THE FOUNDER'S PLAYBOOK

创始人手册:
打造一家 AI 原生创业公司

在 2026,从想法到规模化的路径已被 AI 重写。本手册按四个核心阶段(创意、MVP、发布、规模化)重新梳理创业旅程,并给出每个阶段该如何使用 AI 工具。

Anthropic 官方手册 · 中文译本
原文发布于 claude.com/blog/the-founders-playbook

目录

第 1 章

创业生命周期,为 2026 重启

AI 正在重塑创业公司的构建方式。今天,从没写过一行代码的创始人也能交付可用于生产环境的应用;曾经只是"草根逆袭故事"的"10 人独角兽",如今已成为一种可以刻意规划的路线。

在 2026,AI 能够编写生产级代码、做市场调研、梳理竞争格局、起草投资材料、自动化运营流程。过去即便是经验丰富的技术型创始人,要把想法落地,也要面对集成各种工具、平台与系统的陡峭学习曲线;AI 把这些曲线几乎抹平,最重要的是,它拉平了"谁能创业、谁能做产品"的门槛。

在 2026,一个好想法能带创始人走得比以往任何时候都远。智能体式编码(agentic coding)把过去需要一整支工程团队才能完成的工作,压缩成创始人自己就能交付的活儿。

传统的创业增长弧线假设:从想法到规模化的路径是"验证 → 融资 → 招人 → 构建 → 再融资 → 增长 → 招更多人 → 循环"。如今,AI 已经打破了"每进入一个新阶段都需要更大团队、不同技能、新一轮融资"的预设。

本手册按这些新现实,重新梳理创业旅程的四个核心阶段——创意(Idea)、MVP、发布(Launch)、规模化(Scale)。我们会探讨:当 AI 成为技术与组织发展的核心时,每个阶段是什么样子;每个阶段对应的正确工具是什么;以及正在使用这些工具的创始人,是如何压缩时间线的。如果你已经准备好规划出"从想法到退出"的最短路径,请继续往下读。

第 2 章

创始人的定义正在改变

过去,创始人由"他能做什么"来定义:技术型创始人写代码,非技术型创始人做业务运营、谈成交易。但 2026 年创始人可用的模型、系统与 AI 智能体,已经溶解了"能构建的人"与"有好想法的人"之间的那堵墙。

AI 原生创业公司正在从根本上改变"成为创始人"的含义。现在,毫无工程背景的人也能构建出把想法落地的生产级软件;而一个技术娴熟、却缺乏商业知识的创始人,也能轻松产出 go-to-market 策略、财务模型和高度打磨的融资路演稿。

从历史上看,创始人把大部分时间花在"执行模式"上:写代码、管人、处理日常运营。在 AI 原生创业公司里,创始人的角色不再那么像"亲力亲为的执行者",而更像"智能体的指挥者"——这些专门的 AI 助手能读取文件、运行命令、执行代码,甚至浏览网页。创始人的注意力随之向上移动,转向更高阶的工作:产生想法,并指挥那些执行想法的系统(AI 智能体、工具,以及现有的小团队)。

不过,AI 作为核心基础设施最具革命性的结果,是它解放了那些拥有行业专长的非技术型创始人。当创始人的来源不再局限于有工程背景的人,你会看到一批由"经历截然不同的人"创办的公司,他们在解决传统技术创始人通道从未优先关注(甚至从未注意到)的真实问题。

精益创业公司的 AI 能力

传统创业模型假设:你需要招工程师来构建、招销售来卖货、招运营来运转公司。人头数被当作组织势能和产品成熟度的标志。

2026 年的早期创业公司则截然不同。它们在设计上就极度精益,常常只有创始人一个人,或加上寥寥数人。通过把技术与组织发展都建立在"AI 作为基础设施"之上,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 尤其在三个领域,能让创业公司运转得像一家大得多的组织:调研、智能体式编码,以及关键业务运营的流程自动化。

对话式智能与调研

可以理解为:每个领域都随时待命的专家。

想想一个创始人在第一年里几乎必然不懂、却必须知道的所有事:我该怎么设置工资发放?怎么规划产品开发冲刺?怎么写一份精炼的投资备忘录?

过去,这类早期创业问题的答案都一样:"去找个懂的人"。对一个自筹资金或种子前的创始人来说,这意味着把本该用于构建的时间花在搜集知识上,甚至要烧掉一部分早期资金请顾问。而现在,他们有了一个跨越各种领域、随叫随到的 AI 专家。

- **深度调研:**竞品分析、市场规模估算、财务建模。
- **文档起草:**融资路演稿、案例研究、投资备忘录、PRD。
- **战略思考伙伴:**唱反调式分析、事前验尸(pre-mortem)、情景规划、路线图优化。

智能体式编码

可以理解为:那个永远在线、永不被阻塞的工程师。

过去,构建软件需要一位技术联合创始人、一家外包开发公司,或一条足够长的现金跑道,好在写出第一行生产代码之前就先招起一支工程团队。

智能体式编码工具如今让每一位有志创业者都能用自然语言描述想构建的东西,并指挥 AI 以一整支工程团队的速度和规模来生成、测试、调试、重构一套生产级代码库。从"我有个想法"到"我有个产品"的时间线已被压缩。创始人的角色如今聚焦于"构建什么、为什么构建",而 AI 负责真正搭建出可供真实用户使用的基础设施。

工作流自动化

可以理解为:按需调用、自动运转的运营团队。

即便一个创始人能像顾问一样做调研、像工程团队一样做构建,仍有一整类工作既不属于战略规划也不属于产品开发,却照样得做:排日程、更新 CRM、拉每周报表、维护文档、发布内容、跟踪合规要求、维护公司各类工具与系统之间的衔接。在精益创业公司里,这些负担主要落在创始人头上——它对本该用于更高阶决策的时间与注意力,是一笔不小的"税"。

用 AI 工具做工作流自动化,能卸下这笔税。重复性的运营任务可以被配置为自动发生:交易状态一变 CRM 就自动更新、周报自己编译出来、产品文档随产品变更同步更新。而且关键在于,Claude Cowork 能与创业公司赖以运转的相互关联的系统(项目管理工具、沟通栈、数据源)集成,而无需专人去搭建和维护这些集成。在刚起步、一切从零的创业公司里,这个"专人"几乎总是创始人本人。

时机与编排就是一切

能有效驾驭 AI 在调研、自动化和智能体式编码上能力的创始人,可以打造出一家"杠杆远超其人头数"的创业公司。同时,他们也能把绝大部分时间和精力,投入到真正重要的工作上。

但这些不会自动驾驶完成:编排这些 AI 工具的创始人,需要懂得如何(以及何时)使用它们。本手册余下的部分,正是致力于探讨创始人在 AI 原生创业之路上会遇到的目标与挑战,以及如何在旅程的每个阶段有效运用 AI 工具。

第 3 章

创意阶段(Idea)

每个创始人都从同一个地方出发:一个挥之不去、停不下来思考的问题。这是想法与现实相遇的阶段。在 2026,创业成功需要一种纪律:在证据足以支撑之前,不要动手构建。这个阶段的工作是调研、客户发现、竞品分析,以及对"反面证据"的诚实评估——所有这些,都要在你让 Claude Code 生成第一行生产代码之前完成。

创意阶段目标

在创意阶段,创始人的主要目标是"以调研为导向的验证":在投入资源构建之前,先攒齐扎实的证据,证明一个真实的问题确实存在,并且你提出的方案能有效地解决它。

实际操作上,创意阶段是一连串创始人必须按大致顺序回答的问题:

- 这个问题足够真实、具体、高频,值得围绕它来构建吗?
- 究竟是谁有这个问题?他们构成一个市场吗?
- 有没有别人在解决它?如果有,他们怎么解决的、解决得多好?
- 一个方案到底需要做到什么才能解决这个问题?我的想法做到了吗?

