# win10 WSL2

## 基本概念

WSL2 (Windows Subsystem for Linux 2) 是微软开发的一个功能，让你可以在 Windows 10/11 上直接运行 Linux 环境。把 WSL2 想象成在你的 Windows 电脑里安装了一个"迷你版 Linux 电脑"，但它们可以很好地配合工作。

## 主要特点

### 1、真正的 Linux 内核

（1）WSL2 使用完整的 Linux 内核，兼容性很好。

（2）可以运行几乎所有 Linux 程序。

### 2、性能优秀

（1）文件系统性能比 WSL1 快很多。

（2）启动速度快，资源占用少。

### 3、无缝集成

（1）可以直接访问 Windows 文件

（2）在 Windows 中可以调用 Linux 命令

（3）支持图形界面程序

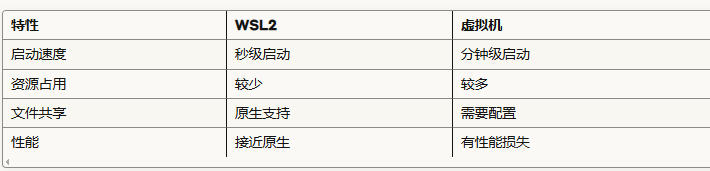
## 适用人群

1、开发者：需要 Linux 开发环境但主要用 Windows。

2、学习者：想学习 Linux 但不想安装双系统。

3、运维人员：需要使用 Linux 工具和命令。

## 与虚拟机的区别



## 总结

总的来说，WSL2 就是让 Windows 用户能够方便地使用 Linux 功能，不需要重启切换系统或者运行笨重的虚拟机。

# WSL2和WSL1的区别

## 核心区别

### WSL1：像是"翻译官"

1、把 Linux 命令翻译成 Windows 能理解的操作

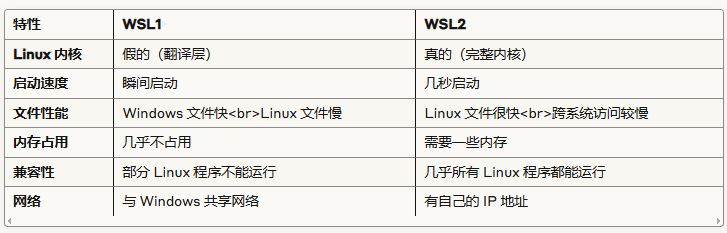
2、没有真正的 Linux 内核。

### WSL2：像是"真正的 Linux 电脑"

1、运行完整的 Linux 内核

2、在轻量级虚拟机中运行

## 详细对比



## 选择建议

### 选择WSL2

1、需要运行 Docker

2、大量使用 Linux 文件系统

3、需要完整的 Linux 兼容性

4、主要在 Linux 环境中工作

### 选择 WSL1

1、主要编辑 Windows 文件

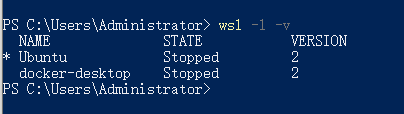
2、电脑配置较低

3、只是偶尔使用简单的 Linux 命令

4、需要最快的启动速度

## 查看当前版本

wsl -l -v



## 总结

WSL1 = 快速启动 + 简单使用 + 有限功能

WSL2 = 完整功能 + 更好性能 + 稍多资源占用

对于大多数用户，建议使用 WSL2，因为它提供了更完整和真实的 Linux 体验。

# 怎么在win10安装WSL

https://blog.csdn.net/m0\_51233386/article/details/127961763

# 官方文档

https://docs.anthropic.com/en/docs/claude-code/overview

# 中文文档

https://docs.anthropic.com/zh-CN/docs/claude-code/overview

# Kimi API Key

sk-pjfGR5kdeRXSd1zCpvxdzRjqYhubYXRlMrf0Qtz8YOdSPlmC

# win10原生系统安装Claude Code

## 前提条件

1、Windows 10/11

2、Node.js 18+（如果没有，去nodejs.org下载LTS版本）

3、Git for Windows（如果没有，去git-scm.com下载）

## 安装步骤

### 1、打开PowerShell或CMD

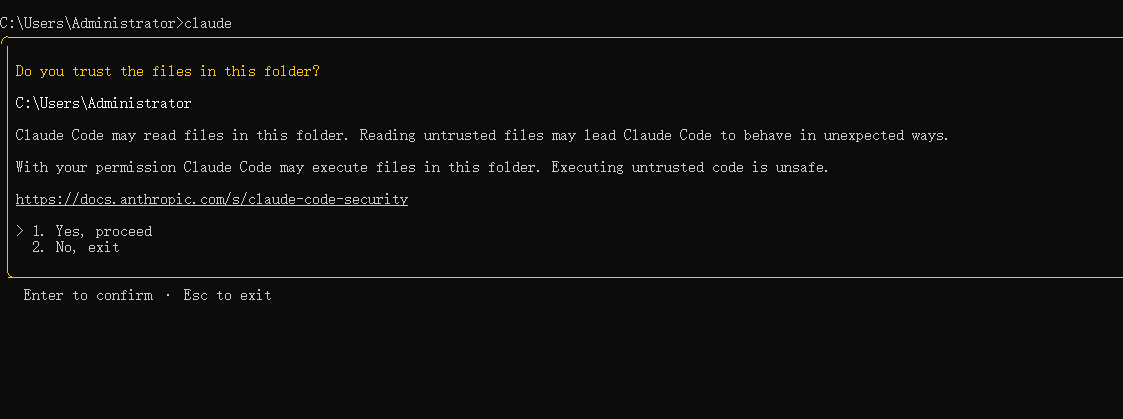
### 2、安装Claude Code

执行下面的命令：

npm install -g @anthropic-ai/claude-code

### 3、启动使用

安装完后，在命令行输入claude启动。就是这么简单！不需要WSL，不需要Ubuntu，不需要任何Linux知识。



## 重磅替代方案：用Kimi K2（不推荐用这个）

### 基本说明

kimi-k2是一款具备超强代码和Agent能力的MoE架构基础模型，总参数1T，激活参数32B。在编程、数学、Agent等主要类别的基准测试中，K2模型的性能超过其他主流开源模型。重点是：成本只有Claude的1/5。

### 配置方式

#### 第一步：获取Kimi API Key

访问Moonshot AI开放平台：https://platform.moonshot.cn/console/api-keys，注册并获取API Key。



#### 第二步：修改.claude.json文件

前提说明：这个文件需要你在命令行打开claude才会有。

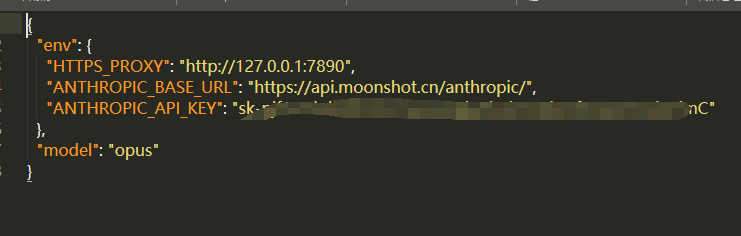
Windows用户在“C:\Users\你的用户名”目录下，找到.claude.json文件。

在customApiKeyResponses中的approved里面，加上你的Kimi API Key



#### 第三步：修改settings.json文件

Windows用户在“C:\Users\你的用户名＼.claude”目录下，找到settings.json文件，如果没有这个文件，就自己新建。在里面添加Kimi的URL和API Key



