，完成后流向哪里。

把安全和合规变成产品工作流

使用 Claude Code 浮现代码层面常出现在 SOC 2、GDPR 或 HIPAA 审计中的问题，以及目标市场要求的标准。这会同时浮现漏洞和合规缺口。把这些发现交给 Claude，帮助你优先排序修复工作，并设计企业买家签约前会要求的控制、审计日志和访问管理。

提示：AI 扫描是一种辅助，但不能替代合格的合规审查。

接下来，把合规工作流嵌入开发周期，而不是把它当作一次性项目；合规文档需要持续维护和更新。对于正在接近企业合同或国际市场的创始人来说，这也是 Claude Code 安全扫描帮助你为独立安全评估做准备的时刻。

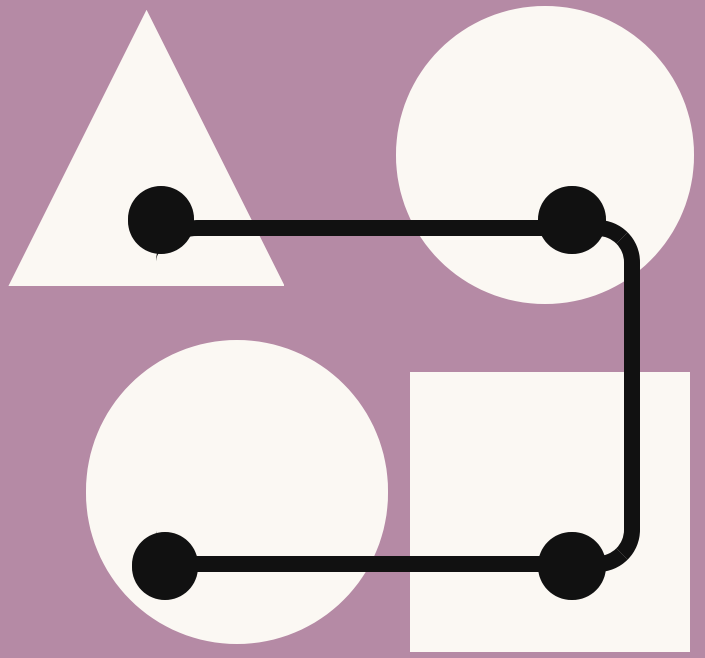
- 练习：用 Claude Code 运行一次代码级安全审查，面向目标市场要求的框架。把输出交给 Claude，并要求它产出两样东西：一个按优先级排序的安全修复顺序，以及一份为满足潜在企业买方合规审查而需要准备的文档和控制清单。

建立你一直跳过的产品管理流程

发布阶段需要一套轻量、可重复的流程，它们可以在不需要创始人干预触发或运行的情况下运转。使用 Claude 设计你的产品时间线和工作周期如何组织，一个规格说明在 Claude Code 接触某个功能之前需要包含什么，bug 报告如何分流和路由，以及每周指标报告覆盖什么、如何分发。

流程设计完成后，使用 Claude Cowork 构建并运行运营层：安排冲刺仪式，把进入的 bug 报告路由到正确位置，从连接的数据源汇总每周指标，并维护让用户信号持续流入产品决策的反馈循环。

- 练习：要求 Claude 设计一个轻量产品管理操作系统：明确的冲刺节奏、最低限度的规格模板、bug 分流决策树，以及从实际数据源拉取数据的每周指标简报。然后设置 Claude Cowork 执行并运行系统中的重复运营元素，如排期、路由和报告汇总，让它们按时发生，而不需要你介入。



Chapter 6

第 6 章：规模化阶段

Chapter 6

第 6 章：规模化阶段

在规模化阶段，创始人的角色会从构建者重新聚焦为面向公众的高管。产品仍然是核心，但你的个人日常工作会越来越围绕公司本身展开。你的注意力必须扩展到新的规模化阶段活动，如分析师简报和 IPO 路演，同时努力保持精益、以 AI 为中心的结构性优势。

规模化阶段目标

扩展技术基础设施的工作会继续进行，并且现在还会加入扩展组织本身、把它成熟为一家业务公司的工作。

到了规模化阶段，你要面对的是从数千用户到数百万用户、从一个市场到多个市场的跃迁。在之前每个阶段，增长都可以通过贴近用户、根据紧密反馈循环中的数据加上健康的创始人直觉来摸索。现在，目标变成了构建由成熟组织运营支撑的系统性增长。