这些追问加在一起,最终回答一个终极问题:**这值得构建吗?**

这意味着:在行动之前先把事情说具体。"人们在报销上很头疼"是一个观察。"中型公司的财务经理每周要花 4 小时以上去核对提交的报销,因为他们现有的工具无法与会计软件打通"——这才是一个可被检验的假设。

创意阶段的退出标准

创意阶段的退出条件,是找到"问题–方案契合"(problem-solution fit)。你已经建立了定性的证据(主要来自真实的人际对话),证明在动手构建解决方案之前,你确实是在为真实的人解决一个真实的问题。

当你能对以下三点都回答"是"时,就可以离开创意阶段了:

1. **问题是否真实且具体?**要回答"是",你必须能准确说出:谁会遇到这个问题、他们多频繁地遇到、它对他们的影响有多严重、他们现在是怎么应对的。
2. **你的方案是否解决了真正的问题?**不是你最初假设的那个问题,而是验证过程揭示出来的那个。有时两者一致,但并非总是如此。
3. **你是否有足够的信号支撑你去构建?**这个阶段你永远不会有确定性,而苦等确定性本身就是一种失败模式;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,而非一次信仰之举。

创意阶段的挑战

创意阶段是整段创业旅程中最重要的工作所在,因为这里也是最关键的错误最容易发生的地方:此刻搞错某件事,会迅速让你刚萌芽的事业脱轨。不过,创意阶段的大多数挑战都源于"跑得比你的理解更快",所以那些深思熟虑、审慎推进的创始人,会获得稳健的进展。

把"构建"误当成"验证"

挑战:当技术障碍被移除,一个满腔热情的创始人很容易跳过创业旅程中最重要的工作——验证他的想法是否真的是人们需要、并且会用的方案。

即便在智能体式编码出现之前,就有 42% 的创业公司因为"造了没人想要的东西"而失败。如今,像 Claude Code 这样的智能体式编码方案,已经大幅压缩了"我有个想法"到"我有个产品"之间的距离,这个失败率只会上升。

对一个拥有绝妙好想法的创始人来说,从没有比现在更好的时代;但"快速、轻松地搭出一个看起来很像产品的原型",反过来也给 AI 原生创业公司带来了一种真实而危险的生存风险。

直到不久前,构建还需要真金白银的开发时间和预算,搭个基础原型通常要好几个月。而现在技术开发的门槛几乎消失,AI 让创始人太容易"还没在真实世界里验证它有没有用,就直接扎进构建"。

达成问题–方案契合,需要先验证假设再构建;但很多首次创业者(甚至老手)误以为 AI 可以绕过这个要求,把流程变成了:有想法 → 立刻搭原型 → 把"原型存在"当成验证。原型反过来成了"假设本来就对"的理由,而你从未真正检验它是否成立。

一个能跑的原型,很容易被误认为是"你在解决真实问题"的确凿证据——但它不是。原型真正的用途,是在与潜在用户对话时,当一个用来试探、检验想法的"靶子"。这些对话本身,才是真正的证据。

过早规模化

挑战:当构建变得毫不费力、即刻完成,你可能会让"执行"远远跑在"业务真正需要"的前面。

过早规模化,意味着在你真正验证某条产品路径值得投入之前,就一头扎了进去。

这一直是创业杀手,但 AI 让创始人更容易在不知不觉中掉进这个陷阱。智能体式编码助手太强大了,以至于你很容易在从未"有意识地决定偏离航线"的情况下,就让执行远远超前于"问题–方案契合"的验证。它会以同样的热情,围绕一个根本错误的前提去生成、测试、调试、重构代码库——和它对待一个绝妙想法时一模一样。系统里的智慧来自你。这个阶段的首要指令,是让你的"理解"始终跑在"构建"前面,尤其是当构建如此之快、如此轻松的时候。

丧失客观性

挑战:你让 AI 工具去找支持你既有信念的证据,它就一定能找到。如今,确认偏误配上了一台研究引擎。

确认偏误(只盯着支持自己想法的证据、自动忽略相反信息)一直是创业中的职业病:创始人天然偏爱自己的点子。现在,AI 工具给这种偏误装上了助推器。让 AI 去验证你的创业想法,它会找到支持证据;让它去估算你的潜在市场,它会找到那个让你的 TAM 看起来"值得投"的数字。

AI 听从你的指令,这意味着一个不问难题的创始人,如今可以比以往任何时候都更快地、为一个糟糕的想法,构建出一套精致、看起来研究充分的论证,而且全程自信地以为自己正在做尽职调查。解药是同一个工具,只是把枪口反过来指:AI 用来"挑刺、拷问一个想法"时,会和"帮你背书"时一样卖力。当调研和结构化的对抗性思考揭示出"你的想法需要修正"的证据时,这就是该转向(pivot)的信号。

Claude 如何帮助创意阶段的创始人

推动你的 AI 原生创业概念走过创意阶段,可能会让人觉得遥遥无期。你是创始人,你只想赶紧构建。但这个至关重要的开局阶段,本质上是一次调研与验证的练习,意味着你要拿起那些"帮你在写代码之前更严谨地思考"的工具。下面是在创意阶段尽快推进、同时做好尽职调查的方法,涵盖 Claude 的各个产品形态(Chat、Claude Cowork、Claude Code)。

Chat、Claude Cowork 还是 Claude Code:选对 Claude 形态

AI 让创始人更容易更快交付、自动化繁琐流程、规模化运营,但你用哪种形态很关键。

Chat 适合不离开当前应用的快速交流。用它处理运营公司时那些持续不断的小任务:从一份密密麻麻的投资备忘录里拎出一句话的要点、在董事会前快速核对一个说法、读懂团队里一条冗长的 Slack 讨论串。

Claude Cowork 适合那些真正费时的知识工作:从大量来源中汲取、消化,并产出一份成品,比如一份文档、一份演示稿或一张表格。比如把一文件夹的客户通话记录变成下次产品评审用的主题发现文档、在融资前从十几个供应商网站搭出一张竞争格局图,或设置一个每周一早上的固定任务,从你接入的工具里拉取指标、把每周 KPI 简报投到共享文件夹里。

Claude Code 是面向团队工程师的智能体式编码环境:直接访问代码库、Plan Mode、git 集成,以及本地、IDE 或沙箱云环境。它是精益团队在不断增长的代码库上交付功能、迁移 MVP 时期遗留代码、在不增加人头的情况下从原型走向生产的地方。

如果任务是……	就用	原因
一个问题、一次改写、一次快速头脑风暴	Chat	快、对话式、无需配置
基于你的文件和系统做调研、分析,或产出一份成品文档	Claude Cowork	文件夹访问、连接器、技能、定时运行
编写、测试或交付软件	Claude Code	代码库访问、diff、git、开发环境

三者底层是同一个 Claude;变的是围绕它的工作空间。

界定并反复拷问你的"问题假设"

你的领域专长和前期调研已经产生了一个假设。第一项工作,是把它打磨到真正可被检验。Claude 在这里尤其有用,因为它会逼你说具体:究竟谁有这个问题、多频繁、多严重、现在怎么应对?一个无法精确回答这些问题的问题陈述,还不具备验证的条件。

> **练习**
>
> 和 Claude 一起打磨你的问题陈述,直到它成为一个可被检验的假设。例如,"合同评审太花时间"几乎无法检验;而"中型公司的内部法务团队,每个合同评审周期要花 3 天以上,因为红线修订散落在邮件往来里、而非一个单一的版本受控文档中"——这就非常可检验。

下一步,让 Claude 反驳你的想法,去寻找能推翻你假设的反面证据。这能浮现出负面的市场信号、失败过的竞品、客户行为模式,以及那些"支持性综述"会悄悄轻描淡写的结构性障碍。目标是:在进入客户发现之前,你已经把自己的假设拿去和最有力的反方论点对撞过,这样真正的用户访谈才会是真正开放式的,而不是一场"找确认"的搜寻。

