从零到全自动：一个人用 OpenClaw 重新定义"一人公司"

从零到全自动：一个人用 OpenClaw 重新定义"一人公司"

当所有系统合为一体，你就是一支军队

引言：你已经拥有了所有零件

如果你跟着这个系列一路走到了这里——

你已经拥有了：

- **X-01**：一条完整的内容流水线（Scout → Editor → Quill → Pixel → Metric）
- **X-02**：一套统一的信息操作系统（Radar → Cortex → 三管道分发 → 数据闭环）
- **X-03**：一个开发与内容的双轮驱动系统（Ops/Coder/Reviewer → Miner → 内容轮）
- **X-04**：一条从市场发现到产品上线的全链路（Sentinel → Recon → Strategist → Build → Launch → Pulse）
- **X-05**：一个 CTO 级别的技术决策支持体系（Horizon → Vitals → Compass → Bench → Scribe）
- **X-06**：一套通用的 Agent 人格设计框架（七模块模型 I-C-C-C-W-O-G）
- **X-07**：一份完整的安全防护体系（4 场景 × 20 条铁律）
-

散落在你面前的是 **30+ 个 Agent 设计、7 套子系统、上百页 SOUL.md 模板**。

但它们还是零件。

这篇文章要做的最后一件事： **把所有零件焊接成一台机器。**

不是简单地"同时运行所有 Agent"——那只会得到一堆互相打架、重复采集、成本爆炸的混乱。而是经过精心设计的统一架构——信息只流动一次，每个 Agent 各司其职，成本可控，安全可靠。

这台机器有一个名字—— **一人公司操作系统（Solo Company OS）** 。

一个人用它，同时做四件事：

1. **获取信息** ——比别人早 3 个月知道行业在发生什么
1. **创作内容** ——每周稳定产出 2-3 篇高质量文章
1. **开发产品** ——从发现机会到产品上线只需要 6 周
1. **做技术决策** ——每一个选型都有数据支撑和文档沉淀
2.

一个人，活成一支军队。

一、为什么不能简单地"全部跑起来"

1.1 天真方案的三个致命问题

假设你直接把 X-01 到 X-05 的所有 Agent 同时运行：

text

代码块

```
Agent 总数: 30+
每日 API 成本: $40-80
```

```
月度成本：$1,200-2,400
信息源重复采集：约 60%
你每天要看的 Telegram 消息：30+ 条
```

问题一：信息源重复采集。

X-02 的 Radar 在扫 RSS 和 HN。X-04 的 Sentinel 也在扫 HN 和 Reddit。X-05 的 Horizon 还在扫 HN 和 GitHub Trending。三个 Agent 扫同一批源，每天多花 $3-5 的冤枉钱。

问题二：推送轰炸。

Cortex 给你推晨报。Editor 给你推选题。Sentinel 给你推市场信号。Horizon 给你推技术趋势。Vitals 给你推健康报告。Pulse 给你推用户反馈。Metric 给你推数据日报。

你每天要处理 10+ 条 Telegram 长消息。到第三天你就开始忽略它们了——然后你花了大价钱搭建的系统，变成了一个高级通知垃圾桶。

问题三：决策碎片化。

信息在不同系统中各自分析，没有交叉验证。Sentinel 发现了一个市场机会，但它不知道 Horizon 刚报告了那个方向的核心技术正在衰退。Miner 挖到了一个好素材，但它不知道 Pulse 的反馈显示读者对这类话题已经审美疲劳。

你需要的不是 30 个独立的 Agent，而是一个有统一大脑的系统。

1.2 设计原则

在开始合并之前，明确四条设计原则：

原则	含义	落地方式

单点采集	每条信息只被采集一次	统一信息采集层
统一调度	你每天只看一个"仪表盘"	中枢 Agent 合并推送
按需激活	不是所有子系统每天都要运行	模式切换机制
共享记忆	一个子系统的洞察可以被其他子系统利用	共享知识库

二、Solo Company OS：统一架构

2.1 四层架构

text

代码块

```
╔═══════════════════════════════════════════════════════════════════════════╗
║                                                                           ║
║  Layer 0: 统一感知层 (Unified Perception)                                 ║
║  ┌─────────────────────────────────────────────────────┐                  ║
║  │                                                     │                  ║
║  │              OmniRadar                              │                  ║
║  │  合并: Radar + Sentinel + Horizon                   │                  ║
║  │  全网信息一次采集，打标签后分流                      │                  ║
║  └──────────────────────┬──────────────────────────────┘                  ║
║                         │                                                 ║
║  Layer 1: 统一决策层 (Unified Intelligence)                               ║
║  ┌──────────────────────┴──────────────────────────────┐                  ║
║  │                                                     │                  ║
║  │              Nexus (中枢)                           │                  ║
║  │  合并: Cortex + 部分 Editor 职能                    │                  ║
║  │  分拣 → 四条管道 → 统一仪表盘推送                   │                  ║
║  ├──────────┬──────────┬──────────┬────────────────────┤                  ║
```

2.2 Agent 合并清单

从 30+ 个 Agent 精简到 **12 个核心 Agent**：

合并后	合并前的来源	使命	运行模式
OmniRadar	Radar + Sentinel + Horizon	全网统一信息采集	每日定时 5:30
Nexus	Cortex + 部分 Editor	信息分拣 + 四管道分发 + 统一推送	每日定时 6:15
Editor	Editor (精简版)	内容选题评估	每日 19:30

Quill	Quill	内容撰写	每日/按需
Pixel	Pixel	视觉设计	跟随 Quill
Miner	Miner	从开发活动提取内容素材	每日 19:00
Coder	Coder	编码辅助	交互式
Reviewer	Reviewer	代码审查	Webhook
Recon	Recon	深度竞品/技术分析	按需触发
Compass	Compass + Strategist	技术选型 + 产品决策	按需触发
Scribe	Scribe + Launcher	文档/文案/发布协调	按需 + 定时
Oracle	Metric + Pulse + Vitals(监控)	全局数据分析 + 反馈闭环	每日 21:00

