

Taskmaster AI - 通过AI任务管理来提升开发效率

📅 2025年7月9日 ⌚ 2 分钟阅读

#Taskmaster

#MCP

#AI Task Management

Taskmaster AI - 通过AI任务管理来提升开发效率，支持多种开发工具，如Cursor、Windsurf、VS Code、Claude Code CLI等。

简介

主 页 : <https://www.task-master.dev/> github:
<https://github.com/eyaltoledano/claude-task-master> npm:
<https://www.npmjs.com/package/task-master-ai>

Claude Task Master是一个用于基于AI任务管理的开源项目，适配多种开发工具（如Cursor、Windsurf、VS Code、Claude Code CLI等）。主要功能包括：

核心特点：

可以利用Claude及其他AI模型（如OpenAI、Anthropic、Google Gemini）高效生成任务并管理项目。

也可以委托MCP Client端的AI模型来完成大模型调用而不需要额外大模型配置。

支持任务解析、自动规划下一步、任务扩展以及实时研究。

配置与使用：

提供快速安装方式，支持MCP协议集成及命令行使用。

要求设置环境变量及API密钥（支持多个模型和提供商）。

无需API密钥即可通过Claude Code CLI使用Claude Code模型，或通过MCP Client的AI模型来完成大模型调用而不需要额外大模型配置。

主要命令：

项目初始化 (`task-master init`) 。

目录

文章信息

字数

阅读时间

发布时间

更新时间

标签

#Taskmaster

#MCP

#AI Task Management

解析需求文档生成任务。 (`task-master parse-prd`)

显示任务列表及指定任务。

执行研究或生成任务文件。

文档与使用指南：

提供全面的配置教程、任务结构说明及常见交互示例。

包括迁移说明、教程及环境变量设定。

许可协议：

遵循MIT协议，允许个人及商业使用、修改与分发，但禁止出售软件或作为托管服务。

支持：

最新版本为v0.19.0，支持多语言模型集成，适合复杂项目，通过详细PRD提升任务生成质量。

这是一个功能强大的任务管理系统，适用于AI驱动的开发流程，简化任务规划与实施。

主要功能

支持AI Code Assistant CLI

Claude Code CLI： 在0.18.0版本中支持。

Gemini CLI： 在0.19.0版本中支持。

调用MCP Client端的LLM

TBD： 预计在0.20.0版本中支持。

该功能允许Taskmaster通过MCP客户端（比如cursor）进行任务委托，而无需直接调用API密钥, 为已经为cursor等工具付费的用户减少了额外LLM API调用的费用。github PR 参考： [Delegate Task Master LLM calls to an MCP client without the need for API keys](#), 好处是：

简化设置：无需在Taskmaster中管理多个API密钥

成本控制：代理自行处理大语言模型（LLM） 计费和速率限制

灵活性：通过代理使用任何大语言模型（LLM） 提供商或本地模型

统一接口：无论大语言模型（LLM） 来源如何，相同的Taskmaster命令均可使用

该PR实现如下功能：

引入AgentLLMProvider类，通过MCP接口实现LLM任务委托，支持多种AI角色和混合提供商。

新增MCP工具，支持双向通信及状态跟踪，增强交互管理和错误处理机制。

提供详细的配置和使用示例，支持多种模型和定制化设置。

测试覆盖多种场景，包括Claude Desktop等多个客户端，确保兼容性和稳定性。

提供文档和规则支持，确保代理LLM工具的正确工作流程。

保持向后兼容性，支持传统提供商与新功能的结合使用。

安装

开始使用Taskmaster AI的最快方法是通过npm安装它，并在你的项目中初始化它：

1

```
npm install -g task-master-ai
```

配置

方式一：作为CLI单独使用

这个一般和第三方系统结合使用或让用户单独直接使用。这里不做过多介绍，可以参考 npm 相关网页：<https://www.npmjs.com/package/task-master-ai> 注意，CLI方式会自动读取.env文件。

CLI主要命令如下：

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

```
[  
  # Initialize a new project  
  task-master init  
  
  # Parse a PRD and generate tasks  
  task-master parse-prd your-prd.txt  
  
  # List all tasks  
  task-master list  
  
  # Show the next task to work on  
  task-master next  
  
  # Show specific task(s) - supports comma-separated IDs  
  task-master show 1,3,5  
  
  # Research fresh information with project context  
  task-master research "What are the latest best practices for JWT authentication?"  
  
  # Generate task files  
  task-master generate  
  
  # Add rules after initialization  
  task-master rules add windsurf,roo,vscode  
]
```