注:把 Claude 当作"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。

市场调研与绘制竞争格局

掂量你的竞争对手。有一种创业特有的现象叫"竞品忽视":过度专注于自己的愿景和执行,以至于系统性地低估了同一空间里别人在做什么。好在 AI 提供了解药:让 Claude 给出"为什么这个空间里的某个竞品会成功、而你不会"的最有力论证。Claude 能分析:为什么他们的路子其实更好、为什么客户会选他们、为什么你那些自以为是的差异点也许并不像你想的那么有护城河。

> **练习**
>
> 让 Claude 按层级绘制你的竞争格局:直接竞品、间接竞品、潜在收购方,以及可能切入你领域的相邻玩家。然后让它论证每一层为何对你的成功构成真实威胁——是真实的威胁,而不是最容易被你打发掉的那个版本。

市场调研。Claude Code 能综合公开的客户反馈,浮现反复出现的抱怨和未被满足的需求。附带好处:这本质上是对你竞争对手的客户做了一次免费的定性调研。

> **练习**
>
> 让 Claude Cowork 综合你各关键来源上的竞品评价,找出现有方案尚未解决的头部抱怨。如果你的假设解决了其中一个或多个,这就是问题–方案契合的有力证据;如果没有,这同样值得知道。

Claude Cowork 也能从密集的行业报告、分析师备案文件和市场研究文档中,提取相关信息与数据;这些干净、综合过的输入,正是 Claude 后续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,并反复检验其背后的假设。判断市场是在扩张、整合还是已经成熟——这会影响你对时机和差异化的思考。同时绘制买方格局: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。用 Claude 倾听早期指标,判断你是否在正确的时机进入。追踪那些已经在讨论你这个问题的 subreddit 和 LinkedIn 群组,以及用户描述自身困扰时所用的确切措辞。让 Claude 找出"类似问题曾被解决"的可类比市场,提取其中行得通和行不通的部分。浮现可能加速或威胁这个机会的监管、技术或人口趋势。

> **练习**
>
> 让 Claude 找出未来两年可能显著影响你市场的三个外部趋势(监管、技术或人口),并评估每一个对你的具体假设而言,是顺风还是逆风。

注:本节的市场调研与竞争绘制不是一次性练习。你会在 MVP 和发布阶段持续有新发现、不断演进思考,所以每当你的假设演变时,都要重做这些练习。

规划并设计客户发现

你从与潜在用户对话中学到的东西,其质量取决于:(1) 你提问的质量,(2) 你是否在问对的人。Claude 在客户发现上尤其有用,包括该和谁谈、问什么、以及如何理解你听到的内容。

该和谁谈。一份精准的目标画像,远比一长串联系人名单有价值,它包含最可能切身感受这个问题的具体职位、公司类型、团队结构和资历层级。接着,找出这些人真正可触达的地方——他们聚集的社群、活动、LinkedIn 群组和 Slack 工作区——并根据"离问题有多近"建立一个"先联系谁"的优先级框架。

问什么。目标确定后,用 Claude 来搭建访谈框架本身:正确的问题、正确的顺序,且要能揭示人们"实际做了什么",而非"他们觉得自己会做什么"。新手创始人常犯的错,是问一个泛泛的、面向未来的开放问题("你会用这样的东西吗?"),而不是具体地追问相关的过去("跟我讲讲你上一次处理这个问题的经历")。Claude 还能标出你草拟的问题中哪些是诱导性的、太宽泛的,或可能产生噪音而非信号的;它也能帮你设计追问,挖出对方回避带过的地方,或就重要问题上含糊的回答继续深挖。如果你的假设涉及不止一种人群画像,Claude 还能为每一种设计不同的问题集。财务经理和 CFO 对同一个问题有着不同的关系,而单一的访谈框架会抹平这种区别。

> **练习**
>
> 先自己手写访谈问题,再让 Claude 审核。专门让它标出任何诱导性的、面向未来的、太宽泛的,或更可能引出"社交期望式"而非诚实回答的问题。然后让它针对访谈中对方最可能含糊带过、避而不答的两三个时刻,各建议一个追问。

访谈后分析。每次对话后,用 Claude 做复盘:把你的笔记喂给它,让它指出哪些印证了你的假设、哪些挑战了它、哪些是真正出乎意料的。攒够一批访谈后,把全部访谈笔记跑一遍 Claude Cowork,浮现反复出现的主题、矛盾之处,以及两个方向上最强的信号。然后把综合后的输出再交回 Claude,让它指出:你自己对数据的解读,哪里可能是在"往你想听的方向"做模式匹配,而非数据真正所呈现的。

> **练习**
>
> 每做完五次访谈,就让 Claude Cowork 综合你的笔记,产出两份清单:支持你假设的证据,以及挑战你假设的证据。如果第一份明显比第二份长,就问 Claude:这种不对称是数据里真实存在的,还是你期望看到的?

客户触达与排期。用 Claude Cowork 自动化建联系人名单、做触达、约访谈这些运营杂活。Claude Cowork 能用你和 Claude 一起定义的目标画像(含职位、公司类型、资历层级)去调研并整理出一份结构化的潜在客户名单和经过核实的联系方式;接着批量起草个性化触达邮件,逐封贴合对方的角色和情境。随着回复进来,它通过 MCP 连接 Gmail 和 Google Calendar 来管理邮件线程、处理排期请求、把访谈排进日历。工作流还会继续:Claude Cowork 按设定节奏生成跟进草稿(比如对七天未回复的联系人发第 7 天跟进),并在每一步完成时更新你的追踪表,让你随时知道每个潜在客户处在管线的哪个环节。

> **练习**
>
> 把你验证过的访谈目标画像交给 Claude Cowork,让它构建潜在客户名单、起草一套个性化触达序列,并建立一张含"触达状态、跟进节奏、访谈完成"等列的追踪表。然后让它去跑这些协调工作,你专注于为对话本身做准备。

设计你最终的方案概念

你已经做完了验证工作:问题真实、你知道谁有这个问题、并且有一个被证据支持的方案概念。用 Claude 从各个角度来发展并挑战你的方案概念:有哪些缺口?有哪些替代方案?要让这个方案在规模上成立,什么必须为真?这是一个重要的"现实检查点":这个设计真的解决了验证过程揭示出来的那个问题,而不是你一开始假设的那个问题吗?

> **练习**
>
> 把你的方案概念呈给 Claude,让它找出你的设计最依赖的三个假设。然后问:每个假设要成立,什么必须为真;如果其中任意一个不成立,后果是什么。

用 Claude Code 搭一个轻量原型

现在到了有意思的部分:有了经过验证的假设和反复推敲过的方案概念,你终于可以动手造点东西了。这正是创意阶段里 Claude Code 登场的时刻。即便你一路上一直在折腾,现在才是你生成"正式轻量原型"的节点:做出能摆到真人面前、获得真实反应的最小那一块就够了。

你(还)不是在造一个真实产品;你是在构建一个能用的"想法样品",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做出的反应,会告诉你十几次问题–方案发现访谈都问不出来的东西。之前,你在确立"你要解决的问题是真实的";现在,你是在请潜在用户去与提出的方案互动。

> **练习**
>
> 定义你的方案所依赖的那个唯一的核心交互。让 Claude Code 只构建那一个。做出来后,把它摆到五个来自你验证过的目标画像的人面前,请他们试用。你在这五次对话里学到的,将决定你是继续构建,还是回到画板重来。

走到创意阶段的终点,是这场 AI 创业竞赛中的一大跃进,因为现在你不再是押注一个直觉,而是在依据证据执行。接下来是 MVP 阶段,创始人的指导性问题从"这值得构建吗?"变成"我们究竟该先构建什么?",而 AI 的主要角色也从调研伙伴,转向了施工队。