对于 AI 原生创业公司，你的目标应该是通过累积深度构建可防御护城河。这个深度来自你嵌入产品的专业知识、产品与用户依赖的其他工具和平台之间的集成深度，以及专有系统数据和工作流。那些一直朝一个方向、在一致基础设施上持续构建的创始人，现在拥有了真正难以复制的东西。

在这个阶段，公开市场投资人、分析师、监管者、企业采购团队和收购方都会施加更大压力，也会带着更强怀疑，因为风险更高了。你的产品和组织必须经得起外部审视：不仅是你所构建能力本身，还包括围绕它的治理、合规姿态、财务控制和战略叙事。

规模化阶段退出标准

规模化阶段的退出条件不再是单一里程碑，而是一个阈值事件：即使创始人越来越不直接运营日常事务，公司仍然可持续。你已经证明了系统性增长；构建了能满足最严苛外部审查者的组织治理和合规基础设施；并且对这个问题有扎实回答：“如果一个资金充足的在位者今天复制了你的产品，你的用户会留下来吗？”

实践中，这个阈值通常会以三种形式之一出现：在不再需要外部资本的规模上实现可持续盈利，达到 IPO 准备状态，或被收购。三者都要求你的增长是系统性且可审计的，你的产品护城河经得起审查，你的组织在运营上成熟且可持续。

当这件事成真，就值得庆祝了：你的创业公司已经从一个赌注变成了一门生意。

规模化阶段挑战

委派运营层

挑战：规模化阶段的运营系统必须可靠、可持续地运转，而不需要人盯着。对于一个从第一天起就亲力亲为的创始人来说，这种转变既是结构挑战，也可能是心理挑战。

你在发布阶段的工作是创建系统；到了规模化阶段，工作变成两件事：让这些系统成熟到完全值得信任，然后真正信任它们。

这比听起来更难。即使你是一个擅长委派的创始人，也并不总是显而易见哪些应该交出去，哪些应该留在自己盘子里。交得太多太快，尤其是交给 AI 自动化系统，关键决策可能会在缺少只有创始人知道的关键上

下文时被做出。但抓得太久，你又会变成瓶颈。

这里的根本挑战，是识别那些只存在于创始人脑子里或未文档化工作流中的机构知识，然后把它编码进有文档、可审计、可转交的系统中。

扩展技术运营

挑战：客户不再只评估你的产品；他们还想知道你的组织是否能成为一个可靠的基础设施伙伴。

前三个创业阶段的技术挑战集中在代码库：构建正确解决方案，不积累技术债，然后为真实用户加固安全和合规。进入规模化阶段后，挑战变成代码库周围的一切：创建支持基础设施、文档和可靠性保证，以传达成熟度。

签署多年合同的大型客户和机构买家，在签约前就会需要这些，也会在签约后继续要求你兑现。帮助你走到这一步的同一套 AI 基础设施，也能帮助你构建具备明确响应时间和文档的专门支持功能，让新客户的工程团队真正能使用。

扩展组织职能

挑战：一家规模化阶段的公司通常需要招聘、工资、会计和法务运营等组织基础设施，不管实际由多少人运转。

在发布阶段，系统化运营意味着自动化那些消耗创始人注意力的工作流。规模化阶段的创业公司现在需要发展更广泛、在某些方面更关键的一系列运营职能，例如财务报告、合规监控、合同管理和客户支持等。

构建 GTM 职能

挑战：有机增长存在天花板，而大多数规模化阶段创始人在真正构建过上市职能之前就会撞到它。

构想、MVP 和发布阶段的增长，经常来自创始人主导的销售：可能是一条时机恰好的 Product Hunt 帖子，也可能是与早期客户的个人关系。这种有机增长只能在一定程度上有效，而大多数创业公司会在规模化阶段触及上限。迹象包括用户曲线趋平、客户获取成本上升，以及只有当创始人亲自参与时管道才会推进。

规模化阶段的增长要求构建一个专门增长引擎，让产品触达新的、更广泛的受众。但大多数创业公司创始人，可能以前从未运行过营销、销售、分析师关系等项目。真正的 GTM 动作不仅需要建立新系统和流程，还需要创建品牌声音和叙事，说明你想如何谈论自己的产品。因为在创业生命周期的这个阶段，你不仅需要触达个体新用户，也需要触达投资人和企业买家等完整目标受众。

