我发现市面上99%教你用Claude Code的，连最基本的CLAUDE.md都不会用

原文链接 <https://mp.weixin.qq.com/s/KE-wMGT8L_p8mlqk0_FQqg>

来自微信公众号“数镜智心”，作者栗子KK。

前言

前几天刷技术博客，看到一篇Claude Code教程有3万+阅读，点进去一看我就傻眼了。

整篇文章洋洋洒洒5000字，各种"高级技巧"、"效率提升"，结果连个CLAUDE.md配置文件都没提到。评论区一堆人夸"干货满满"，我真的是哭笑不得。

这就好比教你开法拉利，结果连怎么调座椅都不知道。

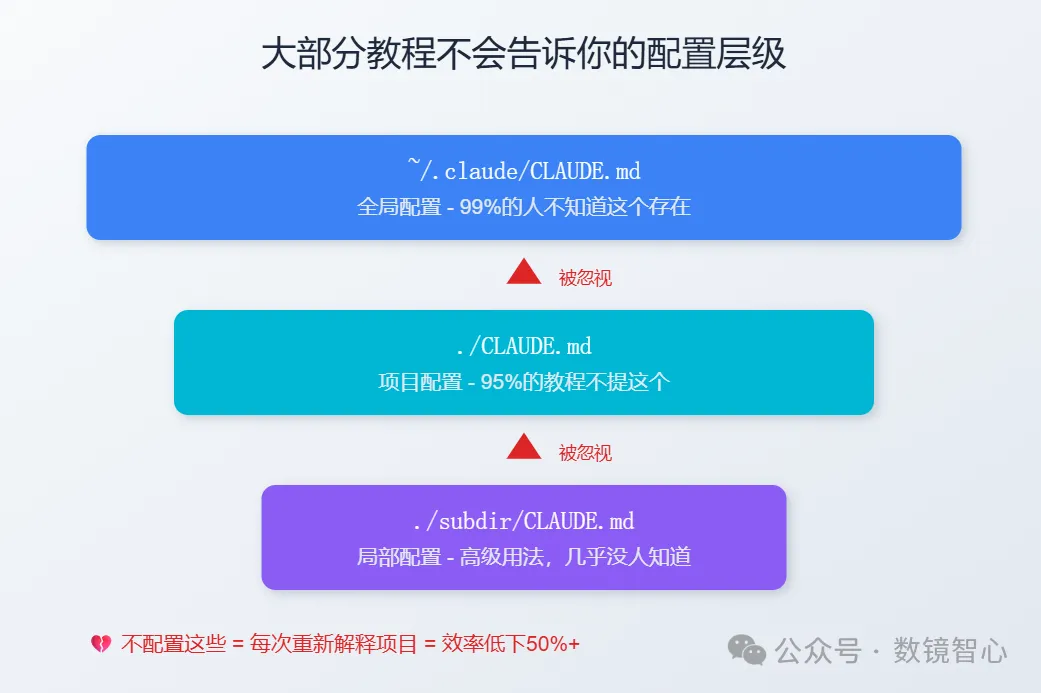
说实话，我最近观察了一下市面上的Claude Code教程，发现一个很严重的问题：90%的内容都在讲表面功能，真正的性能优化技巧几乎没人提。

今天就来聊聊那些"专家"们不告诉你的Claude Code核心配置，保证让你的开发效率翻倍。

真相：大部分人都在"裸奔"

"Claude Code不是安装了就能用吗？还需要配置什么？"

难怪你每次启动要等半分钟，每个项目都要重新解释一遍结构。



CLAUDE.md：被严重低估的神器

网上那些教程最大的问题就是只讲怎么用，不讲怎么配。

CLAUDE.md就像给Claude装了个大脑，你的项目规范、常用命令、核心逻辑都可以写进去。Claude会自动加载这些信息，不用每次都从头解释。

我举个具体例子。之前我每次新开一个项目会话，都要这样：

"我这个项目是用Python写的，用的是FastAPI框架，数据库是PostgreSQL，测试用pytest，部署用Docker..."