第 4 章

MVP 阶段

很多创始人把 MVP 阶段当作一个施工阶段,但 MVP 阶段本质上仍是一次"证据收集"练习。区别在于,你现在收集的是关于"方案"的证据,而非"问题空间"的证据;具体说,就是一群真实、可识别的人,是否觉得它有价值到愿意使用、回访、付费,和/或推荐给别人。

MVP 阶段目标

作为 AI 原生创业公司的创始人,你的目标是把一个经过验证的问题,转化为真实用户真正会用的可工作产品。这不是带着路线图上每个功能的完整版本,而是你想法中最小、最聚焦的那次迭代:把一个真实方案摆到真实用户面前,并产出关于"产品–市场契合"(PMF)的真实证据。

与此同时,你现在怎么构建,决定了之后什么是可能的。这意味着 MVP 阶段还有第二个同等重要的目标:在不积累那种"会复利、并会在真实用户大量涌入时反噬你"的技术债的前提下,快速推进。

最后,从第一天起就积累"持久的上下文",才能让 AI 一直是帮你放大产出的助力,而不是越帮越乱、不断添乱的源头。在 AI 原生创业公司里,你的代码库是你一次次会话中与 AI 协作的对象,这让"可读性"成为根基。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,会撞上一堵可预见的墙:每次新会话都要重新解释一遍代码库,而 AI 生成的改动也会逐渐偏离最初的愿景。

MVP 阶段退出标准

MVP 阶段的退出条件,是产品–市场契合的真实证据:证明一群特定、可识别的用户,觉得这个产品有价值到愿意回访(留存)、付费(收入)或推荐给别人(转介)。

MVP 阶段的挑战

在 MVP 阶段,创始人的首要指令是速度与判断力。这里的挑战集中在:你能否足够快地、以正确的方式构建正确的东西,而不去抄那些日后会让你付出代价的近路。

智能体式技术债

挑战:因为 AI 实质上移除了过去控制"什么能进生产环境"的每一个自然瓶颈,速度有了保证。但当速度成为创始人在 MVP 构建中唯一考量的变量时,他们就有可能积累难以偿还的技术债。

在 MVP 阶段,一些技术债是恰当的,前提是明白它必须在规模化之前被管理掉。它会逐渐累积,可以随时间或在一个专门冲刺中清理。但 AI 技术债会复利。如果没有把规格和架构约束写在 AI 能读到的某处,每次会话都会从零重新推导基础决策,而这些决策会漂移。你最终得到一个"背后没有连贯心智模型"的代码库——不是因为任何单块代码不好,而是因为这些碎片从一开始就不是为彼此契合而设计的。这是个真问题,而且往往很晚才浮现。

掉进"虚假的产品–市场契合"

挑战:AI 工具能制造出亮眼的早期数字,但这些并不保证市场需要你的产品。

早期势头是创始人能体验到的最具心理冲击力的事情之一。在数周乃至数月的验证工作和审慎构建之后,交付产品的那一刻,会让人觉得"我从一开始就是对的"得到了印证。智能体式编码工具能帮你比以往任何时候都更快地抵达这一刻,但早期的增长势头并不等于产品–市场契合。发布时的能量来自一些转瞬即逝的力量,比如创始人的朋友、你投资人旗下其他被投公司里的潜在买家,或一个把流量推上去的 Hacker News 标题。遗憾的是,这些都无法可靠地预测第 6 周或第 12 周、当初始助推褪去后会发生什么。

毫无阻力的范围膨胀(想加就能加,收不住)

挑战:当构建感觉毫不费力、几乎免费,总会有"再加一个酷功能"或"再处理一个边界情况"。这种范围蔓延弊大于利。

需求/功能的无限膨胀一直是创业风险。区别在于,当加一个功能从一个冲刺变成一个下午,过去逼着你克制的那个约束——工程时间的真实成本——如今基本不复存在了。这里的挑战是:每一处单独的增项都有道理。产品当然该处理那个边界情况;用户当然会想要那个工作流。在当下,它们都不像"功能失控",因为用智能体式编码每一个都只要那么一点点功夫;但随着你的产品越过原本的边界不断膨胀,你会有失去方向和势头的风险。解药是一份在动手构建之前就写好的范围定义,描述产品做什么、刻意不做什么,以及"什么样的真实用户证据才足以证明应当新增某项功能"。这把决策点从"我们该不该构建这个?"挪到了"是否已有相当规模的用户告诉我们,没有这个功能他们就无法从产品中获得价值?"

因经验不足而不安全

挑战:创始人用 AI 工具把应用赶上市,却没先理解基本的安全原则,最终把用户暴露在本可预防的风险之下。

残酷的事实是:智能体式编码工具生成的是"能工作"的代码,不是"本质安全"的代码。功能性代码很容易判断,因为要么这个功能能用、要么不能。而安全漏洞在被利用之前是隐形的,这意味着没有一个自然的反馈回路来提醒首次创业者"哪里出了问题"。然而,把一个真实的 MVP 交付给真实用户,意味着真实的数据、真实的暴露面,以及一旦出事真实的后果。轻视安全并非 AI 原生项目独有。每个时代的自筹资金创业公司,常常把安全考量拖到构建很晚才做,有时一直拖到接近生产上线。在任何用户接触你的应用或方案之前做一次安全审查,是把一个最小可行产品放进真实世界的最低负责任门槛。

Claude 如何帮助 MVP 阶段的创始人

在构建之前先定义你的架构

在 Claude Code 写下任何一行生产代码之前,用 Claude 来定义并记录那些将统领本阶段所有构建的架构决策:要遵循的模式、要避免的依赖、正在做出的权衡及其原因。这份输出会成为一份聚焦的架构上下文文档,并确立 Claude Code 将在其中运作的护栏。

没有这份上下文,每次会话都从零开始,Claude Code 被迫推断出它自己的一套结构性假设。让 Claude Code 在没有护栏的情况下构建,会产出一个"能跑但结构不连贯"的代码库,而在不连贯的代码库上迭代和扩展,最终是对时间和 token 的浪费。迟早会有一刻代码不可避免地崩塌,迫使你从头重建。

> **练习**
>
> 打开 Claude Code 之前,先打开 Claude,描述你在构建什么:它解决的核心问题、它服务的用户,以及你在未来六个月内现实预期的规模。让它帮你定义应当统领 MVP 构建的架构原则、在你约束下要避免的依赖,以及你在这个阶段有意识接受的权衡。
>
> 接着,把这份输出存为 CLAUDE.md 文件。这是你的架构上下文文档:你构建的第一件产物,也是后续每次会话都依赖的那一件。CLAUDE.md 文件作为项目级指令,提供项目专属的上下文与说明,Agent SDK 在某目录中运行时会自动读取。从功能上说,它们是你项目的持久"记忆"。

定义并强制执行你的 MVP 范围

这种几乎不费力气、想加就加的功能膨胀,是 AI 时代 MVP 的标志性失败模式之一。就像你定义并记录了产品的应用架构一样,你也需要在第一个功能被构建之前,先定义好 MVP 的范围。Claude 能帮你创建一份范围文档,描述:你的 MVP 产品做什么、刻意不做什么,以及功能修订标准——什么样的真实用户证据,才足以证明此刻应该新增某项功能。当新的功能想法冒出来时(它们一定会冒出来),你用 Claude 去拷问它:这究竟是来自用户的真实信号,还是披着产品思维外衣的创始人热情。

用 Claude Code 构建你的 MVP

架构和范围定义好之后,Claude Code 就成为主要的 MVP 构建工具。用它来生成、测试、调试、迭代你的代码库,但要把每次会话当作"执行你已经做出的产品决策",而不是"顺手塞进一些新决策的机会"。每次 Claude Code 会话开始时:(1) 重温你的范围文档,(2) 把你的 CLAUDE.md 架构上下文文档提供给模型。每次会话结束时,用本次会话浮现的任何决策来更新它。目标是一个"你能解释其结构"的代码库,而不仅仅是一个"能跑"的代码库。

