# 模型上下文协议（MCP）深度解析：Agent互操作性的新纪元

2025年4月29日 5 分钟阅读

#AI #Agent #MCP #Protocol

> *本文介绍了模型上下文协议（MCP），并对其技术原理、主要贡献、当前优劣、生态系统现状，并与Google A2A等相关技术进行比较，展望其未来发展趋势。*

## MCP Hub 列表

Anthropic 官网提供的MCP Server 列表 - MCP Services

DockerHub的MCP Server列表 - 探索精心挑选的100多个安全、高质量的MCP服务器Docker镜像集合，涵盖数据库解决方案、开发工具、生产力平台和API集成

Mcp 相关的热门 GitHub AI项目仓库

### 国内的MCP服务器列表

魔搭MCP广场 - 平台验证可托管的MCP服务，已通过标记。 更多社区MCP服务验证中。

阿里pay百宝箱

AIBase

## 常用MCP Server

### 测试用MCP Server - Everything MCP Server

Dockerhub Everything MCP Server -> Docker Image: mcp/everything

Github Everything MCP Server

目录

文章信息

字数

阅读时间

发布时间

更新时间

标签

#AI #Agent #MC

```
{
  "mcpServers": {
    "everything": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-everything"
      ]
    }
  }
}
```

## Github MCP Server

github官方推出的封装Github API的MCP Server github mcp server 魔搭上也有相应的说明：https://www.modelscope.cn/mcp/servers/@modelcontextprotocol/github
Dockerhub上也有相应的说明：https://hub.docker.com/r/mcp/github-mcp-server

## Playwright MCP Server

使用 Playwright 提供浏览器自动化功能的MCP Server 能使 LLM 能够通过结构化的辅助功能快照与网页进行交互，而无需屏幕截图或视觉调整模型。

- Playwright的MCP Server
- Playwright MCP Server Dockerfile
- Playwright MCP Server NPM

### Playwright MCP Server 客户端配置示例

```
{
  "mcpServers": {
    "playwright": {
      "command": "npx",
      "args": [
        "@playwright/mcp@latest"
      ]
    }
  }
}
```

### Playwright MCP Server 本身完整配置示例

Playwright MCP 服务器可以通过一个 JSON 配置文件进行配置(参考Playwright MCP Server 配置文件)。以下是完整的配置格式：

```
{
  // Browser configuration
  browser?: {
    // Browser type to use (chromium, firefox, or
webkit)
    browserName?: 'chromium' | 'firefox' | 'webkit';

    // Path to user data directory for browser profile
persistence
    userDataDir?: string;

    // Browser launch options (see Playwright docs)
    // @see https://playwright.dev/docs/api/class-
browsertype#browser-type-launch
    launchOptions?: {
      channel?: string;        // Browser channel (e.g.
'chrome')
      headless?: boolean;      // Run in headless mode
      executablePath?: string; // Path to browser
executable
      // ... other Playwright launch options
    };

    // Browser context options
    // @see https://playwright.dev/docs/api/class-
browser#browser-new-context
    contextOptions?: {
      viewport?: { width: number, height: number };
      // ... other Playwright context options
    };

    // CDP endpoint for connecting to existing browser
    cdpEndpoint?: string;

    // Remote Playwright server endpoint
    remoteEndpoint?: string;
  },

  // Server configuration
  server?: {
    port?: number;  // Port to listen on
    host?: string;  // Host to bind to (default:
localhost)
  },

  // List of enabled capabilities
  capabilities?: Array<
    'core' |    // Core browser automation
    'tabs' |    // Tab management
    'pdf' |     // PDF generation
    'history' | // Browser history
    'wait' |    // Wait utilities
    'files' |   // File handling
    'install'   // Browser installation
  >;

  // Enable vision mode (screenshots instead of
accessibility snapshots)
  vision?: boolean;

  // Directory for output files
  outputDir?: string;

  // Tool-specific configurations
  tools?: {
    browser_take_screenshot?: {
      // Disable base64-encoded image responses
      omitBase64?: boolean;
    }
  }
}
```

用户可以使用–config命令行选项指定配置文件：

```
npx @playwright/mcp@latest --config path/to/config.json
```

另外playwright支持两种模式，这两种模式代表了自动化测试的两种不同方法：

快照Snapshot模式（缺省模式）：这个是基于 DOM 的测试，通过访问页面的 DOM 结构来识别和操作元素，速度快且可靠，但在某些复杂或动态界面可能受限。

视觉Vision模式：通过图像识别和坐标定位来操作元素，能够处理传统选择器难以识别的元素，但可能更消耗资源。

```
{
  "mcpServers": {
    "playwright": {
      "command": "npx",
      "args": [
        "@playwright/mcp@latest",
        "--vision"
      ]
    }
  }
}
```