幸运的是，GTM 职能不必庞大才有效。构建产品的同一套 AI 基础设施，也可以自举产品进入市场的过程。

Claude 如何帮助规模化阶段的创始人

早期创业阶段把 Claude 用作产品本身的基础设施：它是验证想法的研究伙伴，是设计并构建原型的工程团队，也是让单人创业公司成为可能的 AI 运营层。到达规模化阶段的 AI 原生创业公司创始人，现在可以继续用 Claude、Claude Code 和 Claude Cowork，以构建时相同的方式扩展。

把日常任务交给 Claude Cowork

以清醒视角开始规模化阶段：此时你最需要把时间和注意力投在哪里？这对第一次创业、从未构建过公司的创始人来说可能很难。Claude 可以帮助你列出这个阶段只有你应该做的事情，例如产品叙事决策、董事会关系、企业级交易，以及创始人之间的对话。不在这张清单上的一切，都是委派或 Claude Cowork 自动化候选项。

- 练习：使用 Claude 产出当前运营层的瓶颈地图：每个目前经过你的工作流、决策和审批。然后要求 Claude 推演，如果你一周不可用，每一项会发生什么。会停滞的工作流，就是你仍然足以亲自拖慢进展的地方。

这些与 Claude 帮你梳理出的创始人优先事项和职责清单如何对应？

接下来，是时候压力测试你已经构建的系统是否真的准备好随着业务增长而扩展。

- 练习：使用 Claude 绘制当前工作流地图，然后问它，如果你一周不可用，每个工作流会发生什么。会停滞的工作流，就是交接标准、升级路径或异常处理仍需收紧的地方。Claude 可以帮助分析失败

点，并建议适当修复，让你按需更新或替换 Claude Cowork 自动化。

把技术运营扩展为企业级基础设施

随着你扩展，买家需要确认你的产品和组织可以被信任为长期基础设施。代码库内部的技术工作仍会照常继续，但现在也需要处理代码库周围的技术工作。

第一步，是把机构知识转化为可扩展系统。使用 Claude 起草和维护企业采购期望看到的书面基础设施，包括产品文档、支持手册和 SLA。

与此同时，指示 Claude Code 根据企业合同要求的特定可靠性和安全标准，对代码库进行审计和加固，并构建 Discord 社区支持从来不需要提供的技术支持基础设施：日志、监控、事件响应工具，以及让 SLA 真正可执行的可观测性层。

然后，Claude Cowork 会运行企业支持本身的运营层：工单路由、升级工作流、由产品变更触发的文档更新、续约跟踪，以及企业客户成功所依赖的报告节奏。三者结合，让一个小团队拥有大得多的组织才有的支持姿态，而签下多年企业合同正要求你证明这一点。

- 练习：挑选最苛刻的三个潜在客户，或识别三个你最想签下的理想客户。要求 Claude 产出差距分析：这些客户的企业采购团队在签署多年合同前，预期看到哪些文档、SLA 和支持基础设施？你当前还有哪些不足？用输出结果安排 Claude Code 和 Claude Cowork 的技术与文档工作顺序。

构建真正的 GTM 职能

创始人的拼劲把你带到了这里，但扩展创业公司需要创建并实施真正的上市策略。AI 可以帮助你构建并运行完整 GTM 引擎。

Claude 可以协助从零构建基础 GTM 资源：市场细分、信息架构、分析师关系策略、销售手册，以及当你开始与公开市场投资人、企业买家和华尔街分析师交流时重要的投资人指标叙事。每类受众都有自己的词汇，并根据自己的标准评估你；Claude 的工作是把产品价值主张翻译成适合每个受众细分的产品营销方法。

现在，Claude Cowork 可以成为你的战术执行层：内容管道、外呼序列、分析师简报安排、新闻室和 PR 节奏、CRM 卫生、管道报告，以及把 GTM 策略转化为真实商业动作的许多重复周期。

当 GTM 动作需要产品营销基础设施时，比如交互式演示环境、集成文档、沙盒租户、API 参考、技术单页，Claude Code 可以为你构建。买家希望从技术角度评估你的产品，而在规模化阶段，一段 Loom 视频和一份销售 Deck 已经不够。这也是让 GTM 动作异步运行的基础设施：一个构建良好的演示环境，可以在你参加董事会会议时继续帮你成交。