精简比例：30+ → 12，减少 60%。

2.3 为什么这样合并？

每一次合并都有明确的理由：

合并一：Radar + Sentinel + Horizon → OmniRadar

三者的核心能力都是"从外部采集信息"。区别只是采集的维度不同：

- Radar 采集个人数据（日历/邮件/天气）和新闻
- Sentinel 采集市场/痛点信号
- Horizon 采集技术趋势
-

合并后，OmniRadar 用 **一次扫描** 同时采集所有维度的信息，然后打标签。标签决定信息流向哪条管道。消除了 60% 的信息源重叠。

合并二：Cortex + 部分 Editor → Nexus

Cortex 做信息分拣，Editor 做选题评估。在分开的系统里，Cortex 先分拣出"选题素材"，然后 Editor 再评估。但实际上，"这条信息是不是好素材"的判断在分拣时就可以初步完成。

Nexus 在分拣时同步做初步评分，给每条流入管道 C（内容素材）的信息标注"初始分数"。Editor 只需要在此基础上做精细评估和排序——工作量减少 40%。

合并三：Metric + Pulse + Vitals(监控) → Oracle

三者都是"追踪数据并生成反馈"：

- Metric 追踪内容数据
- Pulse 追踪用户反馈
- Vitals 追踪技术健康度
-

合并后，Oracle 成为系统的"全局大脑"——它知道内容表现、用户声音和技术状态。最重要的是，它可以做 **跨维度关联分析**：

text

```
代码块
1  "本周发布的 LlamaIndex 技术文章阅读量是均值的 3 倍 (Metric 数据)
2  + 用户评论中有 8 个人追问相关话题 (Pulse 数据)
3  + LlamaIndex 的 GitHub Star 本月增长 15% (Vitals 数据)
4  → 结论：这个方向有强烈的市场信号，建议做系列"
```

单独的 Metric、Pulse、Vitals 各自只看到一个维度。Oracle 看到全局。

合并四：Compass + Strategist → Compass

两者的本质都是"给候选方案打分并推荐"。区别只是一个评估技术方案，一个评估产品方案。合并后的 Compass 支持两种模式：

text

```
代码块
1  你：@Compass 评估技术选型：Kafka vs NATS
2  → Compass 使用技术评估框架
3
4  你：@Compass 评估产品方案：基于 Recon 报告出产品方案
5  → Compass 使用产品评估框架
```

同一个 Agent，两套评估框架，按指令切换。减少了一个 Agent 的运维成本和记忆隔离开销。

合并五：Scribe + Launcher → Scribe

两者都是"产出面向外部的文字内容"——Scribe 写文档/ADR，Launcher 准备发布文案。合并后的 Scribe 成为"所有正式文字的出口"：技术文档、ADR、Landing Page、发布文案、Changelog——都由 Scribe 负责。

三、OmniRadar：万源归一

3.1 OmniRadar 的信息源架构

OmniRadar 的信息源按 **维度** 而非 **来源** 组织。每个来源被标注多个维度标签：

YAML

```yaml
# OmniRadar 信息源配置 — sources.yaml

dimensions:
  - personal      # 个人数据（日历/邮件/天气/健康）
  - news          # 新闻和热点
  - market        # 市场/商业信号
  - tech_trend    # 技术趋势
  - community     # 社区讨论/用户声音
  - competitor    # 竞品动态

sources:
  # ── Tier 1: 个人数据（每日必扫）──
  - name: "Google Calendar"
    type: api
    dimensions: [personal]
    frequency: daily

  - name: "Gmail（专用账号）"
    type: api
    dimensions: [personal]
    frequency: daily
    read_only: true

  - name: "天气 API"
    type: api
    dimensions: [personal]
    frequency: daily

  - name: "Apple Health / WHOOP"
    type: api
    dimensions: [personal]
    frequency: daily

  # ── Tier 2: 综合信息源（每日必扫，多维度标签）──
  - name: "Hacker News Top 30"
    type: rss
    dimensions: [news, tech_trend, community]  # 一个来源，三个维度
    frequency: daily

  - name: "Reddit r/SaaS"
    type: rss
    dimensions: [market, community]
    frequency: daily

  - name: "Reddit r/programming"
    type: rss
    dimensions: [tech_trend, community]
```

```yaml
    frequency: daily

  - name: "Twitter/X 关键词监控"
    type: api
    dimensions: [news, market, tech_trend, community]
    keywords: [你的关键词列表]
    frequency: daily

  - name: "Product Hunt 日榜"
    type: scraper
    dimensions: [market, competitor]
    frequency: daily

  - name: "GitHub Trending"
    type: api
    dimensions: [tech_trend]
    frequency: daily

  # —— Tier 3: 专项信息源 ——
  - name: "竞品官网监控"
    type: scraper
    dimensions: [competitor]
    targets: [竞品 URL 列表]
    frequency: daily

  - name: "技术栈官方博客"
    type: rss
    dimensions: [tech_trend]
    feeds: [React blog, FastAPI releases, ...]
    frequency: daily

  - name: "微博热搜 / 即刻 / 36氪"
    type: scraper
    dimensions: [news, market]
    frequency: daily

  # —— Tier 4: 低频深度源（每周）——
  - name: "arXiv"
    type: api
    dimensions: [tech_trend]
    categories: [cs.SE, cs.AI]
    frequency: weekly

  - name: "Crunchbase 融资新闻"
    type: rss
    dimensions: [market]
    frequency: weekly
```

```
95
96      - name: "招聘市场关键词"
97        type: scraper
98        dimensions: [tech_trend, market]
99        frequency: weekly
```

3.2 OmniRadar 的输出

OmniRadar 不做任何分析——它只采集和打标签。输出是一个结构化的 JSON：

JSON

代码块

```
1   {
2     "date": "2026-03-03",
3     "collection_stats": {
4       "total_sources": 45,
5       "successful": 43,
6       "failed": 2,
7       "total_items": 892,
8       "after_dedup": 534
9     },
10    "personal": { ... },
11    "items": [
12      {
13        "id": "omni-2026-03-03-00142",
14        "source": "Hacker News",
15        "title": "Show HN: LlamaIndex 0.12 with streaming RAG",
16        "url": "https://...",
17        "summary": "...",
18        "published_at": "2026-03-03T04:22:00Z",
19        "dimensions": ["news", "tech_trend", "community"],
20        "raw_metrics": { "points": 342, "comments": 87 },
21        "dedup_group": null
22      },
23      ...
24    ]
25  }
```

