创始人实战手册：
打造 AI 原生创业公司

国内理解中文版 / 本土化意译版

说明：本文件根据英文 PDF《The Founder’s Playbook: Building an AI-Native Startup》整理为中文。为了便于国内创业者、AI 产品负责人、个人创业者和教育/内容创业者理解，本文采用“准确传达原意 + 国内语境转写”的方式，不做生硬逐字机翻。原文围绕 Claude、Claude Code、Claude Cowork 展开；在国内阅读时，可把它们分别理解为：通用 AI 对话助手、AI 编程/代码代理、能连接文件/邮箱/日历/表格/项目管理工具的 AI 工作台。

版本：2026 国内阅读版。适用对象：AI 原生创业者、独立开发者、产品经理、非技术创始人、小团队负责人。

目录

第 1 章 2026 年，创业生命周期被 AI 重写

AI 正在重塑创业公司的构建方式。过去，没有写过代码的人很难真正做出可上线的软件；现在，一个非技术创始人也可以借助 AI 把想法变成可运行的产品。以前人们说“10 人独角兽”像是一个励志故事，现在它越来越像一种可以主动设计的组织方案。

到 2026 年，AI 已经可以写生产级代码、做市场研究、整理竞争格局、起草融资材料、自动化日常运营流程。它真正改变的，不只是效率，而是创业门槛：谁能开始做产品、谁能启动一家公司，正在被重新定义。

过去的创业路径通常是：验证 - 融资 - 招人 - 开发 - 再融资 - 增长 - 再招人，如此循环。现在，AI 正在打破一个旧前提：每进入一个新阶段，都必须扩大团队、补齐新能力、再融一轮钱。

这本手册把创业过程拆成四个核心阶段：想法阶段、MVP 阶段、发布阶段、规模化阶段。我们会看清楚：当 AI 成为技术建设和组织建设的基础设施后，每个阶段的目标、风险、工具和节奏会发生什么变化。

> **国内语境：**这份中文版本的核心理解是：AI 原生创业不是“用 AI 做一个功能”，而是把 AI 当成公司早期的研发、研究、运营和知识管理基础设施。创始人的工作会从“亲手做每件事”，转向“定义方向、组织系统、校准判断”。

第 2 章 创始人的定义正在变化

过去，创始人常常被他们“会做什么”定义：技术创始人写代码，非技术创始人负责业务、销售、融资、运营。但 2026 年的模型、系统和 AI Agent，正在拆掉“会做产品的人”和“只有想法的人”之间的高墙。

AI 原生创业最根本的变化是：没有工程背景的人，也可以做出能表达自己想法的生产级软件；有技术能力但不熟悉商业的人，也可以借助 AI 生成 go-to-market 方案、财务模型和高质量融资材料。

历史上，创始人大量时间花在执行上：写代码、管人、处理运营琐事。在 AI 原生公司里，创始人更像“系统指挥官”：调度各种专门的 AI 助手，让它们读文件、运行命令、写代码、查资料、处理流程。创始人的注意力会被推到更高层：提出问题、判断方向、设计系统、协调人和 AI 的工作。

更重要的是，AI 会释放一批过去没有进入科技创业主流的人：有行业经验、有一线痛点、有真实生活经验的人。他们未必会写代码，但他们知道问题在哪里。这会带来更多来自真实场景的创业项目。

AI 给小团队的三类核心能力

传统创业模型默认：你需要工程师来开发，需要销售来卖产品，需要运营来跑业务。团队人数往往被当成公司进展的标志。

但 2026 年的早期 AI 原生公司会非常精简，可能只有创始人一个人，或者创始人加两三个人。只要把 AI 放在技术和组织的中心，小团队也可以在扩张团队前完成产品验证、早期收入，甚至盈利。尤其有三类能力会让小团队像大公司一样运作。

1. 对话智能与研究能力：随叫随到的领域专家

创业第一年，创始人会遇到大量自己不懂的问题：怎么做用户访谈？怎么设计产品 sprint？怎么写投资人 memo？怎么理解竞品？过去答案通常是“找懂的人”。这会消耗大量时间，或者消耗早期现金。现在，AI 可以承担一个随叫随到的多领域顾问角色。

- 深度研究：竞品分析、市场规模估算、财务模型、行业报告梳理。
- 文档起草：融资材料、案例研究、投资人 memo、PRD、产品说明。
- 战略思考伙伴：反方论证、预演失败、情景推演、路线图优化。