##### 其他说明

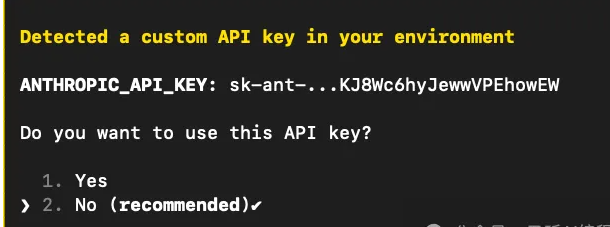
也可以在cmd命令行中配置环境变量，不用在这些文件中配置，参考下面的图片：

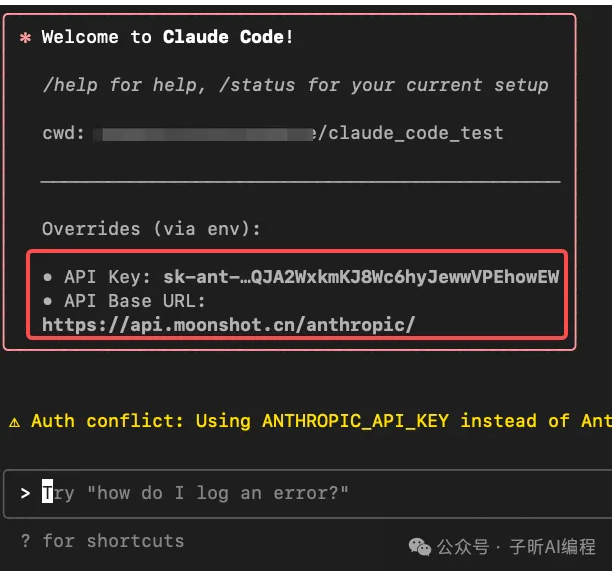
setx ANTHROPIC\_BASE\_URL https://open.bigmodel.cn/api/anthropic  
setx ANTHROPIC\_AUTH\_TOKEN 智谱apikey(需要修改)



#### 第四步：重启生效

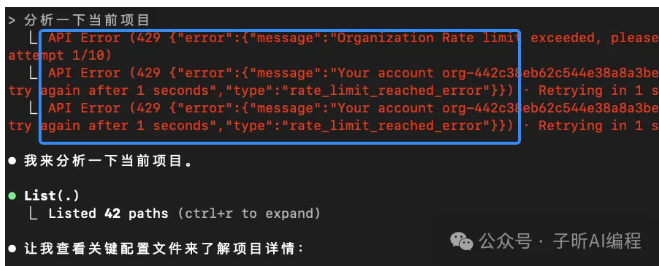
重新打开命令行，输入claude启动Claude Code，第一次启动会让你确认修改API Key，选择yes即可。





### 注意事项

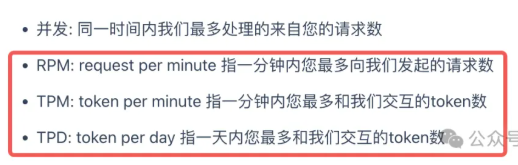
使用中可能遇到429错误，这是因为Kimi对免费用户有RPM限制（3次/分钟）。 如果想要取消这个限制，需要充值。不过就算不充值，多重试几次也能正常使用。



### 充值与限速



### 限速概念解释



### 怎么在cmd窗口中设置环境变量

#### 永久设置（系统级）

setx 变量名 值

#### 临时设置（当前会话有效）

set 变量名=值

#### 示例

##### 临时设置

set JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_191

##### 永久设置

setx JAVA\_HOME "C:\Program Files\Java\jdk1.8.0\_191"

添加到path

setx PATH "%PATH%;C:\Program Files\Java\jdk1.8.0\_191\bin"

#### 核心区别

1、set - 仅当前cmd窗口有效

2、setx - 永久保存，新开cmd窗口生效。

### 怎么在cmd窗口中获取环境变量

#### 1、查看单个环境变量

echo %变量名%

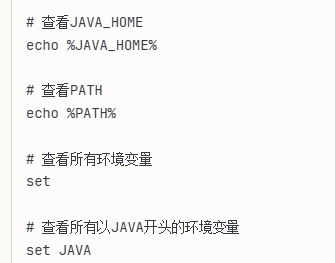
#### 2、查看所有环境变量

set

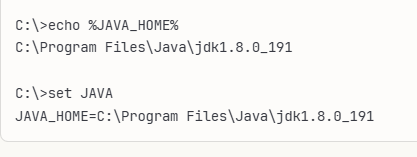
#### 3、查看特定前缀的环境变量

set 前缀

#### 示例



输出结果：

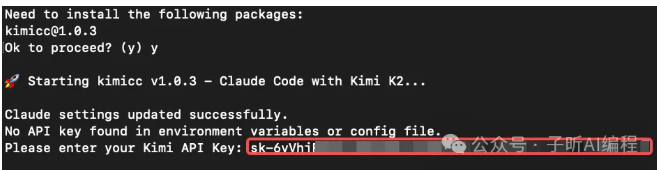


## 更简单的方案：一键安装Kimicc

它可以让你“一步命令使用Kimi K2运行Claude Code”。 只需要执行：

npx kimicc

首次运行会提示输入API Key，之后就不需要重复输入了。



这样就自动帮你装好了Kimi版本的Claude Code，只需要输入kimicc启动即可开始使用。

# ANTHROPIC\_API\_KEY与ANTHROPIC\_AUTH\_TOKEN的区别

## ANTHROPIC\_API\_KEY

**用途**：这是用于调用 Anthropic API的标准密钥。

**获取方式**：在 Anthropic Console (console.anthropic.com) 中创建

**格式**：通常以sk-ant-开头

**作用域**：用于直接调用 Claude API 进行对话、文本生成等功能。

**计费**：基于API使用量计费（按 token 数量）

**使用场景**：适用于直接集成 Claude API 到应用程序中。

## ANTHROPIC\_AUTH\_TOKEN

**用途**：这是用于访问Claude网页版的认证令牌。

**获取方式**：从浏览器中的Claude网页版会话中提取。

**格式**：是一个会话令牌，格式与 API Key 不同。

**作用域**：允许 Claude Code 模拟网页版 Claude 的访问。

**计费**：使用网页版的计费模式（通常是订阅制）

**使用场景**：当你想在 Claude Code 中使用网页版的功能或配额时。

## 主要区别总结

### 1、来源不同

API\_KEY：来自开发者控制台

AUTH\_TOKEN：来自网页版会话

### 2、计费模式不同

API\_KEY：按使用量付费

AUTH\_TOKEN：使用网页版订阅

### 3、功能访问

API\_KEY：标准 API 功能

AUTH\_TOKEN：网页版功能（可能包括一些网页版特有的特性）

## 配置建议

通常建议使用 ANTHROPIC\_API\_KEY，因为它更稳定、官方支持更好。只有在特定需求（比如想使用网页版特定功能或配额）时才考虑使用 ANTHROPIC\_AUTH\_TOKEN。需要注意的是，AUTH\_TOKEN 可能会因为网页版会话过期而失效，需要定期更新。