关键设计：每条信息的 `dimensions` 字段决定了 Nexus 会把它分拣到哪些管道。一条信息可以同时进入多条管道。

3.3 成本对比

	合并前（三系统分别采集）	合并后（OmniRadar）
每日 API 调用	~450 次	~200 次
每日 LLM Token	100K-150K	30K-50K
日均成本	$2.5-5.0	$0.8-1.5
月度节省	—	**$50-105**

四、Nexus：统一的中枢大脑

4.1 Nexus 的四条管道

Nexus 接收 OmniRadar 的结构化输出，按维度标签分拣到四条管道：

text

代码块

```
管道 A: 个人晨报 (Morning Brief)
  ← dimensions 包含 "personal" 的所有条目
  ← dimensions 包含 "news" 且相关度 > 70 的条目
  → 输出：晨报 Markdown
  → 推送至你的 Telegram

```

```
管道 B: 深度情报 (Intelligence Digest)
  ← dimensions 包含 "market" 或 "competitor" 的条目
  ← dimensions 包含 "tech_trend" 且涉及你的技术栈的条目
  → 输出: 情报日报 Markdown
  → 推送至你的 Telegram (可选: 只在有重要信号时推送)

管道 C: 内容素材 (Content Feed)
  ← dimensions 包含 "news" 或 "community" 且有爆款潜力的条目
  ← Miner 的开发素材卡
  → 输出: 选题素材池 Markdown
  → 传递给 Editor

管道 D: 产品&技术信号 (Product & Tech Signals)
  ← dimensions 包含 "market" 且有产品机会信号的条目
  ← dimensions 包含 "tech_trend" 且涉及你的技术栈健康变化的条目
  ← dimensions 包含 "competitor" 的条目
  → 输出: 产品&技术信号 Markdown
  → 推送至你的 Telegram (只在有强信号时推送)
```

4.2 统一仪表盘：一天只看一条消息

Nexus 最重要的创新： **四条管道的输出合并成一条统一的仪表盘消息。**

text

代码块

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
🌅 Solo Company OS — 2026年3月3日 周二
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

📋 YOUR DAY
• 09:00 周会 | 14:00 客户通话 | 19:00 健身
• 📧 3 封重要邮件待处理
• 💪 睡眠 7.2h | 恢复 78%
• ☀️ 晴 12°C → 适合跑步

🔥 TODAY'S TOP 3
1. LlamaIndex 0.12 发布，支持流式 RAG [tech][content📝]
2. Stripe 宣布 AI Agent 支付 API [market][product🎯]
3. Redis 核心维护者离职声明 [tech⚠️]
```

```
📊 SIGNALS
• 🟢 内容：上周文章均阅读 +23%，asyncio 系列特别好
• 🟡 技术：Redis 健康分 75→72（连续 3 周 ↘）
• 🟢 产品：DocFast 本周 3 个新付费用户
• 🟡 市场：竞品 Mintlify 发布新功能

✅ YOUR DECISIONS TODAY
□ 审核选题（Editor 将在 19:30 推送）
□ 审阅终稿（如有 Quill 产出）
□ Redis 评估要不要启动？（回复 "评估 Redis"）

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
回复 "情报" 看完整情报日报
回复 "素材" 看今日内容素材
回复 "技术" 看技术健康报告
回复 "评估 [主题]" 触发 Compass 深度分析
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
```

一条消息，三分钟看完，一天的全局尽在掌握。

注意那些标签：

- `[tech]` = 进入了管道 D（技术信号）
- `[content📝]` = 进入了管道 C（可以写成文章）
- `[market]` = 进入了管道 B（商业情报）
- `[product🎯]` = 管道 D 中的产品机会信号
- `[tech⚠️]` = 管道 D 中的技术风险信号
-

一条信息可以同时有多个标签——LlamaIndex 0.12 发布既是技术趋势也是好的内容选题。Nexus 用标签让你一眼看到每条信息的多维价值。

4.3 Nexus 的交互式命令

Nexus 不只是"推一次就完了"的定时任务——它支持交互式指令：

text

```
你：情报
Nexus：[推送完整情报日报]

你：素材
Nexus：[推送今日内容素材池]

你：技术
Nexus：[推送技术健康仪表盘]

你：评估 Redis
Nexus：收到。我将触发 Compass 对 Redis 替代方案进行深度分析。
       预计 25 分钟后推送报告。

你：本周关注 AI Agent 安全
Nexus：已记录。本周 OmniRadar 将提升 "AI Agent 安全" 相关信息的优先级。
       管道 B 和 C 中的相关内容将置顶。
       管道 D 将关注这个方向的产品机会和技术风险。
       调整于明日生效，持续至本周日自动恢复。
```

Nexus 是你和整个系统的 **唯一交互入口** 。你不再需要分别和 10 个 Agent 对话——你只和 Nexus 对话，它帮你分发指令。

五、模式切换：不是每天都需要跑全量

5.1 三种运行模式

一个人不可能每天同时在四条线上高强度工作。现实是：

- 有些日子你在写代码（Build 模式）

- 有些日子你在做内容（Content 模式）
- 有些日子只是正常工作不需要特别投入（Cruise 模式）
-

Solo Company OS 支持 **模式切换** ——不同模式下，不同的 Agent 被激活或降频：

text

代码块