2. Agentic Coding：永远在线、不会被堵住的工程师

过去，做软件通常需要技术合伙人、外包团队，或者足够资金招聘工程师。现在，AI 编程工具让创始人可以用自然语言描述目标，然后让 AI 生成、测试、调试、重构代码。

“我有一个想法”到“我有一个可用产品”的距离被压缩了。创始人的重心也从“怎么写代码”转向“到底应该做什么，为什么做”。

3. 工作流自动化：按需出现的运营团队

即使 AI 能帮你研究、能帮你写代码，公司仍然有很多琐碎但必要的事情：日程安排、CRM 更新、周报、文档同步、内容发布、合规追踪、工具之间的数据流转。小团队里，这些工作通常压在创始人身上。

AI 工作流自动化可以把这些注意力税转移出去：商机状态一变，CRM 自动更新；每周数据自动整理成报告；产品变更后文档自动同步。对于“Day Zero”式早期公司，能否把这些流程自动化，直接决定创始人能不能把时间用在真正重要的判断上。

时机与编排比工具本身更重要

能同时用好 AI 的研究、自动化和编程能力的创始人，会获得远超团队人数的杠杆。但这不会自动发生。创始人必须知道何时用 Chat 类工具，何时用代码代理，何时用工作流代理，何时必须自己判断。

> **关键提醒：**AI 不是自动驾驶创业公司。它更像一支极强的虚拟团队，但方向、边界、取舍和验收标准仍然来自创始人。

第 3 章 想法阶段：先验证，再动手

每个创业项目都从一个创始人反复思考的问题开始。但在 2026 年，成功创业更需要一种纪律：在证据足够之前，不要急着开发。

想法阶段的工作包括研究、用户发现、竞品分析，以及诚实地寻找反证。也就是说，在让代码代理写第一行生产代码之前，先证明这个问题值得做。

想法阶段的目标

这个阶段的核心目标是研究型验证：在投入资源开发之前，找到足够扎实的证据，证明一个真实问题存在，并且你的解决方案确实可能解决它。

创始人需要按顺序回答几个问题：

- 这个问题是否真实、具体，并且发生频率足够高？
- 到底是谁有这个问题？这些人能否构成一个市场？
- 有没有其他人在解决？他们怎么解决，解决得怎么样？
- 真正能解决这个问题的方案需要具备什么能力？我的想法是否符合？

这些问题最终汇总成一个终极问题：这件事值得做吗？

因此，创始人必须从模糊观察走向可验证假设。“大家做报销很麻烦”只是观察；“中型公司的财务经理每周花 4 小时以上核对报销，因为现有工具无法与会计系统打通”才是可以验证的创业假设。

想法阶段的退出标准

想法阶段的退出条件是找到问题-方案匹配。也就是：你已经通过真实人的对话和定性证据，证明自己在为真实人群解决真实问题。

当你能对以下三个问题都回答“是”时，才适合进入下一阶段：

1. 问题是否真实且具体？你能说清楚是谁遇到这个问题、多久遇到一次、影响有多严重、他们现在如何处理。
2. 你的方案是否解决了验证过程中发现的真实问题？注意，不一定是你最开始以为的问题。
3. 证据是否足够支撑开始做 MVP？这个阶段不可能有绝对确定性，但必须有足够定性证据，说明动手开发是理性决策，而不是信仰跳跃。

想法阶段的主要挑战

1. 把“做出来”误认为“验证过”

当技术门槛被 AI 降低，最危险的事情反而是跳过验证，直接开始做。很多创业失败并不是因为做不出来，而是因为做了一个没人真正想要的东西。

在 AI 编程工具出现前，一个原型可能需要几个月和真金白银。现在，一个看起来像产品的原型可能几个小时就能做出来。这种便利会诱导创始人把流程变成：有想法 - 立刻做原型 - 因为原型存在，所以相信想法已经被验证。

但可运行原型不是证据。它只是一个很好的“访谈道具”，可以帮助你和潜在用户进行更真实的对话。真正的证据来自这些对话，而不是原型本身。

2. 过早扩张

当开发变得轻松、快速、低成本时，你很容易在商业需求还没成立前，就把执行规模拉得过大。

过早扩张的本质是：在没有真正验证方向之前，就把资源、代码、功能和流程压到一个产品路径上。AI 代码代理会非常积极地帮你围绕一个错误前提生成、测试、调试、重构代码。它不会自动知道你的商业假设是错的。系统中的智能，仍然主要来自你。

> **原则：**想法阶段最重要的原则：你的理解速度必须始终领先于你的开发速度。

3. 失去客观性

如果你让 AI 帮你找支持自己观点的证据，它通常能找到。确认偏误现在拥有了一台强大的研究引擎。

AI 会沿着你的方向执行。你问“为什么这个项目值得做”，它会给出一份看似充分的支持材料；你问“这个市场多大”，它也可能给出一个让融资故事看起来很漂亮的数字。

解决办法不是少用 AI，而是反过来用：让 AI 充当反方，寻找否定证据，列出失败案例，分析结构性障碍。当证据说明你的想法需要修改时，那不是坏消息，而是该转向的信号。

