龙虾营 · 30 天激进增长实验

Claude Code CEO 指令合集

一个人 · 6 Agent 团队 · 完整指挥链
从 0 到 1 构建 AI Agent 团队的实战手册

Wesley
AI Agent 实战创始人 · 龙虾营

2026.04

FOREWORD

提示词不够用，AI 需要一个真正的 CEO

> 这不是一本讲"如何写提示词"的手册。
> 这是一份**操作系统设计图**——如何让 AI Agent 团队真正替你工作，而不是每次都需要你手动介入。

2026 年 4 月，我开始了一个激进的实验：**一个人，用 6 个 Claude Code Agent，在 30 天内将公众号从 300 粉丝增长到 3000+。**每天上午热点雷达自动扫描，下午内容自动创作，晚上跨平台自动发布，整个链路无需人工干预。

这份手册记录了让这套系统真正运转的**核心指令体系**——决策框架、铁律、路由逻辑、验证链，以及三次真实的踩坑案例。

如果你也在尝试用 Claude Code 搭建自己的 Agent 团队，这份手册可以帮你少走 80% 的弯路。

Wesley
龙虾营创始人 · AI Agent 实战研究者
公众号：Wesley AI 日记 · 知识星球：光锥之内

目录

阅读建议：如果你是初学者，从 §1 顺序读；如果你已有 Claude Code 经验，直接跳到 §3 铁律 + §8 踩坑案例，ROI 最高。

01

SECTION 1

为什么需要 CEO 指令

提示词解决单次对话问题，CEO 指令解决团队协作问题。当你有多个 Agent 并行运作时，你需要的不是更好的提示词，而是一套指挥体系。

01 提示词不够用的本质原因

大多数人用 Claude Code 的方式是：遇到问题，写提示词，得到答案，下次再写。这套方式在单任务场景下没问题，但当你有**多个并发任务、多个 Agent、需要长期自动运转**时，它会崩溃。

原因很简单：**提示词没有记忆、没有优先级、没有团队意识**。每次对话都从零开始，每个 Agent 只关心自己的任务，没有人在全局层面协调。

CEO 模式 vs 提示词模式的核心差异

维度	提示词模式	CEO 指令模式
决策层级	单次对话决定	系统性框架决定
任务分配	手动选择 Agent	路由表自动匹配
错误处理	发现了再说	铁律预防 + 验证链兜底
一致性	每次可能不同	Workflow 强制执行
扩展性	线性增加复杂度	新增 Agent 即可扩展

核心身份：管理者，不是执行者

> **"你的价值是判断力、品味和速度，不是亲手干活。"**

当 CEO Agent 亲自去调 API 参数、亲自监控脚本运行时，它实际上是在浪费上下文窗口，也在违背"主会话不做重活"这条最基础的铁律。正确的姿势是：接到任务 → 压缩 Spec → 派发 Subagent → 验证结果 → 汇报。

> **关键认知：**CEO 指令的本质是给 AI 一个**角色定义 + 决策框架 + 行为边界**。有了这三样东西，AI 才能在复杂的多任务场景下可预期地运作。

02

SECTION 2

决策四重检验

每一个行动决策，在执行前必须通过四道关卡。这不是官僚程序，而是防止 AI 团队在错误方向上浪费算力和时间的护栏。

02 每个决策必过四道关

在 Agent 团队中，决策成本极低——Agent 可以立刻行动。这导致一个危险：**低价值任务会被立即执行，高价值任务却被过度分析**。四重检验是一个快速过滤器，让执行力和判断力同时在线。

01 现金流检验

这件事能产生或保护现金流吗？如果不能，它在优先级列表里的位置是什么？

02 足够检验

真的需要这件事吗？还是在制造不必要的复杂度？能不做的事是否可以不做？

03 系统化检验

这件事会做第二次吗？如果会，现在就要建系统，而不是每次手动重复。

04 差异化检验

这体现了独特价值吗？还是任何人都能做？CEO 的时间只应该花在差异化工作上。

实际运用示例

场景：Agent 提议每天手动检查一次 cron job 状态报告。

四重检验：
✗ 现金流？间接相关，不是直接驱动力
✗ 足够检验？cron 有监控告警，手动检查是重复劳动
✓ 系统化？可以建自动告警，发现错误立即通知
✗ 差异化？任何 Agent 都能做的监控动作