```
┌──────────────────────────────────────────────────────────────┐
│                        三种运行模式                          │
├──────────┬──────────────┬──────────────┬─────────────────────┤
│ Agent    │ 🚢 Cruise    │ 📝 Content   │ 💻 Build            │
│          │ (巡航模式)    │ (内容模式)    │ (开发模式)           │
├──────────┼──────────────┼──────────────┼─────────────────────┤
│OmniRadar │ ✅ 全量       │ ✅ 全量       │ ✅ 全量              │
│Nexus     │ ✅ 全量       │ ✅ 全量       │ ✅ 全量              │
│Editor    │ ⏸️ 暂停       │ ✅ 激活       │ ⏸️ 暂停              │
│Quill     │ ⏸️ 暂停       │ ✅ 激活       │ ⏸️ 暂停              │
│Pixel     │ ⏸️ 暂停       │ ✅ 激活       │ ⏸️ 暂停              │
│Miner     │ ✅ 低频       │ ✅ 全量       │ ✅ 全量              │
│Coder     │ ⏸️ 待命       │ ⏸️ 待命       │ ✅ 激活              │
│Reviewer  │ ⏸️ 待命       │ ⏸️ 待命       │ ✅ 激活              │
│Recon     │ ⏸️ 待命       │ ⏸️ 待命       │ 按需触发             │
│Compass   │ ⏸️ 待命       │ ⏸️ 待命       │ 按需触发             │
│Scribe    │ ⏸️ 待命       │ ✅ 低频       │ ✅ 低频              │
│Oracle    │ ✅ 全量       │ ✅ 全量       │ ✅ 全量              │
├──────────┼──────────────┼──────────────┼─────────────────────┤
│日成本     │ $3-6         │ $8-15        │ $6-12               │
│你的决策点 │ 1个/天        │ 3-4个/天      │ 1-2个/天             │
└──────────┴──────────────┴──────────────┴─────────────────────┘
```

5.2 模式切换方式

通过 Telegram 命令切换：

text

代码块

```
你：@Nexus 模式 内容
Nexus：已切换至 📝 内容模式。
       激活：Editor, Quill, Pixel, Miner
       降频：Coder, Reviewer
       今日内容管道将获得更多注意力分配。

你：@Nexus 模式 开发
Nexus：已切换至 💻 开发模式。
       激活：Coder, Reviewer, Miner
       降频：Editor, Quill, Pixel
       今日开发素材将自动积累至内容素材池。

你：@Nexus 模式 巡航
Nexus：已切换至 🛳️ 巡航模式。
       核心运行：OmniRadar, Nexus, Oracle
       其他 Agent 待命。
       你只需要看每日仪表盘。
```

5.3 自动模式建议

Oracle 会基于你的行为模式自动建议切换：

text

代码块

```
Oracle 在每日仪表盘中：

"💡 模式建议：你已连续 3 天处于巡航模式，
内容素材积压池有 8 条高分素材（3 条即将过期）。
建议明天切换至 📝 内容模式，消化积压素材。"

"💡 模式建议：Pulse 收到 5 条用户请求 PDF 导出功能，
Strategist 评估该功能可带来 $95/月 MRR 增长。
建议本周切换至 💻 开发模式，优先实现此功能。"
```

六、Oracle：无所不知的全局分析师

6.1 Oracle 的三重职能

Oracle 是整个系统的"后视镜"和"教练"：

text

```
职能一：内容追踪（原 Metric）
  追踪每篇内容的全平台数据
  分析什么选题/风格/时间表现最好
  反馈给 Editor 和 Quill

职能二：产品追踪（原 Pulse + 部分 Vitals）
  追踪用户反馈和产品数据
  追踪技术栈健康度
  反馈给 Compass 和 Coder

职能三：系统追踪（新增）
  追踪每个 Agent 的运行状态和成本
  检测异常行为
  生成每周安全复盘报告
  监控 ADR 触发条件
```

6.2 Oracle 的跨维度关联分析

Oracle 最独特的能力——也是合并三个追踪 Agent 的核心价值——是 **跨维度关联分析** 。

Markdown

代码块

```
# Oracle 跨维度分析规则

## 内容 × 产品
- 某篇技术文章带来了异常多的产品注册 → 通知 Editor 做系列
- 用户反馈中反复提到某个话题 → 提升该话题在内容素材池中的优先级

## 内容 × 技术
- 关于某项技术的文章阅读量持续增长 → 这项技术可能在上升期
- 关于某项技术的文章阅读量骤降 → 该话题可能已饱和

## 产品 × 技术
- 用户请求某功能 + 该功能依赖的技术健康分在下降 → 风险预警
- 竞品使用某技术 + 该技术 Star 增速很快 → 技术机会信号

## 内容 × 产品 × 技术（三维关联）
- 写了一篇技术文章 → 带来了产品注册 → 该技术也是你的栈
  → 建议：深耕这个方向，形成 "内容引流 → 产品转化 → 技术深化" 飞轮
```

6.3 Oracle 的输出

每日输出（21:00）：

Markdown

代码块

```
### 📊 Oracle 每日报告 — [日期]

#### 内容维度
- 今日发布：[文章标题] on [平台列表]
- 数据快照：阅读 X | 点赞 X | 收藏 X | 新增关注 X
- vs 均值：[+/-X%]
- 趋势：[↑上升 / →持平 / ↘下降]

#### 产品维度
- 今日新注册：X | 活跃用户：X | 付费转化：X
- 用户反馈：X 条（🐛X | 💡X | 👍X | 👎X）
```

```
- 紧急：[P0 Bug 或关键反馈，如有]

#### 技术维度
- 技术栈异动：[有/无] [详情如有]
- ADR 触发条件：[无触发 / 有触发 + 详情]

#### 系统维度
- 今日总成本：$X（预算 $Y，使用 Z%）
- Agent 运行状态：全部正常 / [异常详情]
- 安全事件：无 / [事件详情]

#### 🔮 跨维度洞察
[Oracle 发现的跨维度关联，如有]

#### 📌 明日建议
- 模式建议：[保持/切换]
- 重点关注：[什么值得你明天花 5 分钟看]
- 选题建议：[基于数据的内容方向建议]
```

每周输出（周日 20:00）：

Markdown

代码块

```
### 📊 Oracle 周报 — W[周号]

#### 一周数据总览
[各维度的周度汇总]

#### 信源质量审计
[哪些信源产出了最终被使用的信息，哪些应该降级]

#### 内容 ROI 排名
[本周各篇内容的投入产出比]

#### 产品进展摘要
[功能上线/Bug 修复/用户增长]

#### 技术健康变化
[技术栈各项目的健康分周度变化]

```

```
#### 系统健康报告
[成本/Agent 运行/安全审计]