> **练习**
>
> 为你的 Claude Code 工作创建一个简单的会话模板,包含架构上下文文档、本次会话的具体任务,以及要遵循的任何约束或模式。每次会话结束时,在上下文文档里加一条简短日志:构建了什么、做了哪些决策、本次会话引入了哪些假设。每次会话花五分钟做文档,是对抗"复利成不可管理代码库"的架构漂移的廉价保险。

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,你的责任是:知道代码库里有什么、理解任何潜在的暴露途径,并且不把明显的漏洞交付给那些信任你、把数据交给你的真实用户。Claude 能对 AI 生成的代码做一次有用的初步安全审查,帮助识别常见漏洞。把它纳入"交付前"的循环是个好习惯。但它不是安全工具的替代品,在

更高风险下,也不是人类审查者的替代品——把它当作替代品的创始人,正是会出现在数据泄露报道里的那些人。

Claude Code Security 走得更远:它扫描代码库中的安全漏洞,并给出供人类审查的针对性补丁,浮现传统方法可能遗漏的问题。

> **注:**本手册发布时,Claude Code Security 还是有限的 beta 版本,所以把它纳入工作流之前,请先确认当前可用性。

> **练习**
>
> 在部署给任何真实用户之前,把你的核心应用代码喂给 Claude,并给出明确指令:审查认证与会话处理、API 响应中的数据暴露、输入校验与注入风险,以及有已知漏洞的依赖。认真对待每一项发现,评估它是否需要修复,凡涉及认证、密钥或数据处理的,都要有人类审查。

在发布前建好你的度量框架

那些把早期的增长势头误认为产品–市场契合的创始人,通常也是那些"发布后才开始追踪数据"的人——而且选的指标是为了评估"什么在起作用",而非浮现"什么没在起作用"。解药是在第一个用户出现之前,就确立你的度量框架。用 Claude 定义:对你这个特定产品而言哪些指标重要、基准是什么、数据里什么样的模式才构成真正的产品–市场契合(而非讨人喜欢的噪音)。具体来说:在发布 MVP 之前,就设好你的留存基准、激活标准,以及第 7 天和第 30 天的目标。接着,为你这个特定产品定义什么是"虚假的好兆头":有注册无激活、有收入无留存,或有初始热情却无重复使用,等等。数据进来后,让 Claude 对你自己的增长数据唱反调:一个怀疑者会怎么看这些数字?

管理客户发现与用户反馈的事务性工作

真实用户进入产品后,运营层会迅速膨胀。Claude Cowork 处理那些"重要但繁琐"的工作:建立并维护用户联系人名单、跑触达序列、安排反馈会议、分诊 bug 报告、追踪迭代周期。创意阶段管理客户发现事务的那套 MCP 集成,在这里同样适用。处理那些需要细致解读的用户反馈时,一定要让真人留在这个环节里,别全交给 AI。比如用户说"这很棒,不过我希望它还能……",这就需要解读:这是核心需求还是锦上添花?是这个客户特有的,还是代表了一个细分群体?缺的那个功能才是真问题,还是上游 onboarding 里出了别的问题?没有工具能回答这些。

> **练习**
>
> 配置 Claude Cowork 来跑你 MVP 阶段的反馈回路:向早期用户名单起草触达、安排反馈会议、为 bug 报告和功能请求设计结构化的收集流程,并每周写一份"进来了什么"的综合摘要。先自己审阅这份摘要;之后,你可以让 Claude 分析这些信息,捕捉你可能漏掉的重要点。

朝"证据"迭代,而非朝"完整度"迭代

MVP 阶段在你拿到产品–市场契合的真实证据时结束,无论产品感觉上有多"完成"。宣布你已达成 PMF、可以从 MVP 阶段进入发布阶段,终究是一次结合创始人直觉与所收集证据的判断练习。不过有一些有用的试金石:

- **Sean Ellis 测试:**问你的活跃用户:"如果你再也不能用这个产品,你会作何感受?"如果超过 40% 回答"非常失望",这是一个有意义的 PMF 信号。
- **努力程度测试:**在 PMF 之前,留存需要持续干预,包括频繁触达、激励、个人跟进,以及创始人为维持用户活跃而拼命付出的精力。在 PMF 之后,产品开始自己做这些工作。当事情开始"拉"而不是"推"时,这种努力程度的转变,是"某种真实的东西已经改变"最清晰的信号之一。

归根结底,没有任何单一数据点能确认产品–市场契合,因为它是一种"必须在多个迭代周期里持续成立"的模式,然后你才能明确地这样说。

当证据要求时,就转向(pivot)

如果投入了所有这些工作,你却始终到不了产品–市场契合,怎么办?你的结果没有印证你最初的方向,这不是失败,而是系统在起作用:MVP 阶段的设计初衷,正是在你对错误答案过度投入之前,把这个信息浮现出来。当数据不支持你当前的产品时,用 Claude 去厘清数据在告诉你什么。

- **探索其他客户细分**。也许那些没转化的用户,从一开始就不是对的目标。正确的受众往往已经在你的数据里,只是被低估了。
- **调整你产品的价值主张**。也许你有对的受众,但你的 MVP 没能与用户共鸣。对 onboarding、信息传达或核心功能侧重的调整,有可能在不改变你已构建之物的前提下解决这个问题。

同时也要对一种可能性保持开放:这种脱节,深到需要一次更根本的改变。

> **练习**
>
> 如果你已完成三轮或更多迭代周期,却没有朝 PMF 基准有意义地移动,在决定下一步之前,用 Claude 做一次诊断。把你的留存数据、用户反馈和最初的问题假设喂给它,问它三个问题:
> · 数据里是否有某个细分群体的反应和其余人不同?
> · "设计的价值"与"体验到的价值"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的 PMF,什么必须为真;鉴于你所看到的,这个情景现实吗?
> 让答案来决定你是调整、转向,还是回到创意阶段。

第 5 章

发布阶段(Launch)

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,创始人必须把早期的增长势头,变成一台可重复、可持续的增长引擎。除了让产品达到生产就绪,你还必须加固其下的基础设施,同时围绕产品真正建立起一家公司。

在创意和 MVP 阶段,创业公司天然以创始人为中心,因为你需要对全局了如指掌,以及紧密的反馈回路。但到了现在,仍试图亲自抓住每一根线头的创始人,会成为发布阶段的瓶颈。目标不是把自己从公司里抽离,而是建立运营系统,把你的注意力释放出来,留给那些只有创始人才能做的决策。

发布阶段退出标准

发布阶段的退出条件有三个要素:

1. **增长可重复、由渠道驱动**。你不只是在留住用户,而是在通过特定渠道、以可理解的单位经济模型可预测地获取他们:CAC、LTV 和回收周期是你已知且能解释清楚的数字。
2. **产品能扛住生产负载**。基础设施已加固,安全与合规就位,可靠性在真实生产条件下(而不仅是你测试过的条件)依然成立。
3. **运营不依赖创始人这个瓶颈**。流程已存在,自动化已就位。你不再是那个亲自处理支持、分诊、冲刺规划或报表的人。

发布阶段的挑战

找到产品–市场契合,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"保住它"。发布阶段,是那些产品已获得真实增长的公司,仍可能因为"围绕并支撑产品的组织跟不上"而分崩离析的地方。以下是要警惕的失败模式。

技术债到期

挑战:那个为速度和验证而建的 MVP 代码库,跑得足够好、证明了产品行得通;但生产流量、新功能和不断增长的复杂度,正在暴露当初抄的近路。

在 MVP 阶段,为了速度积累一些技术债是合理的权衡。到了发布阶段,这笔债开始计息,拖得越久、修起来越贵。解法包括:一次系统性的架构审计以识别结构性弱点、针对最严重问题的定向重构,以及有意义地扩展测试覆盖,好让下一轮功能开发不会重新引入同样的问题。

