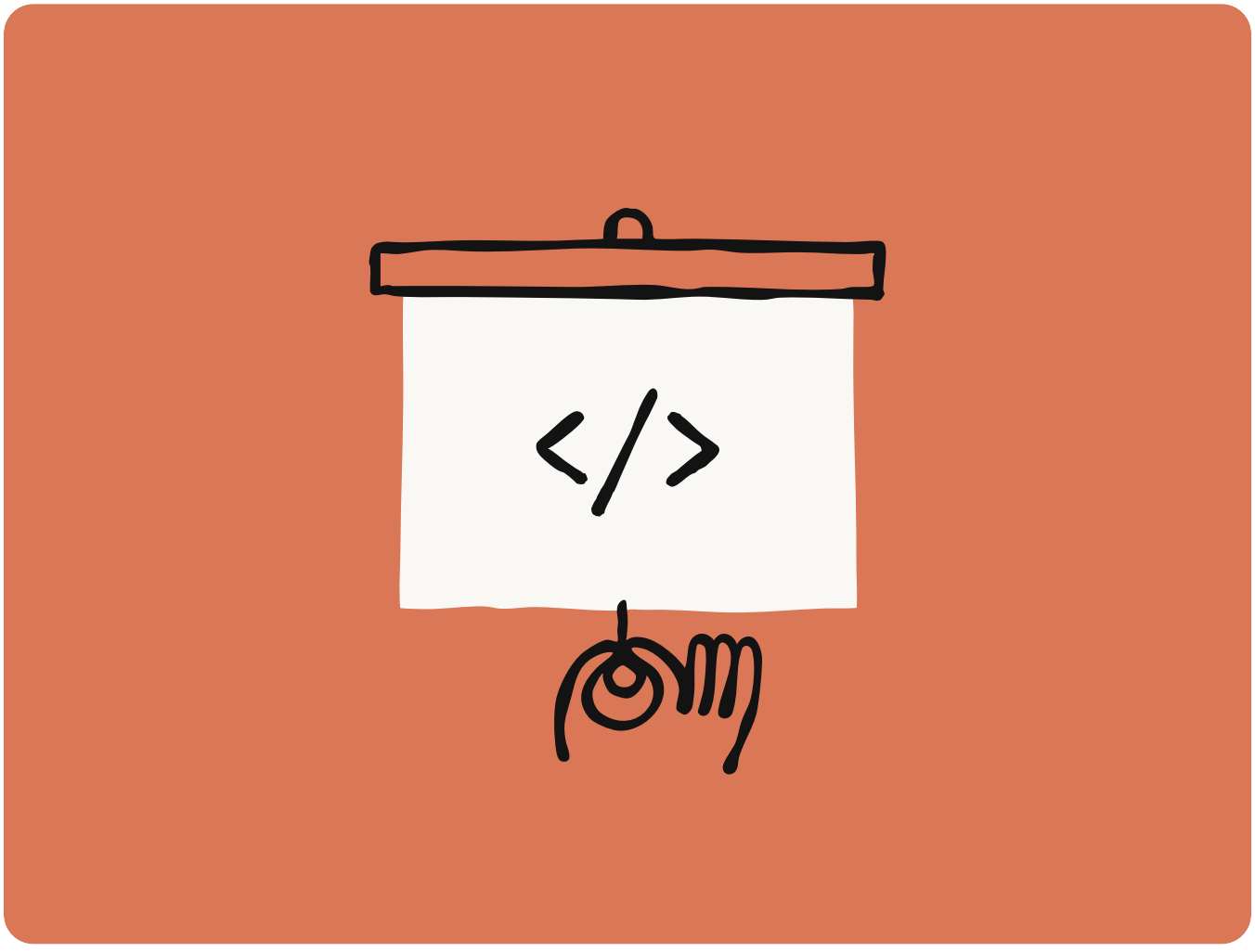
ANTHROP\C

# Anthropic团队如何运用Claude Code



Anthropic内部团队正通过Claude代码革新工作流程，使开发人员和非技术人员能够处理复杂项目、实现任务自动化，并弥合曾限制其生产效率的技能鸿沟。

通过访谈我们内部的Claude代码核心用户，我们收集了不同部门如何利用Claude代码、其工作影响以及给考虑采用该技术的组织的建议等多维度洞见。

# 目录

# Claude代码在数据基础设施中的应用

数据基础设施团队负责整合全公司的业务数据，为各部门提供统一支持。他们使用Claude Code来自动化常规数据工程任务、排查复杂基础设施问题，并为技术和非技术团队成员创建文档化工作流程，使其能够独立访问和操作数据。