AI 如何帮助想法阶段的创始人

想法阶段很容易让创始人焦躁，因为你会想马上开始做。但这个阶段本质上是研究和验证，不是编码冲刺。AI 最适合帮你更快、更严谨地思考。

不同 AI 工作界面的用途

原文以 Claude、Claude Cowork、Claude Code 为例。放到国内语境，可以理解为三类 AI 使用形态：

任务类型	适合使用	为什么
提问、改写、快速脑暴	通用聊天助手	响应快，无需复杂设置，适合日常小任务。
研究、分析、生成完整文档	能连接文件和系统的 AI 工作台	可读取资料夹、表格、邮件、项目工具，适合产出报告、文档、表格。
写代码、测试、部署、改造软件	AI 编程代理 / 代码代理	可以访问代码库、生成 diff、运行命令、接入 git 和开发环境。

三者底层都可以是同类大模型，差别在于模型周围的工作空间：有没有文件、代码、工具、权限、自动化流程。

定义并压力测试问题假设

先把你的问题假设变得足够具体。AI 在这里的价值，是逼你回答：谁遇到这个问题？多频繁？多严重？现在怎么解决？如果一个问题陈述无法回答这些问题，就还不适合验证。

- 练习：把“合同审查太慢”改写为可验证假设。例如：“中型企业法务团队每个合同审查周期要花 3 天以上，因为红线修改散落在邮件线程中，而不是集中在一个版本可控的文档里。”

下一步，让 AI 反驳你。要求它寻找能推翻假设的证据，包括负面市场信号、失败竞品、客户行为模式和结构性障碍。这样你进入用户访谈时，才不是寻找确认，而是真正开放地学习。

市场研究与竞品地图

创业者很容易陷入“竞品忽视”：太专注自己的愿景和执行，而低估别人已经在做什么。AI 的用法之一，就是请它认真论证为什么竞品会成功，而你不会。

- 练习：让 AI 按层级绘制竞争格局：直接竞品、间接竞品、潜在收购方、可能跨界进入的邻近玩家。然后让它说明每一类为什么会对你构成真实威胁。
- 练习：让 AI 汇总竞品用户评论，提取反复出现的抱怨和未满足需求。如果你的假设正好解决这些问题，就是强信号；如果没有，也要知道。
- 练习：基于公开资料搭建 TAM/SAM/SOM 模型，并让 AI 压力测试其中的假设：市场是在扩张、整合还是成熟？谁掌握预算？谁影响购买？
- 练习：让 AI 找出未来两年会影响该市场的三个外部趋势：监管、技术、人口结构或行业变化，并判断它们是顺风还是逆风。

这些研究不是一次性工作。只要你的假设发生变化，就应该重新做一次竞品、市场和趋势分析。

设计用户访谈

用户访谈的质量取决于两件事：问对人，问对问题。AI 可以帮你定义访谈对象、设计问题、分析反馈。

首先，你需要明确目标画像：具体职位、公司类型、团队结构、资历层级，以及他们最可能在哪里出现，例如社群、活动、LinkedIn/脉脉群、行业 Slack/飞书/微信群、线下活动等。

其次，设计问题时要避免问“你会不会用这样的产品？”这类未来假设问题。更好的问法是：“上一次你遇到这个问题时发生了什么？”你要问的是过去真实行为，而不是未来口头承诺。

- 练习：先自己写访谈提纲，再让 AI 审核。要求它标出诱导性、面向未来、过于宽泛、容易得到社交礼貌答案的问题，并给出追问设计。

访谈后，把笔记交给 AI 复盘：哪些内容支持假设，哪些挑战假设，哪些令人意外。每完成 5 个访谈，就让 AI 汇总支持证据和反对证据。如果支持列表远长于反对列表，要问 AI：这是数据真实反映，还是我只看见了想看见的东西？

用户邀约与排期

AI 工作台可以承担很多用户发现中的运营工作：整理潜在访谈对象名单、生成个性化邀约邮件、跟进未回复的人、安排日程、更新追踪表。

- 练习：把目标访谈画像交给 AI，让它建立候选名单、写邀约序列、建立状态表，字段包括：邀约状态、跟进节奏、是否完成访谈、访谈重点。创始人把精力放在真正的对话上。

形成最终方案概念

当验证工作完成后，你已经知道问题真实存在，也知道谁有这个问题。此时再让 AI 帮你完善方案：还有哪些缺口？有哪些替代方案？这个方案要规模化成立，必须满足哪些前提？

