
今年是agent的一年

2025年是智能体的元年，
也注定是智能体集中爆发的一年！

两个互联领域的重大挑战：

第一、Agent 与 Tools（工具）的交互

Agent 需要调用外部工具和API、访问数据库、执行代码等。

第二、Agent 与 Agent（其他智能体或用户）的交互

Agent 需要理解其他 Agent 的意图、协同完成任务、与用户进行自然的对话。

两个互联领域的重大挑战：

第一、Agent 与 Tools（工具）的交互

Agent 需要调用外部工具和API、访问数据库、执行代码等。 ⇨ **MCP**

第二、Agent 与 Agent（其他智能体或用户）的交互

Agent 需要理解其他 Agent 的意图、协同完成任务、与用户进行自然的对话。

两个互联领域的重大挑战：

第一、Agent 与 Tools（工具）的交互

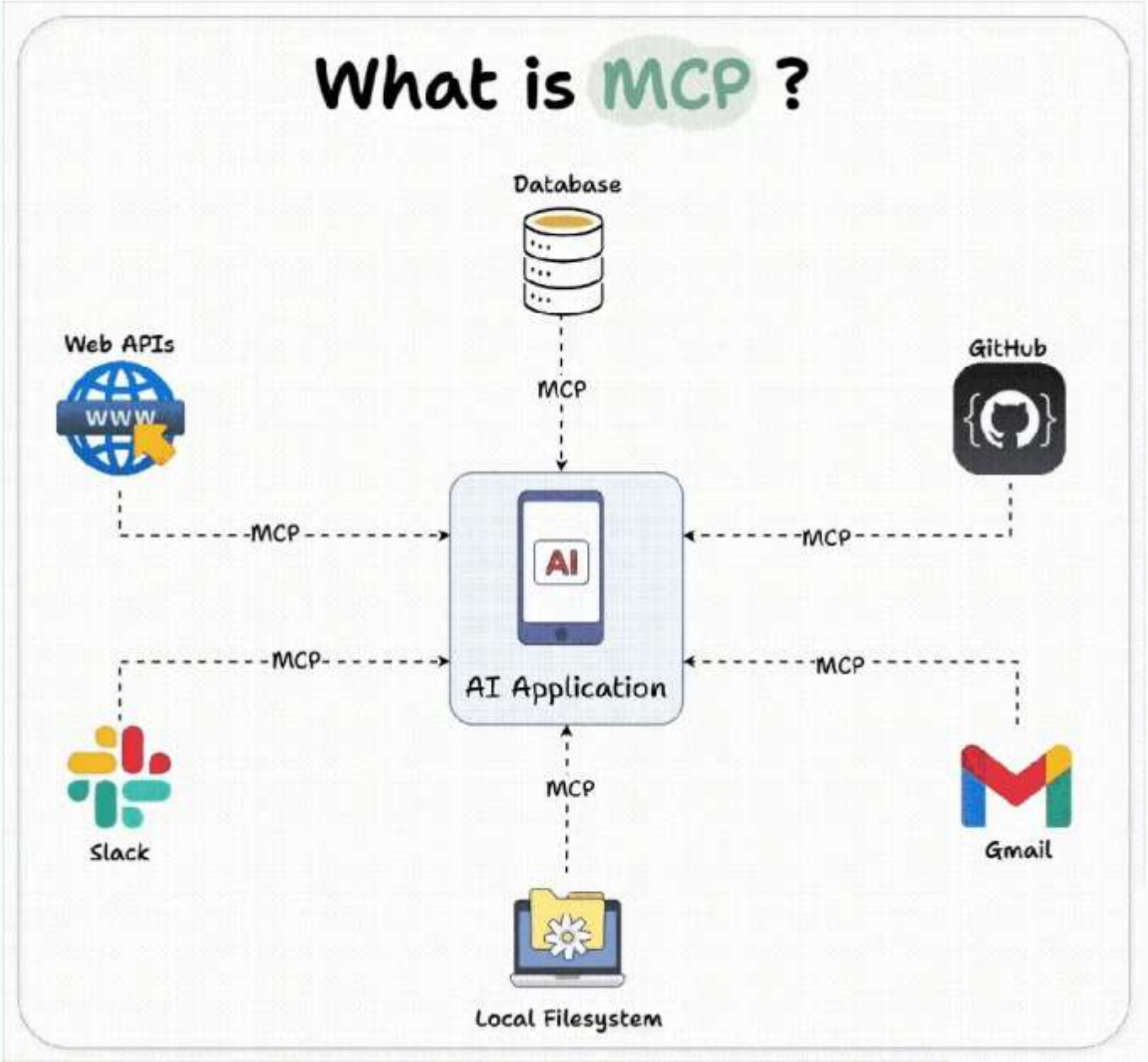
Agent 需要调用外部工具和API、访问数据库、执行代码等。 ⇨ **MCP**