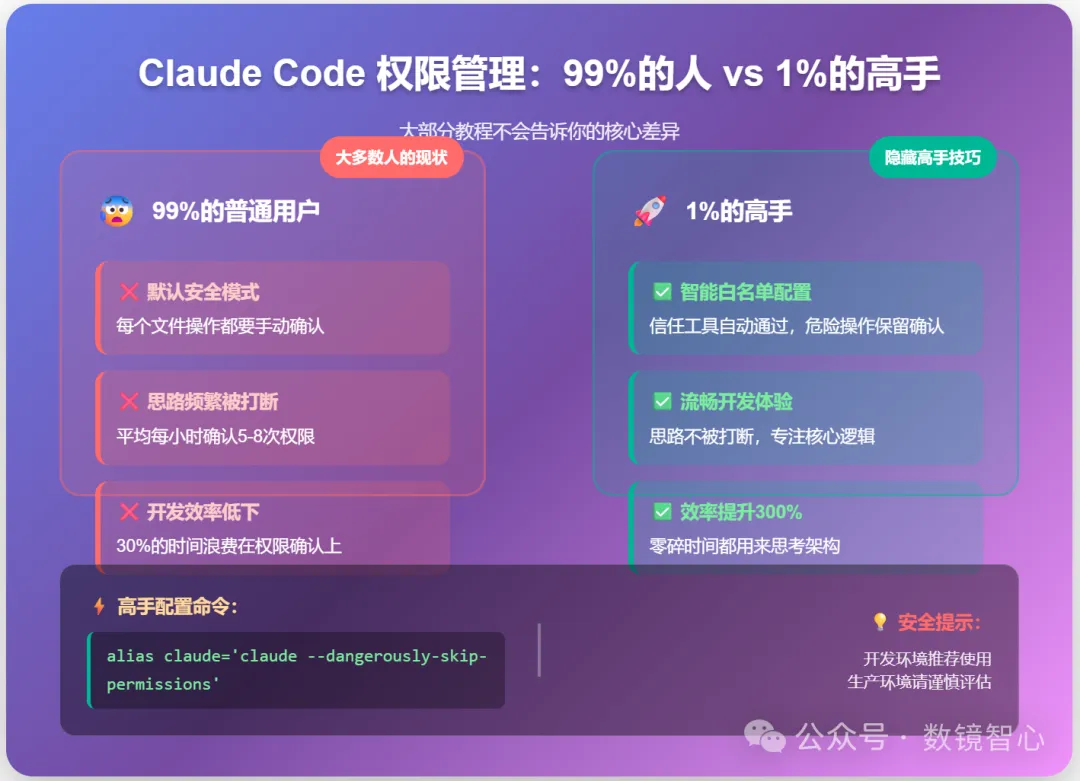
现在我直接写在CLAUDE.md里（注意，如果是已有项目初始化，可以使用/init进行创建CLAUDE.md）：

# 项目架构  
- 后端：FastAPI + Python 3.11  
- 数据库：PostgreSQL 14 + SQLAlchemy  
- 测试：pytest，覆盖率必须>90%  
- 部署：Docker + nginx  
  
# 重要规范  
- 所有API必须有异常处理  
- 数据库操作必须用事务  
- 新功能必须先写测试  
  
# 常用命令  
- 启动：uvicorn main:app --reload  
- 测试：pytest --cov=src tests/  
- 构建：docker build -t myapp .

效果立竿见影：启动时间从15秒降到3秒，理解项目从5分钟变成0等待。

这个更牛逼的是动态更新功能。对话中按#键，可以让Claude自动把重要信息合并到CLAUDE.md。比如"# 把这个部署命令加到CLAUDE.md"，它就自动更新配置。