- 练习：把方案交给 AI，让它找出这个设计最依赖的三个假设。再分别分析：每个假设要成立，现实中必须发生什么？如果不成立，后果是什么？

用 AI 编程代理做轻量原型

现在才进入真正好玩的部分：当假设被验证、方案被压力测试之后，你可以做一个轻量原型。它不是完整产品，而是一个可以让真实用户触摸、体验、反馈的最小交互样本。

- 练习：定义你的方案最核心的一次交互，只让代码代理做这一件事。然后拿给 5 个目标用户试用。接下来的 5 次对话会决定你继续开发，还是回到画板。

当你走完想法阶段，你就不再是在赌直觉，而是在根据证据执行。下一阶段的问题会从“这件事值不值得做？”变成“我们到底应该先做什么？”

第 4 章 MVP 阶段：最小产品，不是最小工程

很多创始人把 MVP 阶段理解为开发阶段。但 MVP 阶段仍然是证据收集阶段，只不过你收集的不再是“问题是否存在”的证据，而是“这个方案是否真的有价值”的证据。你要看真实用户是否愿意使用、回来、付费，甚至主动推荐。

MVP 阶段的目标

AI 原生创始人在 MVP 阶段的目标，是把已经验证过的问题转化为真实用户愿意使用的工作产品。MVP 不是带有完整路线图的最终版本，而是最小、最聚焦、能把真实解决方案放到真实用户面前，并产生产品-市场匹配证据的版本。

同时，这一阶段还有第二个同样重要的目标：快，但不能快到积累无法承受的技术债。MVP 的建设方式会决定后面能不能扩展。

第三个目标是从第一天开始建立持久上下文。AI 原生创业中，代码库会被你和 AI 在一轮又一轮会话中共同修改。如果你没有规格说明、架构决策、上下文文件，每次新会话都要重新解释项目，AI 生成的改动也会逐渐偏离最初设计。

MVP 阶段的退出标准

MVP 阶段的退出条件是真正的产品-市场匹配证据：某一类明确用户发现产品有价值，愿意反复使用、付费，或推荐给别人。

MVP 阶段的主要挑战

1. AI 时代的技术债

AI 几乎移除了过去控制代码进入生产环境的自然阻碍。速度被保证了，但如果创始人只看速度，就会积累以后很难偿还的技术债。

MVP 阶段允许一定技术债，因为你在追求验证。但 AI 技术债的特殊之处在于它会快速复利：如果没有可读的规格说明、架构约束和上下文文件，每次 AI 会话都会重新推导基础决策，并逐渐漂移。最后你会得到一个能运行但没有统一心智模型的代码库。问题不一定出在某一段代码，而是整体从未被设计成一体。

2. 假性产品-市场匹配

早期数据很容易让创始人兴奋：朋友注册、投资人资源带来的试用、一次社媒曝光、一次社区推荐，都可能制造“看起来很有市场”的错觉。

但早期热度不等于产品-市场匹配。真正的问题是第 6 周、第 12 周之后，当初始流量消退，用户是否还回来、是否继续用、是否愿意付费。

3. 零摩擦功能蔓延

当增加功能只需要一个下午，而不是一个开发 sprint，创始人就会不断想加“顺手”的功能：多支持一个场景、多处理一个边界、多做一个看起来很酷的能力。每个新增都可以被解释为合理，但产品会逐渐失去边界。

解决方案是在开发前写下 MVP 范围：产品做什么、明确不做什么，以及什么样的真实用户证据才能证明值得加新功能。这样你讨论的就不是“想不想做”，而是“是否有足够用户没有这个功能就无法获得价值”。

4. 因经验不足导致的不安全

AI 代码代理生成的是能工作的代码，不代表它天然安全。功能是否可用容易判断，安全漏洞往往在被利用之前不可见。

只要 MVP 面向真实用户，就涉及真实数据、真实风险和真实后果。最低限度是在任何真实用户接触产品前做一次安全审查，尤其关注鉴权、会话、数据暴露、输入校验、依赖漏洞、密钥管理。

AI 如何帮助 MVP 阶段的创始人

先定义架构，再开始写代码

在 AI 写第一行生产代码之前，先用 AI 定义和记录架构决策：采用哪些模式，避免哪些依赖，接受哪些权衡，为什么这么做。这份输出应成为项目的架构上下文文档。

没有上下文，代码代理每次都会从头推断结构假设。它能做出可运行代码，但不一定能做出结构一致的代码。结构不一致的代码库最终会浪费大量时间和 token，甚至不得不重写。

- 练习：打开 AI，描述你要做什么、解决什么问题、服务哪些用户、未来 6 个月大概会到什么规模。让它帮你定义 MVP 架构原则、应避免的依赖、当前阶段主动接受的权衡。