把领域专业知识和机构知识转化为 AI 上下文

许多超精益创业公司创始人，正在为自己在某个特定行业中亲身经历或一手观察到的现实问题，构建高度具体的应用或工具。智能体 AI 现在让从未写过一行代码的创始人，也能用自己的领域专业知识，构建解决复杂问题的产品。Claude、Claude Code 和 Claude Cowork 会共同把创始人知识转化为不断复合的产品具体性。

使用 Claude 捕捉、组织并精炼创始人知识，会把领域专业知识放到产品能够触达的地方。通过长期对话、项目和记忆，创始人可以把自己知道的一切分享进去：行业术语、监管陷阱、边界情况、挫折、那些看似显而易见的答案为什么不能解决问题的原因，并把它们转化为结构化、可搜索的上下文。Skills 随后可以把重复工作流编码成可复用流程，例如“我如何审计商业租约”“我如何分流患者入院表”，让 Claude 每次都以同样方式运行。几个月后，这会成为通用 AI 无法匹敌的专有知识底座。

用 Claude 外化你的领域知识，对把行业特定边界情况编码进产品非常有价值。例如，一个通用 AI 医疗计费工具可能会在 340B 药品项目申报上出错，但你的工具有对应的具体逻辑。Claude Code 帮你把同领域其他专业人士经历的常见挫折，转化为验证逻辑、提示词改进，或与你竞争对手甚至没听过的小众行业系统的 MCP 集成。因此，你的应用或工具在深度和广度上都会持续复合，形成竞争对手无法复制的东西。

- 练习：识别一个通用竞争者在你的垂直领域中肯定会弄错的边界情况。和 Claude Code 一起基于你真实见过的场景，为它构建一个专门测试用例，而不是单元测试。每当类似边界情况浮现，就把它加进去。你的测试套件会变成护城河地图。

把累积用户数据复合成可防御优势

当用户与你的产品互动时，他们会产生行为信号，例如他们接受哪些输出、拒绝哪些输出，这些信号会影响产品路线图。随着时间推移，你会了解特定用户群体的具体模式、偏好和边界情况。这就是复合价值的含义：每次改进让产品更有用，推动更多使用，产生更多反馈，再推动更多改进。

这些数据是时间锁定的、特定上下文中的，而且复制者不可能重建：你无法购买成千上万用户在你的产品中不断精炼工作流所形成的行为指纹。

Claude 可以帮助审计你已经收集的任何用户互动数据，识别其中信号最高的行为模式，并设计反馈循环，把持续使用转化为系统性模型改进。

- 练习：把你的产品互动数据摘要交给 Claude：你一直在收集什么、收集了多久，以及你知道用户如何随时间与产品互动。要求它识别这些数据中三个信号最高的行为模式，并为每个模式设计一个反馈循环，将其转化为系统性模型改进。然后要求它帮助你起草一页护城河叙事用于产品营销：讲清你的数据飞轮如何运转、已经运转多久，以及为什么一个资金充足、今天才开始的竞争者无法在两年内复制它。