创始人成为瓶颈

挑战:在 MVP 阶段,创始人身处每个回路是一种资产;到了发布阶段,随着支持量增长、产品决策堆积、运营复杂度倍增,同样的本能变成了约束。

从"亲自做工作"到"设计做工作的系统",是创业生命周期里最难的转变之一。因为它很少有一个清晰的发生时刻,风险在于你完全错过它,继续停留在"构建者模式",而组织在你周围停滞。这种情况正在发生的征兆包括:本该一小时就能做的决策,现在要等你一周才轮得上;支持请求堆积,因为只有你知道答案;运营任务只在你亲自想起时才会发生。补救之道,是对你亲自处理的一切——从最小的任务到风险最高的决策——做一次彻底审计,以识别什么可以系统化、什么可以委派、什么确实仍值得创始人投入时间和注意力。

安全与合规不再可以拖延

挑战:在 MVP 阶段把安全与合规保持简单是没问题的;但现在,有了真实用户、真实数据,以及桌面上可能出现的企业合同,它就成了一项负债。

在 MVP 阶段,只有少数 beta 用户、生产环境里没有敏感数据,安全漏洞还是理论风险。然而,一旦你的产品带着依赖它的真实用户进入生产,这个假设就变成了非常真实的暴露风险。此外,那些不适用于原型的合规要求,在你开始处理客户数据、处理支付,或销售进入受监管行业的那一刻,就一定适用了。补救之道,是在生产规模到来之前(而非之后)做一次系统性的安全与合规审查,并把浮现出来的一切都当作"必须修复的整改项"——而不是"建议"——在下一波用户到来之前完成。

在准备好之前就扩张

挑战:新市场和新融资机会看起来像增长机会,它们也可能是产品–市场契合的葬身之地。

你已建立的初始增长势头是真实的,但它也特定于你的早期受众。过早扩张进一个与你最初市场显著不同的市场,会引入新的用户行为、合规要求、支付基础设施和基线预期,而这些都不是你产品当初设计时考虑的。突然之间,新变量太多,你失去了清晰解读自己数据的能力。你还会面临在追逐一个新的、未经验证的受众时,忽视原有用户群的风险。

Claude 如何帮助发布阶段的创始人

在发布阶段,Claude 的三种形态全部投入使用,并相互支撑:每个工具产出的成果,都成为另外两个的输入。结果有机地复利,一个同时使用三个工具的创始人,得到的远不止三者之和。这正是"超精益创业模型"在结构上得以成立的原因。当 Claude Code 构建产品、Claude Cowork 围绕它构建公司、Claude 帮助把这些产品与组织上的知识沉淀进日常运营时,一个小团队就能像一家规模大它好几倍的公司那样运转。

在技术债复利之前整改它

你的 MVP 代码库能跑,但它也需要一次系统性的整改,排查任何可能变成结构性负债的技术债。首先,用 Claude Code 跑一次完整的架构审计:找出代码库哪里脆弱、哪些近路日后维护起来会很贵、哪里

测试覆盖薄到下一轮功能开发会重新引入同样的问题。把 Claude Code 的审计发现交回 Claude,做分诊和排序:什么必须在下次发布前修、什么可以等一个冲刺、什么属于在你当前阶段可接受的长期债务。这也是把你 MVP 阶段做出的架构决策(那些因为没时间写下来、只活在你脑子里的)记录下来的时刻。现在把它们写进 CLAUDE.md,确保未来每次 Claude Code 会话都从"对系统如何设计、为何如此设计"的共同理解出发。

> **练习**
>
> 让 Claude Code 审计你的 MVP 代码库,产出一份按优先级排序的清单:结构性弱点、测试覆盖缺口、重构候选项。然后把这份清单交给 Claude,让它把整改工作排进你接下来的若干冲刺:哪些重大问题必须先处理、哪些可与功能开发并行、哪些可以等。

构建替代创始人注意力的系统

要建立能"释放你注意力、去处理只有创始人能做的责任"的运营系统,前提是确切知道你的注意力流向了哪里。用 Claude Cowork 对你当前的运营负载做一次结构化审计,记录每一项重复任务、每一个落到你桌上的决策、每一个只因为你亲自记得才会发生的工作流。然后让 Claude Cowork 把这份清单分类为:可以完全自动化的、需要一个人但不一定是你的、真正需要创始人判断的。审计完成后,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、完成后去往何处。

把安全与合规变成一条产品工作线

用 Claude Code 浮现那些在 SOC 2、GDPR 或 HIPAA 审计中经常出现的代码级问题,以及你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,帮你为整改工作排优先级,并设计企业买家签约前会要求的控制项、审计日志和访问管理。然后,把合规工作线建进你的开发周期,而不是把它当作一次性项目来跑;合规文档需要持续维护和更新。对接近企业合同或国际市场的创始人,这也是 Claude Code 安全扫描能帮你为独立安全评估做准备的时刻。

> **注:**AI 扫描是辅助,而非合格合规审查的替代品。

> **练习**
>
> 用 Claude Code 跑一次面向你目标市场所需框架的代码级安全审查。把输出交给 Claude,让它产出两样东西:一份按优先级排序的安全整改顺序,以及一份你需要产出的文档与控制项清单,以满足潜在企业买家的合规审查。

把你一直在跳过的产品管理流程搭起来

发布阶段需要一套轻量、可重复、无需创始人介入即可触发或运转的流程。用 Claude 设计:你的产品时间线和工作周期如何组织、在 Claude Code 动一个功能之前一份规格需要包含什么、bug 报告如何分诊和路由、你的每周指标报告涵盖什么以及如何分发。流程设计完成后,用 Claude Cowork 构建并

运行运营层:安排冲刺仪式、把进来的 bug 报告路由到正确的地方、从你接入的数据源编译每周指标,并维护把用户信号持续输入产品决策的反馈回路。

> **练习**
>
> 让 Claude 设计一套轻量的产品管理操作系统:定义好的冲刺节奏、一份最小规格模板、一棵 bug 分诊决策树,以及一份从你真实数据源拉取的每周指标简报。然后设置 Claude Cowork 去实现并运行该系统中的重复性运营环节(排期、路由、报告编译),让它们按计划自动发生,无需你。

第 6 章

规模化阶段(Scale)

在规模化阶段,创始人的角色从构建者,重新定位为面向公众的高管。产品依然核心,但你个人的日常工作越来越关于公司本身。即便你努力维持精益、以 AI 为中心的结构性优势,你的注意力也必须扩展到分析师沟通会、IPO 路演等规模化阶段的新活动上。

规模化阶段目标

扩展技术基础设施的工作仍在继续,如今又加上了扩展组织本身、并把它成熟为一门生意的工作。在规模化阶段,你面对的是从数千用户到数百万、从一个市场到多个市场。在此前的每个阶段,增长是你可以"凭感觉摸索"的:靠贴近用户、靠紧密反馈回路里的数据再加一份健康的创始人直觉来调整航向。但现在,目标是构建由成熟组织运营所支撑的系统化增长。

对一家 AI 原生创业公司,你的目标应当是通过"积累的深度"构建一道可防御的护城河,它来自你注入产品的专长、你产品与用户所依赖的其他工具和平台的集成深度,以及专有的系统数据和工作流。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,公开市场投资者、分析师、监管者、企业采购团队和收购方,会施加更大的压力——以及更大的怀疑——因为现在赌注更高了。你的产品和组织必须经得起外部审视:不只是你所构建之物的能力,还有围绕它的治理、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:即便创始人越来越不直接运营日常,公司依然可持续。你已展示出系统化增长;已构建出能满足最严苛外部审查者的组织治理与合规基础设施;并且对"如果一个资金雄厚的在位者今天复制你的产品,你的用户会留下吗?"有了扎实的答案。实践中,这个阈值通常表现为三种形式之一:不再需要外部资本的、规模化的可持续盈利;IPO 就绪;或被收购。三者都要求你的增长系统化且可审计、你的产品护城河经得起审视、你的组织在运营上成熟且可持续。当这一切为真,值得恭喜:你的创业公司已从一场赌注,变成了一门生意。