然后，把这份输出保存为项目上下文文件，例如 CLAUDE.md、README-AI.md、ARCHITECTURE.md 或类似文档。它的作用是项目记忆：让之后每次 AI 编码都能先读懂项目边界。

定义并执行 MVP 范围

AI 时代 MVP 最常见的失败模式之一，就是没有摩擦的范围蔓延。因此，除了架构文档，还需要范围文档：MVP 做什么，不做什么，新增功能需要满足什么证据条件。

当新点子出现时，让 AI 帮你判断：这是用户真实信号，还是创始人自己的兴奋感伪装成产品洞察。

用 AI 编程代理构建 MVP

当架构和范围明确后，代码代理才是主要建设工具。它负责生成、测试、调试、迭代代码。但每次会话都应该执行已经做出的产品决策，而不是临时发散出一堆新功能。

- 练习：建立一个简单的 AI 编码会话模板，包括架构上下文、本次任务、约束条件、要遵守的模式。每次结束后，更新一小段会话日志：做了什么、做了哪些决定、引入了什么假设。每次花 5 分钟，能显著降低长期架构漂移。

真实用户接触前做安全审查

AI 可以做第一轮安全审查，找常见问题，但不能替代专业安全工具或高风险场景下的人类审核。

- 练习：上线给真实用户前，让 AI 按清单审查核心应用代码：身份认证和会话、API 响应中的数据泄露、输入校验和注入风险、依赖漏洞、密钥暴露。涉及鉴权、密钥、数据处理的发现，要认真评估是否必须修复。

发布前先设计指标体系

把早期热度误认为 PMF 的创始人，通常是在产品上线后才开始补数据。他们会选择能证明自己做对了的指标，而不是能暴露问题的指标。正确做法是在第一个用户出现前，就定义好衡量框架。

你需要提前设定激活标准、留存标准、Day 7 和 Day 30 目标，并定义什么是假阳性：注册但不激活，有收入但无留存，有初始兴奋但没有重复使用。

当数据出现后，让 AI 做反方：如果一个怀疑者看这些数字，会如何解释它们?

管理用户反馈和迭代流程

当真实用户开始使用产品后，运营层会迅速膨胀：用户列表、反馈访谈、bug 报告、功能请求、迭代节奏。AI 工作台可以承担很多重要但琐碎的部分。

- 练习：让 AI 搭建 MVP 阶段反馈循环：给早期用户发邀约，安排反馈会，设计 bug/功能请求收集流程，每周整理一次反馈综合报告。创始人需要先亲自读这份报告，再让 AI 分析可能遗漏的重点。

朝证据迭代，而不是朝“完整”迭代

MVP 阶段的结束不是因为产品看起来完整，而是因为你有真实产品-市场匹配证据。判断 PMF 通常需要多轮迭代和多种信号，而不是一个单点数据。

- Sean Ellis 测试：问活跃用户“如果以后不能再使用这个产品，你会有什么感受？”如果超过 40% 选择“非常失望”，这是重要的 PMF 信号。
- 努力程度测试：PMF 前，留存依赖创始人大量干预、补贴、私聊、强推；PMF 后，产品开始自己拉动用户回来。当增长从“推”变成“拉”，说明发生了实质变化。

证据要求你转向时，就转向

如果三轮以上迭代后仍然没有明显接近 PMF，不要把它理解为失败。这正是 MVP 阶段的作用：在你过度投入错误答案之前暴露问题。

- 练习：把留存数据、用户反馈、原始问题假设交给 AI，问三个问题：是否存在一个细分人群反应明显不同？价值设计和用户感知之间的差距，是定位问题还是产品问题？当前产品要找到真正 PMF，必须满足什么条件？这些条件现实吗?

答案会帮助你判断：是微调、转向，还是回到想法阶段重新验证。

第 5 章 发布阶段：从有人用，到能增长

如果说 MVP 阶段是在证明产品值得存在，那么发布阶段就是证明这家公司值得增长。

发布阶段的目标

发布阶段，创始人需要把早期 traction 转化为可重复、可持续的增长引擎。除了让产品达到生产可用状态，还要加固底层基础设施，并围绕产品建立真正的公司运转系统。

在想法和 MVP 阶段，创始人深度参与每个环节是优势，因为需要强烈的现场感和快速反馈。到了发布阶段，如果创始人仍然试图抓住所有线头，就会成为瓶颈。目标不是把创始人从公司里拿走，而是建立系统，让创始人只处理真正需要创始人判断的事情。