## Claude Code主要应用场景

**附带截图的Kubernetes调试**

当Kubernetes集群宕机且无法调度新Pod时，团队借助Claude Code诊断问题。他们将仪表板截图输入Claude Code，系统通过逐步指导Google Cloud用户界面的各个菜单，最终发现提示Pod IP地址耗尽的告警信息。Claude Code随即提供创建新IP池并添加到集群的具体命令，省去了联系网络专家的环节。

**财务团队纯文本工作流程**

团队向财务人员演示如何编写描述数据工作流程的纯文本文件，将其导入Claude Code即可实现全自动执行。无编程经验的员工只需描述步骤（如"查询该仪表板→获取信息→执行这些查询→生成Excel输出"），Claude Code就能自动执行完整流程，包括要求输入日期等必要参数。

**新员工代码库导航指南**

当新数据科学家加入团队时，他们会被引导使用Claude Code来导航庞大的代码库。Claude Code会读取他们的Claude.md文档文件，识别特定任务相关的文件，解释数据管道依赖关系，并帮助新人理解哪些上游数据源为仪表板提供支持。这替代了传统的数据目录和可发现性工具。

**会话结束时的文档更新**

团队要求Claude Code在每次任务结束时总结已完成的工作会话并提出改进建议。这形成了持续改进闭环：Claude Code基于实际使用情况帮助优化Claude.md文档和工作流程说明，使后续迭代更加高效。

**跨多实例的并行任务管理**

在处理长期运行的数据任务时，他们会在不同项目的代码仓库中开启多个Claude Code实例。每个实例都保持完整上下文，因此当工程师在数小时或数天后切换回来时，Claude Code能准确记忆其工作进度，实现真正的无上下文丢失并行工作流程管理。

# Claude代码在数据基础设施中的应用

## 团队影响力提升

**无需专业背景即可解决基础设施问题**
通过Claude代码诊断问题并提供精确修复方案，解决通常需要系统或网络团队成员介入的Kubernetes集群问题

**加速新成员融入**
数据分析师与团队新成员无需大量指导即可快速理解复杂系统，并作出实质性贡献

**支持流程优化升级**
可处理海量数据并识别异常指标（如同时监测200个数据看板），这种规模的人工审核已超出人类能力范畴

**实现跨团队自助服务**
零编码经验的财务团队现可独立执行复杂数据工作流程

## 来自数据基础设施团队的顶级建议

**编写详尽的Claude.md文档**
在Claude.md文件中对工作流、工具和预期目标记录越细致，Claude代码的执行效能就越高这种模式使Claude代码在常规任务中表现卓越，例如基于现有模板快速搭建新数据管道

敏感数据处理建议使用MCP服务器替代CLI 团队建议采用MCP服务器而非BigQuery命令行界面，以加强对Claude代码访问权限的安全管控，特别是在处理需要审计记录或涉及隐私风险的敏感数据时

**共享团队使用案例会议**
团队举办内部研讨会，成员们互相演示各自使用Claude Code的工作流程。这种交流有助于推广最佳实践，并展示了用户可能尚未自行发现的多种工具使用方法。

# Claude Code赋能产品开发

Claude Code团队使用自家产品进行功能迭代，持续扩展产品的企业级功能与代理循环能力。

## Claude Code主要应用场景

**自动接受模式加速原型开发**

工程师通过启用"自动接受模式"（shift + tab）建立自主循环流程，利用Claude进行代码编写、测试运行和持续迭代，实现快速原型开发。工程师向Claude提出不熟悉的抽象问题，让其自主运行解决问题，在80%完成度时介入审查解决方案，最后进行最终优化。

团队强调保持git仓库初始洁净状态，并定期提交检查点，确保在Claude偏离轨道时可快速回撤错误变更。

**同步协作开发核心功能**

对于涉及应用程序业务逻辑的关键功能开发，团队与Claude Code保持同步协作，通过包含具体实施指令的详细提示词进行交互。他们通过实时监控流程来确保代码质量、风格指南合规性及架构合理性，同时将重复性编码工作交由Claude处理。