#### 下周建议
[综合所有维度的行动建议]
```

七、完整时间线：Solo Company OS 的一天

7.1 巡航模式的一天

text

代码块

```
05:30  🛰️ OmniRadar 启动
       → 扫描 45 个信息源
       → 去重: 892 → 534 条
       → 每条打标签（personal/news/market/tech_trend/community/competitor）
       → 输出 raw-intel JSON
       ⏱️ 约 35 分钟

06:15  🧠 Nexus 启动
       → 读取 OmniRadar 输出
       → 四管道分拣
       → 生成统一仪表盘
       → 推送至 Telegram
       ⏱️ 约 12 分钟

06:30  📱 你收到一条 Telegram 消息：今日仪表盘
       → 3 分钟浏览
       → 看到 Redis 连续第 3 周健康分下降 🟡
       → 看到 Stripe AI Agent API 是个有趣的产品信号
       → 决定今天不需要做什么特别的（巡航模式足够）

19:00  ⛏️ Miner 低频运行
       → 检查今天有没有编码日记（你今天没写代码 → 无产出）

```

```
21:00  📊 Oracle 运行
       → 采集全维度数据
       → 生成日报
       → 推送至 Telegram

21:05  📱 你看日报
       → "今天没有发布内容，素材积压池 +2 条"
       → "DocFast 今日 1 个新注册，活跃 23"
       → "系统成本 $3.2，正常"
       → 回复 "OK"，关手机

——— 你的总决策时间：8 分钟 ———
```

7.2 内容模式的一天

text

代码块

```
05:30  🛰️ OmniRadar 启动 → 同上

06:15  🧠 Nexus 启动 → 同上 + 内容管道加权

06:30  📱 你看仪表盘
       → 注意到 2 条标了 [content📝] 的信息
       → "今天要消化素材，写一篇文章"

09:00  💻 你做日常工作 / 写代码（Coder 待命模式，不主动打扰）

19:00  ⛏️ Miner 全量运行
       → 提取今日开发素材（如有）
       → 整合 OmniRadar 中的内容素材
       → 更新素材积压池

19:30  📋 Editor 激活
       → 读取 Nexus 管道 C + Miner 素材卡 + Oracle 反馈
       → 6 维评分 → Top 3 推荐
       → 推送至 Telegram

19:45  📱 你看选题推荐
       → 回复 "1"（asyncio 深潜系列第 3 篇）
```

```
20:00  ✍️ Quill 激活
       → 读取选题 + 素材 + 系列上下文
       → 调研 → 大纲 → 初稿 → 自检 → 终稿
       ⏱️ 约 50 分钟

20:50  📱 你审阅终稿
       → "第二段太长，压缩。结尾加个下篇预告。"
       → Quill 修改 → 5 分钟后推送修订版
       → 你回复 "OK"

21:00  🎨 Pixel 激活
       → 生成封面 + 配图
       ⏱️ 约 10 分钟

21:10  📱 你选择封面方案 B → 确认

21:15  📊 Oracle 运行 → 日报

21:30  🚀 你手动发布到掘金 + 知乎
       → 使用 Scribe 预生成的多平台版本（直接复制粘贴）

——— 你的总决策时间：25 分钟 ———
——— 你的总写作时间：0 分钟（Quill 完成，你只审阅）———
```

7.3 开发模式的一天

text

代码块

```
05:30  🛰️ OmniRadar → 同上
06:15  🧠 Nexus → 同上 + 产品/技术管道加权

06:30  📱 你看仪表盘
       → 注意到 Pulse 反馈："5 个用户请求 PDF 导出"
       → 注意到 Oracle 跨维度洞察："该功能可带来 $95 MRR"
       → 回复 "@Nexus 模式 开发"

08:00  💻 你开始写代码
       → Coder 激活，辅助实现 PDF 导出功能
       → 每次重要提交，Coder 自动记录编码日记

```

```
11:30  📋 Reviewer 自动触发（PR 创建时）
       → Code Review + 安全检查

14:00  💻 继续编码

17:00  💻 功能完成，Push 代码

19:00  ⛏️ Miner 全量运行
       → 提取今天的编码日记（PDF 生成相关技术决策）
       → 评分：78 分（有文章潜力，但不急）
       → 加入素材积压池

19:30  📋 Scribe 低频运行
       → 检测到新功能完成
       → 自动更新 Changelog
       → 更新产品文档中的功能列表
       → 草拟发布公告（待你审阅）

21:00  📊 Oracle 运行 → 日报
       → "今日开发：PDF 导出功能完成"
       → "Miner 新增 1 条素材（PDF 技术决策），建议下周内容模式时消化"
       → "Scribe 已更新 Changelog，发布公告待审阅"

——— 你的总决策时间：10 分钟 ———
——— 你的总编码时间：~6 小时（核心工作）———
```

八、共享知识库：让一个子系统的洞察流向所有子系统

8.1 知识库架构

Solo Company OS 的核心创新之一是 **共享知识库** ——一个所有 Agent 都可以读取（部分可写入）的结构化知识存储。

text

代码块

```
shared-knowledge/
├── audience-profile.md          # 目标读者/用户画像（Editor, Quill, Scribe 引用）
├── brand-voice.md               # 品牌调性指南（Quill, Scribe 引用）
├── tech-stack.yaml              # 技术栈配置（OmniRadar, Oracle, Compass 引用）
├── product-context.md           # 产品定位和当前状态（Compass, Scribe, Oracle 引用）
├── competitor-map.md            # 竞品图谱（OmniRadar, Recon, Compass 引用）
├── content-calendar.json        # 内容发布日历（Editor, Quill, Oracle 读写）
├── series/                      # 系列连载规划
│   ├── python-async-deep-dive.json
│   └── ...
├── adrs/                        # 架构决策记录（Scribe 写入，全员读取）
│   ├── ADR-001.md
│   └── ...
├── topic-backlog.json           # 内容素材积压池（Miner 写入，Editor 读取）
├── feedback-trends.json         # 用户反馈趋势（Oracle 写入，Compass 读取）
└── signal-history.json          # 历史信号数据（Oracle 写入，OmniRadar 读取）
```

8.2 知识是怎么"流动"的

一个具体的例子：

text

代码块