# Claude Code在Windows中的主要配置文件

## .claude.json文件

### 位置

C:\Users\Administrator

### 功能

1、.claude.json文件包含的内容远不止MCP服务器配置。

2、包含全局和项目特定的设置。

3、MCP（Model Context Protocol）服务器配置。

4、API密钥和端点设置。

5、工具和权限配置。

## settings.json文件

### 基本说明

正在将全局配置迁移到settings.json文件，claude config命令将被弃用，改用settings.json文件。

settings.json 是Claude Code中的全局配置文件，它定义了工具的默认行为和全局设置。

### 位置

C:\Users\Administrator\.claude

### 作用

1、**全局默认设置**：为所有用户和项目提供基础配置。

**2、系统级配置**：控制 Claude Code 的核心行为和功能。

**3、默认模板**：为新用户提供标准的配置模板。

### 配置示例

# 怎么升级claude code

## 1、使用包管理器升级（推荐）

# 全局升级

npm update -g @anthropic-ai/claude-code

# 或者重新安装最新版本

npm install -g @anthropic-ai/claude-code@latest

## 2、使用命令（使用这种方法）

claude update

# plan模式

## 基本概念

Plan 模式是 Claude Code 中的一个特殊操作模式，它创建了一个只读的研究和规划阶段，在进行任何代码更改之前进行分析。可以把它理解为将 AI 助手切换到"架构师模式"，在这种模式下它只能观察、分析和规划，但不能执行代码更改。

## 如何激活plan模式

激活方式非常简单：按两次 Shift+Tab 即可立即进入 Plan 模式。

## Plan 模式的工作流程

### 1、研究和分析

读取文件、搜索代码、理解代码库结构

### 2、制定计划

开发全面的实施策略

### 3、提交审批

使用 exit\_plan\_mode 工具展示建议的更改

### 4、等待确认

在得到确认之前不会进行任何更改。

## 主要优势

### 1、安全性

提供安全的研究和分析，避免意外更改。

### 2、深度思考

可以配合"think"指令触发扩展思考模式，给 Claude 额外的计算时间来更彻底地评估替代方案。

### 3、结构化规划

提供结构化的规划而不会产生不必要的更改。

## 使用技巧

一个推荐的工作流程是：使用 Opus 进行规划，使用 Sonnet 进行执行。可以通过 Shift + Tab 在两者之间切换。

## 意义

Plan 模式不仅仅是一个功能特性，它代表着软件工程未来的发展方向。正如文档中提到的，"工程界正在发生一场巨大变革，获胜的工程师和团队正在为他们的代理（agents）而不是为他们自己设置代码库和工具。"

这种模式特别适合：

1、大型代码更改前的规划。

2、复杂项目的架构设计。

3、代码库分析和理解。

4、风险评估和变更影响分析。

## 总结

Plan 模式本质上提供了一个安全的"沙盒"环境，让你可以与 Claude 一起探索和规划复杂的编码任务，而不用担心意外修改你的代码。

# claude.md文件作用（重点）

## 基本概念

claude.md 文件是用来设定约束、建立项目结构，并教 AI 如何在你的技术栈中操作——而不会让代码库变得臃肿或依赖脆弱的注释，它是 Claude Code 理解你项目的关键配置文件。

## 主要作用

### 1、项目上下文提供

（1）告诉Claude你的项目是做什么的。

（2）解释项目的技术栈和架构。

（3）描述团队的开发约定和规范。

### 2、开发规范定义

（1）代码风格和命名约定

（2）文件结构偏好

（3）测试方法和标准

（4）质量要求和编码标准

### 3、常用命令记录

（1）构建命令

（2）测试命令

（3）开发服务器启动命令

（4）部署和发布流程

## 文件层级结构

Claude Code 提供四个层级的内存位置。

### 企业级策略 (/Library/Application Support/ClaudeCode/CLAUDE.md)

1、组织范围的编码标准

2、安全策略和合规要求

3、全公司共享

### 项目内存 (./CLAUDE.md)

1、团队共享的项目指南。

2、项目特定的架构信息。

3、通过版本控制共享。

### 用户内存 (~/.claude/CLAUDE.md)

1、个人偏好设置。

2、适用于所有项目的个人工具配置。

### 本地项目内存 (./CLAUDE.local.md)

1、个人的项目特定设置。

2、**已弃用**，推荐使用导入功能。

## 实用功能

### 文件导入

可以使用@path/to/import 语法导入其他文件，支持相对和绝对路径。

### 快捷添加

使用#开头的输入可以快速添加内存内容。

### 内存管理

1、使用 /memory 命令查看加载的内存文件。

2、使用 /memory 命令直接编辑内存文件。

## 最佳实践

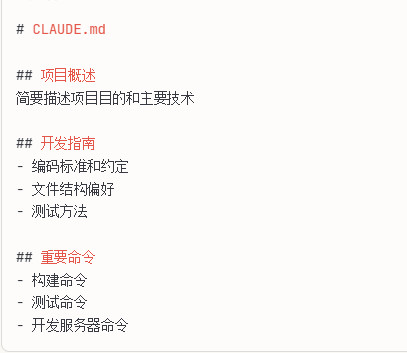
### ****内容要求****

1、要具体："使用 2 空格缩进"比"正确格式化代码"更好

2、使用结构化组织：用标题分组相关内容。

3、定期审查和更新。

## 典型结构示例



## 核心价值

当你开始新的 Claude 对话时，它会将你的 Claude.md 文件拉入上下文窗口，让 Claude 能够：

1、理解你的项目结构和约定。

2、生成符合项目标准的代码。

3、提供更精准的建议和修改。

4、帮助团队成员快速上手项目。

## 总结

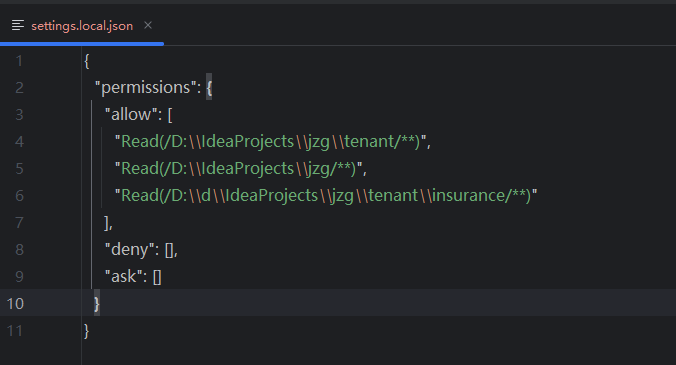
claude.md 文件本质上是让Claude从一个通用的编程助手转变为你项目专属的智能开发伙伴。