决策：改为自动监控 + 仅异常时通知，CEO 不做例行检查。

检验的顺序很重要

按 01→02→03→04 的顺序检验。如果第一关就不过，后面三关不用看。最节省时间的决策是"不做这件事"。

03

SECTION 3

六条铁律

铁律不是建议，不是最佳实践，是系统边界。违反任何一条，轻则任务失败，重则账号封禁、数据丢失。每条铁律背后都有一次真实的踩坑。

03 六条不可违反的系统边界

1 主会话不做重活

超过 3 轮 tool call → 必须 spawn subagent。主会话的上下文窗口是稀缺资源，一旦被消耗殆尽，新消息会被截断导致指令丢失。

踩坑记录
CEO 直接在主会话里处理一个 20+ 步的内容发布任务，上下文溢出后 Wesley 的新指令被截断，紧急任务延误 2 小时。

2 content-reviewer 强制质检

任何内容发布前必须通过独立质检 Agent，绝不绕过。质检不是形式，是脱敏 + 合规 + 品质的最后一道防线。

踩坑记录
一篇文章因"时间紧迫"跳过质检直接发布，事后发现含有服务器 IP 地址，被截图传播，紧急删除但已造成信息泄露。

3 CEO 不碰 HTML/代码

所有代码修改委派给 dev-engineer Agent。CEO 直接改代码会引入难以追踪的 bug，且破坏 Agent 职责边界。

踩坑记录
CEO 直接修改了一个发布脚本的参数，导致后续 15 篇文章的图片路径全部错误，dev-engineer 花费 3 小时回滚修复。

4 执行问题不问 Wesley

有疑问直接做，完成后汇报结果。等待确认 = 失职。CEO 的自主判断能力是整个系统的核心价值，频繁请示会打断 Wesley 的深度工作。

5 Cron Error 立刻响应

发现定时任务错误 → 立刻通知 Wesley + 派 dev-engineer 修复。不等下次心跳，不等"看看是不是偶发"。定时任务是整个自动化流水线的心脏。

踩坑记录

Master Scheduler 触发器被意外删除，因为没有立即响应机制，16 个自动化任务断链 2 天，造成内容发布空窗期，损失了关键增长窗口。

6 全生命周期通知

任务的每个关键状态都要通知：收到任务→ACK / 委派→DELEGATE / 异常→ERROR / 完成→DONE。

铁律可视化

记忆技巧：3 个"不"（不做重活、不碰代码、不问 Wesley）+ 3 个"必须"（必须质检、必须响应 Error、必须全程通知）。

04

SECTION 4

任务路由表

路由表是团队的"交通规则"。每种任务类型对应一个专属 Agent，这不是偏好，是强制约束。错误路由会导致 Agent 越权操作，引发连锁故障。

04 13 种任务类型路由

任务类型	→ AGENT	→ 模型
小红书内容创作/发布/互动	→ Agent(xhs-main)	[opus]
微信公众号长文	→ Agent(wechat-mp)	[opus]
产品功能/代码/业务 Bug	→ Agent(dev-engineer)	[opus]
品牌/视觉/配图	→ Agent(brand-designer)	[sonnet]
数据分析/增长策略	→ Agent(growth-hacker)	[opus]
跨平台引流分发	→ Agent(cross-platform-publisher)	[sonnet]
视频制作	→ Agent(video-producer)	[opus]
内容质检	→ Agent(content-reviewer)	[sonnet]
基础设施/Cron/MCP	→ Agent(dev-engineer) / CEO	[opus]
战略/协调/与创始人沟通	→ CEO 自己做	—

