目　录

- ai工具推荐
  - AI 助手管理工具: Cherry Studio
  - Coding
    - Claude Code + CC Switch 使用指南
      - CC GUI - AI 编程助手插件
    - Codex
      - Codex Auth Helper
      - opencodex 介绍
    - DeepSeek Harness 介绍
- 基础知识
- MCP 和 CLI
- AI网关
  - OmniRoute 项目介绍

ai工具推荐

AI 助手管理工具: Cherry Studio

Cherry Studio —— 多模型 AI 助手管理工具

> *一站式 AI 智能体管理平台，内置 300+ 预配置助手，支持多模型并行对话*

一、项目简介

Cherry Studio 是一款开源的 AI 助手管理工具，内置超过 300 个预配置 AI 智能体，涵盖学习辅导、编程开发、内容创作、数据分析等各个领域。用户还可根据需求轻松创建专属 AI 助手，实现个性化工作流。

特性	说明
多模型并行	同时打开多个大模型对话窗口，同界面高效协作
丰富智能体	300+ 预配置助手，覆盖全场景需求
自定义助手	支持创建专属 AI 助手，定制提示词与参数
多格式支持	导入 txt、png、docx、pdf 等多种文件
云端备份	数据安全同步，跨设备无缝衔接
实用工具	全局搜索、话题管理、AI 翻译、代码高亮
主题切换	明暗双主题，适配不同使用环境
跨平台	Windows、macOS、Linux 全平台支持

二、下载安装

2.1 官网下载（推荐）

https://cherry-ai.com/download

提供各平台开箱即用的安装包，无需复杂配置。

2.2 GitHub 源码

```
git clone https://github.com/kangfenmao/cherry-studio.git
```

2.3 Gitee 镜像（国内加速）

```
git clone https://gitee.com/mirrors/Cherry-Studio.git
```

三、核心功能

3.1 多模型并行对话

- 同时开启多个大模型对话窗口
- 在同一界面对比不同模型的回答
- 支持 GPT、Claude、Gemini、文心一言、通义千问等主流模型
- 灵活切换，高效辅助完成复杂任务

3.2 智能体市场

分类	示例助手
编程开发	代码审查、Bug 修复、算法优化
学习辅导	论文润色、知识点讲解、考试复习
内容创作	文案撰写、标题生成、SEO 优化
数据分析	数据清洗、可视化建议、统计解读
翻译助手	多语言翻译、术语校对、本地化
生活助手	旅行规划、菜谱推荐、健身计划

3.3 自定义 AI 助手

创建流程：

```
点击「新建助手」→ 设置名称与图标 → 编写系统提示词 → 调整模型参数 → 保存使用
```

支持自定义：

- 系统提示词（System Prompt）
- 温度参数（Temperature）
- 最大 token 数
- 对话上下文长度

3.4 文档处理

支持格式	功能
`.txt`	纯文本导入，快速分析

支持格式	功能
`.png` / `.jpg`	图片识别，OCR 提取文字
`.docx` / `.doc`	Word 文档解析，保留格式
`.pdf`	PDF 全文提取，支持扫描件

操作：直接拖拽文件到对话窗口，AI 自动分析内容。

3.5 云端备份

- 一键备份对话记录与助手配置
- 加密传输，保障数据安全
- 多设备登录，配置实时同步

四、实用工具

功能	说明
全局搜索	快速检索所有对话历史与助手
话题管理	对话分类归档，支持标签与收藏
AI 翻译	选中文字一键翻译，支持多语言互译
代码高亮	自动识别代码块，语法着色显示
主题切换	明亮/暗黑模式，保护视力
快捷键	支持自定义键盘快捷键，提升效率

五、快速上手

5.1 首次启动

1. 安装完成后打开 Cherry Studio
2. 选择主题（明/暗）
3. 配置 API 密钥（支持 OpenAI、Azure、Claude、国产大模型等）
4. 从智能体市场选择助手开始使用

5.2 添加自定义模型

```
设置 → 模型提供商 → 添加 → 填写 API 地址与密钥 → 测试连接 → 保存
```

5.3 创建专属助手

```
智能体 → 新建 → 输入名称 → 编写提示词模板 → 选择默认模型 → 保存
```

示例提示词：

```
你是一位资深 Python 开发工程师，擅长代码优化与性能调优。
请对用户的代码进行审查，指出潜在问题并给出改进建议。
```

六、平台支持

操作系统	安装包	说明
Windows	`.exe` / `.msi`	支持 Win10/Win11
macOS	`.dmg` / `.zip`	支持 Intel 与 Apple Silicon
Linux	`.AppImage` `.deb` `.rpm`	主流发行版兼容

七、相关资源

资源	地址
官网下载	https://cherry-ai.com/download
GitHub	https://github.com/kangfenmao/cherry-studio
Gitee 镜像	https://gitee.com/mirrors/Cherry-Studio.git

> *提示：首次使用建议从智能体市场选择 2-3 个常用助手体验，熟悉后再创建自定义助手，充分发挥多模型并行优势。*

Coding

Coding 简介

什么是 Coding

Coding（编程）是将人类的思维和逻辑，通过计算机语言转化为可执行指令的过程。
它是软件开发、数据分析、人工智能和自动化等现代技术的基础。

Coding 的分类

1. 传统编程

使用经典的编程语言（如 Python、Java、C++）来开发软件和应用程序。
特点：

- 明确的语法和规则
- 高度可控
- 适用于大型项目和复杂系统

2. Vibe Coding

一种创意与直觉驱动的编程方式，通常用于快速原型、互动艺术、小游戏和视觉化项目。
特点：

- 强调快速迭代和“Flow”
- 更关注体验和创意
- 常用工具：Processing, p5.js, Unity

3. AI Coding

利用人工智能技术辅助编码，自动生成、优化或修复代码。
特点：

- 提高开发效率
- 自动生成重复性代码
- 常用工具：GitHub Copilot, ChatGPT, Tabnine

Coding 的作用

- 软件开发：创建桌面应用、Web 应用、移动应用等

- 数据处理：数据清洗、统计分析、可视化
- 自动化：自动执行任务，提高效率
- 创意表达：通过编程实现艺术、游戏和互动项目
- 学习与探索：理解计算机思维和逻辑结构