## Desktop Commander MCP Server

这是面向Claude的MCP服务器，它赋予Claude终端控制、文件系统搜索以及差异文件编辑功能。

Dockerhub 地址

Github 地址

```
{
  "mcpServers": {
    "desktop-commander": {
      "command": "npx",
      "args": [
        "-y",
        "@wonderwhy-er/desktop-commander"
      ]
    }
  }
}
```

## Alipay MCP Server

支付宝Alipay的MCP Server

```
最终用户设备                                    Agent 运行环境
```

+————————————+ +————————————————+ +————————————+ | | 交流 | 支付宝 MCP Server + | | | | 小程序/WebApp |<——>| 其他 MCP Server + |<—->| 支付服务 | | | 支 付 | Agent 开 发 工 具 | | 交 易 / 退 款 / 查 询 | +————————————+ + ————————————————+ +————————————+ 创作服务买家 智能工具开发者 支付宝开放平台

```
（最终用户）                                    （创作者）
```

### 支付宝MCP Server 配置示例

@alipay/mcp-server-alipay 是支付宝开放平台提供的 MCP Server，让你可以轻松将支付宝开放平台提供的交易创建、查询、退款等能力集成到你的 LLM 应用中，并进一步创建具备支付能力的智能工具。

```
{
  "mcpServers": {
    "mcp-server-alipay": {
      "command": "npx",
      "args": ["-y", "@alipay/mcp-server-alipay"],
      "env": {
        "AP_APP_ID": "2014...222",
        "AP_APP_KEY": "MIIE...DZdM=",
        "AP_PUB_KEY": "MIIB...DAQAB",
        "AP_RETURN_URL": "https://success-page",
        "AP_NOTIFY_URL": "https://your-own-server",
        "...其他参数": "...其他值"
      }
    },
    "其他工具": {
      "...": "..."
    }
  }
}
```

**所有环境变量** 支付宝 MCP Server 通过环境变量接收参数。所有参数和默认值包括:

AP_APP_ID=2014...222 # 商户在开放平台申请的应用 ID（APPID）。必需。 AP_APP_KEY=MIIE...DZdM= # 商户在开放平台申请的应用私钥。必需。 AP_PUB_KEY=MIIB...DAQAB # 用于验证支付宝服务端数据签名的支付宝公钥，在开放平台获取。必需。 AP_RETURN_URL=https://success-page # 网页支付完成后对付款用户展示的「同步结果返回地址」。 AP_NOTIFY_URL=https://your-own-server # 支付完成后，用于告知开发者支付结果的「异步结果通知地址」。 AP_ENCRYPTION_ALGO=RSA2 # 商户在开放平台配置的参数签名方式。可选值为 “RSA2” 或 “RSA”。缺省值为 “RSA2”。 AP_CURRENT_ENV=prod # 连接的支付宝开放平台环境。可选值为 “prod”（线上环境）或 “sandbox”（沙箱环境）。缺省值为 “prod”。

# MCP Server 配置

AP_SELECT_TOOLS=all # 允许使用的工具。可选值为 "all" 或逗号分隔的工具名称列表。工具名称包括 `mobilePay`, `webPagePay`, `queryPay`, `refundPay`, `refundQuery` 。缺省值为 "all"。 AP_LOG_ENABLED=true # 是否在 $HOME/mcp-server-alipay.log 中记录日志。默认值为 true。

## Cursor MCP 使用

### 全局MCP服务器配置方法

进入Cursor Settings > MCP >"Add New Global MCP server"。

下面是使用github-mcp-server的配置示例： 该格式是Anthropic MCP服务器的配置格式。

```
{
  "mcpServers": {
    "github": {
      "command": "docker",
      "args": [
        "run",
        "-i",
        "--rm",
        "-e",
        "GITHUB_PERSONAL_ACCESS_TOKEN",
        "ghcr.io/github/github-mcp-server"
      ],
      "env": {
        "GITHUB_PERSONAL_ACCESS_TOKEN": "Your GitHub
Personal Access Token"
      }
    }
  }
}
```

### 单个开发项目MCP服务器配置方法

在.cursor/mcp.json 加入MCP Server的配置，和全局方式类似。

## VS Code MCP 使用

VS Code支持MCP服务器传输的本地标准输入/输出（stdio）和服务器发送事件（sse）。目前，在三个原语（tools, prompts, resources）中，服务器只能向Copilot的代理模式提供工具。工具的列表和描述可以使用列表更改事件动态更新。VS Code使用roots（规范）向服务器提供当前工作区文件夹。