**构建Vim模式**

其最成功的异步协作项目之一是为Claude Code实现Vim键位绑定功能。团队要求Claude完整构建该功能（尽管优先级不高），最终实现方案约70%的内容由Claude自主完成，仅需少量迭代即告完成。

**测试用例生成与缺陷修复**

团队在功能实现后使用Claude Code编写全面测试用例，并处理代码审查中发现的基础缺陷修复。

他们还利用GitHub Actions集成，使Claude能自动处理Pull Request中诸如格式问题或函数重命名等评审意见。

**代码库探索**

当处理陌生代码库（如单体仓库或API端）时，团队通过Claude Code快速理解系统运行机制。团队成员直接询问Claude获取系统解释和代码索引，而非等待Slack回复，这显著降低了上下文切换的时间成本。

# Claude Code赋能产品开发

## 团队影响力提升

**加速功能实现**

已成功实现Vim模式等复杂功能，其中70%的代码由Claude自主编写完成。

**提升开发效率**

可快速构建功能原型并迭代创意，无需深陷具体实现细节。

**通过自动化测试提升代码质量**

Claude能生成全面测试用例并处理常规缺陷修复，在降低人工成本的同时保持高标准。

**优化代码库探索效率**

团队成员无需等待同事回复即可快速理解单体仓库中的陌生模块。

## Claude Code团队的最佳实践建议

**建立自洽工作闭环**

配置Claude自动执行构建、测试和代码规范检查，实现自我验证。该机制使Claude能更长时间自主工作并自查错误，特别是在要求Claude先编写测试用例再开发代码时效果显著。

**培养任务分类直觉**

需区分适合异步处理的任务（外围功能、原型设计）与需要同步监督的任务（核心业务逻辑、关键修复）。 产品边缘的抽象任务可通过"自动接受模式"处理，而核心功能仍需严格的人工监督。

**构建清晰、详尽的指令**

当组件名称或功能相似时，必须在请求中极度明确具体细节。提示语质量越高、细节越丰富，就越能放心让Claude独立工作，避免对代码库的错误部分进行意外修改。

# Claude Code在安全工程领域的应用

安全工程团队专注于保障软件开发生命周期安全、供应链安全及开发环境安全。该团队广泛使用Claude Code进行代码编写与调试工作。

## Claude Code主要应用场景

**复杂基础设施调试**
处理事故时，团队向Claude Code输入堆栈追踪和文档资料，要求其追踪代码库中的控制流程。该技术显著缩短生产问题的解决时间，将原本需要10-15分钟人工代码审查的问题理解过程压缩至约5分钟。

**Terraform代码审查与分析**
对于需要安全审批的基础设施变更，团队将Terraform方案导入Claude Code询问「这会产生什么影响？」 我会后悔这个决定吗？这种模式形成了更紧密的反馈循环，使安全团队能够更快速地审查和批准基础设施变更，从而减少开发流程中的瓶颈。