常用编程语言

- Python：易学，适合 AI、数据分析和 Web 开发
- Java：跨平台，适合大型企业系统
- JavaScript：网页开发必备
- C/C++：性能优化和系统级开发
- Go / Rust：现代高性能编程语言

总结

Coding 不仅是一项技能，更是一种思维方式。
它既可以用于解决实际问题，也可以成为创意表达的工具。
结合不同的方法和工具（如 Vibe Coding 或 AI Coding），可以让编程更高效、更有趣。

Claude Code + CC Switch 使用指南

Claude Code + CC Switch 使用指南

> *从零开始，在 Windows 上安装 Claude Code，通过 CC Switch 接入国产模型*

一、核心概念

1.1 Claude Code 是什么

Claude Code 是 Anthropic 推出的 AI 编程 Agent 框架。它可以：

- 读取项目文件
- 分析代码结构
- 生成和修改代码
- 执行命令
- 自主决定工作流程

关键理解：Claude Code 是"外壳"，需要接入"大脑"（大语言模型）才能工作。

1.2 CC Switch 是什么

CC Switch（Claude Code Switch）是给 Claude Code 换大脑的图形化工具：

- 添加任意兼容 OpenAI API 格式的模型
- 一键切换当前 Claude Code 使用的模型
- 管理多个 API Key 和模型配置

一句话：CC Switch = 给 Claude Code 换大脑的遥控器

1.3 为什么需要 CC Switch

原版 Claude Code 默认搭配官方 Claude 模型，国内用户很难直接使用。通过 CC Switch，可以给 Claude Code 换上国内能直接调用的模型（智谱 GLM、MiniMax、Kimi、DeepSeek 等），无需外国手机号、无需特殊网络、无需 Visa 卡。

二、安装 Claude Code

2.1 方式一：官方脚本安装（推荐）

```
# 在 PowerShell 中执行
irm https://claude.ai/install.ps1 | iex
```

2.2 方式二：winget 安装

```
winget install Anthropic.ClaudeCode
```

2.3 方式三：npm 安装

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

2.4 验证安装

```
claude --version
```

看到版本号即成功。如果提示「claude 不是可识别的命令」，关闭终端重新打开再试。

三、安装 CC Switch

3.1 下载安装包

访问 GitHub Releases：

```
https://github.com/farion1231/cc-switch/releases
```

下载对应平台的安装包：

平台	文件
Windows	`CC-Switch-vX.X.X-Windows.msi` 或 `.zip`
macOS	`CC-Switch-vX.X.X-macOS.dmg`
Linux	`.AppImage` `.deb` `.rpm`

3.2 安装

双击安装包，一路 Next 完成安装。

四、获取 API Key

以 DeepSeek 为例（其他平台类似）：

1. 访问 platform.deepseek.co...
2. 注册账号并登录

3. 进入「API Keys」页面
4. 点击「创建 API Key」
5. 复制生成的 Key（形如 `sk-xxxxxxxxxxxxxxxx`）

其他可选平台：

- 智谱 AI（GLM）：open.bigmodel.cn
- MiniMax：platform.minimaxi.co...
- Kimi（Moonshot）：platform.moonshot.cn
- 硅基流动：cloud.siliconflow.cn

五、CC Switch 配置详解

5.1 打开 CC Switch

在开始菜单找到 CC Switch 并打开。

5.2 添加 Provider

```
点击「添加 Provider」→ 选择「OpenAI Compatible」
```

5.3 填写配置

字段	填写内容
名称	自定义，如 "DeepSeek"
Base URL	API 地址，如 `https://api.deepseek.com/v1`
API Key	从平台复制的 Key
模型	选择或填写模型名称，如 `deepseek-chat`

5.4 常用模型配置参考

DeepSeek：

```
Base URL: https://api.deepseek.com/v1
模型: deepseek-chat / deepseek-coder
```

智谱 GLM：

```
Base URL: https://open.bigmodel.cn/api/paas/v4
模型: glm-5.1 / glm-4-plus
```

硅基流动（多模型聚合）：

```
Base URL: https://api.siliconflow.cn/v1
模型: deepseek-ai/DeepSeek-V3 / Qwen/Qwen2.5-72B
```

5.5 激活配置

配置完成后，点击「激活」按钮，CC Switch 会自动修改 Claude Code 的配置文件，使其使用你设置的模型。

六、启动 Claude Code

6.1 在项目目录中启动

```
# 进入你的项目文件夹
cd D:\projects\my-project

# 启动 Claude Code
claude
```

⚠ 重要：不要在电脑根目录（如 `C:`）启动，信息量太大容易导致分析变慢、误操作。为每个项目建独立文件夹，cd 进去再启动。

6.2 首次启动配置

首次启动会提示：

1. 选择主题（Dark mode / Light mode）
2. 登录验证（如果接的是国产模型，CC Switch 已处理，直接回车）
3. 进入交互界面

七、CLAUDE.md 配置（推荐）

CLAUDE.md 是 Claude Code 的配置文件，用来设定协作规则。

7.1 全局配置

文件位置：

```
Windows: %USERPROFILE%\.claude\CLAUDE.md
macOS/Linux: ~/.claude/CLAUDE.md
```

7.2 项目级配置

在项目根目录创建 `CLAUDE.md` ：

```
my-project/
├── src/
├── CLAUDE.md      ← 项目级配置
└── ...
```

7.3 配置示例

```
# 协作规则

## 语言
- 所有回复使用中文
- 代码注释使用英文

## 安全
- 删除文件前必须确认
- 修改配置文件前必须确认
- 执行危险命令前必须确认

## 代码风格
- 使用 4 空格缩进
- 函数命名使用 camelCase
- 类命名使用 PascalCase

## 项目结构
- 源代码放在 src/ 目录
- 测试文件放在 tests/ 目录
- 文档放在 docs/ 目录
```

八、常用操作

8.1 基本命令

命令	作用
`claude`	启动 Claude Code
`/help`	查看所有命令
`/theme`	切换主题
`/cost`	查看本次会话费用
`/clear`	清空当前对话
`/exit`	退出