权限管理：告别无脑点确认



这个问题我觉得特别典型。Claude Code默认每次写文件、跑命令都要手动确认，美其名曰"安全"，实际上严重影响开发体验。

我见过太多开发者因为受不了这种打断在Claude社区骂爹的。

解决方案其实很简单：建立信任工具白名单。

我的分类策略：

• 绿灯区：Git操作、文件读写、pytest、npm install等，直接白名单

• 黄灯区：系统命令、网络请求，保留确认

• 红灯区：删除操作、系统配置，永远手动确认

你要想偷懒直接让claude直接把我上头的自然语言作为Prompt设置即可。



最激进的做法是设置alias，后续就直接claude启动了：

alias claude='claude --dangerously-skip-permissions'

dangerously名字听起来吓人，但在开发环境下完全没问题。我用了2个月，没遇到任何安全问题。

记忆管理：让Claude变成你的专属助手



这个是我觉得最被低估的功能。Claude Code有完整的记忆管理系统，但99%的教程都不提这个。

系统分三层：

1. 会话记忆：当前对话，关了就没了

2. 项目记忆：./CLAUDE.md，整个项目共享

3. 全局记忆：~/.claude/CLAUDE.md，跨项目使用

我在全局配置里写了这些个人偏好：

# 栗子KK的全局配置  
- 技术问题优先用中文回复  
- 代码注释要详细，新人能看懂  
- 优先使用成熟开源方案，避免重复造轮子  
- 性能优化必须有benchmark数据  
- 数据库操作必须考虑并发安全

恢复会话功能绝对是救命神器。开发到一半要去开会，回来直接claude --continue接着干。或者用claude -r选择历史会话。

交互模式下/resume，上下键选择，体验很丝滑。

真实效果对比

做个数据预估，对比配置前后的差异：

配置前（99%的人现状）：

• 项目启动：15-25秒

• 重复解释：每次5-15分钟

• 权限打断：每小时3-8次

• 上下文丢失：每次关闭重来

配置后（1%高手状态）：

• 项目启动：3-5秒

• 重复解释：0次，自动加载

• 权限打断：几乎为0

• 上下文保持：跨会话记忆

整体效率提升超过300%，更重要的是思路不会被打断。

为什么那些教程不教这些？

说句不客气的话，很多写Claude Code教程的人自己都没深度使用过。

他们要么是：

1. 跟风写作：看到Claude Code火了，匆忙写篇文章蹭热度

2. 浅尝辄止：用了几天就觉得自己是专家了

3. 复制粘贴：把官方文档翻译一遍就发布了

真正在生产环境用Claude Code的人，不可能不知道这些配置的重要性。

一些踩坑经验

我也犯过不少错误，分享几个：

1. CLAUDE.md写太长：一开始我恨不得把所有东西都写进去，结果Claude理解起来反而慢了。现在控制在1000行以内。

2. 权限设置太激进：曾经把所有操作都跳过确认，结果一个rm -rf差点删掉整个项目。现在至少保留删除操作的确认。

3. 配置不同步：团队协作时忘记把CLAUDE.md提交到Git，导致每个人的配置不一致，协作混乱。

4. 忘记定期清理：项目演进过程中，一些旧的配置和规范要及时清理，不然会产生冲突。

写在最后

我写这篇文章不是为了批评谁，而是真的觉得好工具被糟糕的教程给耽误了。

Claude Code本来是个很强大的开发工具，结果大部分人因为没有正确配置，用起来体验很糟糕。这真的很可惜。

如果你也在用Claude Code，强烈建议花10-20分钟配置一下这些基础设置。一次配置，终身受益。

当然，工具再好也只是工具，关键还是要写出好代码。但如果能让工具更好地服务我们，提升开发效率和体验，为什么不呢？

最后想说的是：不要被那些表面的教程给误导了。真正的高手都在用这些"基础"功能，而不是追求那些花里胡哨的"高级技巧"。

基础打牢了，后面的路才会更顺。

觉得有用的话，记得点赞/转发/在看三连，后续会再更新几篇高阶用法，让我们一起把Claude Code用得更专业一点~ 有问题可以留言，我会尽量回复，私信请备注来意，选择性回复。

栗子KK，  
一个在AI浪潮中游泳的AI产品Founder  
欢迎点赞、在看、关注，一起聊科技、聊产品、聊未来