第二、Agent 与 Agent（其他智能体或用户）的交互

Agent 需要理解其他 Agent 的意图、协同完成任务、与用户进行自然的对话。 ⇨ **A2A**

LLM（DeepSeek） + MCP

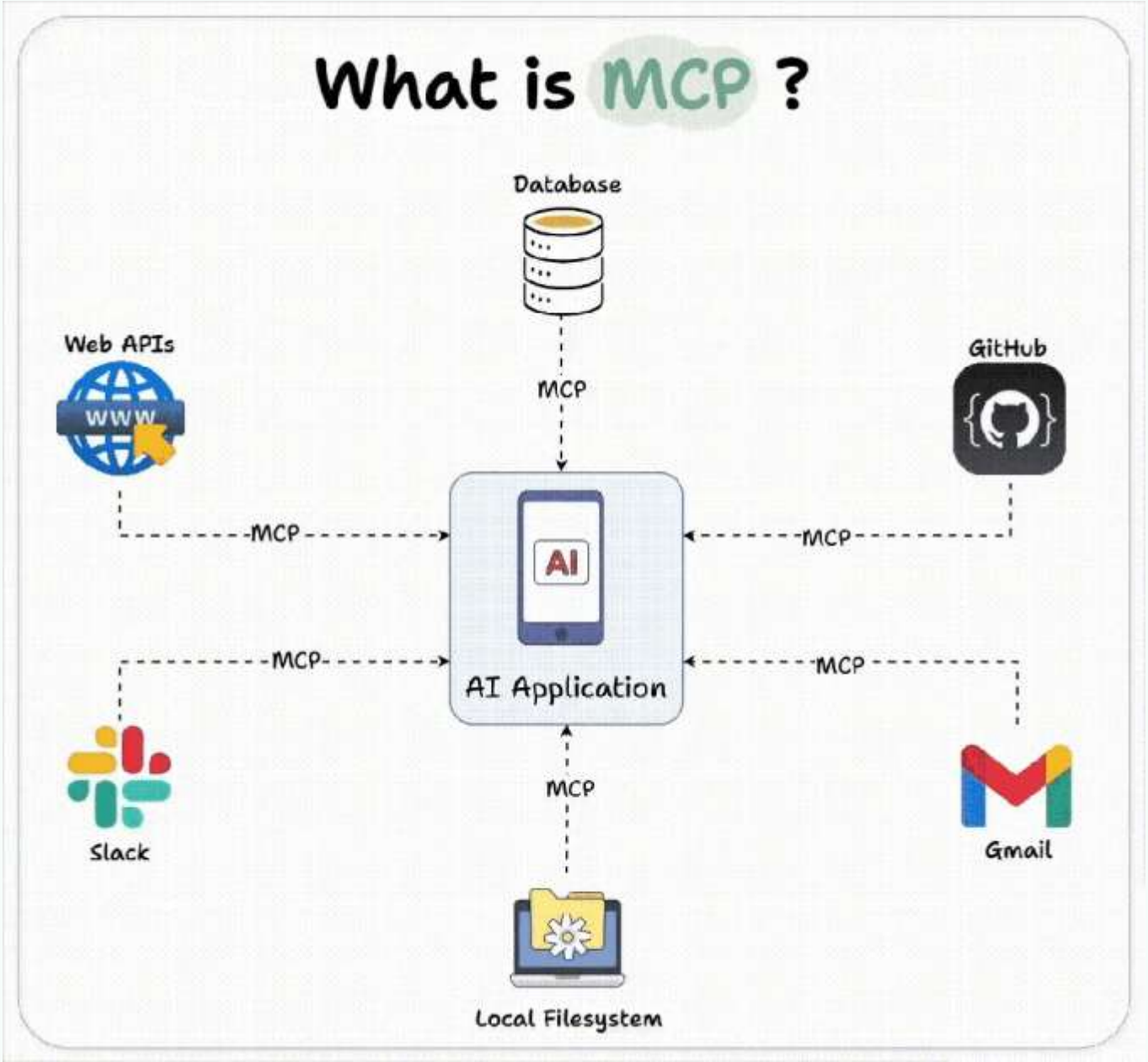
智能体?