8.2 文件操作

```
@file src/main.py      # 引用特定文件
@directory src/        # 引用整个目录
@git diff              # 查看 Git 变更
```

8.3 常用提示词

```
"帮我分析一下这个项目的架构"
"给这个函数加上单元测试"
"把这段代码重构得更简洁"
"解释一下这个报错的原因"
"帮我生成一个 REST API 接口"
```

九、CC Switch 高级功能

9.1 多 Provider 管理

可以同时配置多个平台的 API，随时切换：

- Provider A：DeepSeek（便宜量大）
- Provider B：智谱 GLM（中文优化）
- Provider C：官方 Claude（需要时切换）

9.2 用量统计

CC Switch 提供使用量分析，帮助你监控 API 调用和费用。

9.3 会话管理

- 历史会话记录
- 会话收藏
- 消息导出

十、常见问题

问题	解决
`claude` 命令找不到	关闭终端重新打开，或检查 PATH
API 调用失败	检查 Base URL 和 API Key 是否正确
响应慢	切换更快的 Provider 或模型
中文回复乱码	终端编码设为 UTF-8
费用过高	切换到更便宜的模型（如 DeepSeek）

十一、相关资源

资源	地址
Claude Code 官网	code.claude.com
CC Switch GitHub	github.com/farion123...
CC GUI（IDEA 插件）	github.com/zhukunpen...

资源	地址
DeepSeek 开放平台	platform.deepseek.co...
智谱 AI 开放平台	open.bigmodel.cn
硅基流动	cloud.siliconflow.cn

> *提示：模型会不断迭代更新，但好的 Agent 框架是更持久的基础设施。掌握 Claude Code + CC Switch 的组合，无论底层模型如何变化，你都能快速适配。*

CC GUI - AI 编程助手插件

CC GUI —— IntelliJ IDEA 的 AI 编程助手插件

> *在 JetBrains IDE 内深度集成 Claude Code 与 Codex，支持与 CC Switch 无缝配合*

一、项目简介

CC GUI（原 Claude Code GUI）是一款 IntelliJ IDEA 插件，为 Claude Code 和 OpenAI Codex 提供可视化图形界面。它将 AI 编程能力直接嵌入 IDE，让开发者在不离开编码环境的情况下，享受 AI 辅助编程的便利。

- GitHub：github.com/zhukunpen...
- 原名：Claude Code GUI（为避免商标风险已更名）
- 协议：MIT 开源
- 支持 IDE：IntelliJ IDEA、WebStorm、PyCharm、GoLand、Rider 等 JetBrains 全系

二、核心功能

2.1 双 AI 引擎支持

引擎	提供商	说明
Claude Code	Anthropic	上下文感知编程助手，支持 Opus 系列模型
OpenAI Codex	OpenAI	强大的代码生成引擎

2.2 智能对话系统

- 上下文感知：自动识别当前项目代码上下文
- @file 引用：精准引用特定文件作为对话上下文
- 图片发送：支持上传图片描述视觉需求
- 对话回溯：灵活调整历史对话记录
- 增强提示词：自动优化用户输入，提升 AI 理解准确度

2.3 Agent 自动化

- 内置 Agent：自动化执行复杂多步骤任务

- Skills 命令：斜杠命令系统（ `/init` 、 `/review` 、 `/test` 等）
- MCP 扩展：支持 Model Context Protocol 服务器扩展 AI 能力

2.4 开发者体验优化

功能	说明
代码 DIFF 对比	直观展示 AI 修改前后的代码差异
文件导航	点击即可跳转到相关代码位置
代码跳转	支持跳转到定义、查找引用
主题切换	明暗双主题，适配 IDE 外观
字体同步	自动跟随 IDE 字体设置
国际化	中/英文自动切换

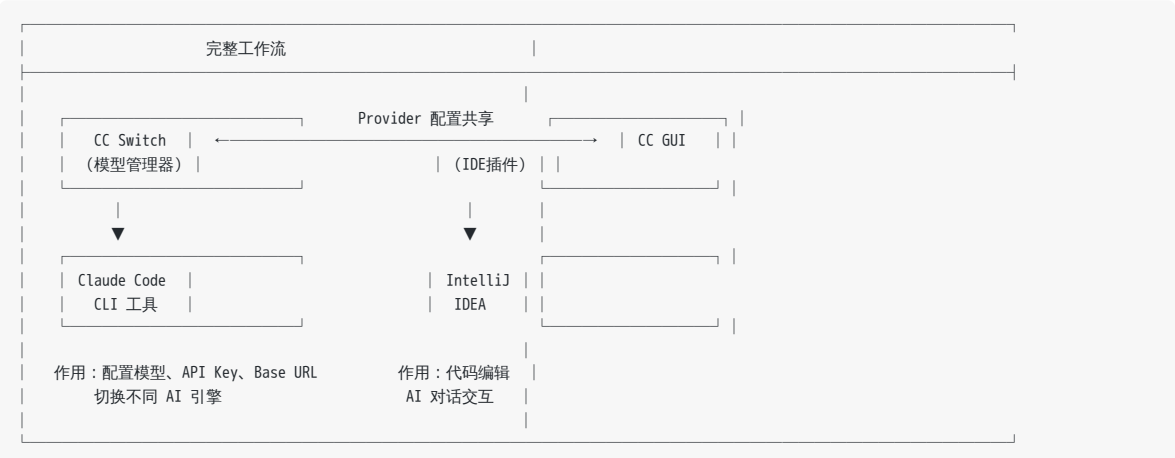
2.5 会话管理

- 历史会话记录与搜索
- 会话收藏（标记重要对话）
- 消息导出（保存为文件）
- 使用量统计分析

三、与 CC Switch 配合使用

3.1 为什么需要配合 CC Switch

CC GUI 是 IDE 内的操作界面，CC Switch 是模型配置管理器。两者配合：



3.2 配合流程

第一步：在 CC Switch 中配置模型

```
打开 CC Switch → 添加 Provider → 填写 API 信息 → 激活
```

支持的 Provider 类型：

- OpenAI Compatible（DeepSeek、智谱、硅基流动等）
- 官方 Claude（需特殊网络）
- 官方 Codex（需特殊网络）

第二步：在 CC GUI 中加载配置

CC GUI 会自动检测 CC Switch 的配置，或手动同步：

```
CC GUI 设置 → Provider → 选择「使用 CC Switch 配置」
```