规模化阶段的挑战

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,而无需被人盯着。对一个从第一天起就亲力亲为的创始人来说,这个转变既是结构性的,也是心理上的。

你在发布阶段的工作是创建这些系统;在规模化阶段,它变成:(1) 让这些系统成熟到完全可信赖,(2) 然后真正去信赖它们。这比听起来更难。即便你是个善于委派的创始人,也不总是显而易见该交出什么、该留在自己手里什么。交出太多太快(尤其是交给 AI 自动化系统),关键决策就会在缺少只有创始人能提供

的关键上下文的情况下被做出;而抓得太久,你又会成为瓶颈。这里的根本挑战,是识别那些只存在于创始人脑中、或藏在没写下来的工作流里的隐性经验,并把它编码进有文档、可审计、可转移的系统。

扩展技术运营

挑战:客户不再只评估你的产品;他们想知道你的组织能否成为一个可依赖的基础设施伙伴。

前三个创业阶段的技术挑战集中在代码库上:在不积累技术债的前提下构建正确的方案,然后为真实用户加固安全与合规。到了规模化阶段,挑战变成了"围绕代码库的一切":创建出能彰显成熟度的支持基础设施、文档和可靠性保证。更大规模的客户和签多年合同的机构买家,会在签约前要这些,签约后也会照此要求你。不过,把你带到这一步的那套 AI 基础设施,也能帮你构建有明确响应时间的专门支持职能,以及一个新客户的工程团队真正用得上的文档。

扩展组织职能

挑战:无论由多少人运营,一家规模化阶段的公司通常都需要招聘、工资、会计、法务运营等组织基础设施。

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,现在需要发展一套更广、在某些方面也更重大的运营职能,例如财务报告、合规监控、合同管理、客户支持等。

构建 GTM 职能

挑战:有机增长有天花板,而大多数规模化阶段的创始人,在从未建过真正的 go-to-market 职能之前就撞上了它。

创意、MVP 和发布阶段的增长,往往来自创始人主导的销售,从一次时机恰好的 Product Hunt 发帖,到与早期客户的私人关系。但这样的有机增长只在某个点之前有效,大多数创业公司在规模化阶段触及这个上限。征兆包括:用户曲线趋平、获客成本上升,以及一条只有创始人亲自介入才会推进的管线。规模化阶段的增长,需要构建一台专门的增长引擎,去触达更新、更广的受众。然而大多数创始人很可能从未运营过营销、销售、分析师关系等项目。一套正经的 GTM 动作,不仅需要建立新的系统和流程,还需要为"你想如何谈论你的产品"创造一种品牌声音和故事。因为在创业生命周期的这个阶段,你需要它来触达的不只是单个新用户,还有投资者和企业买家这样的整个目标受众。好在,GTM 职能不必很大也能有效,而构建了产品的那套 AI 基础设施,也能引导它走向市场。

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:验证想法的调研伙伴、设计并构建原型的工程团队、让单人创业公司得以成立的 AI 运营层。抵达规模化阶段的创始人,现在可以用同样的方式,继续用 Claude、Claude Code 和 Claude Cowork 来扩展规模。

把日常任务交给 Claude Cowork

带着对"此刻你最需要把时间和注意力投到哪里"的清醒认识,开始规模化阶段——这对从未建过公司的首次创始人是个挑战。Claude 能帮你列出"这个阶段只有你该做的事",可能包括产品叙事决策、董事会关系、企业大单、创始人之间的对话。任何不在这份清单上的,都是委派或 Claude Cowork 自动化的候选项。

> **练习**
>
> 用 Claude 为你当前的运营层画一张"瓶颈地图":每一个经由你的工作流、决策和审批。然后让 Claude 推演:当你一周无法处理时,每一个会怎样。那些会停滞的工作流,正是你仍亲手介入、足以拖慢进度的地方。它们如何对应你和 Claude 一起做的那份创始人优先事项与责任清单?

接下来,该反复检验你已构建的系统,是否真的准备好随业务增长而扩展。

> **练习**
>
> 用 Claude 梳理你当前的工作流,然后问它:当你一周不在时,每一个会怎样。那些停滞的工作流,正是交接标准、升级路径或异常处理仍需收紧的地方。Claude 能帮你分析失败点并建议合适的修复,以便你按需更新或替换 Claude Cowork 自动化。

把技术运营扩展成企业级基础设施

随着你扩展,买家需要确信你的产品和组织可以作为长期基础设施被信赖。技术工作一如既往在代码库内部进行,但现在也有围绕代码库的技术工作要处理。第一步是把组织里沉淀的隐性经验,转化为一个可扩展的系统。用 Claude 起草并维护企业采购期望看到的成文基础设施,包括产品文档、支持手册和 SLA。同时,指挥 Claude Code 针对企业合同要求的特定可靠性和安全标准来审计并加固代码库,并搭建那些基于 Discord 的社区支持从未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 来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、续约追踪,以及企业客户成功所依赖的报告节奏。三者合起来,让一个小团队拥有一家大得多组织的支持姿态——而这正是签下一份多年企业合同所要求你展示的。

> **练习**
>
> 挑出你最难搞的三个潜在客户,或确定三个你梦寐以求想签下的理想客户。让 Claude 产出一份差距分析:这些客户的企业采购团队在签多年合同前,期望看到什么样的文档、SLA 和支持基础设施,而你目前在哪里不足?用这份输出,把跨 Claude Code 和 Claude Cowork 的技术与文档工作排好序。

构建一个真正的 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一套真正的 go-to-market 策略。AI 能帮你构建、然后运行这台完整的 GTM 引擎。Claude 能协助从零构建 GTM 的基础资源:市场细分、信息传达体系、分析师关系策略、销售手册,以及当你开始与公开市场投资者、企业买家和华尔街分析师对话时才重要的、面向投资者的指标叙事。每一类受众都有自己的词汇、以自己的标准评判你;Claude 的工作是把你产品的价值主张,翻译成对每个受众细分都贴切的产品营销方式。然后,Claude Cowork 可以成为你的战术执行层:内容管线、外联序列、分析师简报的事务安排、新闻发布与 PR 节奏、CRM 数据维护、销售管线报告,以及把 GTM 策略变成真实商业动作的众多重复周期。在 GTM 动作需要产品营销基础设施的地方——交互式演示环境、集成文档、沙箱租户、API 参考、技术单页——Claude Code 可以为你构建。买家期望在技术上评估你的产品,而在规模化阶段,一个 Loom 视频和一份销售稿已不再够用。这也是让你的 GTM 动作得以异步运行的基础设施:一个搭得好的演示环境,会在你开董事会时帮你成交。

把领域专长和组织经验变成 AI 上下文

许多超精益创业公司的创始人,是在为他们亲身经历或一手观察到的某个真实行业问题,构建高度具体的应用或工具。智能体式 AI 如今让从没写过一行代码的创始人,也能用他们的领域专长去构建解决复杂问题的产品。Claude、Claude Code 和 Claude Cowork 各自在"把创始人知识转化为不断复利的产品独特性"上扮演一个角色。