MCP的官方服务器存储库是一个很好的起点，可用于参考官方和社区贡献的服务器，这些服务器展示了MCP的多功能性。你可以探索具有各种功能的服务器，例如文件系统操作、数据库交互和Web服务。

## VS Code中的MCP服务器配置方法

在Visual Studio Code中配置MCP Server的方法有以下几种：

### 1. 工作区设置

在工作区中添加`.vscode/mcp.json`文件，用于配置MCP服务器，并可与团队成员共享配置。

```
{
  // 💡 Inputs are prompted on first server start, then stored securely by VS Code.
  "inputs": [
    {
      "type": "promptString",
      "id": "perplexity-key",
      "description": "Perplexity API Key",
      "password": true
    }
  ],
  "servers": {
    // https://github.com/ppl-ai/modelcontextprotocol/
    "Perplexity": {
      "type": "stdio",
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-perplexity-ask"],
      "env": {
        "PERPLEXITY_API_KEY": "${input:perplexity-key}"
      }
    }
  }
}
```

### 2. 用户设置

在用户设置中指定服务器配置，这样可以在所有工作区中启用MCP服务器。

```
// settings.json
{
  "mcp": {
    "servers": {
      "my-mcp-server": {
        "type": "stdio",
        "command": "my-command",
        "args": []
      }
    }
  }
}
```

### 3. 自动发现

启用MCP服务器的自动发现功能，可以自动检测在其他工具（如Claude Desktop）中定义的MCP服务器。

通过 chat.mcp.discovery.enabled 设置启用自动发现功能。

## 配置示例

以下代码片段展示了一个示例MCP服务器配置，该配置指定了三台服务器，并为API密钥定义了一个输入占位符。

```
// Example .vscode/mcp.json
{
  // 💡 Inputs will be prompted on first server start,
  //    then stored securely by VS Code.
  "inputs": [
    {
      "type": "promptString",
      "id": "perplexity-key",
      "description": "Perplexity API Key",
      "password": true
    }
  ],
  "servers": {
    // https://github.com/ppl-ai/modelcontextprotocol/
    "Perplexity": {
      "type": "stdio",
      "command": "docker",
      "args": ["run", "-i", "--rm", "-e", "PERPLEXITY_API_KEY",
"mcp/perplexity-ask"],
      "env": {
        "PERPLEXITY_API_KEY": "${input:perplexity-key}"
      }
    },
    //
https://github.com/modelcontextprotocol/servers/tree/main/src/fetch
    "fetch": {
      "type": "stdio",
      "command": "uvx",
      "args": ["mcp-server-fetch"]
    },
    "my-remote-server": {
      "type": "sse",
      "url": "http://api.contoso.com/sse",
      "headers": { "VERSION": "1.2" }
    }
  }
}
```

“servers”：{}字段保存MCP服务器列表，并遵循Claude桌面版的配置格式。

“inputs”：[]字段允许你为配置值定义自定义占位符，避免硬编码敏感信息。

## 在代理模式下使用MCP工具

添加 MCP 服务器后（比如在上面的“2. 用户设置”之后），你可以在代理模式下使用它提供的工具。要在代理模式下使用 MCP 工具：

打开聊天视图（按Ctrl+Alt+I），然后从下拉菜单中选择“Agent”。

选择“工具”按钮以查看可用工具列表。

你也可以通过输入“#”加上工具名称，在提示中直接引用某个工具。在所有聊天模式（提问、编辑和智能体模式）下都能这样做。

现在你可以在聊天输入框中输入提示，留意工具是如何根据需要自动调用的。

默认情况下，调用工具时，需要在运行前确认操作。这是因为工具可能会在你的本地计算机上运行，并且可能会执行修改文件或数据的操作。使用“继续”按钮的下拉选项，可针对当前会话、工作区或所有未来调用自动确认特定工具。

（Optionally），在运行工具之前验证并编辑工具输入参数。）

选择工具名称旁边的箭头，以查看其详细信息和输入参数。在运行该工具之前，你可以编辑输入参数。

## 创建MCP Server

VS Code 拥有开发自己的 MCP 服务器所需的所有工具。虽然 MCP 服务器可以用任何能够处理标准输出的语言编写，但 MCP 的官方软件开发工具包（SDK）是一个很好的起点：

- TypeScript SDK
- Python SDK
- Java SDK
- Kotlin SDK
- C# SDK

## 参考

Use MCP servers in VS Code (Preview)

AI Agent Gateway