LLM（DeepSeek） + MCP

智能体?

MCP就是大模型连接世界的标准、桥梁!



MCP实战指南

讲师：尚硅谷宋红康（江湖人称：康师傅）

如果你是程序员：

电商场景：

- **智能推荐系统：**商品推荐、用户行为分析、个性化推送
- **智能客服与订单管理：**自动问答、订单信息捕捉、需求分析
- **库存预测与动态定价：**库存监控、需求预测、价格策略优化

社交场景：

- **内容审核与情感分析：**敏感词过滤、图片/视频违规检测、用户情绪识别
- **社交关系推荐：**好友推荐、社群匹配、兴趣聚类
- **聊天机器人：**自动回复、上下文理解、多轮对话

物流场景：

- **智能仓储管理：**库存分拣、路径规划、异常检测
- **配送路线优化：**实时路径计算、交通预测、成本控制
- **需求预测与资源调度：**运力分配、仓库选址、峰值预测

金融场景：

- **风险评估与信贷审批：**信用评分、反欺诈、贷款决策
- **智能投顾与财富管理：**资产配置、市场预测、个性化理财建议
- **交易监控与反洗钱：**异常交易检测、合规审查、模式识别

如果你是程序员：

Java语言

+

Spring AI / LangChain / LangChain4J

+

MCP

如果你是程序员：

Java语言

+

Spring AI / LangChain / LangChain4J

+

MCP

==> AI智能落地项目

如果你是大众用户：如何理解MCP

有了DeepSeek，你就有了一个"智能助理"。但是，我们期望 LLM 能够承担更多功能，不仅限于简单对话，还能与外部的多种数据、工具进行交互。有了MCP，成为了现实！

