如何用 Claude Code 提升开发效率

谢威宇

Madsys & 趋境科技

2025 年 10 月

目录

开场白

我的 Claude Code 使用体验

今年最幸运的一件事

用上 Claude Code 是我今年最幸运的事。它给我的震撼不亚于、甚至超过了当初初次使用 ChatGPT 3.5 时的感受。

使用时间：从 8 月至今约 2 个月

完成的项目：

- 系统级代码开发
- 运维自动化
- 各类小工具
- 项目管理工作

核心价值

让我有能力实现各种想法，效率提升 5-10 倍

Claude Code 是什么

Claude Code 定位

核心功能

- AI 驱动的编程助手
- 命令行界面操作
- 深度集成文件系统
- 自主执行任务

和其他工具的区别

- vs IDE：理解更智能
- vs Copilot：能完成整个任务
- vs ChatGPT：直接操作代码

最大优势

基于命令行和文件系统，能拿到海量的上下文信息

核心优势

最优协作模式

- AI：处理海量背景信息
- 人类：提供反馈和创意
- 基于文件系统，上下文获取能力极强

真正能干活

- 不只是提建议，直接动手
- 自动拆解多步骤任务

工具集成

- 编辑器、终端
- Git、包管理器
- 构建、测试工具
- 无缝切换

本质

AI 和人站在同一信息起跑线，充分发挥 AI 实力

输入方式的优化空间

当前瓶颈

- 工位上主要靠打字输入
- 打字速度有限
- 思维 $\rightarrow$ 文字转换慢

语音交互的潜力

- 语音速度远超打字
- 更自然的表达方式
- 解放双手继续操作

洞察

输入方式成为人机协作的效率瓶颈

未来方向

语音 + Claude Code = 更高效的协作方式

注：需要考虑工作环境的噪音和隐私问题

核心能力概览

Claude Code 能做什么

代码开发

- 理解项目结构和代码
- 智能补全和重构
- 诊断和修复 bug
- 性能优化

项目管理

- Git 操作（提交、分支、PR）
- 依赖和构建管理
- 自动化测试

任务协作

- 自动拆解任务
- 追踪执行进度
- 并行处理优化

重点

具体能力通过真实案例来展示更直观

真实案例

案例 1：代码分析 - AWQ 量化格式研究

任务

在 sglang 代码库里找到 AWQ 量化的 swizzle 格式排列方式

挑战

- 陌生代码库
- 非标准排布方式
- 论文描述不清

效率

- Claude Code：几分钟
- 传统方式：几小时

价值

快速理解陌生代码，精准定位关键实现

案例 2：系统级开发 - KTransformers 支持 AWQ

任务

- 实现 group 量化
- 高维矩阵切分
- AVX 指令优化

协作

- AI：写测试、生成框架
- 人：调试关键指令

效率提升

2 周 → 2 天

启示

复杂系统级开发也能高效完成

案例 3：运维 - GPU 监控中心部署

任务

- 多 GPU 集群管理
- DCGM + Grafana
- Docker 容器化
- 批量远程部署

效率

4 小时 vs 2-3 天

价值

- 运维自动化
- 快速部署

案例 4 & 5：文档与管理自动化

立项文档

- 背景资料 → MD
- AI 生成初稿
- 调整后转 Word
- 专注内容不是格式

项目进度管理

- 自动化进度更新
- 多项目信息聚合
- 自动生成日报周报
- 数据可视化

效果

文档编写快速无痛

洞察

系统化、自动化管理工作

最佳实践

怎么问问题

1. 清楚地描述
   - 说清楚想要什么结果
   - 描述现在是什么状态
   - 贴上错误信息

2. 给足上下文
   - 指出相关文件
   - 说明用的什么技术栈
   - 讲清楚有什么限制

3. 一步步来
   - 把复杂任务拆开
   - 每步都验证结果
   - 及时给反馈

优化工作流

工具集成

- Git
- IDE
- 终端
- CI/CD

团队协作