```
Day 1（周一）：
OmniRadar 扫描到："Redis 核心维护者宣布离开"
→ 标签：[tech_trend, tech⚠️]
→ 进入管道 D（技术信号）
→ Nexus 在仪表盘中标注 ⚠️

Day 1：
Oracle 更新 signal-history.json：
  { "signal": "redis_maintainer_departure", "date": "2026-03-03", "strength": "medium" }

Day 3（周三）：
Oracle 的 Vitals 职能检测到 Redis 健康分从 75 降到 72
→ 结合 signal-history 中的维护者离开信号
```

```
→ 跨维度分析："Redis 维护者离开 + 健康分连续下降 = 强风险信号"
→ 在日报中高亮标注
→ 建议："启动 Compass 评估 Redis 替代方案"

Day 4（周四）：
你回复："@Nexus 评估 Redis"
→ Nexus 触发 Compass
→ Compass 读取：
  - Oracle 的 Vitals 数据（Redis 健康分趋势）
  - signal-history（维护者离开事件）
  - tech-stack.yaml（我们怎么用 Redis 的）
  - feedback-trends（用户有没有报告 Redis 相关问题）
→ 生成完整评估报告

Day 4：
你决定迁移到 Dragonfly
→ Compass 通知 Scribe 写 ADR
→ Scribe 写 ADR-008 存入 shared-knowledge/adrs/
→ Oracle 在 Vitals 监控列表中添加 Dragonfly，移除 Redis 至"已废弃"

Day 5（周五）：
Miner 读取编码日记：今天花了 3 小时做 Redis → Dragonfly 迁移调研
→ 评分：85 分（高价值技术决策素材）
→ 存入 topic-backlog.json
→ Editor 下周将推荐"为什么我们从 Redis 迁移到 Dragonfly"

Day 12：
Quill 写了这篇文章并发布
→ Oracle 追踪数据：阅读量是均值的 2.8 倍
→ Oracle 更新 feedback-trends："Redis 迁移话题用户关注度极高"
→ 建议 Editor："可以做一个'数据库选型'系列"

一条信息，从 OmniRadar 采集 → Nexus 分拣 → Oracle 分析 → Compass 评估
→ Scribe 记录 → Miner 提取 → Editor 推荐 → Quill 创作 → Oracle 追踪
→ 形成下一个循环。

这就是知识的流动。
```

九、成本总览

9.1 三种模式的月度成本

成本项	🚢 巡航模式	📝 内容模式	💻 开发模式
OmniRadar	$25-40	$25-40	$25-40
Nexus	$15-25	$15-25	$15-25
Editor	—	$8-15	—
Quill	—	$30-80	—
Pixel	—	$10-20	—
Miner	$5-10	$10-15	$10-15
Coder	—	—	$30-100
Reviewer	—	—	$5-15
Recon	—	—	$5-15 (按需)
Compass	—	—	$5-15 (按需)
Scribe	—	$5-10	$5-10
Oracle	$15-25	$15-25	$15-25
VPS	$10-15	$10-15	$10-15
日均	**$3-5**	**$8-15**	**$6-12**
月均	**$90-150**	**$150-280**	**$130-250**

9.2 实际混合模式月成本

一个典型月份：

- 巡航模式：15 天
- 内容模式：8 天
- 开发模式：7 天

text

```
15 × $4 + 8 × $12 + 7 × $9 = $60 + $96 + $63 = $219/月
```

$219/月——相当于一个中等价位的 SaaS 订阅费。 但你获得的是一个自动化的信息系统 + 内容工厂 + 开发助手 + 技术决策支持。

9.3 对比：如果不用 AI 系统

活动	你自己做的时间投入	有 Solo OS 后的时间	节省
每日信息浏览	30-60 分钟/天	3 分钟/天	27-57 分钟/天
内容创作	4-6 小时/篇	25 分钟/篇（审阅）	3.5-5.5 小时/篇
竞品调研	10-20 小时/次	1 小时审阅	9-19 小时/次
技术选型调研	5-10 小时/次	30 分钟审阅	4.5-9.5 小时/次
数据追踪	2 小时/周	5 分钟/天看报告	1.5 小时/周
月度时间节省			**40-80 小时**

$219/月 ÷ 40 小时 = **$5.5/小时的"虚拟员工"成本。**

世界上没有这个价格的人类员工。

十、完整目录结构

text

代码块

```
~/.openclaw/
├── openclaw.json                              # 主配置（12 Agent）
│
│   ═══════════ Layer 0: 统一感知 ═══════════
│
├── workspace-omniradar/
│   ├── SOUL.md
│   ├── sources.yaml                           # 统一信息源配置
│   └── skills/
│       ├── blogwatcher/
│       ├── airadar/
│       └── browser-automation/
│
│   ═══════════ Layer 1: 统一决策 ═══════════
│
├── workspace-nexus/
│   ├── SOUL.md
│   ├── pipeline-rules.md                      # 四管道分拣规则
│   ├── dashboard-template.md                  # 仪表盘模板
│   └── memory/
│       └── user-preferences.json              # 你的偏好（学习积累）
│
│   ═══════════ Layer 2: 专业执行 ═══════════
│
├── workspace-editor/
│   ├── SOUL.md
│   └── memory/
│
├── workspace-quill/
│   ├── SOUL.md
│   ├── tech-style-guide.md
│   └── memory/
│
├── workspace-pixel/
│   ├── SOUL.md
│   └── brand-assets/
│
├── workspace-miner/
│   ├── SOUL.md
│   └── desensitization-rules.md
```