**文档整合与运维手册**
他们让Claude Code整合多个文档源，生成Markdown格式的运维手册、故障排除指南和系统概览文档。
将这些精炼文档作为调试实际问题的上下文依据，相比检索完整知识库，创造了更高效的工作流程。

**测试驱动开发流程**
团队摒弃了原先「设计文档→临时代码→重构→放弃测试」的模式，改为要求Claude Code生成伪代码，引导其进行测试驱动开发，并在遇到瓶颈时定期检查指导，最终产出更可靠且可测试的代码。

**上下文切换与项目接入**
在参与现有项目（如安全审批流程的Web应用「dependant」）时，团队使用Claude Code编写、审查和执行存储于代码库中的Markdown规范文档，使得新成员能在数日内而非数周内做出实质性贡献。

# Claude Code在安全工程领域的应用

## 团队影响力提升

**缩短事故处理时间**

原本需要10-15分钟手动代码扫描的基础设施调试流程，现仅需约5分钟即可完成。

**优化安全审查周期**

Terraform代码安全审批的审查效率显著提升，
消除了开发人员等待安全团队审批时的流程阻塞。

**强化跨职能协作**

团队成员可在数日内实现有效项目贡献，无需耗费数周进行背景知识储备。

**改进文档工作流**

通过整合多来源的故障排除指南与运维手册，形成了更高效的调试流程。

## 安全工程团队核心建议

**广泛使用自定义斜杠命令**

安全工程部门占据整个单体仓库中50%的自定义斜杠命令应用实例。这些定制化命令有效精简特定工作流程，加速重复性任务执行。

**优先启用Claude自主模式**

团队现采用"随写随提交"指令替代针对性代码片段提问，让Claude Code自主运行并定期检查，从而获得更全面的解决方案。

**利用该工具处理文档工作**

除编写代码外，Claude Code 在整合文档资料和生成结构化输出内容方面同样表现卓越。团队会提供写作样本与格式偏好设置，从而直接生成适用于 Slack、Google Docs 等协作工具的文档，有效避免界面切换疲劳。

# Claude代码在推理领域的实践

推理团队负责管理内存系统，该系统在 Claude 读取用户提示并生成响应时存储相关信息。团队成员（特别是机器学习领域的新人）可充分利用 Claude Code 弥合知识鸿沟，从而加速工作进程。

## Claude Code主要应用场景

**代码库理解与团队融入**

面对复杂代码库时，团队高度依赖 Claude Code 实现架构的快速解析。无需手动检索 GitHub 代码仓库，通过询问 Claude 即可秒级定位实现特定功能的文件，替代传统的人工询问或手动搜索方式。

**具备边缘场景覆盖的单元测试生成**

开发核心功能后，团队会要求 Claude 编写完整的单元测试。Claude 能自动补充遗漏的边缘场景测试用例，将通常需要大量脑力消耗的工作压缩至数分钟完成，如同可供审阅的智能编程助手。

**机器学习概念讲解**

不具备机器学习背景的团队成员依靠Claude解释模型专用功能及参数设置。原本需要1小时谷歌搜索和查阅文档的工作现仅需10-20分钟，研究时间压缩80%。

**跨语言代码转换**

当需要测试不同编程语言的功能时，工程师只需说明测试需求，Claude即可用目标语言（如Rust）编写测试逻辑，无需为测试目的专门学习新语言。

**命令调取与Kubernetes管理**

无需记忆复杂的Kubernetes命令，通过询问Claude获取精确语法，例如输入"如何获取所有pod或部署状态"，即可获得基础设施运维所需的完整命令。

# Claude代码在推理领域的实践

## 团队影响力提升

**加速机器学习概念理解**

研究时间缩减80%——原本1小时的谷歌搜索现仅需10-20分钟

**快速定位代码库**

可在数秒内定位相关文件并理解系统架构，无需依赖同事协助

**全面测试覆盖**

Claude能自动生成包含边界条件的单元测试，在维持代码质量的同时有效减轻心智负担。

**消除语言障碍**

无需学习新语言即可在Rust等陌生编程语言中实现功能开发。

## 推理团队的核心经验分享

**优先测试知识库功能**

尝试提出各类问题，观察Claude能否提供比Google搜索更快捷的解答。若响应更快且更精准，这将成为您工作流程中极具价值的省时工具。

**从代码生成着手**

为Claude提供具体指令要求其编写逻辑代码，随后验证正确性。在使用该工具处理复杂任务前，这有助于建立对其能力的信任基础。

**应用于测试编写**

让Claude编写单元测试可显著缓解日常开发工作的压力。善用此功能以保持代码质量，无需手动设计所有测试用例。

# Claude Code在数据科学与可视化中的应用

数据科学与机器学习工程团队的需求需要精密的可视化工具来解析模型性能，但构建此类工具通常要求掌握陌生语言和框架的专业技能。Claude Code使这些团队无需成为全栈开发人员，即可构建生产级质量的分析仪表板。

## Claude Code主要应用场景

**构建JavaScript/TypeScript仪表板应用**

尽管团队自称对"JavaScript和TypeScript知之甚少"，他们仍使用Claude Code构建完整的React应用程序，用于可视化强化学习模型的性能和训练数据。他们让Claude自主编写完整的应用程序（例如5000行TypeScript应用），而无需自行理解代码实现。这一能力至关重要，因为可视化应用相对低上下文依赖，且无需理解整个单体仓库架构，这使得团队能够在模型训练和评估期间快速开发原型工具来理解模型性能。

