

## 别再手搓Prompt了：Claude Code有一整套模板系统

很多人第一次用Claude Code，会有一种微妙的落差感：

模型很强，但我每次都要“重新教它怎么干活”。

但这不是你的错，这是工具形态的问题。

Claude Code本身只是一个「会话式编程引擎」，为了提升用户的开发效率和开发体验，Anthropic为Claude Code打造了一个可扩展模版/插件生态 + CLI管理工具claude-code-templates，让你可以像装插件一样给Claude Code加上智能agent、自动化命令、外部服务集成、监控工具等，从大大加速搭建、管理和优化你的工作流。



而claude-code-templates做的事情只有一件——

把“一次次重复教AI的经验”，固化成可复用、可安装、可组合的能力模块。

这篇文章，我们就把它彻底拆开。

### 01

#### claude-code-templates简介

claude-code-templates = Claude Code的模板/插件生态 + 官方级CLI管理器

•

<https://github.com/davila7/claude-code-templates>



它不是“再包一层AI”，而是做了三件很工程化的事：

1. 把Claude Code的能力拆成标准化组件
2. 用CLI把这些组件变成可安装、可组合、可复用
3. 给整个生态配了一个Web浏览器（aitmpl.com）

你可以把它理解为：

VS Code+插件市场  
只是编辑器换成了Claude Code，插件换成了AI行为模板。

## 02

### claude-code-templates

#### 解决的到底是什么问题？

先说一个所有Vibe Coding玩家都踩过的坑。

#### 1、上下文是一次性的

你告诉Claude：

- 你是一个严谨的Python reviewer
- 遵循PEP8
- 先分析再给代码

下一个会话，全没了。

templates做的事：

把这些“角色设定+行为约束+输出结构”

→ 固化成一个Agent模板。

## 2、每个项目都在重复造提示词

- 单元测试怎么让AI写得靠谱？
- PR review怎么让它像资深工程师？
- 重构时怎么强制它先给方案？

templates做的事：

把这些套路 → 变成Slash Commands（斜杠命令）。

## 3、Claude Code本身“不会自动做事”

Claude Code默认是被动的，你不叫它，它不动。

templates引入了：

- Hooks（自动触发）
- MCP（外部系统接入）

这一步，才是真正把Claude Code推向Agent化。

# 03

## claudie-code-templates 的核心组成

不搞花架子，直接按能力拆。

### 1、Agents：把“提示词”升级为“角色”

Agent本质 = 长期生效的系统Prompt

典型Agent包括：

- Senior Code Reviewer
- Test Engineer
- Security Auditor
- Refactoring Expert
- Documentation Writer

它解决的不是“会不会写代码”，而是：

AI在这个项目里，默认扮演谁？

工程直觉很简单：

- Prompt是一次性的

- Agent是状态化资产

## 2、Slash Commands：把流程变成命令

Slash Command是claude-code-templates最实用的部分。

比如：

- /review : 代码审查
- /add-tests : 补测试
- /refactor : 结构重构
- /explain : 解释复杂逻辑
- /optimize : 性能优化

它们的关键不是名字，而是内置流程：

- 输入格式
- 分析步骤
- 输出结构

你按下命令，其实是在触发一套预先设计好的推理链。

## 3、Hooks：让AI自动“闻风而动”

Hooks是很多人没意识到的高级能力。

它允许你定义：

- 当文件变化时
- 当测试失败时
- 当PR创建时

自动触发Claude Code行为。

这一步的意义是：

Claude Code不再只是聊天对象，而是项目里的“自动化参与者”。

## 4、MCP ( Model Context Protocol )：接入真实世界

MCP让Claude Code可以：

- 读GitHub
- 查数据库
- 调API

- 接内部工具

这解决了一个根本问题：

AI不再只靠你粘上下文，而是能“自己拿信息”

claude-code-templates提供的是MCP接入模板，帮你少踩坑。

## 04

### claude-code-templates 的怎么用？

#### 1、CLI安装

```
npx claude-code-templates@latest
```

这一步你得到的不是“模板”，而是一个  
交互式安装器。

#### 2、浏览模板（强烈推荐）

打开：<https://www.aitmpl.com>

The screenshot shows the official website for claude-code-templates. At the top, it features the title "CLAUDE CODE TEMPLATES" and a subtitle "Ready-to-use configurations for your Claude Code projects". Below this are three main steps: 1. "Install CLI Tool (Optional)" with a command "\$ npm install -g claude-code-templates" and a "Copy" button; 2. "Build Your Stack" with a sub-section "Add components to cart and generate install command"; 3. "Start Coding" with a command "\$ claude" and a note "// That's it! 🚀". Below these steps is a search bar with the placeholder "Search templates...". Further down are navigation filters for "type" (Agents, Commands, Settings, Hooks, MCPs, Templates) and "category" (All, AI Specialists, API Graphql, Business Marketing, Data AI, Database, Deep Research Team, Development Team, Development Tools, Devops Infrastructure, Documentation, Expert Advisors, FFmpeg Clip Team, Game Development, MCP Dev Team, Modernization, Obsidian Ops Team, OCR Extraction Team, Performance Testing, Podcast Creator Team, Programming Languages, Security, Web Tools).

这是claude-code-templates的官方Web目录。

你可以：

- 按类型筛选 ( Agent/Command/Hook/MCP )
- 查看每个模板的用途
- 直接复制安装命令

Web负责“看清楚”，CLI负责“装进去”。

### 3、安装你需要的能力

例如安装一个代码审查Agent：

```
•  
claude-code-templates install agent senior-code-reviewer
```

安装一个命令：

```
•  
claude-code-templates install command review
```

关键点：

它们不是“示例”，而是直接进入你的Claude Code环境。

- Agent : Senior Engineer
- Command : /review+ /add-tests
- Hook : 文件变更自动review

结果是：

1. 你改完代码
2. Claude 自动review
3. 发现问题给建议
4. 你用 /add-tests 补测试

这已经不是“写代码+AI”，而是：

你在写代码，AI在当同事。

## 和“普通Prompt仓库”的本质区别

很多人会问：

这不就是一堆Prompt吗？

工程上答案很清楚：

### **Prompt Claude Code Templates**

一次性 可安装

手动复制 CLI管理

无状态 持久生效

不可组合 Agent+Command+Hook

差别不是“写得好不好”，而是“能不能长期用”。

这是Claude Code真正“变强”的地方

Claude Code本身是一台发动机。

claude-code-templates才是变速箱、刹车、自动驾驶系统。

它让AI编程从：

一次次即兴发挥

升级为“可积累、可复用、可进化的工程系统”。

好了，今天的内容就分享到这里了，希望对小伙伴们有帮助！

往期文章：

[OpenCode深度体验：一个真正工程友好的AI编程智能体](#)

[Vibe Coding实战指南：把Vibe Coding从玄学变成编程流程](#)

[都说Vibe Coding不看代码，让AI写C或汇编行不行？结论有点残酷](#)

[Open Lovable，第一次让“AI抄网站”变得正大光明](#)

[iFlow CLI和Owen Code CLI怎么选？一篇说清楚](#)