

# OpenCode 介绍与实战

面向软件算法岗位的技术培训

按空格键继续 →



# 议程

第一部分

**为什么需要使用 OpenCode?**

第二部分

**OpenCode 模型配置方案**

第三部分

**OpenCode 实战**



# 第一部分

为什么需要使用 OpenCode?

"不可能把未来押给一家随时可能封号的公司"

# Claude Code 的问题

## ⚠️ 封禁风险

存在封禁中国用户的问题，账号安全无法保障

## 🔒 排斥竞争

切断第三方调用权限、封禁 OpenCode 等竞争工具用户

## 🔗 模型绑定

强绑定 Claude 模型，无法使用 GPT、Gemini 等其他模型

## 📉 服务不稳定

账号随时可能被封禁，严重影响工作流连续性

**核心痛点：**作为开发者，我们需要一个**稳定、可靠、不受单一厂商限制**的工具

# OpenCode 的优势

## 开源且支持多模型

- 可接入 **75+** 个 LLM 提供商
- 支持 GPT、Gemini、Claude、国产模型等
- 支持本地模型部署
- 无需翻墙即可安装使用
- 提供许多免费模型，降低上手门槛

## 中国用户友好

- 无使用障碍
- 华人创始人
- 飞书话题社区技术支持
- 无封号风险

## 高可定制性

- 类似 Android 系统的开放性
- 支持深度定制
- Oh My OpenCode 插件生态
- 多智能体并行协作

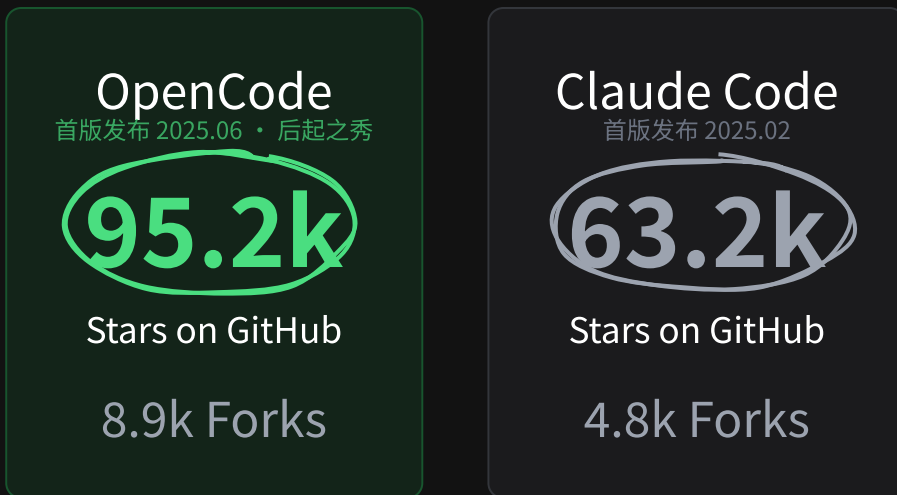
## 交互体验优越

- **Prompt Queue**: Agent 工作时可随时插入新指令
- 无需等待当前任务完成即可打断/补充
- Copilot 等工具目前不支持此功能

## 社区活跃

- 开源透明
- 快速迭代
- 丰富的插件生态

# OpenCode vs Claude Code



截止 2026年2月

# OpenCode vs Cursor 等 AI IDE

## Cursor 模式

- 围绕 IDE 运行，AI 是"高级助手"
- 需关注代码逻辑、项目结构
- 你在写代码

## OpenCode 模式

- 以 AI Agent 为核心
- 只需描述目标和要求
- AI 在写代码

## 底层架构差异

### Cursor (依托 VSCode/IDE)

- 受 Extension API 沙箱限制
- 文件/终端操作需通过 API 边界
- Electron 框架额外内存开销
- 扩展受限于 IDE 插件体系

### OpenCode (依托操作系统)

- 直接调用 `grep` / `git` / `ast-grep` 等原生工具
- 无沙箱限制，可启动任意进程
- 零中间层，工具调用延迟极低
- MCP 协议自由扩展任意 CLI 工具