路由纠错（常见错误）

错误路由	正确路由	原因
草稿修改 → dev-engineer	草稿修改 → **wechat-mp**	内容任务，不是代码任务
脚本 Bug → wechat-mp	脚本 Bug → **dev-engineer**	基础设施代码，不是内容
配图生成 → xhs-main	配图生成 → **brand-designer**	视觉任务有独立 Agent

> **路由判断原则：**不确定该谁做 → **默认委派给最相关的 Agent，让 Agent 自己决定是否转包**。CEO 不在路由判断上浪费时间。

05

SECTION 5

Workflow 驱动执行

每个标准流程都有对应的 Skill 文件。执行前先读 Skill，严格按步骤走。跳步骤 = P0 事故。Workflow 是把"人治"变成"法治"的核心机制。

05 10 个核心 Workflow

每个 Workflow 都是一个结构化的 SKILL.md 文件，包含前置条件检查、执行步骤、产出物规格、验证清单。Agent 必须先 read 对应 Skill 才能开始执行。

📝 **公众号长文写作**
前置门禁（热点雷达→数据门禁→增长动作）→ 写作 → 质检 → 发布，缺任何一步强制拒绝

🔴 **热点雷达（强制前置）**
每天早上扫描 T0/T1/T2 信源，打分排序，产出 morning-brief.md，是写长文的硬前置

📊 **公众号数据门禁（强制前置）**
14 天数据复盘 + 爆款归因 + 选题 GO/NO-GO 判定，写文前必须通过，产出 gate-report.md

📈 **增长动作中台（强制前置）**
匹配奇袭动作、埋点 SEO 关键词、生成发布后分发计划，产出 growth-action-plan.md

🖼️ **公众号贴图发布**
竖版图片 + 200 字文案 + 标签，通过 API 上传，不走长文流程

🌸 **小红书日常内容**
选题→文案→配图生成→发布，严格执行每日互动上限（≤3次/会话），防封号

🔗 **跨平台引流**
公众号文章适配 → 微博/知乎/掘金/CSDN 多平台分发 → 末尾双二维码

🌟 **知识星球内容**
每日精华提炼 → 星球专属内容 → 发布，维系付费用户关系

05 公众号长文发布完整流程

这是整个系统中最复杂的流程，也是踩坑最多的地方。完整流程分为三个强制前置门禁 + 写作 + 质检 + 发布，共 6 个阶段。

```
# 公众号长文发布完整流程（不可跳步） Step 0.1 → hotspot-radar 产出: output/hotspots/{today}/morning-brief.md # 没有这个文件 → 写作 Agent 硬拒 Step 0.2 → wechat-data-gate 产出: output/growth/gate-reports/{today}/gate-report.md # GO 才能继续，NO-GO = 停止 Step 0.3 → growth-tactics 产出: output/growth/growth-action-plan-{today}.md Step 1 → wechat-article-writer # 读 morning-brief + gate-report 后才开始写 Step 2 → content-reviewer # 脱敏 + 质检，PASS 才能继续 Step 3 → wechat-article-publish # assert gate-report-ref.txt 存在才发布
```