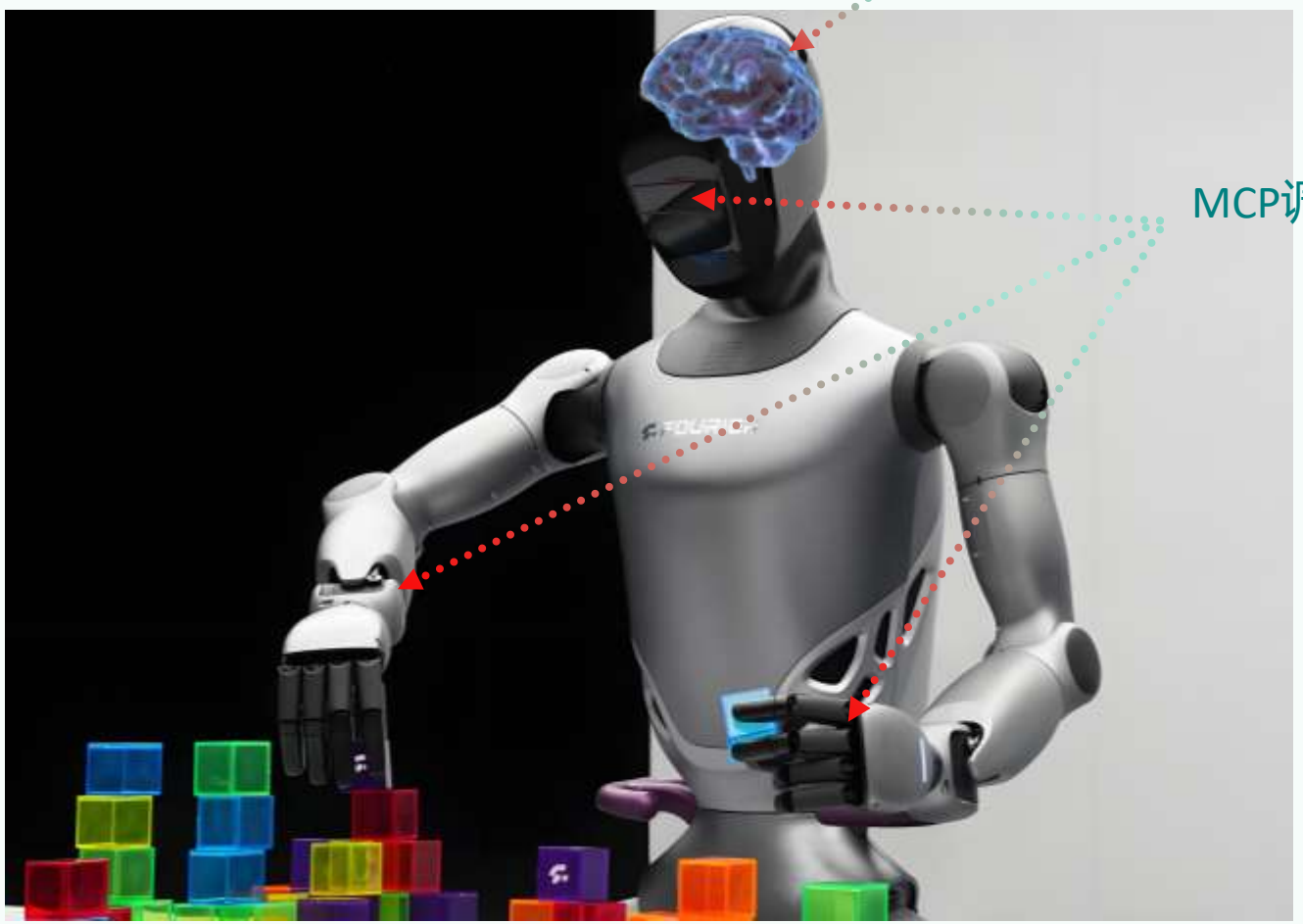
如果你是大众用户：如何理解LLM与MCP

如果你是大众用户：如何理解LLM与MCP

大模型(比如：DeepSeek)