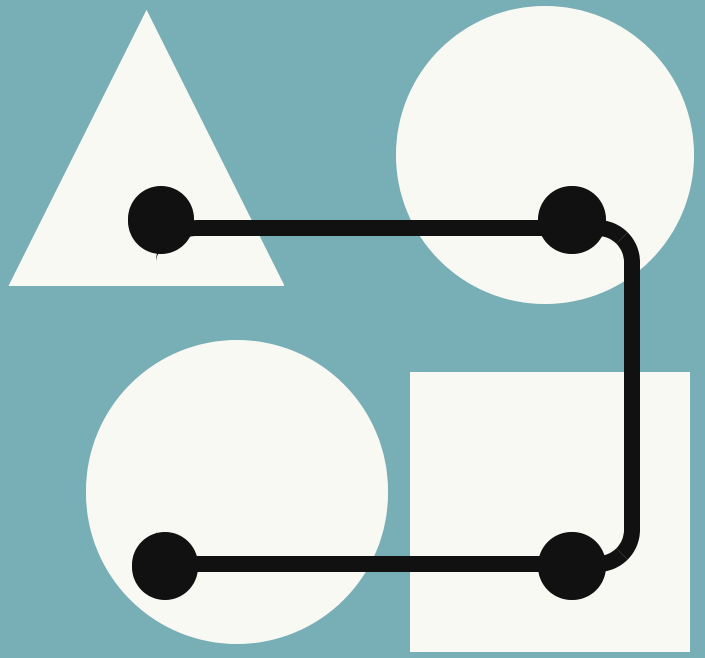
创建工作流锁定

复合数据网络效应让产品更难被复制，但用户工作流锁定让产品更难被离开。用户在日常运营中使用你的产品越久，它就越深地嵌入他们实际工作的方式。他们在它之上构建了自动化，培训了人员使用它，并把它连接到自己的数据源和其他工具。他们开发的提示词、精炼的工作流、标准化的输出，全部围绕你的产品能做什么以及如何做而形成。到了这个点，切换就不再是产品决策，而是完整的运营项目。

创建工作流锁定的第一步，是要求 Claude 按集成深度绘制当前客户群。对每个客户细分，识别他们在你的产品之上构建了哪些工作流，以及依赖哪些集成。这会显示产品在哪里开始粘住客户，以及哪里还需要更深入。

你提供的集成越多，客户就越有空间构建依赖你产品的工作流。Claude Code 可以帮助你快速搭建与目标用户依赖的数据管道、项目管理工具和其他系统的原生集成。Claude Code 还可以构建 API、webhook 和 SDK，让客户不只是使用你的产品，而是在它之上构建。这是最深层的锁定形式。

- 练习：要求 Claude 帮你为前 10 大客户构建工作流集成审计。对每个客户，记录他们构建的自动化、依赖的集成、通过你的产品运行的团队工作流，以及你对其切换成本的估计。然后要求 Claude 识别这组客户中的模式：对你的具体产品而言，哪些集成类型创造最深的锁定？你可以构建或启用什么，让目前还停留在表层的客户获得更深集成？



Chapter 7

第 7 章：同一份工作，新的规则

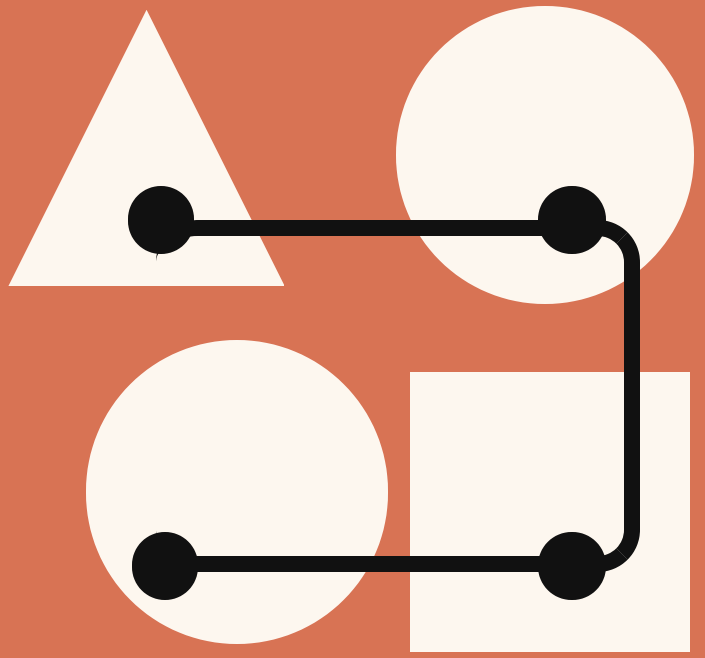
Chapter 7

第 7 章：同一份工作，新的规则

在 AI 时代，创始人的工作并没有改变：找到一个真实问题，构建能解决它的东西，并把它扩展成一家重要的公司。改变的是到达那里的路径。横跨构想、MVP、发布和规模化四个阶段，AI 把几个季度压缩成几周。

过去需要几个月的验证周期，现在可能下午就能完成。一个可工作的原型不再要求你有一位掌握正确技术栈的联合创始人；它要求的是一个清晰问题，以及和编码智能体进行几次聚焦会话。发布准备不再是发布前的混乱冲刺，而是一个连续工作流。到了规模化阶段，过去迫使早期员工进入救火角色的运营重量，越来越可以交给 AI，从而释放团队注意力，让他们投入那些会成为护城河的判断。