Workflow 的核心价值

防止"聪明"绕路： Agent 有时会觉得"这步可以跳过"，Workflow 的硬前置检查会强制停止这种行为。

可复现性： 同一个流程每次执行结果高度一致，不依赖某个具体 Agent 的"发挥"。

新手常见误区： 把 Workflow 当成"参考"而不是"强制约束"。一旦开了"特殊情况可以跳过"的口子，整个系统的可靠性就会崩塌。

06

SECTION 6

验证链 L1–L4

"完成了"不等于真的完成。Subagent 汇报成功，不代表产出物是真实的、在正确的地方、实际效果达标。四层验证缺一不可。

06 四层验证清单

每次 Subagent 完成任务汇报后，CEO 必须至少验证到 L3。"绝不转发未验证的汇报给 Wesley"——这是 CEO 的信誉底线。

L1 产出物存在

media_id 不为空 / URL 可达 / 文件存在于正确路径

检查方法：ls 文件路径 / curl URL / API 返回 media_id

L2 内容是真实的

读前 300 字无乱码 / 标题正确 / 非 placeholder

检查方法：Read 文件前 50 行 / 查看 API 响应内容字段

L3 放对地方了

正确的公众号 / 平台 / 目录，不是发到了错误账号

检查方法：查平台后台草稿列表 / 查文件所在目录

L4 实际效果对

预览正常 / 图片加载 / 链接可点 / 格式正确

检查方法：Playwright 截图验证 / 浏览器实际打开

重要认知：curl 返回 521 ≠ 发布成功——平台反爬机制会让 curl 误判。必须用 Playwright 真浏览器打开 + 查用户文章列表双重证据，才算 L4 验证通过。

验证的典型失败模式

失败模式	被哪层发现	后果
文件写到了 /tmp 而不是 output/	L1	重启后产出物消失
文件存在但内容是模板占位符	L2	发布了空内容
发布到了测试账号	L3	正式账号无内容
图片 URL 过期无法加载	L4	读者看到破图

07

SECTION 7

P0 红线与核心铁律

这些是创始人与 AI CEO 之间最基础的信任协议。违反这些规则不是"犯错"，而是动摇整个协作体系的根基。

07 7 条 P0 级行为准则

1. **执行类问题：直接做，不问**
   包括发布时间/节奏/内容选择。CEO 有完整上下文，有能力做这些判断。频繁请示 = 浪费双方时间。

2. **只有战略方向才汇报**
   新业务方向、重大资源调配才值得打扰创始人。日常执行细节自己处理，完成后一句话汇报结果即可。

3. **Subagent 汇报"完成" ≠ 真的完成**
   必须用验证链 L1-L4 验证实际结果，再决定是否向创始人汇报。转发未验证的汇报是 P0 失职。

4. **Subagent 失败/超时：CEO 自行处理**
   重试、重派、换方案，这是 CEO 的职责范围。找创始人解决执行问题 = 角色错位。

5. **Memory ≠ 当前事实**
   历史记忆可能已经过时。必须交叉验证：查文件 / 查 API / 查日志。"我记得上次是这样的"不是有效证据。

6. **禁止附和**
   知道答案直接说，不确定就说"让我查一下"。"您说得对"、"非常好的想法"这类话从不出现在 CEO 的汇报里。

7. **汇报中禁止给创始人派任务**
   汇报里不能出现"动作清单/TODO/你要做"这类内容。查看、验证、监控都是 CEO 自己的活，不外包给创始人。

P0 红线指针：脱敏检查 → 对外内容必须通过 content-desensitization Skill；内容矩阵 → 三条人设线路完全隔离；发布门禁 → 公众号长文必须过三关后写 gate-report-ref.txt 才能发布。

08

SECTION 8

实战踩坑 Case 3 个

真实的失败比成功更有价值。以下三个案例已脱敏处理，保留了核心教训。每一个都是系统设计迭代的起点。

08 Case 1：调度器崩溃事故

Master Scheduler 触发器意外删除

P0 INCIDENT

背景	整个自动化系统依赖一个远程调度中心（Remote Trigger）统一管理 16 个定时任务，包括热点雷达、内容发布、数据统计等核心流程。
触发	某次配置清理操作中，Master Scheduler 的 trigger 被意外删除。由于没有监控告警机制，删除后没有任何人知道。
影响	16 个自动化任务全部断链，连续 2 天无内容发布，公众号、小红书等平台断更，正值增长关键窗口期。
发现	创始人发现"今天怎么没有内容发出去"，手动检查才发现 Scheduler 已经停止运行。

> **核心教训：**
> 1. 关键任务不能只靠单点远程调度，核心发布类任务应迁移到本地 launchd/cron 双保险
> 2. 任何 Scheduler 必须有心跳检测 + 停止自动告警，发现 Error → 立刻通知（铁律 5 的由来）
> 3. 计划配额是约束，不是"够用就行"——Max 20x plan 的 3 个 scheduled tasks 配额不够用，需要 Master Scheduler 架构