# settings.local.json文件

## 基本概念

这个.claude/settings.local.json文件是Claude Code的本地权限配置文件，专门用来控制老子在这个项目里的访问权限。

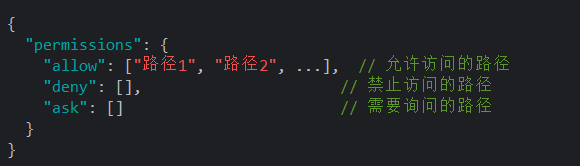
## 文件内容



## 作用

### 1、权限控制核心

这个文件定义了老子能访问哪些目录和文件，就像给老王我发了个工作证，规定了能去哪些车间干活：



### 2、当前配置的权限范围

Read(/D:\IdeaProjects\jzg\tenant/\*\*) - 整个tenant模块都能看

Read(/D:\IdeaProjects\jzg/\*\*) - 整个jzg项目根目录都能访问

Read(/D:\d\IdeaProjects\jzg\tenant\insurance/\*\*) - insurance模块

### 3、实际效果

可以读取整个诚泰报价项目代码

可以查看tenant模块的其他保险项目

能访问jzg项目的公共组件和工具类

不能修改文件（只有Read权限）

不能访问配置中未列出的其他路径

### 4、为什么需要这个文件

1、安全隔离 - 防止老子乱跑看到不该看的东西

2、性能优化 - 限制搜索范围，加快文件查找速度

3、项目聚焦 - 让老子专注于当前保险报价业务

4、权限管理 - 区分不同项目的访问级别

## 生成时机

1. 首次运行Claude Code时

当用户第一次在项目中启动Claude Code时，系统会自动创建.claude目录和这个配置文件。

2. 权限变更时

如果用户通过Claude Code的界面修改了权限设置，系统会自动更新这个文件。

3. 项目初始化时

有些项目模板可能会预先包含这个配置文件，确保开发环境一致性。

## 生成机制详解

Claude Code会智能检测项目结构，然后生成合适的权限配置：

1. 扫描项目根目录 - 发现这是Maven项目

2. 分析目录结构 - 识别出src/main/java等标准目录

3. 推断依赖关系 - 通过pom.xml发现依赖其他模块

4. 生成权限配置 - 自动添加相关模块的读取权限

## 文件管理特点

### 自动维护

1、文件由Claude Code自动维护，用户一般不需要手动修改

2、系统会根据项目变化智能更新配置

3、修改后会自动生效，无需重启

### 版本控制建议

1、这个文件通常不应该提交到版本控制

2、属于本地开发环境配置

3、每个开发者的权限需求可能不同

## 最佳实践

1、如果确实需要调整权限，通过Claude Code的界面操作

2、不要提交到git，避免团队配置冲突

3、定期检查权限配置，确保安全性

## 总结

这文件就是Claude Code的"智能工作证"，自动生成、自动维护，让老子既能高效帮你写代码，又不会越权看到不该看的东西。

# 文件导入功能

## 基本概念

# Claude Code交互模式（重点）

## 基本概念

交互模式是Claude Code的主要使用方式，创建一个持续的对话会话，你可以与 Claude 进行多轮对话，保持上下文状态，完成复杂的开发任务。

## 启动交互模式



## 核心特性

### 1、持续上下文

（1）记住之前的对话内容。

（2）理解项目结构和代码关系

（3）维护工作状态和进度

### 2、实时文件访问

（1）自动读取当前目录的文件。

（2）可以直接编辑和修改代码

（3）实时同步文件变化.

### 3、智能项目理解

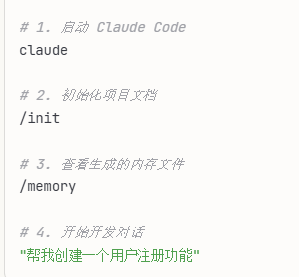
（1）分析技术栈和框架

（2）理解代码架构和模式

（3）提供符合项目规范的建议

## 典型工作流程

### 1、项目初始化流程

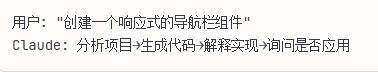


### 2、开发调试流程

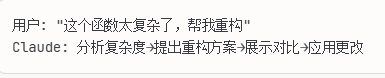


## 强大功能

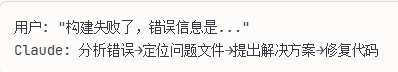
### 1、自然语言编程



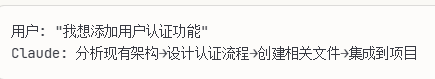
### 2、代码重构



### 3、错误诊断



### 4、架构设计



## 交互模式 vs 其他工具



## 实际使用场景

### ****新功能开发****

1、描述需求："我需要一个文件上传组件"

2、Claude 设计方案并实现

3、测试和调试

4、优化和完善

### ****代码审查****

1、要求审查："检查这个 PR 的代码质量"

2、Claude 分析所有更改

3、指出问题和改进建议

4、应用修复

### ****技术债务处理****

1、识别问题："这个模块耦合度太高"

2、Claude 分析依赖关系

3、提出解耦方案

4、逐步重构

### ****学习和教育****

1、询问原理："解释这个算法是如何工作的"

2、Claude 详细解释并举例

3、提供相关最佳实践

4、推荐进一步学习资源

## 最佳实践建议

### ****会话管理****

1、使用/clear清理长对话

2、使用/compact保留重要上下文

3、定期/export 保存重要对话。

### ****项目配置****

1、先运行 /init 建立项目上下文

2、维护好 CLAUDE.md 文件

3、使用 /memory 添加项目特定信息

### ****高效交流****

1、描述问题要具体明确。

2、提供足够的上下文信息。

3、善用自然语言描述需求。

## 总结

交互模式让 Claude Code 成为真正的编程伙伴，不仅仅是工具，更像是一个理解你项目、能够执行任务的智能助手。

# Claude Code非交互模式

## 基本概念

**非交互模式**是指你可以直接在命令行中提出问题或给出指令，Claude 处理完后立即返回结果并退出，而不需要进入持续的对话会话。

## 使用语法

claude -p "your question or instruction"

其中 -p 是 --prompt 的缩写。

## 示例

### 1、代码分析

claude -p "解释这段代码的作用" main.py

### 2、快速问题

claude -p "如何在 Python 中读取 JSON 文件？"

### 3、代码生成

claude -p "创建一个 React 组件用于显示用户列表"

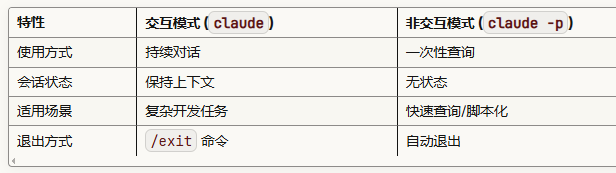
### 4、调试帮助

claude -p "这个错误是什么意思：TypeError: 'NoneType' object is not iterable"

### 5、文件处理

claude -p "重构这个函数使其更易读" utils.js

## 与交互模式的对比



## 优势

### 1、脚本友好

可以集成到 shell 脚本和自动化流程中

### 2、快速查询

适合简单问题的快速回答

### 3、CI/CD 集成

可以在构建流程中使用

### 4、命令组合

可以与其他 Unix 工具链接使用

## 总结

非交互模式让 Claude Code 不仅仅是一个聊天工具，更像是一个可以集成到任何开发工作流程中的智能命令行工具。

# 常用的命令

## claude

启动交互式对话模式，可以持续与Claude交互，直到输入/exit

## claude <prompt>

一次性发送提示并获取回复，例：claude "解释这段代码的作用"

## claude --version

查看版本

## claude auth login

认证登录

## claude auth logout

退出当前登录的 Claude 账号

## claude auth status

查看 Claude CLI当前的登录状态

## claude auth refresh

**刷新当前的登录凭证（token）**，确保你继续保持登录状态。

## claude config list

它的作用类似于一个“配置清单”，主要会显示：

**全局配置** —— 比如 Claude CLI 使用的默认模型、输出格式、是否启用颜色、代理设置等。

**用户配置** —— 如果你在本地通过 claude config set 修改过某些参数，这里都会展示出来。

**配置来源** —— 一般会标明哪些配置来自系统默认，哪些是用户手动设置的。

## claude update

更新claude code

# 交互式会话中的常用命令（重点）

## 目录和文件管理

### /add-dir

添加新的工作目录

### /compact

清除对话历史但保留上下文摘要（可选：/compact [指令摘要]）

### /export

导出当前对话到文件或剪贴板

## 代理和配置管理

### /agents

管理代理配置

### /config

打开配置面板

### /hooks

管理工具事件的钩子配置

### /model

设置 Claude Code 的AI模型

### /permissions

管理工具权限规则

## 项目初始化和文档

### /init

1、生成项目的CLAUDE.md文件。

2、分析项目结构和技术栈。

### /memory

1、查看当前加载的内存文件。

2、编辑现有的内存文件（如 CLAUDE.md）。

## 开发工具集成

### /install-github-app

为仓库设置 Claude GitHub Actions

### /pr-comments

获取 GitHub pull request 的评论

### /review

审查 pull request

### /security-review

对当前分支的待处理更改进行安全审查

## 会话控制

### /exit或/quit

1、退出当前 Claude Code 会话。

2、返回到终端命令行。

### /clear

1、清除当前对话历史

2、重新开始一个干净的会话

### /resume

恢复对话

## 系统和状态

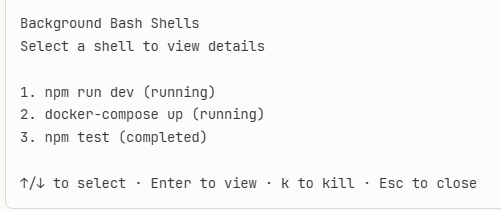
### /bashes

列出和管理后台bash shells。

/bashes是Claude Code中用于管理后台bash shells的内置命令，当你在Claude Code会话中运行这个命令时，会显示一个交互式界面来管理所有正在运行或已完成的后台shell进程。