```
41  │
42  ├── workspace-coder/
43  │   ├── SOUL.md
44  │   └── coding-standards.md
45  │
46  ├── workspace-reviewer/
47  │   └── SOUL.md
48  │
49  ├── workspace-recon/
50  │   └── SOUL.md
51  │
52  ├── workspace-compass/
53  │   ├── SOUL.md
54  │   ├── tech-evaluation-framework.md          # 技术评估框架
55  │   └── product-evaluation-framework.md       # 产品评估框架
56  │
57  ├── workspace-scribe/
58  │   ├── SOUL.md
59  │   ├── adr-template.md
60  │   ├── landing-page-templates/
61  │   └── launch-playbooks/
62  │
63  │   ═══════════ Layer 3: 统一回流 ═══════════
64  │
65  ├── workspace-oracle/
66  │   ├── SOUL.md
67  │   ├── cross-dimension-rules.md              # 跨维度分析规则
68  │   ├── security-audit-rules.md               # 安全审计规则
69  │   └── memory/
70  │       └── historical-data.json
71  │
72  │   ═══════════ 共享层 ═══════════
73  │
74  ├── shared-knowledge/                         # 共享知识库
75  │   ├── audience-profile.md
76  │   ├── brand-voice.md
77  │   ├── tech-stack.yaml
78  │   ├── product-context.md
79  │   ├── competitor-map.md
80  │   ├── content-calendar.json
81  │   ├── series/
82  │   ├── adrs/
83  │   ├── topic-backlog.json
84  │   ├── feedback-trends.json
85  │   └── signal-history.json
86  │
87  ├── shared-output/                            # Agent 输出目录
```

```
88  │   ├── omniradar/
89  │   ├── nexus/
90  │   ├── editor/
91  │   ├── quill/
92  │   ├── pixel/
93  │   ├── miner/
94  │   ├── coder/
95  │   ├── reviewer/
96  │   ├── recon/
97  │   ├── compass/
98  │   ├── scribe/
99  │   ├── oracle/
100 │   └── project-[项目名]/                    # 产品项目目录
101 │
102 ├── docker/                                  # Docker 配置
103 │   ├── docker-compose.yml
104 │   └── .env
105 │
106 ├── security/                                # 安全配置
107 │   ├── shell-whitelist.json
108 │   ├── shell-blacklist.json
109 │   ├── skill-hashes.json
110 │   └── weekly-review-template.md
111 │
112 └── logs/                                    # 审计日志
113     └── audit/
```

十一、渐进式搭建路线图

11.1 不要一次性搭建全部系统

Solo Company OS 有 12 个 Agent、4 层架构——但你不应该第一天就搭建全部。按以下路线图渐进式推进：

text

代码块

```
═══════════════════════════════════════════════════════════
Phase 1: 感知层（第 1-2 周）
  成本：$40-60/月
  Agent: OmniRadar + Nexus + Oracle
  你获得：每天一条仪表盘消息，掌控全局信息
═══════════════════════════════════════════════════════════

  这两周你只需要做一件事：每天早上看 3 分钟仪表盘。
  不需要做任何决策，不需要切换模式。
  目标：验证信息采集和分拣是否准确。
  调整：根据推送质量调整 OmniRadar 的信息源和 Nexus 的分拣规则。

═══════════════════════════════════════════════════════════
Phase 2: 内容轮（第 3-4 周）
  新增：Editor + Quill + Pixel
  成本：$100-180/月
  你获得：每周稳定产出 2-3 篇内容
═══════════════════════════════════════════════════════════

  尝试切换到 📝 内容模式运行一周。
  体验从仪表盘到选题到终稿的完整流程。
  调整：根据 Quill 的输出质量调整 SOUL.md 风格指南。

═══════════════════════════════════════════════════════════
Phase 3: 开发轮（第 5-6 周）
  新增：Coder + Reviewer + Miner
  成本：$130-250/月
  你获得：开发效率提升 + 开发活动自动产出内容素材
═══════════════════════════════════════════════════════════

  尝试切换到 💻 开发模式运行一周。
  体验 Miner 如何从你的编码活动中提取素材。
  调整：根据 Miner 的素材质量调整评分标准。

═══════════════════════════════════════════════════════════
Phase 4: 决策引擎（第 7-8 周）
  新增：Recon + Compass + Scribe
  成本：$150-280/月（按需部分仅在触发时产生费用）
  你获得：完整的竞品分析 + 技术/产品决策支持
═══════════════════════════════════════════════════════════

  找一个真实的决策场景（如技术选型）试用 Compass。
  让 Scribe 写第一份 ADR。
  体验决策过程被完整记录的价值。

```

```
═══════════════════════════════════════════════════════════════
Phase 5: 全系统联动（第 9 周起）
  全部 12 Agent 在线
  三种模式自由切换
  Oracle 跨维度分析开始产出价值
  系统进入自进化状态
═══════════════════════════════════════════════════════════════
```

11.2 每个 Phase 的成功标准

Phase	成功标准	如果没达标
Phase 1	连续 7 天每天的仪表盘你都觉得"有用"	调整信息源和分拣规则
Phase 2	成功发布 2 篇你满意的文章	调整 Quill 的风格指南和 Editor 的评分标准
Phase 3	Miner 至少提取了 3 条你认同的高价值素材	调整 Miner 的评分标准和 Coder 的日记规则
Phase 4	完成 1 次你满意的 Compass 评估	调整评估框架的维度和权重
Phase 5	Oracle 产出了至少 1 条你没预想到的跨维度洞察	需要更多数据积累，耐心等待

十二、这套系统的长期复利

12.1 第 1 个月：磨合期

系统刚搭好的第一个月，体验会比较粗糙。OmniRadar 可能推了太多不相关的信息。Quill 写出来的文章风格不太对。Miner 的素材评分不够准确。

这是正常的。你在这个月做的每一次反馈——"这条信息不相关"、"这篇文章的结尾太生硬"、"这个素材评分应该更高"——都在训练系统。

12.2 第 3 个月：飞轮启动

Oracle 的跨维度分析开始产出真正有价值的洞察。因为它现在有了 3 个月的历史数据。

你会开始体验到"意外收获"——Oracle 告诉你一些你自己没注意到的关联：

text

```
"你上月发布的 3 篇 Python 异步文章累计带来了 127 个产品注册（是平均的 4.2 倍）。
同时 Python 在 Stack Overflow 的月度新问题数连续 6 个月增长。
建议：深耕 Python 方向的内容，并考虑开发一个面向 Python 开发者的工具。"
```

一个人做不出这种跨维度的洞察——因为你的大脑不可能同时追踪内容数据、产品数据和技术趋势数据的关联。但 Oracle 可以。

12.3 第 6 个月：系统成为竞争壁垒

半年后，你的系统积累了：

- 180+ 天的信息信号历史
- 50+ 篇内容的全平台数据
- 完整的技术栈健康档案
- 所有重要决策的 ADR 记录
- 经过校准的素材评分模型
- 了解你写作风格的 Quill

- 了解你决策偏好的 Nexus
-

这些 **积累** 本身就是竞争壁垒。你的竞争对手今天开始搭建同样的系统，也需要 6 个月才能达到你现在的水平。

12.4 第 12 个月：复利全面开花

text