第三步：在 IDE 中使用

```
打开项目 → 点击 CC GUI 面板 → 选择 AI 引擎 → 开始对话
```

3.3 配置共享说明

配置项	存储位置	共享方式
API Key	CC Switch 加密存储	CC GUI 读取调用
Base URL	CC Switch 配置文件	两者同步
模型选择	CC Switch 激活状态	CC GUI 自动跟随
会话历史	CC GUI 本地存储	独立管理

四、安装方法

4.1 方式一：插件市场安装（推荐）

```
IntelliJ IDEA → 设置 → 插件 → 市场 → 搜索 "CC GUI" → 安装 → 重启 IDE
```

4.2 方式二：离线安装

```
1. 从 GitHub Releases 下载插件包（.zip）
2. IDEA → 设置 → 插件 → 从磁盘安装 → 选择 .zip 文件
3. 重启 IDE
```

4.3 方式三：源码构建

```
git clone https://github.com/zhukunpenglinyutong/jetbrains-cc-gui.git
cd jetbrains-cc-gui
./gradlew clean buildPlugin
# 插件包位于 build/distributions/（约 40MB）
```

五、首次使用

5.1 打开 CC GUI 面板

```
IDEA 右侧工具栏 → 点击 "CCG" 图标 → 打开 AI 对话面板
```

5.2 连接 CC Switch

```
CC GUI 面板 → 设置齿轮 → Provider → 选择「CC Switch」→ 测试连接
```

5.3 开始对话

```
1. 在编辑器中选中代码
2. 右键 → "发送到 CC GUI" 或直接输入问题
3. AI 分析后给出建议
4. 点击「应用修改」或「复制代码」
```

六、典型使用场景

场景一：代码审查

```
选中代码 → 输入 "/review" → AI 自动分析代码问题
→ 查看 DIFF → 点击应用修改
```

场景二：生成测试

```
选中函数 → 输入 "给这个函数写单元测试" → AI 生成测试代码
→ 查看 DIFF → 应用到测试目录
```

场景三：重构建议

```
选中类文件 → 输入 "重构这个类，使用设计模式优化"
→ AI 给出重构方案 → 逐步应用
```

场景四：Bug 修复

```
选中报错代码 → 输入 "修复这个报错"
→ AI 分析错误原因 → 给出修复代码
```

七、安全审计

项目承诺定期安全审计：

- 每个小版本发布前进行 `/security-review` 审计
- 每 10 个小版本进行全面 `claude-code-security` 审计

八、常见问题

问题	解决
CC GUI 无法连接 CC Switch	确认 CC Switch 已启动并激活 Provider
模型响应慢	在 CC Switch 中切换到更快的 Provider
中文显示乱码	IDEA 设置 → 编辑器 → 文件编码 → UTF-8
代码应用失败	检查文件是否被其他程序锁定
插件崩溃	检查 IDEA 版本兼容性，更新到最新版

九、相关资源

资源	地址
CC GUI GitHub	github.com/zhukunpen...
CC Switch GitHub	github.com/farion123...
Claude Code 官网	code.claude.com
JetBrains 插件市场	IDEA 内搜索 "CC GUI"

十、与 CC Switch 的关系总结

维度	CC Switch	CC GUI
定位	模型配置管理器	IDE 内 AI 操作界面
运行环境	桌面系统	JetBrains IDE 内
核心功能	管理 API、切换模型	代码交互、AI 对话
是否必须	是（提供模型接入）	否（可选增强体验）

维度	CC Switch	CC GUI
配合效果	配置一次，多处使用	编码不离 IDE，效率最高

> *建议：先安装配置好 CC Switch，再安装 CC GUI，两者配合使用效果最佳。*

Codex

Codex

简介

Codex 是 OpenAI 推出的 AI 编程助手（Codex CLI），是一个运行在终端中的轻量级编程代理。它基于 OpenAI 的编码模型（如 GPT-5-Codex 系列），可以在本地代码库中直接读取、编写和修改代码，支持执行终端命令、运行测试、修复 Bug、编写新功能以及重构代码等任务。

主要特点：

- 终端原生：以命令行工具形式运行，适合开发者日常在终端中的工作流；
- 代码库感知：能够理解当前项目结构，直接在本地仓库中进行多文件修改；
- 支持 ChatGPT 账号登录：可直接使用 ChatGPT（Plus/Pro）订阅进行认证，也可使用 OpenAI API Key；
- 跨平台：支持 Windows、macOS 和 Linux。

下载地址

- GitHub 仓库：https://github.com/openai/codex
- Releases 下载：https://github.com/openai/codex/releases
- 官方文档：https://developers.openai.com/codex/cli

安装方式：

```
# npm 全局安装
npm install -g @openai/codex

# 或使用 Homebrew (macOS)
brew install codex
```

安装完成后执行 `codex` 并按提示完成登录即可开始使用。

Codex Auth Helper

Codex Auth Helper

简介

Codex Auth Helper 是一款专为 Codex 开发者设计的 Chrome 扩展程序（Manifest V3），用于安全地在本地导出已登录的 ChatGPT 会话凭证，并自动转换为符合 Codex 运行规范的 `auth.json` 配置备份文件，方便在 Codex CLI 中免登录复用认证信息。

- 项目地址：https://github.com/zhishile/codex-auth-helper
- 许可证：MIT License

主要功能

- 智能状态检测：自动检测浏览器中 ChatGPT 登录状态，展示头像、邮箱、订阅计划（Free/Plus/Pro）；
- Token 有效期倒计时：实时显示 Token 剩余生存时间；
- 一键生成 auth.json：自动构造 JWT 仿真 id_token 并导出配置文件；
- 100% 纯本地离线处理：数据不经第三方服务器，零上传、隐私安全；权限最小化（仅 `downloads` 与 `chatgpt.com`）。