**处理重复性重构任务**

当面临合并冲突或编辑器宏无法处理但又不值得投入大量开发资源的半复杂文件重构时，他们像使用"老虎机"般运用Claude Code——提交当前状态，让Claude自主运行30分钟后，要么接受解决方案，要么在未达预期时重新开始。

**创建持久化的分析工具而非一次性笔记本**

团队不再构建用完即弃的Jupyter笔记本，而是让Claude创建可永久使用的React仪表盘，这些仪表盘可在未来模型评估中重复使用。这一点至关重要，因为理解Claude的表现是'团队最重要的任务之一'——他们需要了解模型在训练和评估期间的表现，而'这实际上并非易事，简单的工具无法通过观察单一指标的上升获取太多有效信号'。

**零依赖任务委派**

对于完全不熟悉的代码库或编程语言任务，他们将整个实现过程委托给Claude Code，利用其从单体仓库收集上下文的能力，无需参与实际编码即可完成任务。这使得团队能在专业领域外保持高效，无需花费时间学习新技术。

# Claude Code在数据科学与可视化中的应用

## 团队影响力提升

**实现了2-4倍的时间节省**
过去需要手动完成的繁琐重构任务，现在能以更快的速度完成。

**在陌生编程语言环境中构建复杂应用程序**
在JavaScript/TypeScript经验有限的情况下，成功开发了5000行规模的TypeScript应用程序。

**从临时工具转向持久化工具开发**
替代一次性Jupyter笔记本方案，现采用可复用的React仪表盘进行模型分析。

**直接获取模型改进洞见**
通过Claude Code的实际使用经验，为未来模型版本开发更优内存系统和改进用户体验提供直接依据。

**实现可视化驱动决策**
借助先进数据可视化工具，更深入理解Claude在训练和评估阶段的性能表现。

## 数据科学与机器学习工程团队的实用建议

**像对待老虎机一样操作**
在启动Claude工作前保存当前状态，让其持续运行30分钟，随后选择接受结果或重新开始，避免陷入修正调试的泥潭。相比纠错，重启任务往往能获得更高成功率。

**适时干预保持简洁性**
在监督过程中，可随时暂停Claude并询问'为什么要这样做？尝试更简明的方案。' 该模型默认倾向于采用更复杂的解决方案，但能够良好响应关于简化方法的请求。

# Claude代码与API整合

API知识团队致力于开发PDF支持等特性，
文献引用及网络搜索功能，这些可将额外知识注入Claude的上下文窗口。
在庞大复杂的代码库中开展工作，意味着需要持续面对陌生代码段，耗费大量时间理解特定任务需检查的文件，并在修改前构建完整的上下文环境。Claude Code通过充当系统指南显著改善了这一体验，可帮助理解系统架构、识别相关文件并解释复杂的交互逻辑。

## Claude Code主要应用场景

**工作流程规划第一步**
团队将Claude Code作为任务的'首选咨询对象'，要求其识别修复漏洞、功能开发或分析时需要检查的具体文件。这取代了传统工作中需手动遍历代码库收集上下文信息后才能开展工作的耗时流程。

**跨代码库的自主调试**
团队现在有信心独立处理代码库陌生区域的错误，而无需再向他人寻求帮助。他们可以直接询问Claude：“你认为你能修复这个漏洞吗？” 这就是我观察到的行为"，通常能立即取得进展——这在过去由于所需的时间投入成本而根本无法实现。