瓶颈不再是你能构建什么，而是你选择构建什么。



Resources

资源

Resources

资源

使用 Claude 构建

- Building AI Agents for Startups：分享创业公司如何使用智能体，在规模化时减少对创始人的依赖。
- Claude Code 文档：把构建者从初始安装带到高级智能体工作流。提示：从“How Claude Code works”概览开始。
- Claude Code 最佳实践：覆盖 Anthropic 内部和工程团队中有效的模式，包括上下文管理、权限、规划和验证工作流。
- 使用 CLAUDE.md 文件：介绍如何为特定代码库配置 Claude Code。对于正在设置开发环境的 MVP 阶段创始人来说，这是必读内容。
- Claude Code 高阶用户技巧：强调 Claude Code 团队自己的工作流模式，包括并行会话和验证循环。
- Get started with Claude Cowork：分享团队如何设置 Claude Cowork，并开始实施 skills、plugins 和其他能放大其在创业公司中影响力的功能。
- Tutorials：claude.com/resources/tutorials 提供可搜索的动手教程列表，面向具体任务。

创始人故事

- How three YC startups built their companies with Claude Code：考察 HumanLayer（F24）、Ambral（W25）和 Vulcan Technologies（S25）如何使用 Claude 快速把原型推向市场，并通过智能体式编码工作流扩展 AI 驱动的平台。
- GC AI 的创始人使用领域专业知识，构建了一个由 Claude 驱动、响应式的法律平台，服务内部法务团队真实工作的方式：公司特定 playbook、跨职能利益相关者，以及可变风险容忍阈值。
- Carta Healthcare 使用 Claude 驱动其临床抽象平台，每年处理 22,000 例外科病例，并将数据抽象时间减少 66%。
- Anything 由 Claude 和 Agent SDK 驱动，已经帮助 150 万用户在不写代码的情况下把想法变成可运行软件产品，包括一位非技术创始人，他构建并已经在销售一套完整招聘平台。Anything 的 AI 智能体负责完整构建，让独立创业者可以加倍投入自己的领域专业知识。
- Cogent 是一家应用 AI 实验室，构建用于自动化关键企业安全任务的智能体。这家公司使用 Claude 作为智能体的推理层，自动化完整漏洞生命周期中的调查、优先级排序和修复。
- Airtree 使用 Claude Cowork 作为其运营基础设施中心，把过去分散在十几个不同工具和团队中的数据统一起来。现在，当某个人用 skills 构建一个工作流自动化时，组织中的每个人都可以用它完成那些待办清单上一直没完成的事情。
- Duvo 构建 AI 智能体，在 ERP、供应商门户、电子表格、邮件甚至电话之间运行采购、供应链和品类管理流程。Duvo 完全建立在 Claude 之上，使用 Agent SDK 在工作流之间进行编排。
- Zingage 是一个为居家护理机构提供 24/7 自动化运营的 AI 智能体平台。它使用 Claude 的结构化工具调用，在 EMR 和多个沟通渠道之间编排；并使用 Claude 的上下文推理，构建能够给出细腻、面向患者个体结果的智能体，而不是只匹配最常见回答。

- Kindora 是一个由非营利组织高管使用 Claude Sonnet 构建的 AI 驱动平台，用来解决一个迫切需求：智能匹配慈善机构与资助方。在把数千个匹配筛选到少数值得追求的机会之后，Kindora 的 MCP 连接器让非营利组织可以直接在 Claude 中访问其潜在资助方搜索工具。
- Wordsmith 由一位律师转型 CTO 创办，为内部法务团队提供可靠的 AI 驱动法律技术。Claude 是 Wordsmith 合同审查、协议起草和文档审查能力的推理引擎，该创业公司的工程团队也使用 Claude Code 构建和演进平台本身。

创业支持与机会

- Anthropic Startups Program：面向与 Anthropic 的 VC 伙伴合作的创业公司，该项目提供免费 API credits、公开可用最高等级 rate limits，以及参加专属创始人活动和工作坊的邀请。
- Claude community：面向构建者的论坛和社区空间。
- Live learning resources：会议、网络研讨会、直播和录播资源。

译注

翻译：Diors By Codex.

英文版出处：https://claude.com/blog/the-founders-playbook