下载地址

- Release 页面：https://github.com/zhishile/codex-auth-helper/releases/tag/v1.0.0
- 扩展下载包（extension.zip）：https://github.com/zhishile/codex-auth-helper/releases/download/v1.0.0/extension.zip

安装与使用

1. 下载并解压扩展包（或克隆仓库）；
2. 在 Chrome 中打开 `chrome://extensions/`，开启“开发者模式”；
3. 点击“加载已解压的扩展程序”，选择仓库中的 `extension` 文件夹；
4. 登录 ChatGPT 官网后点击插件图标，点击“生成并保存 auth.json”即可下载配置文件；
5. 将生成的 `auth.json` 放入 Codex 的配置目录（如 `~/.codex/`）即可供 Codex CLI 使用。

opencodex 介绍

opencodex

简介

opencodex（lidge-jun/opencodex，npm 包名 `@bitkyc08/opencodex` ，CLI 命令 `ocx` ）是一款通用模型中转代理（Universal Provider Proxy）：它把 OpenAI Codex 的 Responses API 翻译成各厂商自己的协议——流式输出、工具调用、推理 token、图片，双向转换。

一句话概括：让 Codex / Claude Code / Claude Desktop / Grok Build 都能跑任何大模型（Claude、Gemini、Grok、GLM、DeepSeek、Kimi、Qwen、Ollama…）。

核心功能

- 任意模型接入 Codex 系列客户端：40+ 内置 Provider（OpenAI、Anthropic、Google Gemini、xAI、Kimi、Azure OpenAI、Ollama、DeepSeek、Groq、OpenRouter、Together、Fireworks、Cerebras、Mistral、Hugging Face、NVIDIA NIM、MiniMax、通义、硅基流动等），或任意 OpenAI 兼容端点。
- ChatGPT 账号池管理：可添加多个 ChatGPT/Codex 账号，在面板里刷新 5 小时 / 每周 / 30 天配额；新会话自动路由到余量最充足的健康账号，已有会话保持"线程亲和"不跳号（配额重评估、故障切换、401/403/429 恢复时可重绑定）。
- Combos 组合模型：一个虚拟模型 ID，在不同 Provider 间做故障转移或加权轮询。
- 子代理换模型：在 Codex 的 sub-agent 选择器里路由模型，支持 v1/v2 界面控制与回退链。
- 免 API Key 登录：xAI、Anthropic、Kimi 支持 OAuth；也可转发 `codex login` 、直接粘贴 Key、或引用 `${ENV_VAR}` 环境变量。
- Web 搜索 & 视觉侧车：非 OpenAI 模型也能通过你的 ChatGPT 登录态获得真实联网搜索和图片理解能力。
- 可视化面板： `localhost:10100` 上查看 Provider、OAuth 状态、模型选择，以及带缓存 token 统计的实时请求日志。
- 干净退出零残留： `ocx stop` 会把 Codex 配置恢复原样；进程内所有长期状态都有内存上限（默认 256 MiB 预算），无无限增长的 Map/Set。

快速使用

安装与启动（人用）

```
npm install -g @bitkyc08/opencodex   # 需 Node 18+, Bun 运行时会自动随包捆绑
ocx start                            # 启动代理 + 面板，默认 localhost:10100
```

之后打开 http://localhost:10100 在网页面板里配置：添加 Provider、选模型、管理账号。 `ocx gui` 可随时重新打开面板， `ocx service` 以后台服务方式运行。

Agent 用法

```
npm install -g @bitkyc08/opencodex
ocx start     # 或 ocx service
ocx init      # 交互式初始化：写入 ~/.opencodex/config.json 并接好 Codex
```

> *先启动代理再执行其他命令：* `ocx provider add` 、 `ocx combo set` *等无头命令需要连接运行中的代理。* `ocx status` / `ocx doctor` / `ocx health` *可查看运行状态。Agent 安装请阅读仓库里的 AGENTS_INSTALL.md。*

模型路由

用 `provider/model` 语法指定任意已配置的 Provider 和模型：

```
codex -m "anthropic/claude-opus-5" "Explain this stack trace"
codex -m "google/gemini-3-pro" "Write unit tests for auth.ts"
codex -m "ollama/llama3" "Refactor this function"
```

省略 `provider/` 前缀则使用默认 Provider，或按模型名模式自动匹配。

从源码安装（最新 dev 版，Bun canary）

macOS / Linux：

```
curl -fsSL https://bun.sh/install | bash && ~/.bun/bin/bun upgrade --canary
git clone https://github.com/lidge-jun/opencodex.git
cd opencodex && ~/.bun/bin/bun install
~/.bun/bin/bun run src/cli/index.ts start
```

Windows（PowerShell）：

```
irm bun.sh/install.ps1 | iex; bun upgrade --canary
git clone https://github.com/lidge-jun/opencodex.git
cd opencodex; bun install
bun run src/cli/index.ts start
```

平台支持

系统	状态	服务管理
macOS (arm64 / x64)	完整支持	launchd
Linux (x64 / arm64)	完整支持	systemd（用户单元）
Windows (x64)	完整支持	任务计划程序（隐藏）/ 可选原生服务 `--native`（WinSW）

常用 CLI 命令

```
ocx start       # 启动代理与面板（localhost:10100）
ocx service     # 后台运行
ocx stop        # 停止并恢复 Codex 原始配置
ocx init        # 交互式初始化配置
ocx gui         # 重新打开面板
ocx status      # 查看运行状态
```

```
ocx doctor      # 诊断检查
ocx health      # 健康检查
```

文档与地址

- GitHub 仓库：https://github.com/lidge-jun/opencodex
- 完整文档：https://opencodex.me/
- npm 包：https://www.npmjs.com/package/@bitkyc08/opencodex
- 安装文档：https://opencodex.me/getting-started/installation/

> *注：本文档根据 opencodex README 整理，截至 2026-08-19。*

DeepSeek Harness 介绍

DeepSeek Harness 介绍

一、项目简介

DeepSeek Harness（代号 “Harness”）是 DeepSeek 于 2026 年 8 月 13 日晚正式发布的 AI 编程智能体（Agentic Coding）开源框架 开发者预览版，版本号为 v0.1，采用 MIT 协议 开源，定位对标 OpenAI Codex、Anthropic Claude Cowork 等 AI 编程智能体产品。