#### 命令界面展示

当你输入/bashes后，会看到类似这样的界面



#### 功能详解

##### 1、查看后台进程

（1）显示所有Claude Code启动的后台bash会话

（2）显示每个进程的状态：running（运行中）或completed（已完成）

（3）展示正在执行的命令（如npm run dev、docker-compose up等）

##### 2、交互式操作

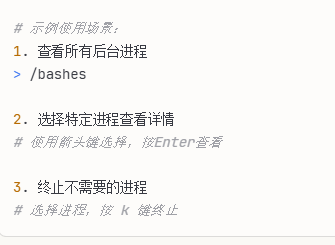
↑/↓ 箭头键：在不同的shell会话间导航选择

Enter键：查看选中shell的详细信息

k键：杀死（终止）选中的shell进程

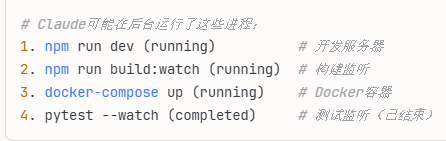
Esc键：关闭/bashes界面，返回正常聊天模式

##### 3、进程管理

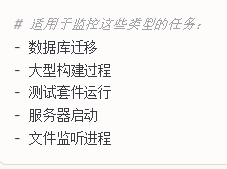


#### 实际使用场景

##### 1、开发服务器管理

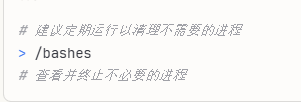


##### 2、长期任务监控

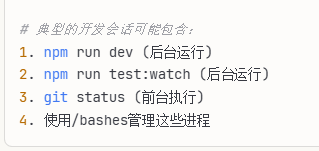


#### 最佳实践

##### 1、定期检查后台进程



##### 2、开发工作流优化



##### 3、资源管理

（1）避免积累过多的后台进程

（2）及时终止已完成或不需要的进程

（3）监控长时间运行的任务状态

#### 总结

/bashes命令是Claude Code中非常实用的进程管理工具，特别适合需要同时运行多个长期任务的开发场景，让你可以在一个统一的界面中监控和管理所有后台进程。

### /bug

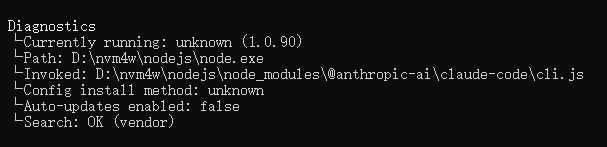
提交 Claude Code反馈

### /cost

显示当前会话的总成本和持续时间

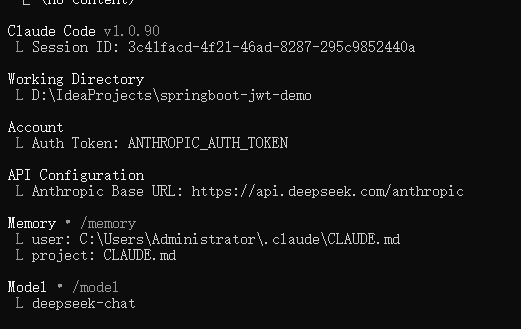
### /doctor

诊断和验证 Claude Code安装及设置。



### /status

显示Claude Code状态（版本、模型、账户、API 连接和工具状态）



### /statusline

设置Claude Code 状态栏UI

## 账户管理

### /login

使用 Anthropic 账户登录

### /logout

从 Anthropic 账户登出

### /upgrade

升级到 Max 以获得更高速率限制和更多 Opus

## 高级功能

### /ide

管理 IDE 集成并显示状态。

### /mcp

管理 MCP 服务器。

### /migrate-installer

从全局npm安装迁移到本地安装

### /output-style

直接设置输出样式或从选择菜单中选择

### /output-style:new

创建自定义输出样式

### /release-notes

查看发布说明

### /vim

在Vim 和 Normal 编辑模式之间切换

## 帮助

### /help

1、显示所有可用的斜杠命令列表

2、查看命令使用说明

## /terminal-setup

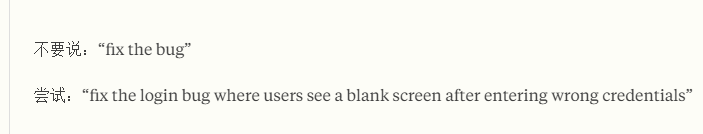
配置终端集成

## 使用@符号引用文件

# 状态栏配置和输出样式（Output Styles）

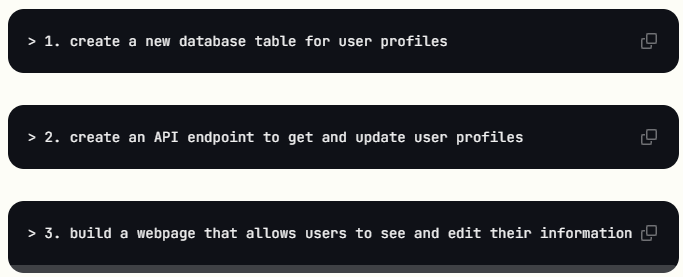
# 初学者专业提示

## 1、对您的要求要具体



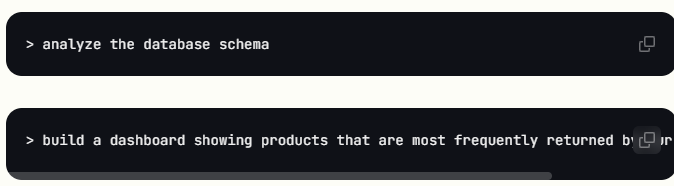
## 2、使用分布说明

将复杂任务分解为步骤

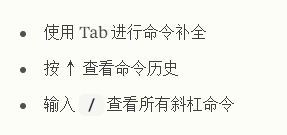


## 3、让Claude先探索

在进行更改之前，让Claude了解您的代码。



## 4、使用快捷方式节省时间



# 使用技巧

## 1、用好 claude.md ：写出“零 Bug”代码的关键

Claude Code 支持自定义规则文件 claude.md，这是许多用户忽视的神技，它是 Claude 的“工作手册”，包含项目规则与流程规范。你可以在项目目录中添加这个文件，写入一套“任务驱动规则”，引导 Claude 以“先规划、再执行”的方式逐步生成代码。这套七条规则能让 Claude 把每一个请求都拆解成清晰可控的任务，还能自动生成任务目录，确保每一步有迹可循。

7 Claude rules  
1. First think through the problem, read the codebase for relevant files, and write a plan to tasks/todo.md.  
2. The plan should have a list of todo items that you can check off as you complete them  
3. Before you begin working, check in with me and I will verify the plan.  
4. Then, begin working on the todo items, marking them as complete as you go.  
5. Please every step of the way just give me a high level explanation of what changes you made  
6. Make every task and code change you do as simple as possible. We want to avoid making any massive or complex changes. Every change should impact as little code as possible. Everything is about simplicity.  
7. Finally, add a review section to the todo.md file with a summary of the changes you made and any other relevant information.