💡 CLI Agent 直接对话操作系统，IDE Agent 需要通过 IDE 这个"中间人"





## 第二部分

OpenCode 模型配置方案

# 模型推荐方案

## 任务类型 → 推荐模型

任务类型	推荐模型
规划	Claude Opus 4.5 ≈ GPT5.2
编程	Claude Opus 4.5 > GPT5.2-Codex
前端设计	Gemini 3 Pro
代码审查	GPT5.2-Codex
多模态	Gemini 3 Pro
简单任务	MiniMax M2.1 / GLM 4.7

## 核心原则

- **规划任务**用最**强**模型
- **编程任务**注重代码质量
- **简单任务**用性价比高的模型
- 根据实际效果动态调整

⚠ **注意：**没有永恒的 SOTA，模型迭代速度极快。国产模型正在快速追赶，建议持续关注最新动态，根据实际效果灵活调整。

# 推荐的订阅方案

## GitHub Copilot ~\$10-39/月

- 与 OpenCode 深度集成
- 支持 GPT-4o、Claude Sonnet、Gemini 等
- 性价比最高的多模型方案

## ChatGPT Plus ~\$20/月

- GPT 系列模型完整访问
- o1/o3 推理模型
- 稳定可靠，适合日常使用

## Claude Max/Pro ~\$20-100/月

- 编程能力最强的模型之一
- ⚠️ 需要稳定的订阅渠道
- 有封号风险，谨慎评估

## 国内大模型套餐 价格友好

- 智谱/Minimax/Kimi Coding Plan
- 部分支持 API 形式使用订阅额度
- 无需翻墙，稳定可靠

💡 **推荐组合：** GitHub Copilot（主力多模型） + 国产套餐（备用/简单任务） = 高性价比方案

# 不推荐的方案

以下是一些用户可能考虑的替代方案，但存在明显缺陷

## ✗ Claude Code + CC-Switch

让 Claude Code 调用其他模型

- 依赖 Claude Code 官方客户端
- 本质是"套壳"，受上游政策约束
- Anthropic 曾封禁此类用法
- 随时可能失效

💡 OpenCode 原生支持多模型

## ✗ API 按量计费

302.ai、OpenRouter 等平台

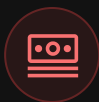
- 对编程深度用户成本过高
- 无法享受订阅优惠
- 费用难以控制
- 重度使用月费可达数百美元

## ⚠️ Google AI Studio

通过 antigravity-auth 等方式接入

- Gemini 免费额度确实很高
- 但非官方支持的接入方式
- 存在封号风险
- 稳定性无法保证

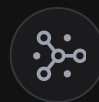
⚠️ 以上方案或成本过高，或存在政策风险，不推荐普通用户尝试



成本不可控



封号风险高



生态封闭



# 第三部分

OpenCode 实战

# workflow 建议

## 推荐 workflow



## 最佳实践

- **上下文管理**: 给 AI 足够的上下文, 但不要过多干扰
- **目标清晰**: 描述清晰的目标, 避免模糊指令
- **步骤化**: 分步骤执行复杂任务, "Step by step"
- **及时反馈**: 及时检查中间结果, 纠正偏差
- **版本控制**: 使用 git 做版本控制, 随时回滚

💡 **核心原则**: 从"写代码"转变为"指挥 AI", 专注于需求理解、架构设计和结果验收

# Oh My OpenCode 插件

## 核心能力

- **Sisyphus 主智能体**：协调整体任务执行
- **多智能体协调**：Oracle、Librarian、Explore 等专业 Agent 并行工作
- **最佳实践集成**：内置专业的开发规范和流程
- **效率提升**：Todo 强制继续、自动代码审查等

## 安装使用

```
1 # 推荐: 让 AI Agent 执行安装
2 # 将以下 URL 发送给 Claude Code / OpenCode:
3 https://raw.githubusercontent.com/code-yeongyu/
4 oh-my-opencode/refs/heads/master/docs/guide/
5 installation.md
6
7 # 或使用 npx 交互式安装
8 npx oh-my-opencode@latest install
```



zsh — 80×24

```
$ npx oh-my-opencode install
```

```
Downloading core components... [100%]
```

```
✓ Sisyphus Agent installed
```