“Harness” 意为“马具 / 线束 / 安全绑带”，官方取其“把复杂能力可靠地束缚、编排在一起”的含义，希望成为开发者驾驭 AI 智能体的通用底座。

二、发布时间与定位

- 发布时间：2026 年 8 月 13 日（v0.1 开发者预览版）
- 项目定位：开源 AI 编程智能体框架 / Agentic Coding 平台
- 开源协议：MIT License
- 对标产品：OpenAI Codex、Anthropic Claude Cowork
- 团队信息：项目负责人为崔添翼（DeepSeek 技术团队），团队于 8 月 2 日左右开始征集内测用户，8 月 13 日正式发布

三、官方地址

- 官网：https://deepseek.com/harness/
- GitHub 仓库：https://github.com/deepseek-ai/deepseek-harness
- npm 包：`@deepseek-ai/dsh`

四、快速开始

方式一：通过 npm / npx 直接运行

```
npx @deepseek-ai/dsh web
```

启动后默认在本地 `http://127.0.0.1:3080` 提供 Web 界面。

方式二：从源码构建运行

```
git clone https://github.com/deepseek-ai/deepseek-harness.git
cd deepseek-harness
pnpm install
```

```
pnpm run build
pnpm dsh web
```

CLI 命令行使用

```
dsh --profile <name>
```

五、核心设计：一切皆插件

DeepSeek Harness 的核心理念是 “一切皆插件（Everything is a Plugin）”，底层基于 Cordis 插件系统 构建，具备高度的时空可组合性，开发者可以像搭积木一样自由组合、编排智能体的各项能力。

官方表示，Harness 旨在成为一个可扩展的智能体基础设施，让用户能够“可靠地驾驭” AI 智能体完成真实世界的编程任务。

六、四种预设模式

DeepSeek Harness 开箱即用，提供了四种预设工作模式：

1. 标准模式（Standard）：适合常规开发任务的默认智能体模式
2. PTC 模式（Plan-Test-Code）：先规划、再测试、后编码的严谨开发流程
3. 极简模式（Minimal）：轻量、快速的简洁模式
4. 创造模式（Creative）：面向创意与探索性开发的模式

七、参考来源

- DeepSeek Harness 官网：https://deepseek.com/harness/
- GitHub 仓库 README：https://github.com/deepseek-ai/deepseek-harness
- IT之家、量子位等科技媒体报道（2026 年 8 月 13 日）

基础知识

AI 基础知识概览

本文整理了大语言模型（LLM）及其核心概念、架构和工作原理，适合初学者快速理解 AI 系统。

一、底层引擎：大语言模型（LLM）

核心定义与架构

- LM 全称：Large Language Model（大语言模型），简称大模型
- 底层架构：基于 Transformer（2017 年 Google 提出的 Attention 机制）
- 工作原理：通过预测下一个最可能的词生成文本

发展里程碑

时间	事件	意义
2017 年	Transformer 架构提出	奠定大模型技术基础
2022 年底	GPT-3.5 发布	首个达到可用级别的大模型
2023 年3月	GPT-4 发布	大幅提升 AI 能力上限
2023 年后	Claude、Gemini 等模型涌现	AI 赛道竞争加剧，OpenAI AI 技术多方竞争

二、数据处理单元：Token

核心特性

- 定义：文本的最小处理单元，通过 Tokenizer 分词
- 编码过程：文本 → Token → Token ID（数字表示）
- 解码过程：Token → 文本

Token 与语言单位关系

语言单位	与 Token 的关系	示例
中文词语	可能拆分	“工作坊” → “工作” + “坊”

语言单位	与 Token 的关系	示例
英文单词	常见单词对应 1 个 Token	"hello" → 1 Token
复杂单词	可能拆分	"helpful" → "help" + "ful"
特殊符号	可能多个 Token 表示	→ 3 Token

Token 参考量

- 1 Token ≈ 0.75 个英文单词
- 1 Token ≈ 1.5-2 个中文字符
- 40 万 Token ≈ 30 万英文单词或 60-80 万汉字

OpenAI Tokenizer 工具

- 网址：https://platform.openai.com/tokenizer
- 功能：
  - 输入文本可直观查看对应 Token
  - 支持不同模型 Token 计算（如 GPT-3、GPT-4、GPT-5）
  - 帮助理解文本长度和 Context Window 消耗
- 用途：
  - 预测 Token 消耗，控制上下文容量
  - 优化 Prompt，避免超过模型限制
  - 调试模型输入输出

三、记忆单元：Context

核心概念

- 定义：模型每次处理任务时能访问的信息总和，类似“临时记忆”
- 组成部分：用户问题、对话历史、工具调用、System Prompt 等
- 容量限制：由 Context Window（上下文窗口）定义

主流模型 Context Window 对比

模型	Context Window（Token）	约合汉字数量
GPT-5.4	105 万	~157.5 万
Gemini 3.1 Pro	100 万	~150 万
Claude Opus 4.6	100 万	~150 万

突破 Context Window 限制

- RAG 技术（检索增强生成）：通过检索相关文档片段，降低 Token 消耗

四、指令交互：Prompt

定义与分类

- Prompt：给 AI 的任务指令，决定输出内容
- 分类：
  - User Prompt：用户具体任务，如“写一首诗”
  - System Prompt：系统设定规则，如“保持幽默风格”

Prompt 工程（Prompt Engineering）

- 核心原则：清晰、具体、明确
- 提示设计技巧：
  - 避免模糊指令
  - 拆分复杂任务
  - 结合上下文提供信息

五、外部能力扩展：Tool

核心作用

- 定义：模型调用外部工具或接口，实现能力扩展
- 工作流程：
  i. 用户输入或平台转发

ii. 模型生成工具调用指令

iii. 工具执行并返回结果

iv. 模型生成最终输出

六、工具标准化：MCP

- 全称：Model Context Protocol（模型上下文协议）
- 作用：统一工具接口，解决多平台调用不一致问题
- 典型示例：OpenAI、Anthropic、Google 各自有接入规范，通过 MCP 统一