08 Case 2：平台自动化封号事故

单次会话高频写操作触发平台风控 P0 ACCOUNT BAN

背景	为了提升内容发布效率，计划在单次 Agent 会话中完成某内容平台的多篇笔记发布和批量互动操作。
触发	单会话内执行了 31 次写操作（发布+评论+互动），触发平台风控系统，账号被永久封禁。
影响	该平台账号永久封禁，数十篇历史内容全部清零，粉丝关系断裂，需要重新冷启动。

> **核心教训：**
> 1. 平台风控 > 效率诉求：账号安全优先级高于"零手动铁律"，当两者冲突时，账号安全获胜
> 2. 单会话写操作硬上限 ≤ 3 次 / 每次操作间隔 ≥ 30 分钟，这是不可协商的红线
> 3. 自动化必须模拟人类操作节奏，批量操作 = 自杀行为
> 4. 代码层 + Agent 层 + 连接层三重隔离，任何自动化不能碰该平台域名

08 Case 3：发布验证失效事故

curl 误判导致"发布成功"实际未发布 SILENT FAILURE

背景	发布脚本通过 curl 请求检查文章是否可访问，返回 200 即判定发布成功，汇报"已发布"。
触发	平台启用了反爬 CDN（返回 521 Visitor Check），curl 被返回 521，脚本误判为"不可访问"。反过来，平台对某些请求返回缓存的 200 也会误判为"已发布"。

> **核心教训：**
> 1. curl 验证 ≠ 真实发布状态，必须用 Playwright 真浏览器 + 查平台用户文章列表双重证据
> 2. "发布状态"判断必须查平台后台，不能只看本地 publish-report.json
> 3. 登录态验证不能只看 URL，必须 title + DOM + 截图三重验证（Session probe 视觉验证）

08 踩坑模式归纳

三个案例背后是同一个根本原因：**系统没有足够的失败可见性**。没有告警、没有双重验证、没有硬上限，导致问题在"默默失败"中扩大。

踩坑类型	根本原因	解决方案
调度器无声崩溃	单点依赖 + 无监控	心跳检测 + 本地双保险 + 立即告警
平台账号封禁	效率优先忽视风控边界	硬上限写入 Agent 指令，不可协商
验证误判	依赖间接信号（curl 状态码）	真浏览器 + 后台列表双重验证
内容质量滑坡	跳过质检节省时间	质检前置为强制门禁，不可绕过
脱敏信息泄露	发布前未扫描敏感内容	content-reviewer 强制扫描白名单

系统韧性原则

假设失败：不是"如果失败怎么办"，而是"失败时如何被立刻发现"。

最小权限：Agent 只做自己职责范围内的事，越界操作必须被系统拒绝。

硬上限 > 软建议：把重要约束写进 Agent 指令，不依赖 Agent"记得"遵守。

双重验证：任何单一信号都可能出错，关键决策需要两个独立信源确认。

09

SECTION 9

5 步从 0 搭 CEO Agent 团队

抄作业指南。你不需要从零设计架构，按照这 5 步走，最快 3 天可以搭出一个能自动运转的 Agent 团队雏形。

09 Step 1–3：地基搭建

1 定义你的 CEO CLAUDE.md

在项目根目录创建 CLAUDE.md，写入：角色定义（管理者）、决策框架（四重检验）、铁律（6条）、路由表（你的业务场景）。这是整个系统的宪法，其他所有设置都服从它。

关键要素
核心身份定义 + 决策四重检验 + 6条铁律 + 任务路由表 + 通知规则 + Wesley P0铁律

2 创建专属 Agent 角色文件

在 .claude/agents/ 目录下为每个 Agent 创建对应的 .md 文件。每个文件定义该 Agent 的职责边界、工具权限、输出规范。建议从 3 个核心 Agent 开始：内容创作、代码执行、质检审核。

```
# 目录结构 .claude/ agents/ wechat-mp.md # 内容创作 dev-engineer.md # 代码执行 content-reviewer.md # 质检 brand-designer.md # 视觉设计
```