发布阶段的退出标准

4. 增长可重复且来自明确渠道。你不只是留住用户，还能通过具体渠道稳定获客，并理解获客成本、客户生命周期价值和回本周期。
5. 产品能承载生产负载。基础设施已加固，安全和合规到位，在真实生产环境下保持可靠。
6. 运营不再依赖创始人亲自推动。支持、分诊、sprint 规划、报表等流程可以按系统运行。

发布阶段的主要挑战

1. 技术债开始到期

MVP 阶段为了速度做出的捷径，在生产流量、功能增长和复杂度上升后会被暴露。发布阶段需要系统性架构审计、针对性重构，以及足够测试覆盖，避免下一轮功能开发再次引入同类问题。

2. 创始人成为瓶颈

在 MVP 阶段，创始人处在每个循环中是资产；发布阶段，这可能变成限制。支持请求堆积、产品决策排队、只有创始人知道答案、运营任务只有创始人记得做，这些都是信号。

解决方案是彻底盘点所有由创始人亲自处理的事情，从小任务到高风险决策，判断哪些可以系统化、哪些可以委派、哪些真的仍需创始人投入。

3. 安全和合规不能再推迟

MVP 阶段可以保持简单，但一旦有真实用户、真实数据，甚至企业客户在谈，安全和合规就不能再是“以后再说”。处理客户数据、支付、进入监管行业，都会带来明确合规要求。

正确做法是在生产规模到来之前做系统审查，而不是事后补救。审查发现的问题应被视为必须修复项，而不是建议项。

4. 还没准备好就扩张

新市场和融资机会看起来像增长机会，但也可能让 PMF 消失。你已有的 traction 往往来自特定早期人群。过早进入差异很大的市场，会引入新的行为、合规、支付、预期和竞争变量，导致你无法解释数据，也可能忽视原始用户群。

AI 如何帮助发布阶段的创始人

发布阶段，通用 AI、代码代理和工作流代理应同时使用，并互相提供输入。代码代理建设产品，工作流代理建设公司周围的运营层，通用 AI 则帮助沉淀产品和组织知识。

在技术债复利前处理它

用代码代理对 MVP 代码库做一次完整架构审计：哪里脆弱，哪些捷径会变成维护成本，哪些地方测试覆盖过薄。然后把审计结果交给通用 AI 排优先级：哪些必须在下一次发布前修，哪些可以等一个 sprint，哪些可以作为当前阶段可接受债务。

- 练习：让代码代理生成结构性弱点、测试缺口、重构候选清单；再让 AI 把这些工作排进几个 sprint，区分紧急问题、可并行处理的问题、可等待的问题。

建立替代创始人注意力的系统

先知道你的注意力去了哪里。让 AI 工作台审计当前运营负担：所有周期性任务、所有需要你拍板的决策、所有只有你记得才会发生的流程。然后分类：完全可自动化、需要人但不一定是你、必须由创始人判断。

审计完成后，为自动化候选设计流程逻辑：触发条件是什么，决策规则是什么，输出长什么样，完成后进入哪里。

把安全和合规纳入产品工作流

让代码代理检查目标市场需要的 SOC 2、GDPR、HIPAA 或国内等保、数据安全、隐私合规相关问题。AI 扫描不能替代专业审查，但可以帮助你提前发现漏洞和文档缺口。

- 练习：围绕目标市场合规框架做代码级安全检查，再让 AI 输出两份东西：安全修复优先级、企业客户合规审查会要求的文档和控制项。

补上过去跳过的产品管理流程

发布阶段需要轻量但可重复的流程：sprint 节奏、功能 spec 最低标准、bug 分诊树、每周指标简报、反馈如何进入产品决策。

- 练习：让 AI 设计一个轻量产品管理操作系统，包括 sprint 周期、最小 spec 模板、bug 分诊规则、每周指标简报。再让 AI 工作台执行其中可周期化的运营部分，如排会、路由、报告生成。

第 6 章 规模化阶段：让公司离开创始人也能运转

规模化阶段，创始人的角色会从 builder 重新转向面向外部的公司领导者。产品仍然核心，但创始人的日常会越来越多地围绕公司本身展开：分析师沟通、重要客户、融资叙事、IPO 或并购准备、组织治理等。

规模化阶段的目标

规模化不只是技术基础设施扩容，也是组织能力成熟。你可能从几千用户走向数百万用户，从一个市场走向多个市场。此前每个阶段都可以靠贴近用户和创始人直觉快速调整；规模化阶段必须建立系统化增长和成熟运营。

对 AI 原生创业公司来说，真正的护城河来自持续积累的深度：你嵌入产品里的行业专业能力、与用户工作系统的集成深度、专有的系统数据和流程。持续朝一个方向、在一致基础设施上建设的团队，会逐渐得到难以复制的东西。