七、命令行工具（CLI）

核心概念

- CLI（Command-Line Interface） 是通过命令行与 AI 或工具交互的方式
- 常用于：
  - 快速调用模型功能
  - 自动化任务
  - 与脚本或系统集成
- 优势：
  - 高效轻量，不依赖图形界面
  - 可组合管道命令完成复杂任务
  - 适合开发者、数据科学家和运维场景

典型 CLI 示例

- OpenAI CLI：调用模型生成、管理 API keys
- Gemini CLI / Claude CLI：调度 Agent 执行任务
- Codex CLI：编程辅助和代码生成

八、自主执行系统：Agent

- 定义：能够自主决策、调用工具、完成任务的系统
- 核心能力：多步骤推理、工具选择、流程控制
- 代表产品：Claude Code、Codex、Gemini CLI 等
- 典型模式：React、Plan and Execute

九、任务定制：Agent Skill

- 定义：给 Agent 的能力模块，包含任务规则和执行步骤
- 核心功能：
  - 名称与描述
  - 任务目标
  - 执行步骤、判断规则、输出格式
- 技术实现：
  - Markdown 文档管理
  - 存放于专用目录，方便调用
  - 可附加用户问题映射规则，实现智能指令

十、概念体系关系

LLM（大脑） → Token（数据单元） → Context（记忆空间） → Prompt（交互指令） → Tool（外部能力）→ MCP（工具标准化） → CLI（命令行工具） → Agent（决策系统） → Agent Skill（任务定制）

补充说明

- Transformer 架构奠定基础
- Token 化处理文本，是模型理解的核心
- CLI 是现实场景中操作 AI 的重要接口
- Agent 与 Skill 扩展了 LLM 的实际应用能力
- RAG 技术可突破 Context Window 限制，提高信息调用效率

> *本文整理了 AI 核心概念、工作流程、扩展模块、CLI 工具及历史发展，便于快速入门和实践。*

MCP 和 CLI

本文分析 MCP 和 CLI 在 AI 流程中的优缺点、使用场景和未来趋势。

一、核心问题

项目	说明
MCP	AI 工具标准，告诉模型一次性“我有这些工具”
CLI	命令行工具，类似 `ffmpeg`、`grep`，直接执行任务

二、CLI 胜在两点

1⃣ 省钱（Token 消耗少）

- MCP 模式：需要把所有工具说明发送给 AI
  - 示例：44 个工具说明，占用 >14,000 Token
- CLI 模式：仅用几行命令即可，Token 消耗大幅降低
- 结论：CLI 更经济，尤其在大模型调用频繁时

2⃣ 高效（执行效率高）

- MCP 流程：
  i. AI 思考 → 调用工具 → 等结果
  ii. 再思考 → 再调用 → 多次往返
- CLI 流程：
  i. AI 生成一条命令 → 本地执行所有步骤 → 输出最终结果
- 类比：CLI 像搭积木，工具自由组合，一条命令完成复杂流程

三、MCP 还有用吗？

优势	说明
更可控	参数明确，文件名包含特殊符号也不会出错

优势	说明
更安全	受限制，企业和云端使用更放心；CLI 可能误执行删除命令

> *MCP 并非过时，仍适用于企业和云端场景*

四、未来走向

方式	适用场景
CLI	个人开发者，需要快速、便宜、轻量化的场景
MCP	企业、云端，对安全要求高的场景

- 两者都会存在，各自占据不同领域

五、总结

- CLI 优势：省钱、执行效率高、灵活组合
- MCP 优势：可控、安全，适合企业级场景
- 策略：个人快速开发用 CLI，企业云端使用 MCP，两者结合可形成完整 AI 流程

AI网关

AI 网关的作用与市场趋势

一、什么是 AI 网关

AI 网关（AI Gateway / LLM Gateway）是位于应用与大模型服务之间的中间层，负责统一入口、智能路由、流量治理与安全管控，类似微服务架构中的 API 网关之于大模型调用。它对上提供统一 API，对下屏蔽数百个模型提供商的差异（认证方式、请求格式、计费模式均不同），让应用无需适配 SDK 细节，也无需关心底层模型由谁提供。

二、核心作用

1. 统一接入

一个端点对接所有模型：统一的 OpenAI 兼容接口（ `/v1/chat/completions` ）即可同时调用 OpenAI、Claude、Gemini、DeepSeek、Kimi、GLM 等上千种模型，应用零改造即可切换供应商。

2. 智能路由与容灾（高可用）

- 健康检查与熔断：自动识别限流（429）、5xx 故障并踢出问题提供商，毫秒级切换
- 自动故障转移（Fallback）：主模型失败/配额耗尽时，自动降级到备用模型链
- 多级路由策略：优先级、权重、轮询、最低成本、最高配额余量、最佳缓存命中率等（如 OmniRoute 提供 19 种策略）

3. 成本治理

- 配额与预算管控：按团队/项目/Key 设置每日预算与速率限制，防超支
- 路由降级链路：订阅账号 → API Key → 低价模型 → 免费档，把成本压到最低
- Token 压缩与语义缓存：压缩工具输出（可省 15–95% token）、缓存重复请求前缀，直接降低账单
- 成本归因：精确统计每次调用花在哪个模型多少钱

4. 可观测性

统一日志、指标、链路追踪与用量看板；每次响应携带路由决策、成本、延迟等元数据，故障可回溯、用量可审计。

5. 安全与合规

- 密钥集中托管、AES 加密存储，业务代码不再接触裸 Key
- Prompt 注入防护、内容过滤、敏感信息（密钥/手机号/身份证）脱敏
- 细粒度访问控制、审计日志，满足企业合规要求