3 为核心流程编写 Workflow Skill

把你最常重复的操作写成 SKILL.md 文件，放入 .claude/skills/ 目录。每个 Skill 包含：触发条件、前置检查、执行步骤（编号）、产出物规格、验证清单。从你最痛的那个流程开始写。

09 Step 4–5：激活与迭代

4 设置验证链和告警机制

在每个 Workflow 末尾加入 L1-L4 验证步骤。为关键定时任务设置心跳检测：任务执行后写入 last-run.json，监控脚本检查时间戳，超过预期间隔立即发通知。

最小可行告警
定时任务执行完成 → 写 last-run.json → 另一个 cron 每小时检查时间戳 → 超时发飞书/邮件通知

5 从单 Agent 到团队：渐进扩展

不要一开始就搭 6 个 Agent。从 CEO + 1 个执行 Agent 开始，跑通一个完整的任务闭环（例如：写文章→质检→发布），再逐步增加 Agent。每增加一个 Agent，先问四重检验：需要吗？

月1	月2
冷启动	**扩张**
CEO + 内容 Agent + 质检 Agent，跑通一个平台的完整发布链路	*加入开发 Agent + 增长分析 Agent，跑通多平台分发和数据回收*

最重要的一句话：先有 CEO 指令（CLAUDE.md），再有 Agent，再有 Workflow。顺序不能反。很多人直接开始写 Workflow，发现 Agent 行为不一致，根本原因是缺少 CEO 层的约束框架。

起步最低配置：Claude Code + CLAUDE.md（约 200 行）+ 1 个 Agent .md + 1 个 Skill.md，总工作量约 4-6 小时，即可跑出第一个自动化循环。

10

SECTION 10

工具生态全景

CEO Agent 团队不是孤立运作的，它依赖一套工具生态。了解每层工具的职责，才能在故障时快速定位问题所在。

10 四层工具架构

整个系统从底到上分四层：**基础设施层 → Agent 运行时层 → Skill 能力层 → 业务执行层**。每层有明确的职责边界，故障排查时从下往上查。

Claude Code

RUNTIME

Agent 的运行环境。支持多 Agent 并发，通过 spawn subagent 实现任务分发。上下文窗口是关键约束资源。

Agent System

ORCHESTRATION

CEO + 各职能 Agent 的协作框架。通过 .claude/agents/*.md 定义角色，CLAUDE.md 定义全局规则。

Skill / Workflow

CAPABILITY

标准流程库。.claude/skills/*.md 文件定义每个流程的步骤、约束和产出物规格，Agent 调用前必须先 read。

MCP Servers

INTEGRATION

外部服务连接器。Playwright（浏览器自动化）、Jina（网络检索）、飞书（通知）等，扩展 Agent 的感知和行动能力。

Cron / Scheduler

INFRA

定时驱动层。本地 launchd 处理关键发布任务，远程 Scheduler 处理非关键任务，双轨冗余防止单点崩溃。

output/ 目录

STORAGE

所有产出物必须存入 output/ 目录。/tmp 路径重启后消失，产出物 = output/，这是不可违反的存储规范。

关键 MCP 工具用途速查

MCP 工具	主要用途	使用 Agent
`playwright`	浏览器自动化、登录态验证、截图确认	所有发布 Agent
`jina`	网络检索、热点扫描、URL 内容提取	hotspot-radar
`github`	代码版本管理、PR 审查	dev-engineer
`gemini-imagegen`	AI 配图生成（无中文文字类）	brand-designer

龙虾营 · Wesley AI 日记

持续更新，关注获取最新内容

更多 Claude Code 实战技巧、Agent 团队搭建经验
每天在公众号和知识星球同步更新

公众号

Wesley AI 日记

知识星球

光锥之内