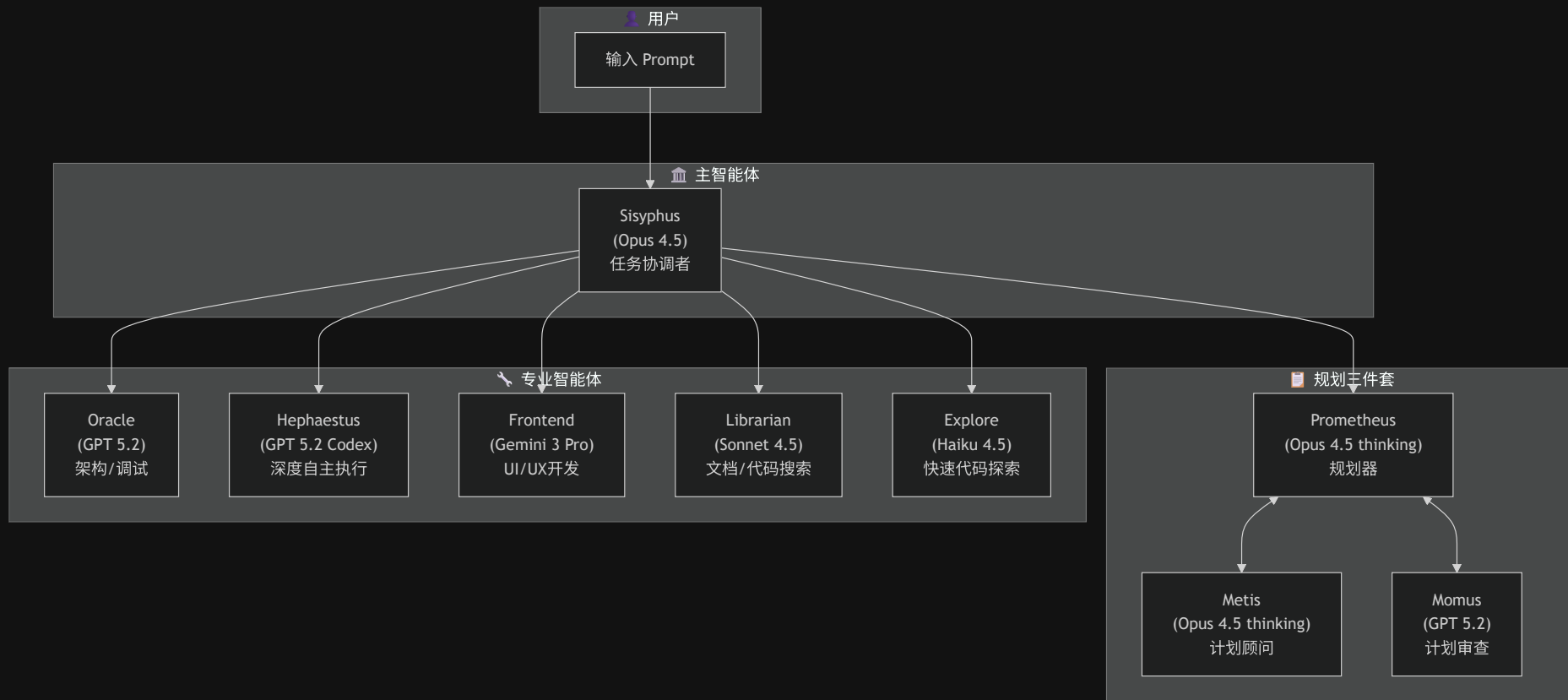
```
✓ Skill: git-master installed
```

```
✓ Skill: playwright installed
```

```
✓ Skill: verification-before-completion installed
```

```
🌟 Ready to work! Type 'ultrawork' to start.
```