三、市场主要产品

类型	代表产品	特点
开源自托管	LiteLLM（BerriAI）	社区最活跃的通用网关，Python 生态，代理 + SDK 双形态
开源自托管	One-API / New-API	中文社区最流行的网关，多模型、多 Key 计费，Docker 一键部署
开源自托管	Higress AI Gateway（阿里）	企业级入口网关的 AI 扩展，K8s/云原生场景
开源自托管	OmniRoute	面向编码 Agent 的免费网关，290 提供商、内置免费额度与 MCP/A2A，本地优先
商业托管	OpenRouter	模型转售市场，一个 Key 买所有模型，按量计费（含抽成）
商业托管	Portkey	企业级网关平台，强于治理、护栏与观测
商业托管	Kong AI Gateway	老牌 API 网关的 AI 扩展，企业集成能力成熟
商业托管	Cloudflare AI Gateway	部署在 CDN 边缘节点，缓存、限流与全球低延迟
观测类	Helicone	专注于 LLM 可观测性与分析

四、市场趋势（2026）

1. 从「接得通」到「接得稳」

多模型接入已不再是难点，竞争转向韧性：三级自愈（提供商熔断 → 连接冷却 → 模型锁定）、防惊群、429 退避等生产级能力成为标配。

2. 成本焦虑推动「自动省钱」

网关从被动代理转向主动省钱：基于实时定价的成本最优路由、订阅/API/低价/免费的分层降级链路、聚合免费额度最大化（OmniRoute 聚合约 15 亿免费 token/月）、Token 压缩与缓存复用成为卖点。

3. Agent 原生成为新战场

面向 Claude Code、Codex、Cursor、Cline 等编码 Agent的网关快速崛起：内置 MCP Server、A2A（Agent-to-Agent）协议、一键配置 30+ 工具、对工具调用密集场景做专项 token 压缩。这回答了「后端早就 Agent 化了，网关如何 Agent 化」的命题。

4. 安全与合规权重上升

数据不出域（本地优先、自托管）、Prompt 注入防护、凭据脱敏、密钥 AES 加密成为标配；企业客户对审计、RBAC、OIDC 的要求驱动网关向治理平台演化。

5. 两条路线分化：模型市场 vs 基础设施

OmniRoute 项目介绍

OmniRoute 项目介绍

项目定位

OmniRoute 是一个 MIT 开源的免费 AI 网关，口号是「Never stop coding」。它把 290 个 AI 提供商（其中 90+ 免费档）、1200+ 模型聚合到一个本地端点后面，让 Claude Code、Codex、Cursor、Cline、Copilot 等所有 OpenAI 兼容工具都能直接跑在免费或低价模型上。

- 仓库地址：https://github.com/diegosouzapw/OmniRoute
- npm 包：`omniroute`，Docker 镜像：`diegosouzapw/omniroute`
- 作者：Diego Souza（巴西），创始开源赞助方为 Kimi（月之暗面）

核心承诺（六大支柱）

1. 一个端点：所有请求走 `http://localhost:20128/v1`，兼容 OpenAI / Claude / Gemini / Responses API
2. 自动故障转移：毫秒级跨 290 个提供商降级，永不停机
3. Token 压缩：RTK + Caveman 双层堆叠压缩，节省 15–95% token（工具密集型会话平均约 89%）
4. $0 起步：90+ 免费档、40+ 永久免费，无需绑卡；聚合统计约 15.3 亿免费 token/月
5. 所有工具可用：一个配置适配 33+ 编码 Agent
6. 生产级能力：熔断器、TLS 指纹隐身、内置 MCP（104 个工具）、A2A、记忆、护栏、评估，25000+ 测试

旗舰功能：Combos（模型链路由）

Combo 是一串模型，OmniRoute 在其间自动路由：配额耗尽、提供商故障或成本飙升时，自动静默切换到下一个模型。

零配置 `auto`

直接设 `model: auto` 即可，另提供变体：

Model ID	优化目标
`auto`	均衡默认（LKGP，记住上次成功的提供商）
`auto/coding`	代码生成质量优先
`auto/fast`	低延迟优先
`auto/cheap`	每 token 成本最低
`auto/offline`	配额余量最多优先
`auto/smart`	质量优先 + 10% 探索

19 种路由策略

可混搭：`priority`、`fill-first`、`weighted`、`round-robin`、`p2c`、`least-used`、`random`、`strict-random`、`cost-optimized`、`headroom`、`reset-window`、`reset-aware`、`context-relay`、`context-optimized`、`cache-optimized`、`lkgp`、`auto`、`fusion`（多模型投票 + 裁判合成）、`pipeline`（链式流水线）。

三级自愈韧性

- Level 1 提供商熔断器：仅 408/5xx 触发，半开探测 + 惰性恢复
- Level 2 连接冷却：指数退避（×2），429 尊重 Retry-After，防惊群
- Level 3 模型锁定：单个模型 $^{429}/_{404}$ 只锁该模型，不波及连接

其他亮点

- Tier 级联降级：订阅账号 → API Key（DeepSeek、Groq）→ 低价（GLM、MiniMax）→ 免费档
- MCP / A2A / CLI：内置 MCP 服务器与 Agent-to-Agent 协议，`setup-*` 一条命令配置 12+ 编码工具
- 安全：每路由 prompt 注入检测、凭证脱敏护栏、本地优先 + AES-256-GCM 加密存 Key、可选 OIDC 登录
- 可观测：每个响应带 `X-OmniRoute-Decision` / `X-OmniRoute-*` 头，展示路由决策、成本与用量
- 媒体生成：图像 / 视频 / 音频 / OCR / 语音翻译
- 远程模式：scoped token 驱动远端实例，支持 VPS 部署

运行环境

Node.js 项目，存储用 `node:sqlite`（Node 22+）或 WASM 版 sql.js。部署形态：

- npm 全局安装：`npm i -g omniroute`
- Docker：AMD64 + ARM64 多架构
- Electron 桌面端：Windows / macOS / Linux
- Android Termux：手机 $^{24}/_{7}$ 运行，无需 root
- PWA、OpenCode 插件、源码运行

Docker 部署（本机已实测）

```
docker run -d --name omniroute --restart unless-stopped --stop-timeout 40 \
  -p 127.0.0.1:20128:20128 -v omniroute-data:/app/data diegosouzapw/omniroute:latest
```

- Dashboard：`http://127.0.0.1:20128/dashboard`
- 常用地址：`/dashboard/free-tiers`（免费额度汇总）
- 装完零配置即可用：

```
curl http://localhost:20128/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{"model":"auto","messages":[{"role":"user","content":"Hello!"}]}'
```