方式二：结合MCP Client使用（我的主要使用方式）

常见的客户端包括：Cursor, Windsurf, Claude Destop, Gemini CLI等。下面以**cursor**为例子。

初始化task master

在cursor的chatbox里面直接说“初始化task master”，然后他就会自动为task master在cursor里面生成相应的.taskmaster目录和相关配置文件,以及在.cursor目录下生成taskmaster目录和mcp.json文件。

```

1
2 .taskmaster/
3 |--config.json # 配置main, research和
4 fallback的模型信息。
5 |--state.json
6 |--tasks/
7
8 .cursor/
9 |--taskmaster/
10 |--mcp.json
11
12 .taskmaster/config.json配置里的**models**部
13 分如下
14 ```json
15 {
16   "models": {
17     "main": {
18       "provider": "agentllm",
19       "modelId": "agent-delegated-model",
20       "maxTokens": 1048000,
21       "temperature": 0.2
22     },
23     "research": {
24       "provider": "agentllm",
25       "modelId": "agent-delegated-model",
26       "maxTokens": 8700,
27       "temperature": 0.1
28     }
29   }
30 }
31 ]

```

在MCP Client里面配置MCP Server 配置mcp.json文件，配置task-master-ai的命令和环境变量。

注意: 如果配置使用cursor（MCP Client）的模型，下面的环境变量的KEY是不用配置的。比如在.taskmaster/config.json的models部分配置了agentllm.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

```
[
{
  "mcpServers": {
    "taskmaster-ai": {
      "command": "npx",
      "args": ["-y", "--package=task-
master-ai", "task-master-ai"],
      "env": {
        "ANTHROPIC_API_KEY":
"YOUR_ANTHROPIC_API_KEY_HERE",
        "PERPLEXITY_API_KEY":
"YOUR_PERPLEXITY_API_KEY_HERE",
        "OPENAI_API_KEY":
"YOUR_OPENAI_KEY_HERE",
        "GOOGLE_API_KEY":
"YOUR_GOOGLE_KEY_HERE",
        "MISTRAL_API_KEY":
"YOUR_MISTRAL_KEY_HERE",
        "OPENROUTER_API_KEY":
"YOUR_OPENROUTER_KEY_HERE",
        "XAI_API_KEY":
"YOUR_XAI_KEY_HERE",
        "AZURE_OPENAI_API_KEY":
"YOUR_AZURE_KEY_HERE",
        "OLLAMA_API_KEY":
"YOUR_OLLAMA_API_KEY_HERE"
      }
    }
  }
}
```

在cursor里面配置MCP Server, 参考: [taskmaster-ai的github](#)

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

```
[
  "taskmaster-ai": {
    "command": "npx",
    "args": ["-y", "--package=task-
master-ai", "task-master-ai"],
    "env": {
      "ANTHROPIC_API_KEY":
"YOUR_ANTHROPIC_API_KEY_HERE",
      "PERPLEXITY_API_KEY":
"YOUR_PERPLEXITY_API_KEY_HERE",
      "OPENAI_API_KEY":
"YOUR_OPENAI_KEY_HERE",
      "GOOGLE_API_KEY":
"YOUR_GOOGLE_KEY_HERE",
      "MISTRAL_API_KEY":
"YOUR_MISTRAL_KEY_HERE",
      "OPENROUTER_API_KEY":
"YOUR_OPENROUTER_KEY_HERE",
      "XAI_API_KEY":
"YOUR_XAI_KEY_HERE",
      "AZURE_OPENAI_API_KEY":
"YOUR_AZURE_KEY_HERE",
      "OLLAMA_API_KEY":
"YOUR_OLLAMA_API_KEY_HERE"
    }
  }
]
```

配好后，enable taskmaster-ai MCP server，当左下角显示绿点时，表示配好并运行成功了。

分享这篇文章



相关文章推荐

DeerFlow - 字节跳动...

DeerFlow - 字节跳动开源的De...

模型上下文 协议...

本文介绍了模型
上 下 文 协 ...

模型上下文 协议...

本文介绍了模型
上 下 文 协 ...