- 代码规范
- 文档同步
- 知识共享
- 最佳实践

提效技巧

- 快捷命令
- 用模板
- 批量操作
- 自动化脚本

挑战与解决方案

上下文限制问题

现状

Claude Code 的上下文一般在几百 K 级别，开发大项目时可能一个特性就用完

解决方案

- 让 AI 写文档记录状态
- 定期重构项目结构
- 模块化开发
- 分阶段处理任务

为什么不能简单扩展

- 长上下文成本太高
- 信息检索效率降低
- 多 agent 是更优解

未来趋势

多 agent 系统会成为主流（Haiku 4.5 已采用）

核心思考

Claude Code 的本质

信息处理加速器

任何需要快速处理和输出信息的工作都可以交给它

优化流程的思维

人负责思考整个流程哪里可以优化

解放生产力

从重复的信息处理工作中解放出来，专注创造性思考

成本与价值

官方订阅

- 月费：200 美元
- 效率提升：5-10 倍
- 学习成本低

实际挑战

- 个人负担较重
- 支付手段受限
- 可考虑拼单

性价比

收益远超成本

建议

公司投资 / 团队分摊 / 用替代方案

经济替代方案

GLM-4.6

- 60 元/季度
- Function call 榜单第一
- 日常使用足够

Kimi K2

- 性能更强
- 价格适中

团队分摊

- 5 人：40 美元/月
- Claude Relay
- 需运维能力

推荐

预算有限优先 GLM-4.6，团队可考虑分摊方案

Claude Relay Service: `https://github.com/Wei-Shaw/claude-relay-service`

AI 能力的本质思考

框架也是能力的一部分

Claude Code 证明了 AI 的能力不完全由基础模型决定

传统观点

- AI 能力 = 模型能力
- 更大的模型 = 更强的能力
- 追求参数规模

新的认知

- 工作范式很重要
- 框架设计是核心
- 多 agent 协作

实证

Haiku 4.5 发布后，看代码功能完全用子 agent 实现

多模型协作的未来

Claude Code 的实践

- 小模型：快速阅读代码
- 大模型：深度思考总结
- 多 agent 分工协作
- 任务自动分配

对系统的要求

- 多模型调度能力
- 任务智能分配
- 统一接口标准
- 成本优化策略

洞察

不同任务用不同模型，性价比最优

趋势

多模型场景会越来越常见

对程序员角色的影响

技能广度 vs 技能深度

Claude Code 显著增加了程序员的技能广度

1. **分工方式的变化**
   - 从技能出发 → 从需求出发
   - 更接近产品经理的工作方式
2. **FAE 角色越来越重要**
   - 现场应用工程师（Field Application Engineer）
   - 理解客户需求
   - 转化成 AI 能处理的形式
3. **变革正在发生**
   - 从一个人用到许多人用
   - 当共识形成，下一步变革就会开始

Agent 应用：爆发前夜

Claude Code 的启示

- 第一个 Killer App
- 处于爆发前夜
- 主要用户：程序员

更广阔的需求

- 其他人也需要智能 agent
- 知识库管理
- 专业领域应用
- 各行各业的效率工具

市场机会

Agent 应用爆发 → AI Infra 需求爆发式增长

基础设施需求

- 多模型调度平台
- 高性能推理引擎
- 成本优化方案
- 企业级部署

趋境科技

在 AI Infra 领域占有一席之地

总结与展望

核心价值

- 全新协作范式
- 效率提升 5-10 倍
- 适用信息密集型工作

未来方向

- 多 agent 系统成熟
- 降低技术门槛
- 重塑团队结构

变革正在发生

从开发、运维到管理，工作方式正在改变

愿景

解放更多人的生产力

Q&A

Q&A

提问与讨论

感谢聆听!

谢谢！

本演示文稿使用 Claude Code 制作