这一阶段，公众投资人、分析师、监管方、企业采购和收购方都会以更高标准审视你。不只是产品能力，还包括治理、合规、财务控制、战略叙事。

规模化阶段的退出标准

规模化阶段不再是一个单一里程碑，而是一个门槛：即使创始人逐渐不直接管理日常运营，公司仍然可持续运转。

实践中，这个门槛通常表现为三种形式之一：稳定盈利且不再依赖外部资本、具备 IPO 准备度，或具备被收购价值。三者都要求增长可系统化、可审计，产品护城河经得起审查，组织成熟可持续。

规模化阶段的主要挑战

1. 交出运营层

发布阶段是在创建系统；规模化阶段则是让系统成熟到可信，并且创始人真的信任它。

这对长期亲手参与的创始人既是结构挑战，也是心理挑战。交得太快，关键决策可能丢失只有创始人知道的上下文；交得太慢，创始人又会成为瓶颈。核心在于把创始人脑内知识和未文档化流程转化为可审计、可传递的系统。

2. 技术运营规模化

客户不再只评估你的产品，还会评估你这个组织能否成为可靠的基础设施伙伴。大客户会要求支持体系、文档、可靠性承诺、SLA、监控、事故响应机制。

3. 组织职能规模化

无论团队人数多少，规模化公司都需要招聘、薪酬、财务、法务、合同、合规监控、客户支持等组织基础设施。AI 可以帮助小团队先搭出这些职能的自动化骨架。

4. 建立真正的 GTM 能力

创始人亲自销售、社群爆发、Product Hunt 或国内平台的一次曝光，都有增长上限。规模化阶段需要真正的 go-to-market 体系：市场细分、品牌叙事、销售手册、分析师关系、投资人指标叙事、内容管线、CRM 管理、外呼序列、产品营销材料。

AI 如何帮助规模化阶段的创始人

把日常任务交给 AI 工作台

规模化开始时，先重新定义“只有创始人应该做的事”：产品叙事、董事会关系、关键企业交易、创始人与创始人之间的高价值对话等。不在清单上的事，都应该考虑委派或自动化。

- 练习：让 AI 制作当前运营瓶颈图：所有经过你的流程、决策、审批。再让它推演如果你离开一周，哪些流程会卡住。卡住的地方，就是移交标准、升级路径、异常处理还不够清楚。

把技术运营升级为企业级基础设施

规模化客户需要确认：你的产品和组织可以被长期信任。通用 AI 可以帮助起草和维护企业采购会看的材料：产品文档、支持手册、SLA。代码代理可以建设日志、监控、事故响应、可观测性层。AI 工作台可以运行工单路由、升级流程、文档同步、续约追踪、客户成功报告节奏。

- 练习：选出 3 个最重要潜在客户，或 3 个理想客户画像。让 AI 做差距分析：这些企业采购团队签多年合同时，会要求哪些文档、SLA 和支持基础设施？你现在缺什么？

建立真正的 GTM 引擎

创始人冲劲可以帮你走到这里，但规模化需要完整的商业化引擎。AI 可以帮助从零搭建市场细分、消息架构、分析师关系策略、销售手册、投资人叙事。不同受众有不同语言：个人用户、企业买家、投资人、行业分析师关心的东西并不一样。

AI 工作台负责战术执行：内容管线、外呼序列、PR 节奏、CRM 清洁、pipeline 报告。代码代理可以搭建产品营销基础设施：交互式 demo、集成文档、沙盒租户、API 文档、技术一页纸。

把领域知识和组织知识变成 AI 上下文

很多超精简创业公司做的是非常垂直的问题。创始人未必会写代码，但拥有行业经验、流程经验、监管细节、边界案例。这些知识应该被沉淀为产品可以读取、AI 可以复用的结构化上下文。

通过长期对话、项目、记忆和技能，创始人可以把行业黑话、监管坑、边界条件、真实痛点、为什么常规方案不可行等内容，整理成可搜索的知识底座。久而久之，这会成为通用 AI 或泛化竞品无法轻易复制的专有上下文。

- 练习：找一个通用竞品一定会做错的行业边界案例，用代码代理为它建立专门测试场景。之后每出现一个类似边界案例，就加入测试集。你的测试集会逐渐变成护城河地图。

把用户行为数据复利成优势

用户使用产品时，会产生行为信号：哪些输出被接受，哪些被拒绝，哪些流程反复出现，哪些边界情况最常发生。长期积累后，你会理解特定用户群的偏好、模式和工作流。

这就是复利：每次改进让产品更好用，更好用带来更多使用，更多使用带来更多反馈，反馈再推动改进。这类数据是时间锁定、场景特定的，竞争者很难快速买到或复制。

- 练习：把你收集的用户交互数据摘要交给 AI：收集了什么、多久、用户如何使用。让它找出三个最高信号的行为模式，并设计把每个模式转化为系统改进的反馈循环。最后写一页护城河叙事：数据飞轮如何运转，已经运转多久，为什么强竞争者从今天开始也难以在两年内复制。