**通过内部试用进行模型迭代测试**
Claude Code自动采用最新的研究模型快照，成为团队体验模型变更的主要途径。这使得他们能在开发周期中直接获得模型行为变化的反馈，这是他们在以往版本发布过程中从未体验过的。

**消除上下文切换的开销**
无需复制代码片段或将文件拖入Claude.ai进行冗长的问题说明，他们可以直接在Claude Code中提问而无需额外收集上下文，这显著降低了认知负荷。

# Claude代码与API整合

## 团队影响力提升

**增强应对陌生领域的信心**
团队成员能够独立调试陌生代码库中的漏洞并调查事故。

**大幅节省上下文收集时间**
消除了复制代码片段和拖拽文件至Claude.ai的开销，减轻了心理上下文切换的负担。

**加速轮岗培训进程**
工程师轮换至新团队时，能够快速熟悉陌生的代码库并做出实质性贡献，无需频繁咨询同事即可高效开展工作。

**提升开发者幸福感**
团队反馈显示，日常工作流程阻力减少后，成员工作愉悦感显著增强且生产效率明显提升。

## API知识团队核心实践指南

**将其定位为迭代协作伙伴，而非一次性解决方案**
与其期待Claude立即解决问题，不如将其视为可反复推敲的协作伙伴共同迭代优化。这种方式比首次尝试就追求完美解决方案更为有效。

**用于建立陌生领域的技术自信**
无需犹豫处理专业范畴外的故障或事件调查——Claude Code使在通常需要大量背景构建的领域独立工作成为可能。

**从最小信息量启动**
仅需提供最基础需求信息启动流程，让Claude引导后续操作，而非预先加载冗长解释说明。

# Claude Code赋能增长营销战略

增长营销团队专注于构建涵盖付费搜索、付费社交广告、移动应用商店、邮件营销及搜索引擎优化的全域绩效营销渠道体系。作为非技术型单人团队，他们利用Claude Code实现营销任务自动化，创建出传统上需要大量工程资源才能实现的智能工作流程

资源。

## Claude Code主要应用场景

**自动化生成Google广告创意**

该团队构建了一个智能工作流程：处理包含数百条历史广告及效果指标的CSV文件，识别需优化的低效广告，并生成符合严格字符限制（标题30字符，描述90字符）的新版本广告。通过部署两个专用子代理（分别负责标题和描述生成），该系统可在数分钟内批量生成数百条新广告，无需跨多个营销活动进行人工创建。

这使得团队能够实现规模化测试与迭代，而过去要达成同等效果需要耗费大量时间。

**用于批量创意生产的Figma插件**

团队开发了一款Figma插件替代人工复制编辑静态图片的繁琐流程：该插件可识别框架并编程生成最多100个广告变体（通过自动替换标题和描述），将原本需要数小时复制粘贴的工作压缩至每批次仅需半秒。该方案可实现10倍的创意输出提升，使团队能够在核心社交渠道测试更多创新广告变体。

**用于广告活动分析的Meta Ads MCP服务器**

团队构建了与Meta Ads API集成的MCP服务器，可在Claude桌面应用中直接查询广告活动表现、投放数据与广告效果，消除了跨平台分析数据的需求，关键时效的提升直接转化为更优的投资回报率。

**基于内存系统的进阶提示工程**

团队部署了基础内存系统用于记录广告迭代中的假设与实验数据，使系统在生成新变体时能调用历史测试结果构建上下文，形成自我优化的测试框架。这使得人工难以追踪的系统化实验成为可能。

# Claude Code赋能增长营销战略

## 团队影响力提升

**重复性任务时效显著提升**
广告文案创作时间从2小时缩减至15分钟，释放战略规划时间。

**创意输出量级提升10倍**
通过自动化生成与Figma集成，团队现可实现跨渠道海量广告变体测试。

**像更大规模的团队一样运作**
该团队能够处理传统上需要专门工程资源支持的任务。

**战略重心转移**
团队可以将更多时间投入整体战略制定和构建自主自动化系统，而非手动执行任务。

## 来自增长营销团队的顶级建议 |

**识别支持API的重复性任务**
重点关注那些涉及重复操作且具备API接口工具的工作流程（如广告平台、设计工具、分析平台）。 这些正是自动化改造的最佳对象，也是Claude Code能提供最大价值的领域。

**将复杂流程分解为专业子智能体**
与其试图在单一提示或流程中处理所有事务，不如为特定任务创建独立智能体（如专门的标题生成智能体与描述生成智能体）。 这种方法不仅便于调试，还能在处理复杂需求时显著提升输出质量。

在编码前进行彻底的头脑风暴和提示规划。建议使用Claude.ai预先投入充足时间通盘思考完整工作流，然后让Claude.ai创建可供Claude Code参考的综合性提示模板与代码架构。此外，应采用逐步推进的工作方式而非要求一次性解决方案，以避免Claude因复杂任务而不堪重负。

# Claude代码优化产品设计

产品设计团队负责支持Claude Code、Claude.ai及Anthropic API，专注于构建人工智能产品体系。即使非开发人员也可利用Claude Code弥合设计与工程间的传统鸿沟，无需与工程师反复沟通即可直接实现其设计愿景。