# Oh My OpenCode 多智能体架构





# 多智能体架构的核心优势



## 模型各取所长

不同模型擅长不同任务：

- **Claude Opus** → 复杂规划、整体协调
- **GPT 5.2 Codex** → 深度调试、架构设计
- **Gemini 3 Pro** → 前端 UI/UX、视觉设计
- **Haiku 4.5** → 快速探索、简单任务



## 上下文保持整洁

主 Agent 委派低级任务给 Subagent：

- 探索代码 → Explore Agent
- 查阅文档 → Librarian Agent
- **主 Agent 专注整体任务编排**
- **节省 70%+ Token 消耗**



## 并行处理加速

多个 Agent 同时工作：

- 前端后端并行开发
- 代码搜索与文档查阅同步
- **构建速度提升 2.5x**
- 后台任务不阻塞主流程



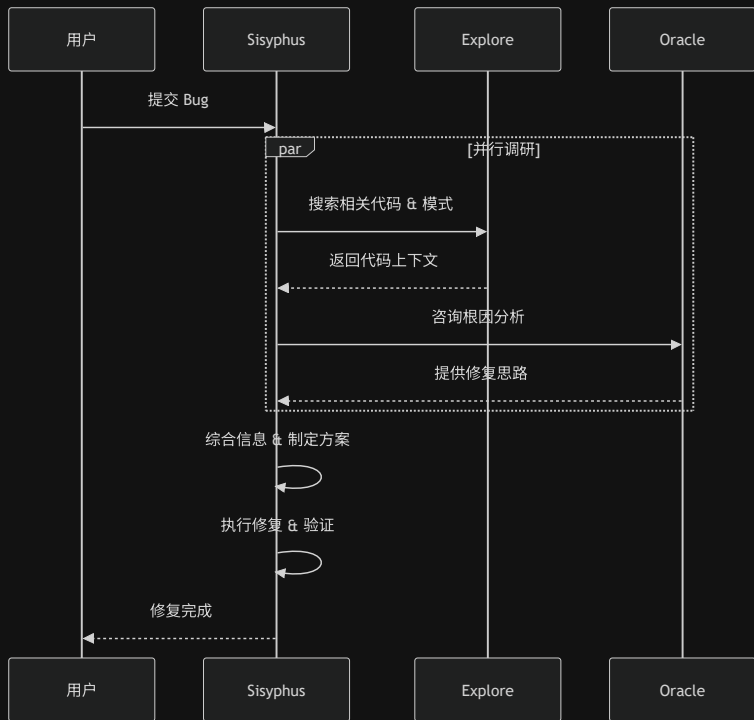
## 成本智能优化

分级调用策略：

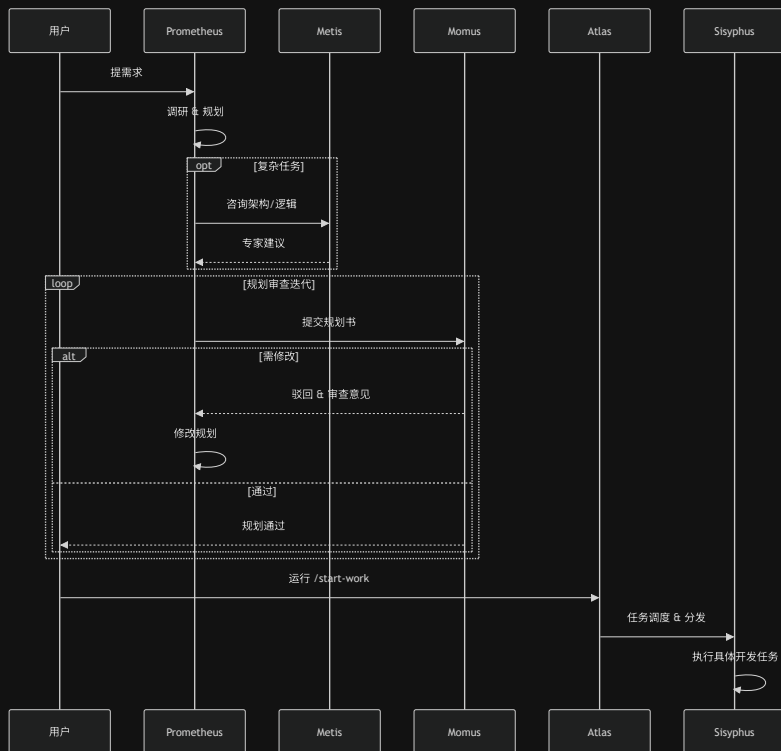
- **FREE:** grep、LSP、AST 工具优先
- **CHEAP:** Explore、Librarian 高频使用
- **EXPENSIVE:** Oracle 仅复杂问题调用
- 自动选择性价比最优模型