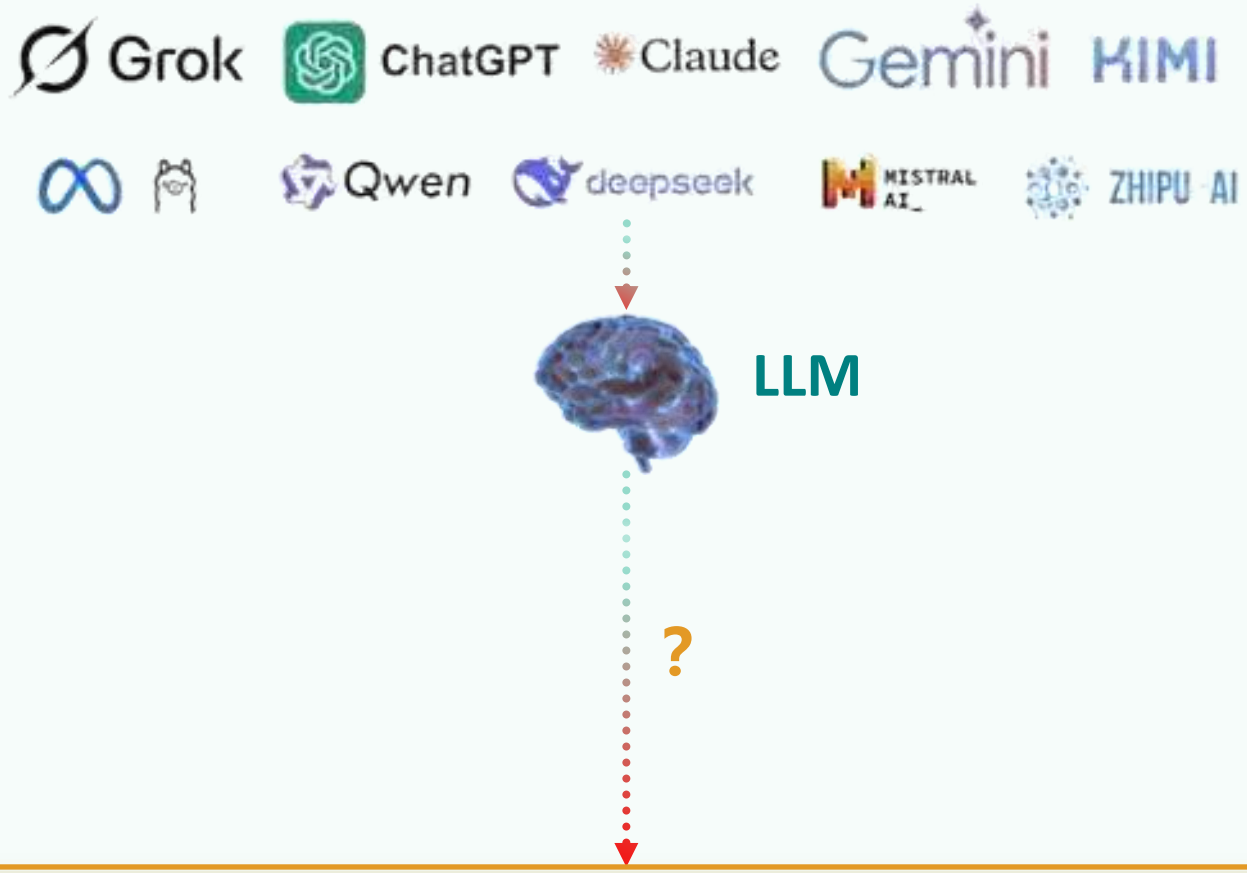
如果你是大众用户：如何理解LLM与MCP

大模型(比如：DeepSeek)