建立工作流锁定

数据网络效应让产品难以复制，工作流锁定让产品难以离开。当用户把产品嵌入日常流程、基于它建立自动化、训练团队、连接数据源和工具后，切换产品就不再是买不买的问题，而是一次组织迁移工程。

第一步是用 AI 盘点客户集成深度：每类客户把哪些流程跑在你的产品上，依赖哪些集成，切换成本多高。代码代理可以帮助快速建设原生集成、API、Webhook 和 SDK，让客户不只是使用你的产品，而是在你的产品上继续搭建。

- 练习：为前 10 个客户做工作流集成审计：他们建立了哪些自动化、依赖哪些集成、哪些团队流程经过你的产品、切换成本估计如何。再让 AI 找出共性：哪类集成最能形成深锁定？你还能建设什么让浅层客户变得更深？

第 7 章 工作没变，规则变了

在 AI 时代，创始人的根本工作没有变：找到真实问题，做出能解决问题的东西，再把它规模化成一家有意义的公司。变化的是到达目标的路径。

在想法、MVP、发布、规模化四个阶段中，AI 会把过去需要几个季度完成的事情压缩到几周。验证循环从几个月缩短到几个下午；可运行原型不再必须依赖懂特定技术栈的合伙人；发布准备从一次上线前冲刺变成持续工作流；规模化阶段，那些过去迫使公司早早招人的运营负担，也越来越可以交给 AI 系统。

新的瓶颈不再是“你能不能做出来”，而是“你选择做什么”。

> **最终结论：**AI 原生创业的核心竞争力，不是会使用某一个工具，而是能持续做出正确判断：判断问题、判断证据、判断边界、判断时机、判断哪些事该交给 AI，哪些事必须由人负责。

资源与参考方向

原文资源主要围绕 Claude 生态。对于国内读者，可以按“能力类型”理解，而不是只按某个产品名称理解。

一、AI 编程与代码代理

- Claude Code / Codex / Cursor / GitHub Copilot / Continue / DeepSeek CLI 等：重点不是谁更热，而是是否能访问代码库、理解项目上下文、运行命令、生成 diff、配合 git 和测试。
- 关键实践：维护项目上下文文件，如 CLAUDE.md、AGENTS.md、README-AI.md、ARCHITECTURE.md；每次 AI 编码前让它读上下文，每次结束后更新决策记录。

二、AI 工作台与自动化

- 原文中的 Claude Cowork 可以理解为能连接文件、邮箱、日历、表格、项目管理工具和内部系统的 AI 工作台。国内可根据实际工具环境，用飞书/钉钉/企业微信/语雀/Notion/Google Workspace/MCP 连接器等组合实现。
- 关键实践：把重复运营工作做成触发器、规则、输出和检查点，而不是每次临时让 AI 聊天。

三、创始人研究与战略思考

- 适合用通用 AI 做：竞品地图、用户访谈提纲、行业趋势、反方论证、融资 memo、PRD、商业模式推演、失败预演。
- 关键实践：不要只问“为什么我对”，要经常问“我哪里可能错”“什么证据能推翻我”“如果竞品赢了，原因是什么”。

四、创业案例理解方式

- 原文列举了多个使用 Claude 构建公司、法律平台、医疗数据抽象、采购供应链 Agent、非营利资助匹配等案例。国内读者更应关注案例背后的模式：领域专家 + AI 编程 + 工作流自动化 + 数据飞轮 + 组织系统。

附录：国内语境下的术语对照

英文/原文概念	中文理解	国内语境解释
AI-native startup	AI 原生创业公司	不是加一个 AI 功能，而是用 AI 重构研发、研究、运营、组织。
Agentic coding	AI 编程代理 / 代码代理	AI 不只是补全代码，而是能规划、写、测、改、运行命令。
MVP	最小可行产品	最小不是粗糙，而是最小闭环验证。
PMF	产品-市场匹配	用户真的反复使用、愿意付费或推荐。
GTM	商业化/增长打法	如何定位、获客、销售、转化、续费。
CAC	获客成本	获得一个客户所花的钱。
LTV	客户生命周期价值	一个客户在整个生命周期贡献的收入。
SLA	服务等级协议	对稳定性、响应时间、支持承诺的正式约定。
Technical debt	技术债	为了速度留下的结构性成本。
Moat	护城河	竞品难以复制、用户难以离开的长期优势。
Workflow lock-in	工作流锁定	产品嵌入用户日常流程后，迁移成本变高。