立即复制这套规则并放入你的 claude.md 文件中，你会立刻感受到代码质量的巨大提升。

## 2、掌握Plan模式：Claude真正发挥实力的关键

Plan模式是Claude Code 最强大的特性之一，却经常被忽略或误用。  
只需按下 Shift + Tab 两次，即可进入 Plan 模式。在这里，你应该用自然语言清晰描述自己希望 Claude 执行的任务。Claude 会先帮你生成一个详细的执行计划，而不是直接生成代码。  
两个核心建议：

1、每一步都用Plan模式：别偷懒，每个微任务都应该规划。

2、搭配模型更高效：规划用Opus，执行用Sonnet，质量更高还省钱。

## 3、用GitHub模拟“撤回”功能：Claude没有回溯按钮怎么办

Claude Code 不像 Cursor 那样有“撤回到某步”的 Checkpoint 功能，因此你需要手动保存版本。

建议做法：每当 Claude 成功执行一个任务，立即 commit 到 GitHub；出错时，回退到上一个版本重新开始。这样能让你拥有几乎等效的“时间穿梭机”。

## 4、善用截图：灵感输入和 UI 调试的利器

Claude 支持图片输入功能，这让它在 UI 开发中如虎添翼。

### **场景一：界面灵感输入**

用截图工具（如 Mac 上的 Command + Shift + 4）截取目标 UI，然后上传至 Claude，配合提示语“请参考这个截图构建界面”，Claude会精准复刻。

### 场景二：调试Bug或优化UI

截图错误页面或丑陋界面，交给 Claude，它能帮你自动修复。

## 5、用好 /clear：节省成本、降低幻觉的法宝

长时间与 Claude 交互，会积累大量上下文信息，导致回复变慢、误差增多，甚至消耗大量 token。

建议：每完成一个阶段性任务，立即使用/clear命令，清除上下文。这不仅能让 Claude 表现更精准，还能显著节省月度额度。

## 6、安全审查：AI 代码最大风险来自“安全盲区”

大多数新手开发者会忽视安全问题，而 Claude 默认并不会主动防御所有安全漏洞。  
解决方法：每写完一段功能后，发送如下Security prompt:

Please check through all the code you just wrote and make sure it follows security best practices. make sure there are no sensitive information in the front and and there are no vulnerabilities that can be exploited.（请检查您刚刚编写的所有代码，并确保它遵循安全最佳实践。确保前面没有敏感信息，并且没有可以利用的漏洞。）

这是防止 API 密钥暴露、XSS 漏洞等问题的有效手段，尤其适合准备上线的产品。也可以在任务结束后，要求它检查是否破坏其他功能、尝试找找边缘情况，确认是否一切正常。它总能找到我们忽略的东西，这可以让我们对它生成的代码多一点心安。

## 7、向 Claude 学习：提升提示词水平的秘密武器

Claude 不只是工具，也是你的 AI 教练。每完成一个功能，不妨这样问 Claude：

Please explain the functionality and code you just built out in detail. Walk me through what you changed and how it works. Act like you’re a senior engineer teaching me code.（请详细解释一下你刚刚实现的功能和代码。告诉我你做了哪些改动以及它们是如何工作的。表现得像一位资深工程师在教我代码一样。）

这样做能显著提升你对代码逻辑的理解，从而更精准地下达指令，最终获得更高质量的输出。

## 8、注入额外上下文：执行任务一路开挂

### **场景一：拖入代码库中的文件夹**

不限于当前的代码库，也包括其他代码库中的其他文件夹。比如可以把后端目录一起拖进来，让它理解整体架构，甚至直接改后端。用这种方法也可以增加处理多个代码库的功能。

### 场景二：通过 URL 访问文档

Claude Code 自带浏览器能力，可以粘贴文档链接，它会访问那个网站，阅读文档并获取它需要的任何上下文；或直接告诉它“用最新 Google Calendar API”，它会自行搜索并查阅文档获取 API 使用方式。

## 9、运行子代理：效率倍增

Cloud Code 有能力启动子代理，在做非常庞大的任务时，可以让它拆分问题并同时跑多个 sub-agent。每个sub-agent拥有它们自己的上下文，会并行地执行任务，速度快非常多。

## 10、等待Claude运行时保持专注的秘诀：用 Claude 当“陪伴教练”

Claude 生成代码有时需要数分钟。在此期间，你是否会刷视频、走神、浪费时间？一个巧妙的方式可以帮你改变这个习惯：每当 Claude 正在执行任务，你就打开另一个 Claude 会话，与之探讨产品创意、复盘任务目标、制定下一步计划，进行头脑风暴。不仅避免了“精神内耗”，还极大提升了效率和心态。

## 11、强化工作流程

将 Claude Code 与 Cursor 或 VS Code 等文件编辑器结合使用，可以实现文件访问和编辑同步进行，提高工作效率。

## 12、管理上下文

频繁清理对话上下文，或使用 “上下文移交” 功能，确保模型每次都处理清晰、相关的输入，从而提升性能。

## 13、规划先行

编写高质量的产品需求文档（PRD），明确产品愿景而非技术细节，能有效指导AI代理，减少后期迭代修正。

## 14、自定义命令

创建项目专属和全局自定义斜杠命令，自动化重复任务、标准化工作流程，极大地提升效率和团队协作。

## 15、Git 检查点

虽Claude Code无内置检查点，但通过 Git 频繁提交可实现版本控制和回溯，尤其在代理编码中能有效应对模型跑偏。

## 16、超越代码边界

将 Claude Code 应用于非编程任务，如视频大纲生成或文档处理，展现其作为通用智能编辑工具的巨大潜力。

# claude-code-router

## 基本概念

一款强大的工具，可将 Claude Code 请求路由到不同的模型，并自定义任何请求。

## github地址

https://github.com/musistudio/claude-code-router

## 功能

### 1、模型路由

根据您的需求将请求路由到不同的模型（比如：后台任务、思考、长上下文）。

### 2、多提供商支持

支持 OpenRouter、DeepSeek、Ollama、Gemini、Volcengine 和 SiliconFlow 等各种模型提供商。

### 3、请求/响应转换

使用转换器为不同的提供商自定义请求和响应。

### 4、动态模型切换

在 Claude Code 中使用 /model 命令动态切换模型。

### 5、GitHub Actions 集成

在您的 GitHub 工作流程中触发 Claude Code 任务。

### 6、插件系统

使用自定义转换器扩展功能。

## 安装

首先，请确保您已安装 [Claude Code](https://docs.anthropic.com/en/docs/claude-code/quickstart)：

npm install -g @anthropic-ai/claude-code

然后，安装 Claude Code Router：

npm install -g @musistudio/claude-code-router

## 配置

创建并配置您的 ~/.claude-code-router/config.json 文件

config.json 文件有几个关键部分：

PROXY\_URL (可选): 您可以为 API 请求设置代理，例如："PROXY\_URL": "http://127.0.0.1:7890"。

LOG (可选): 您可以通过将其设置为 true 来启用日志记录。当设置为 false 时，将不会创建日志文件。默认值为 true。

LOG\_LEVEL (可选): 设置日志级别。可用选项包括："fatal"、"error"、"warn"、"info"、"debug"、"trace"。默认值为 "debug"。