## Claude Code主要应用场景

**前端优化与状态管理变更**

他们不再需要创建冗长的设计文档并与工程师进行多轮反馈来调整视觉元素（字体、颜色、间距），现在可直接通过Claude Code实施这些变更。工程师指出，他们正在进行'通常设计师不会涉及的大规模状态管理变更'，从而实现其理想中的精准质量要求。

**GitHub Actions自动化工单系统**

通过Claude Code的GitHub集成功能，团队只需提交描述变更需求的工单，Claude即可自动生成代码解决方案——无需启动Claude Code即可创建无缝衔接的错误修复与功能优化流程，有效处理长期积压的优化任务。

**快速交互式原型设计**

通过将设计稿图像粘贴至Claude Code，团队可生成工程师能立即理解并迭代的全功能原型，取代了传统静态Figma设计需大量解释和代码转化的流程。

边缘案例发现与系统架构理解：借助Claude Code绘制错误状态、逻辑流程和系统状态图，使团队能在设计阶段识别边缘案例而非开发后期发现，从根本上提升初始设计质量。

**复杂文案修改与法律合规**

在执行'从整个代码库移除'研究预览'提示语'这类任务时，团队运用Claude Code定位所有实例、审查关联文案、与法务部门实时协调变更，并完成更新——整个过程仅需两次30分钟电话会议，而非传统长达一周的反复协调。

# Claude代码优化产品设计

## 团队影响力提升

**核心工作流转型**

Claude Code已升级为主要设计工具，团队80%工作时间同时开启Figma和Claude Code进行协同设计。

**执行速度提升2-3倍**

以往需要与工程师反复沟通的视觉设计和状态管理变更，现在可直接自主实施。

**周期时间从数周缩短至数小时**

类似GA发布信息这样的复杂项目，原本需要一周协调时间，现在通过两次30分钟的电话会议即可完成。

**两种差异化用户体验**

开发者获得「增强型工作流」（执行提速），非技术人员则体验「开发者级工作流」（获得前所未有的全新能力）。

**设计-工程协作效率提升**

由于设计师能预先理解系统约束条件和可能性，团队沟通更高效，问题解决更迅速。

## 产品设计团队的顶尖技巧

**获取工程师的专业配置支持**

建议由工程团队成员协助完成初始仓库配置和权限设置——技术层面的入门流程对非开发人员颇具挑战性，
但一旦完成配置，将彻底改变日常工作效率。

**使用自定义记忆文件引导Claude行为**

创建明确指令告知Claude您是不熟悉编码的设计师，需要详细解释和小幅度增量修改——这将显著提升Claude的响应质量，降低使用门槛。

**利用图像粘贴技术进行原型设计**

使用 Command + V 将截图直接粘贴到Claude Code中——该工具擅长解析设计稿并生成功能性代码，对于将静态效果图转化为工程师可立即理解并扩展开发的交互式原型具有重要价值。

# Claude代码在强化学习工程中的应用

强化学习工程团队专注于高效强化学习采样算法和集群间的权重迁移技术。团队主要使用Claude Code进行中小型功能开发、调试和复杂代码库解析，采用包含频繁检查点设置和回滚机制的迭代式开发方法。

## Claude Code主要应用场景

**监督式自主功能开发**

团队允许Claude Code自主编写中小型功能的大部分代码并实施监管，例如在权重迁移组件中实现身份验证机制。他们采用交互式工作模式，让Claude主导编码过程，同时在其偏离方向时及时纠正。

**测试生成与代码审查**

在自行实施变更后，他们会要求Claude Code添加测试用例或审查代码质量。这种自动化测试流程为常规但重要的质量保证任务节省了大量时间。

**调试与错误排查**

他们使用Claude Code进行错误调试的效果参差不齐——有时能立即识别问题并添加相关测试用例，但在其他情况下难以理解问题本质。总体而言，该工具在成功运行时仍能提供有效价值。