# 多智能体协作示例

## Bug 修复场景



## 新功能开发场景



# Skills 生态：扩展 OpenCode 能力

通过 Skills 可以快速为 Agent 注入领域专业知识

## ⚡ obra/superpowers

多智能体协作增强包（Oh My OpenCode 使用）

```
npx skills add obra/superpowers
```

## 📄 anthropics/claude-plugins-official

官方插件集：PDF/PPTX/XLSX 处理等

```
npx skills add anthropics/claude-plugins-official
```

## ☆ ComposioHQ/awesome-claude-skills

社区精选 skills 合集

```
npx skills add ComposioHQ/awesome-claude-skills
```

## 📁 vercel-labs/agent-skills

React/Next.js 最佳实践、组件模式

```
npx skills add vercel-labs/agent-skills
```

💡 使用 `/find-skills` 命令让 Agent 帮你搜索和安装所需 skill

# MCP 服务与实用工具

## MCP 服务推荐

**Playwright MCP** - 浏览器自动化测试、截图、表单填写

**Context7** - 实时查询框架/库官方文档

**GitHub MCP** - Issues、PR 管理、代码搜索

## 其他实用工具

**ralph-loop** - 自动化循环执行，直到任务完成

**opencode-antigravity-auth** - Google AI Studio 认证

**ultrawork 模式** - 完整多智能体协作关键词



## 安装 MCP 示例

# 在 opencode 配置中添加 MCP

```
opencode mcp add playwright-mcp -- npx @anthropic/mcp-playwright
```

# 常用操作快捷键

快捷键	功能说明
<code>Ctrl + X, L</code>	切换会话（多会话处理，如同时开发前后端）
<code>Ctrl + T</code>	切换模型 Variant（Thinking 模式、GPT 的 low/medium/high/xhigh）
<code>Ctrl + X, M</code>	切换模型
<code>Ctrl + P</code>	打开命令面板
<code>Ctrl + X, &lt;/&gt;</code>	查看 Subagent
<code>Ctrl + Enter</code>	换行输入



## 小技巧

多会话处理是一个强大的功能，可以让你在一个窗口写前端代码，另一个窗口写后端代码，大幅提升开发效率。

# Demo Time

现场演示 OpenCode 的实际使用

## Demo 1: 基础对话

- 启动 OpenCode
- 基本命令操作
- 模型切换

## Demo 2: 代码生成

- 描述需求
- 生成代码
- 迭代优化

## Demo 3: 多会话

- 创建多个会话
- 并行开发
- 会话切换

## Demo 4: 插件使用

- 安装插件
- 配置和使用
- 效果对比

结语

# 为什么选择 OpenCode



## 风险规避

闭源平台存在封号、服务中断等风险  
开源工具更适合长期使用



## 趋势把握

AI 发展明确走向智能体协助  
自动执行任务将成为主流



## 效率提升

从"写代码"到"指挥 AI"  
专注于创造性工作

## AI 编程的未来

以对话和调度为核心的工具将越来越主流

典型案例：Clawdbot（Mac 后台智能体，通过即时通讯沟通）

💡 **建议：** 尽早熟悉和掌握这类工具，建立新的工作范式



# 谢谢大家！

欢迎提问和交流

GitHub

[opencode-ai/opencode](https://github.com/opencode-ai/opencode)

文档

[docs.opencode.ai](https://docs.opencode.ai)

社区

飞书话题社区



# 培训结束

感谢参与！