日志系统: Claude Code Router 使用两个独立的日志系统：

服务器级别日志: HTTP 请求、API 调用和服务器事件使用 pino 记录在 ~/.claude-code-router/logs/ 目录中，文件名类似于 ccr-\*.log

应用程序级别日志: 路由决策和业务逻辑事件记录在 ~/.claude-code-router/claude-code-router.log 文件中

APIKEY (可选): 您可以设置一个密钥来进行身份验证。设置后，客户端请求必须在 Authorization 请求头 (例如, Bearer your-secret-key) 或 x-api-key 请求头中提供此密钥。例如："APIKEY": "your-secret-key"。

HOST (可选): 您可以设置服务的主机地址。如果未设置 APIKEY，出于安全考虑，主机地址将强制设置为 127.0.0.1，以防止未经授权的访问。例如："HOST": "0.0.0.0"。

NON\_INTERACTIVE\_MODE (可选): 当设置为 true 时，启用与非交互式环境（如 GitHub Actions、Docker 容器或其他 CI/CD 系统）的兼容性。这会设置适当的环境变量（CI=true、FORCE\_COLOR=0 等）并配置 stdin 处理，以防止进程在自动化环境中挂起。例如："NON\_INTERACTIVE\_MODE": true。

Providers: 用于配置不同的模型提供商。

Router: 用于设置路由规则。default 指定默认模型，如果未配置其他路由，则该模型将用于所有请求。

API\_TIMEOUT\_MS: API 请求超时时间，单位为毫秒。

如下是我本地的配置示例：





## 使用 Router 运行 Claude Code

使用 router 启动 Claude Code：

ccr code

注意: 修改配置文件后，需要重启服务使配置生效：

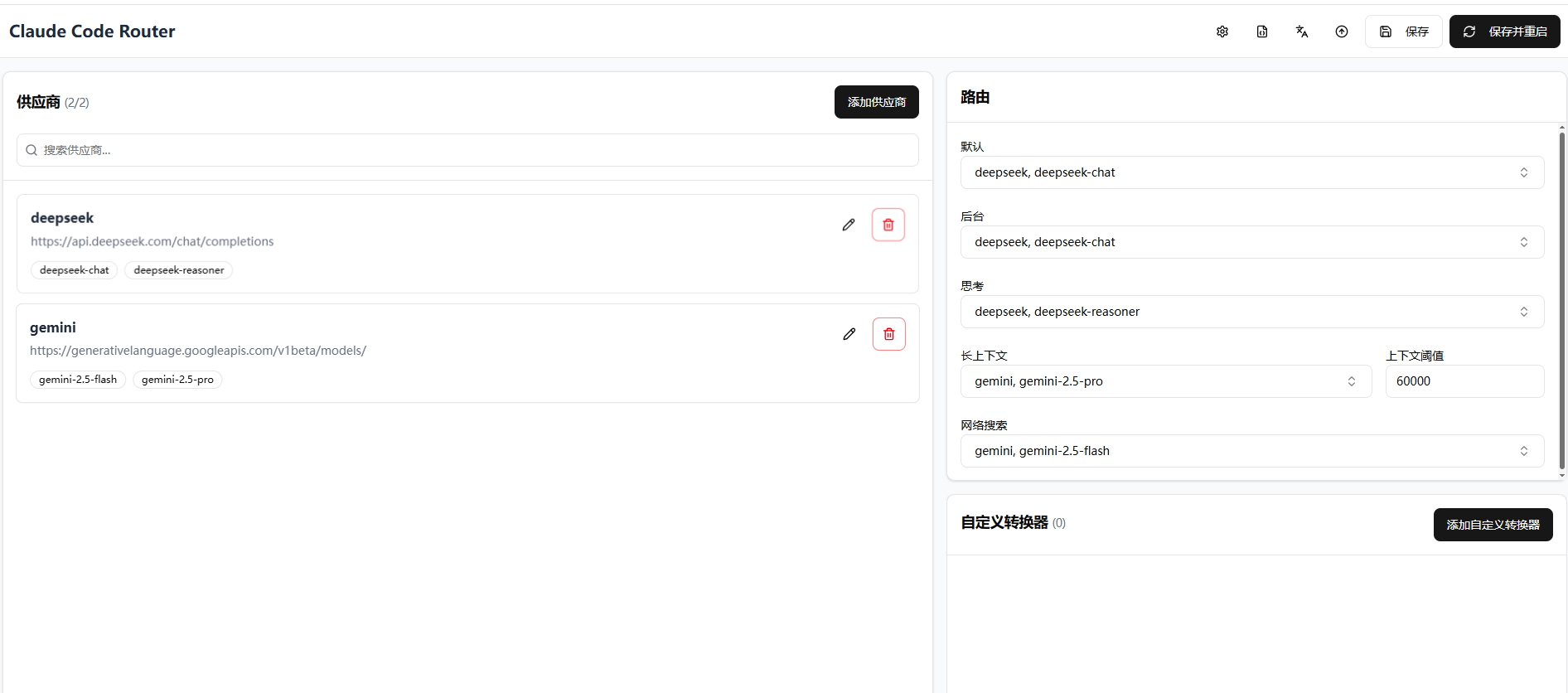
ccr restart

## UI 模式

为了获得更直观的体验，您可以使用 UI 模式来管理您的配置：

ccr ui

这将打开一个基于 Web 的界面，您可以在其中轻松查看和编辑您的 config.json 文件。



### Providers

Providers 数组是您定义要使用的不同模型提供商的地方。每个提供商对象都需要：

name: 提供商的唯一名称。

api\_base\_url: 聊天补全的完整 API 端点。

api\_key: 您提供商的 API 密钥。

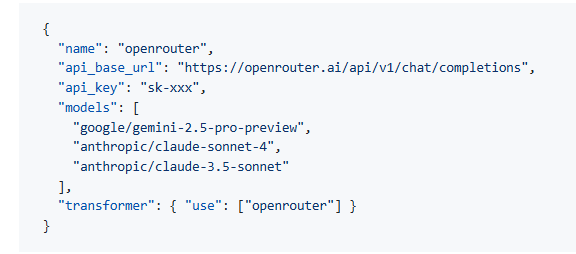
models: 此提供商可用的模型名称列表。

transformer (可选): 指定用于处理请求和响应的转换器。

### Transformers

Transformers 允许您修改请求和响应负载，以确保与不同提供商 API 的兼容性。

**全局 Transformer**: 将转换器应用于提供商的所有模型。在此示例中，openrouter 转换器将应用于 openrouter 提供商下的所有模型。



**特定于模型的 Transformer**: 将转换器应用于特定模型。在此示例中，deepseek 转换器应用于所有模型，而额外的 tooluse 转换器仅应用于 deepseek-chat 模型。



### Router

Router 对象定义了在不同场景下使用哪个模型：

default: 用于常规任务的默认模型。

background: 用于后台任务的模型。这可以是一个较小的本地模型以节省成本。

think: 用于推理密集型任务（如计划模式）的模型。

longContext: 用于处理长上下文（例如，> 60K 令牌）的模型。

longContextThreshold (可选): 触发长上下文模型的令牌数阈值。如果未指定，默认为 60000。

webSearch: 用于处理网络搜索任务，需要模型本身支持。如果使用openrouter需要在模型后面加上:online后缀。

您还可以使用 /model 命令在 Claude Code 中动态切换模型： /model provider\_name,model\_name 示例: /model openrouter,anthropic/claude-3.5-sonnet