```
内容维度：
• 发布了 80-120 篇高质量文章
• 有 3-5 个完整的系列连载
• 搜索流量每月 10 万+ PV
• 成为某个技术方向的"知名博主"

产品维度：
• 至少 1 个产品上线并有付费用户
• 完整的用户反馈循环在运转
• 产品迭代速度是竞争对手的 2-3 倍

技术维度：
• 技术栈每季度有雷达更新
• 所有重大决策有 ADR 记录
• 技术债务被持续监控和偿还

系统维度：
• Oracle 的跨维度洞察越来越精准
• 你每天只需要花 10-25 分钟做决策
• 系统每月自我优化（信源质量、模型选择、成本控制）
```

一个人，一年时间，$2,500-3,500 的 AI 成本（约 ¥18,000-25,000），建立了相当于一个 5-10 人小团队的运营能力。

十三、安全配置总结

Solo Company OS 整合了所有子系统，安全面也随之扩大。在启动前，确保 X-07 的安全配置已经就位。

13.1 最小安全配置（必须）

Markdown

```
在启动 Phase 1 之前，必须完成：

✅ 铁律 #1:  专用 OS 用户
✅ 铁律 #2:  Docker 容器隔离（12 个容器）
✅ 铁律 #5:  每个 Agent 独立 API Key
✅ 铁律 #6:  Telegram 白名单
✅ 铁律 #7:  Shell 命令审批
✅ 铁律 #10: 审计日志
✅ 铁律 #16: 成本上限 + 熔断器
```

13.2 Oracle 安全职能

Oracle 合并了安全审计职能。每周自动执行：

Markdown

```
Oracle 安全审计（每周日 20:00）：

1. 审计日志异常检测
2. Skill 哈希验证
3. 成本异常分析
4. Agent 运行状态检查
```

```
5. ADR 触发条件检查
6. 生成安全周报

输出: shared-output/oracle/security-weekly-[日期].md
```

尾声：一个人就是一支军队——但这支军队为谁而战？

读到这里，你已经拥有了一个完整的操作系统。12 个 Agent、4 层架构、3 种运行模式、共享知识库、统一仪表盘、安全防线、成本控制。

技术上，它已经完备了。

但在你按下 `docker-compose up` 之前，我想请你思考一个问题：

这支"一人军队"要去往哪里？

系统可以帮你高效地做很多事——采集信息、写文章、开发产品、做技术决策。但"高效地做错误的事"比"低效地做正确的事"更危险。

你多出来的 40-80 小时/月，用来做什么？

- 如果你用它来发更多低质量的内容——你只是更快地消耗了读者的信任
- 如果你用它来同时做 5 个产品——你只是更快地做了 5 个半成品
- 如果你用它来追逐每一个市场信号——你只是更快地迷失了方向
-

Solo Company OS 是一个放大器。 它放大你的判断力——如果你的判断是对的，它帮你更快地到达目的地；如果你的判断是错的，它帮你更快地走进死胡同。

所以，这个系统中最重要的组件，不是 OmniRadar，不是 Nexus，不是 Oracle——

是你。

你是唯一不能被自动化的部分。你决定做什么、不做什么、为谁做、做到什么程度。

系统帮你看清了路。但选择走哪条路——这永远是你的事。

我花了 8 篇文章、超过 10 万字，从一个简单的内容流水线（X-01）一路搭建到一个完整的一人公司操作系统（X-08）。

但它的真正意义不是这些技术细节——而是一个信念：

在 AI 时代，一个有想法、有判断力的个体，应该拥有和团队一样的执行力。

不是因为"团队不重要"——团队永远重要。而是因为在你还没有团队的阶段、在你验证想法的阶段、在你一个人扛着梦想往前跑的阶段——你不应该被"执行力"卡住。

你应该被"想法"和"判断"卡住——因为那才是真正难的、真正值得你花时间的事情。

剩下的，交给系统。

📦 完整系列资源汇总

资源	说明
🎁 Solo Company OS 完整配置包	openclaw.json + docker-compose.yml + 全部 SOUL.md

🎁 OmniRadar sources.yaml 模板	45 个信息源的统一配置
🎁 Nexus 仪表盘模板 + 交互命令清单	含三种模式的切换配置
🎁 Oracle 跨维度分析规则	含所有关联分析配置
🎁 共享知识库模板全套	11 个文件的完整空白模板
🎁 渐进式搭建脚本	Phase 1-5 的自动化安装脚本
🎁 X-01 至 X-08 全系列 PDF 合集	离线阅读版
🎁 七模块 SOUL.md 设计框架速查卡	可打印 A4
🎁 20 条安全铁律速查卡	可打印 A4
🎁 Solo Company OS 架构海报	高清可打印的系统架构图

OpenClaw 实战工作流

编号	标题
X-01	**《终极工作流：情报聚合 → 选题决策 → 内容生产 → 发布分发 → 数据复盘，5 个 Agent 跑完一整条链》**
X-02	**《晨报 + 情报站 + 内容工厂：三套系统联动后，我的自媒体上了一个台阶》**
X-03	**《开发者也做自媒体：技术博客 × 开发效率的双轮驱动工作流》**
X-04	**《竞品情报 → 产品决策 → 自动开发 → 发布：一个创业者的全 AI 工作流》**
X-05	**《技术趋势监控 + 自动技术选型：CTO 的 AI 副驾驶》**
X-06	**《4 大系列 × 1 个 SOUL.md 框架：如何设计通用的 Agent 人格系统》**
X-07	**《安全终极指南：4 大场景 × 20 条安全铁律》**

X-08	**《从零到全自动：一个人用 OpenClaw 重新定义"一人公司"》**

从 X-01 的第一个 Agent 到 X-08 的完整操作系统——感谢你读到了这里。

_下一步不是继续读——是打开终端，输入第一条命令。 🦞