用 Claude 来捕捉、组织和提炼创始人知识,把领域专长放到产品能够触及的地方。通过延展的对话、项目和记忆,创始人可以把自己所知的一切——行业术语、监管陷阱、边界情况、痛点、为什么对这个问题显而易见的答案行不通——汇入一个结构化、可搜索的上下文。技能(Skills)随后能把反复出现的工作流(例如"我如何审一份商业租约""我如何分诊一份病人接诊表")编码成可复用的例程,让 Claude 每次都以同样方式运行。数月之后,这就成了一个任何通用 AI 都无法匹敌的专有知识底层。借 Claude 把你的领域知识沉淀下来,对于把行业特有的边界情况写进你的产品极其宝贵:一个通用的 AI 医疗账单工具会在 340B 药品计划的理赔上崩溃,而你的工具对此有专门逻辑。Claude Code 帮你把你所在领域其他专业人士常遇到的痛点,翻译成校验逻辑、提示词优化,或与一个你竞品闻所未闻的小众行业系统的 MCP 集成。结果是,你应用的深度和广度都在持续复利,而这是竞品根本无法复制的。

> **练习**
>
> 找出一个在你垂直领域里、通用竞品一定会搞错的边界情况。和 Claude Code 一起,基于一个你真实见过的场景,为它构建一个专门的测试用例(不是单元测试)。每次出现类似边界情况,就加进去。你的测试套件会变成你护城河的地图。

把累积的用户数据复利成可防御的优势

用户与你产品交互时,会产生行为信号(即他们接受了哪些输出、拒绝了哪些),这些信号反哺产品路线图。久而久之,你会摸清你特定用户群的具体模式、偏好和边界情况。这就是我们所说的复利价值:每一次改进都让产品更有用,从而带来更多使用、产生更多反馈、又驱动更多改进。这种数据是时间锁定、

情境特定、复制者无法重造的:你根本买不到成千上万用户在你产品里打磨工作流所形成的"行为指纹"。Claude 能帮你审计你已收集的任何用户交互数据,识别其中信号最强的行为模式,并设计把持续使用转化为系统化模型改进的反馈回路。

> **练习**
>
> 把你产品交互数据的摘要喂给 Claude:你一直在收集什么、收集了多久、你对"用户如何随时间使用产品"了解多少。让它识别这些数据里信号最强的三个行为模式,并为每一个设计一条把它变成系统化模型改进的反馈回路。然后让它帮你起草一页"护城河叙事"以指导产品营销:你的数据飞轮如何运转、它转了多久、为什么一个今天才起步的资源雄厚竞品在两年内都无法复制它。

制造工作流锁定

复利的数据网络效应让你的产品更难被复制,而用户的工作流锁定让你的产品更难被离开。用户在日常运营中运行你产品的时间越长,它就越深地嵌入他们实际的工作方式。他们在它之上搭了自动化、培训了人去用它、把它连到了自己的数据源和其他工具。他们开发的提示词、打磨的工作流、标准化的输出,全都围绕你产品做什么、怎么做而成型。到这一步,切换就从一个产品决策,变成了一个全面的运营工程。制造工作流锁定的第一步,是让 Claude 按集成深度给你当前客户群画像。对每个客户细分,识别他们在你产品之上搭了哪些工作流、依赖哪些集成。这能显示你的产品在哪里黏住了,以及哪里还需要扎得更深。你提供的集成越多,客户能在你产品上搭出各种依赖性工作流的空间就越大。Claude Code 帮你快速搭起与目标用户依赖的数据管线、项目管理工具及其他系统的原生集成。Claude Code 还能构建 API、webhook 和 SDK,让客户不只是用你的产品,还能在它之上构建——这是最深的锁定形式。

> **练习**
>
> 让 Claude 帮你为你的前十大客户做一次"工作流集成审计"。对每一个,记录他们搭建的自动化、依赖的集成、贯穿你产品的团队工作流,以及你对其切换成本的估计。然后让 Claude 找出这一组的共性:对你这个特定产品而言,哪类集成造就最深的锁定,以及你可以构建或开放什么,来为那些目前还停留在表面的客户加深集成。

第 7 章

工作没变,规则变了

在 AI 时代,创始人的工作并没有变:找到一个真实的问题、构建出能解决它的东西、并把它扩展成一家有分量的公司。变的是抵达终点的路径。在创意、MVP、发布、规模化这四个阶段中,AI 把数个季度压缩成数周。

过去要数月的验证周期,现在只要一个下午。一个能工作的原型,不再需要一个"技术栈正合适"的联合创始人;它只需要一个清晰的问题,和几次与编码智能体的专注会话。发布就绪,从发布前的手忙脚乱,压缩成一条持续的工作线。而在规模化阶段,那些过去逼着公司早早招人去救火的运营重担,如今越来越能交给 AI,把你团队的注意力释放出来,投到那些将成为你护城河的判断决策上。

瓶颈不再是你能构建什么,而是你选择构建什么。

附录

资源

用 Claude 构建

- **为创业公司构建 AI 智能体:**分享创业公司如何用智能体,随着规模化而降低对创始人的依赖。
- **Claude Code 文档:**带建设者从初次安装走到高级智能体工作流。小贴士:从"How Claude Code works"概览开始。
- **Claude Code 最佳实践:**涵盖在 Anthropic 内部及各工程团队中行之有效的模式——上下文管理、权限、规划与验证工作流。
- **使用 CLAUDE.md 文件:**讲解如何为你的特定代码库配置 Claude Code。MVP 阶段搭建开发环境的创始人必读。
- **Claude Code 高阶用户技巧:**来自 Claude Code 团队本身的工作流模式,包括并行会话和验证回路。
- **Claude Cowork 入门:**分享团队如何设置 Claude Cowork,并开始落地技能、插件等功能,把它的影响扩展到整个创业公司。
- **教程:**claude.com/resources/tutorials 提供一份可搜索的、针对具体任务的实操演练清单。

创始人故事

- **三家 YC 创业公司如何用 Claude Code 构建公司:**HumanLayer(F24)、Ambral(W25)、Vulcan Technologies(S25)如何用 Claude 快速把原型推向市场,并用智能体式编码工作流扩展 AI 驱动的平台。
- **GC AI:**创始人用领域专长,构建了一个响应迅速、由 Claude 驱动的法律平台,贴合内部团队的真实工作方式——公司特定的操作手册、跨职能利益相关方、可变的风险容忍阈值。
- **Carta Healthcare:**用 Claude 驱动其临床抽提平台,每年处理 22,000 例外科手术案例,把数据抽提时间减少 66%。
- **Anything:**由 Claude 和 Agent SDK 驱动,已帮助 150 万用户把想法变成可工作的软件产品而无需写代码,包括一位非技术创始人构建并已在销售一个完整的招聘平台。
- **Cogent:**一家应用 AI 实验室,构建自动化关键企业安全任务的智能体,用 Claude 作为推理层,在完整漏洞生命周期中自动化调查、优先级排序和修复。
- **Airtree:**把 Claude Cowork 作为其运营基础设施的中枢,统一了过去散落在十几个工具和团队中的数据。
- **Duvo:**构建跨 ERP、供应商门户、表格、邮件甚至电话来运行采购、供应链和品类管理流程的 AI 智能体,完全基于 Claude,用 Agent SDK 跨工作流编排。

- **Zingage:**为家庭护理机构构建 7×24 自动化运营的 AI 智能体平台,用 Claude 的结构化工具调用跨 EMR 和多个沟通渠道编排。
- **Kindora:**一个 AI 驱动平台,由一位非营利组织高管用 Claude Sonnet 构建,用于智能匹配慈善机构与资助方;其 MCP 连接器让非营利组织在 Claude 内直接访问其潜在客户开发工具。
- **Wordsmith:**由一位"律师转 CTO"创办,为内部法务团队提供可靠的 AI 法律技术;Claude 是其合同评审、协议起草和文档审查能力的推理引擎,其工程团队用 Claude Code 构建和演进平台本身。

创业支持与机会

- **Anthropic 创业计划:**对于与 Anthropic 的 VC 伙伴合作的创业公司,该计划提供免费 API 额度、公开可得的最高一档速率限制,以及独家创始人活动与工作坊的邀请。
- **Claude 社区:**面向建设者的论坛与社区空间。
- **实时学习资源:**会议、网络研讨会、直播和录播。