## 总结

Claude Code 本身体验极好，但受限于昂贵的订阅费用和网络环境，想要长期使用成本并不低。而 Claude Code Router 提供了一种更灵活、低成本的替代方案：保留 Claude Code 的使用习惯，同时自由接入任意大模型提供商，实现模型的路由、切换与请求适配，大大提升了自由度与可控性。

通过简单配置，你可以：

1、接入 DeepSeek、Gemini、Ollama 等多个模型平台；

2、针对不同任务分发不同模型，提升效率与响应质量；

3、使用本地或国产模型完成大部分需求，显著降低 API 成本；

4、借助 UI 界面，可以方便便捷地管理模型与配置项；

在不改动 Claude Code 原始使用方式的基础上，获得更强模型能力；总的来说，Claude Code Router 是一款非常不错的 Claude Code 代理工具，不仅提升了灵活性，又保留了 Claude Code 高效编码的优势，对于有不同的需求的人还是挺香的。

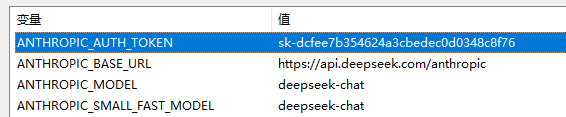
# 将deepseek模型接入Claude Code

## 1、安装claude code



## 2、配置环境变量





## 3、执行claude命令

进入项目目录，执行claude命令，就可以使用了



# 订阅建议

1、用量不大/预算有限：可以尝试 Claude Code Pro，偶尔在官网使用 Opus。

2、个人感觉：Claude Code + Git 的组合在使用体验和额度方面都优于 Cursor。

3、用 Claude 做大规模开发：Claude Code Max + Git。单终端高频编码以 Sonnet 为主，起手用 Opus 拉背景。多终端协作或高并发任务，可能需要开 Max 20x。

4、Max 计划一定要安装 claude-monitor 实时观测消耗，这会大幅增加你的开发速度和使用量（每一次会话的进度条都会提醒你“浪费”了多少额度）。

5、其实最划算的使用方法是开两个 Pro，或者和朋友一起开 Max 20x。

6、更喜欢多模型 + IDE 集成：Cursor Pro + Claude，Claude 负责长上下文/重构与代码质量，Cursor 负责文档浏览/多模型。

7、如果你是一个“新手”，那么我完全不建议订阅 Claude Code Max，可以说 Cursor Pro 会更好。只有当你真正知道自己的开发需求，能够看懂生成的代码，明白 AI 只是工具而非权威的时候，Max 的高额度才会成为助力。记住，掌控方向的始终需要是你自己。

8、AI 订阅确实非常贵，这是事实。但它为我省下的时间更加珍贵，这才是我选择订阅的真正原因。

# 问题

## 1、Tip:Use # to memorize shortcut to quickly add to CLAlDE.md

### 基本概念

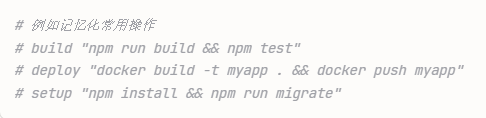
这意味着你可以使用#符号来**记忆化**常用的命令或操作，将它们快速添加到 CLAUDE.md 文件中作为快捷方式。

### 创建快捷方式



# shortcut\_name "你的命令或描述"

### 典型使用场景



### CLAUDE.md文件的作用

1、项目上下文：Claude Code会读取项目根目录的CLAUDE.md文件

2、快捷命令存储：你用#创建的快捷方式会被添加到这个文件

3、持久化：下次使用时可以直接引用这些快捷方式

### 使用已保存的快捷方式



### 实际工作流程

1、创建快捷方式：# test "运行所有测试用例"

2、自动保存：命令被添加到 CLAUDE.md文件中

3、后续使用：直接输入test即可执行完整命令。

这个功能让你可以**标准化**常用的项目操作，提高开发效率，特别适合复杂的多步骤命令。

## 2、auto-accept edits on(shift+tab to cycle)

"auto-accept edits on (shift+tab to cycle)"

auto-accept edits on：自动接受编辑功能已开启

shift+tab to cycle：按 Shift + Tab 键可以切换这个功能的开/关状态

## 3、"Auth conflict: Using ANTHROPIC\_API\_KEY instead of Anthropic Console key. Either unset ANTHROPIC\_API\_KEY, or run claude /logout."

这个警告是关于**API密钥认证冲突**的问题。

系统检测到两种认证方式同时存在：

1、环境变量：ANTHROPIC\_API\_KEY

2、Console登录：通过 claude 命令登录的密钥

Claude Code优先使用环境变量中的API密钥，这可能导致计费混乱。

### 解决方法

#### 方案1：移除环境变量（推荐）

# 临时移除（当前会话）

set ANTHROPIC\_API\_KEY=

# 永久移除

setx ANTHROPIC\_API\_KEY ""

#### 方案2：退出Claude登录

claude /logout

然后重新登录：

claude

### 为什么要解决这个问题

**1、计费问题**：环境变量可能指向不同的账户

**2、权限问题**：两种认证方式可能有不同的访问权限

**3、一致性**：避免认证混乱

解决后，Claude Code将使用Console登录的认证方式，确保与你的Pro会员账户一致。

## 4、"Run /terminal-setup to enable convenient terminal integration like Shift + Enter for new line and more"

这个提示是关于Claude Code的**终端集成设置**功能。建议运行 /terminal-setup 命令来启用便捷的终端集成功能，包括 Shift + Enter 换行等。

### 使用方法

#### 1、运行设置命令

/terminal-setup

#### 2. 启用的便捷功能

#### ****Shift + Enter 换行****

**普通Enter**：发送消息给Claude

**Shift + Enter**：在输入框中换行，继续编辑

适合输入多行代码或长文本

其他可能的集成功能

更好的代码高亮显示

智能缩进和格式化

快捷键支持

更流畅的交互体验

### 设置建议

1、首次使用：立即运行 /terminal-setup

2、提升效率：习惯使用 Shift + Enter 进行多行输入。

3、更好体验：享受更流畅的终端交互。

这个功能让Claude Code的使用体验更接近专业的IDE或终端环境。

## 5、Context left until auto-compact: 7%

1、当前对话的上下文缓冲区还剩余 7% 的可用空间

2、当这个百分比降到更低时，系统将自动启动"压缩"(compact)程序

### 什么是上下文缓冲区

Claude Code 需要维护对话历史、文件内容、代码更改等信息在内存中，但这个"上下文窗口"是有限的。上下文包括：

1、对话历史记录

2、当前打开和编辑的文件内容

3、之前的代码更改和操作

4、项目结构信息

5、MCP 服务器交互记录

### 自动压缩机制

当上下文空间不足时（通常在剩余空间很少时），Claude Code 会自动执行"压缩"操作：

#### 1、保留重要信息

（1）最近的对话内容

（2）当前正在编辑的文件

（3）重要的项目状态

#### 2、压缩或移除

（1）较早的对话历史

（2）不再需要的文件内容

（3）过时的操作记录

### 为什么会出现这个提示

这个提示出现说明：

1、你的对话已经相当长了

2、或者处理了大量的文件和代码

3、系统即将需要清理空间来继续工作

## 6、claude : 无法加载文件 D:\nvm4w\nodejs\claude.ps1，因为在此系统上禁止运行脚本。



### 解决方法

1、以管理员身份运行PowerShell

2、检查当前执行策略

Get-ExecutionPolicy

3、临时更改执行策略

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

4、确认更改

当提示时输入 Y 或 A 确认

5、现在可以运行claude命令

claude