**代码库理解与调用栈分析**

其工作流程最显著的改变体现在利用Claude Code快速获取相关组件及调用栈的摘要，这替代了传统的手动代码阅读或生成大量调试输出的工作方式。

## Kubernetes操作指导

团队频繁通过Claude Code咨询Kubernetes运维问题，这些原本需要大量网络搜索的操作疑问，现在能直接获取配置部署的即时解答。

# Claude代码在强化学习工程中的应用

## 开发
## 工作流程影响

**实验性方法的启用**
团队现采用'尝试与回滚'机制，通过频繁提交检查点来测试Claude的自主实现方案，必要时执行回滚操作，从而支持更激进的实验性开发模式
。

**文档编写加速**
Claude Code自动生成的有效注释显著节省文档编写时间，但需注意其偶尔会在非预期位置插入注释，或采用值得商榷的代码组织方式。

**在限制条件下实现提速**
虽然Claude Code能够以'相对较少的时间'实现中小型拉取请求(PR)，但团队承认其首次尝试成功率仅约三分之一，通常需要额外指导或人工干预。

## 来自RL工程团队的
## 顶级实践建议

**针对特定模式定制您的Claude.md文件**
在Claude.md文件中添加指令以防止重复工具调用错误，例如明确要求'直接使用正确路径运行pytest，无需切换目录'。 这一改进显著提升了操作一致性

**采用检查点密集型工作流程**
在Claude进行代码修改时保持定期提交，便于实验失败时快速回滚版本这种机制支持无风险的实验性开发模式

**先尝试单次生成，再进行协作调整**
给予Claude简洁指令，让其首先尝试完整实现方案若成功（约三分之一概率），即可大幅节省时间成本若条件不满足，则转向一种更具协作性和指导性的实施方法。

# Claude代码赋能法务工作

法律团队通过实验性探索及了解Anthropic产品矩阵的内在驱动力，逐步发掘了Claude Code的潜在价值。值得注意的是，某位成员还基于家庭场景开发了无障碍工具原型，通过工作场景实践验证了该技术对非技术人员的技术赋能效果，充分展示其技术实力。

## Claude Code主要应用场景

**为家庭成员定制无障碍解决方案**

团队成员已成功构建通信辅助工具，帮助因医疗诊断导致语言功能障碍的家庭成员实现有效沟通。团队仅用一小时就开发出基于原生语音转文本技术的预测式应答应用，通过语音库实现智能应答建议与语音播报功能，有效弥补语言治疗师推荐现有辅助工具的不足。

**法律部门工作流自动化实践**

团队创建了原型"电话树"系统，帮助员工精准对接Anthropic内部法务专员，验证了法律部门无需传统开发资源即可为常规事务构建定制化工具的可行性。

**团队协同工具创新**

管理层开发了G Suite自动化应用，实现周度团队更新自动汇总及跨产品法律评审状态跟踪，律师仅需点击按钮即可快速标记待审事项，取代传统电子表格管理模式。

**快速原型开发实现方案验证**

团队利用Claude Code快速构建功能性原型（例如向加州大学旧金山分校专家展示无障碍工具），在投入更多时间前通过领域专家验证构想并识别现有解决方案。

# Claude代码赋能法务工作

**工作方式与影响 在Claude.ai规划方案，通过Claude Code实现构建**

团队采用两步流程：先在Claude.ai进行头脑风暴和规划，随后转入Claude Code实施阶段，要求其逐步推进而非一次性输出所有内容。

**视觉优先方法论**

频繁使用屏幕截图向Claude Code展示界面设计预期，基于视觉反馈进行迭代优化，而非通过文字描述功能需求。

**原型驱动的创新模式**

强调克服对分享「幼稚」或「玩具级」原型的心理障碍——这些演示能激发他人看到未曾设想的可能性。

## 安全与合规意识

**MCP集成的安全顾虑**

作为产品法律顾问，他们立即识别深度MCP集成的安全隐患，指出当AI工具接入更敏感系统时，保守的安全策略将形成实施壁垒。

**合规工具开发优先级**

他们主张在AI能力扩展时快速开发合规工具，认识到创新与风险管理之间的平衡关系。

## 法律部门的重要实务建议

**首先在Claude.ai中全面规划**

在转入Claude Code前，充分利用Claude的对话式界面完善整体构思。然后要求Claude将完整方案提炼为分步实施指令。

**渐进式可视化开发**

让Claude Code逐步分解操作步骤，通过复制粘贴实现可控开发节奏。大量使用界面截图直观展示设计需求。

**勇于分享未臻完美的原型**

克服隐藏「玩具项目」或未完成工作的心理障碍——原型共享能激发跨部门创新思维，让非日常协作团队看见可能性。