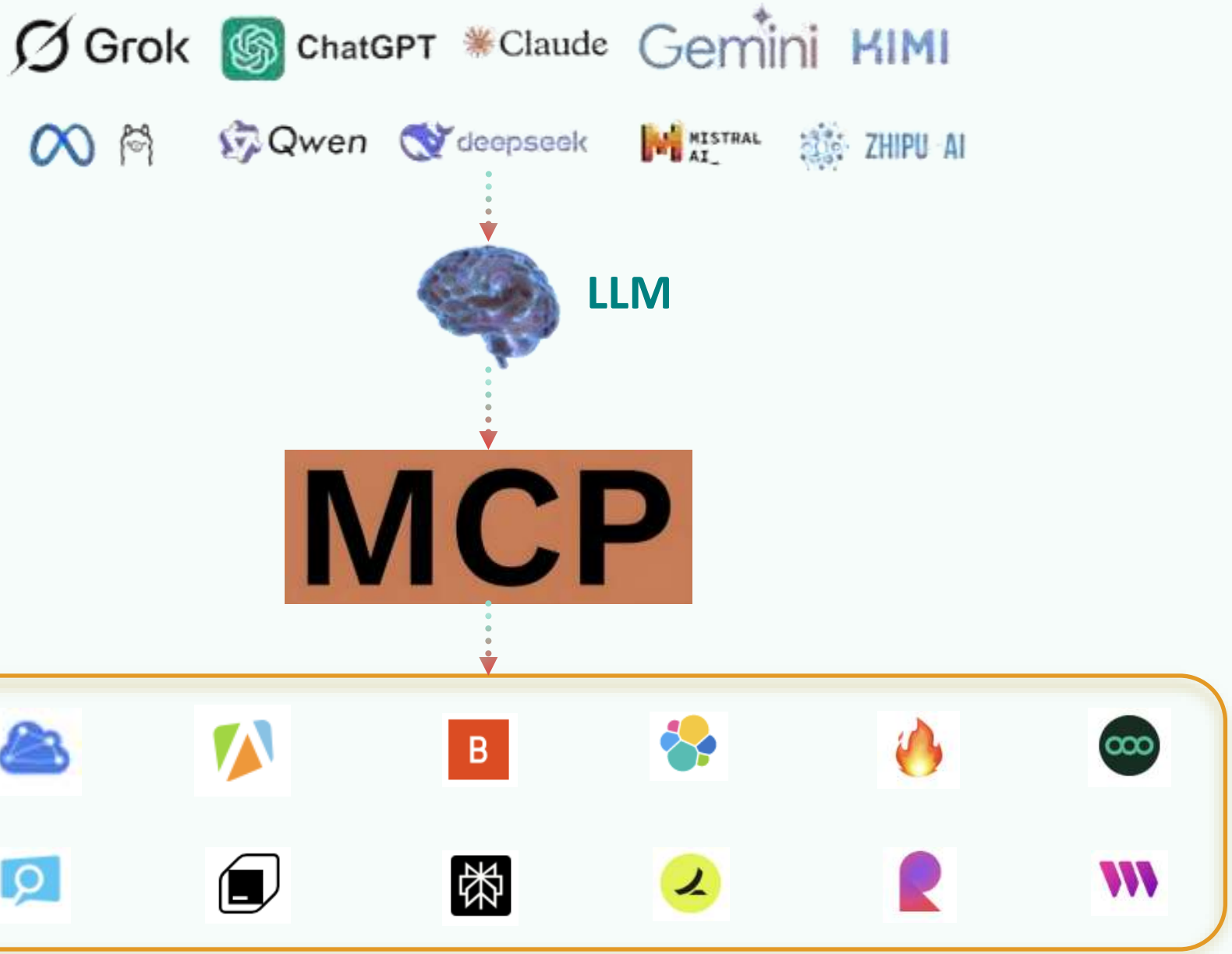


MCP调用工具

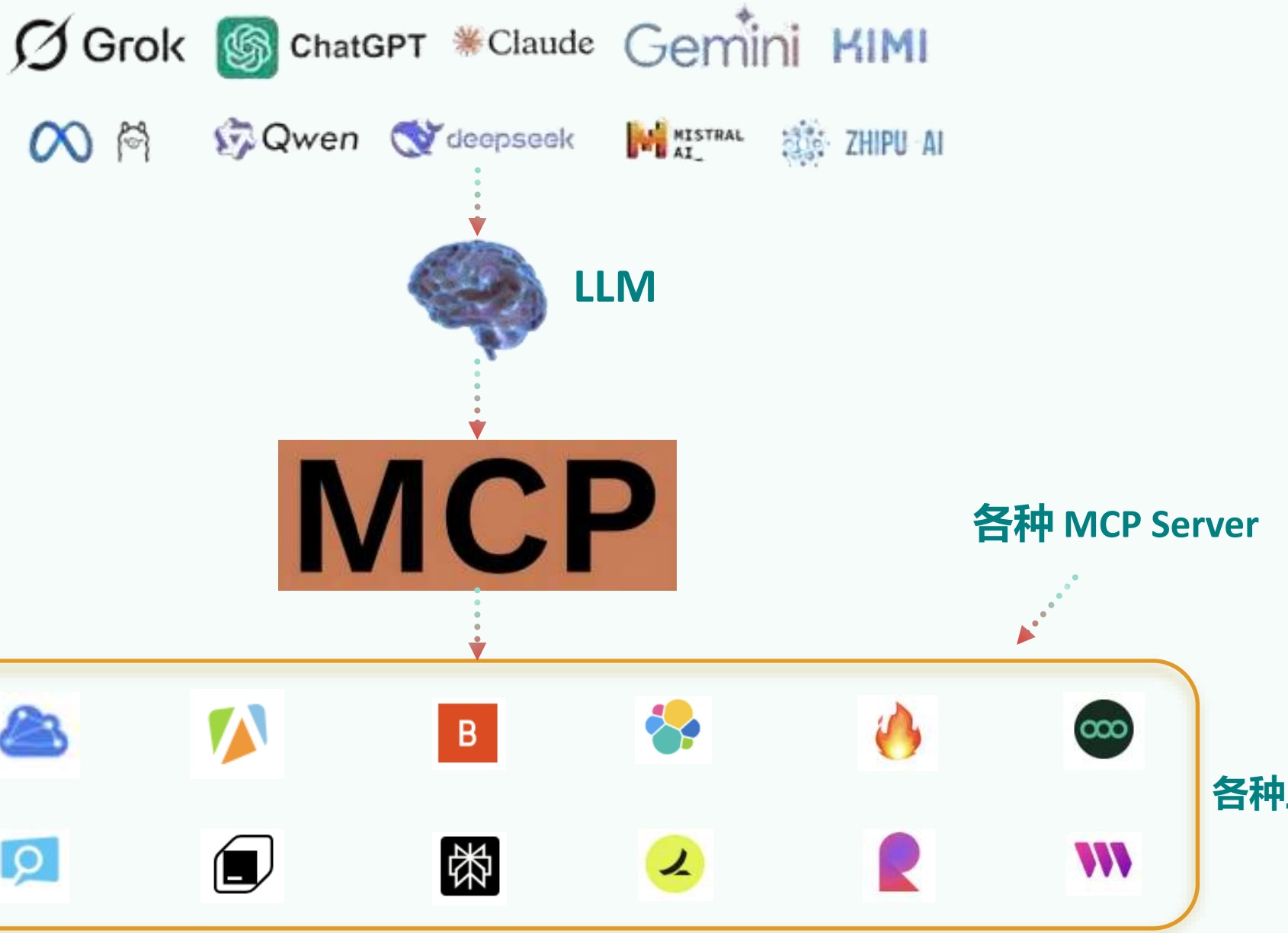
如何理解LLM与MCP



如何理解LLM与MCP



如何理解LLM与MCP



感慨一句

没想到千帆过境的大模型之争，竟然被一个MCP标准协议统一了。

deepseek

尚硅谷 www.atguigu.com

MCP实战指南 目录

1. MCP能干什么？

对于程序员来说

举例1：开发部署

开发者通过自然语言指令“部署新版本到测试环境”，触发 MCP 链式调用 GitLab API（代码合并）、Jenkins API（构建镜像）、Slack API（通知团队）。

举例2：SQL查询

开发者通过自然语言输入，比如“查询某集团部门上个季度销售额”，就能查询出数据库的数据，并结合大模型进行回答，不再需要编写 SQL，MCP 自动转换为精准 SQL 语句并执行。

对于程序员来说

举例3：manus智能体

Manus的每一次任务处理都至少需要调用网页搜索、网页访问、网页信息获取、本地文件创建、代码解释器等几十个外部工具。

这里就暴露了两个问题。

问题1：可供大模型调用的工具不足。

对于程序员来说

举例3：manus智能体

这里就暴露了两个问题。

问题2：调用工作量很大。

借助 MCP，只要支持了该协议，就能轻松将各种数据源和工具连接到 LLM。

对于大众用户来说

举例1：旅游规划

当我要去旅行时，旅行规划助手通过 MCP 同时调用天气 API（获取目的地气象）、交通 API（查询航班动态）、地图 API（规划路线），AI 自动生成带实时数据的行程方案。

对于大众用户来说

举例2：联网搜索

我们在与 LLM 交互时，经常需要联网搜索最新信息以减少幻觉。然而，这里也存在问题：

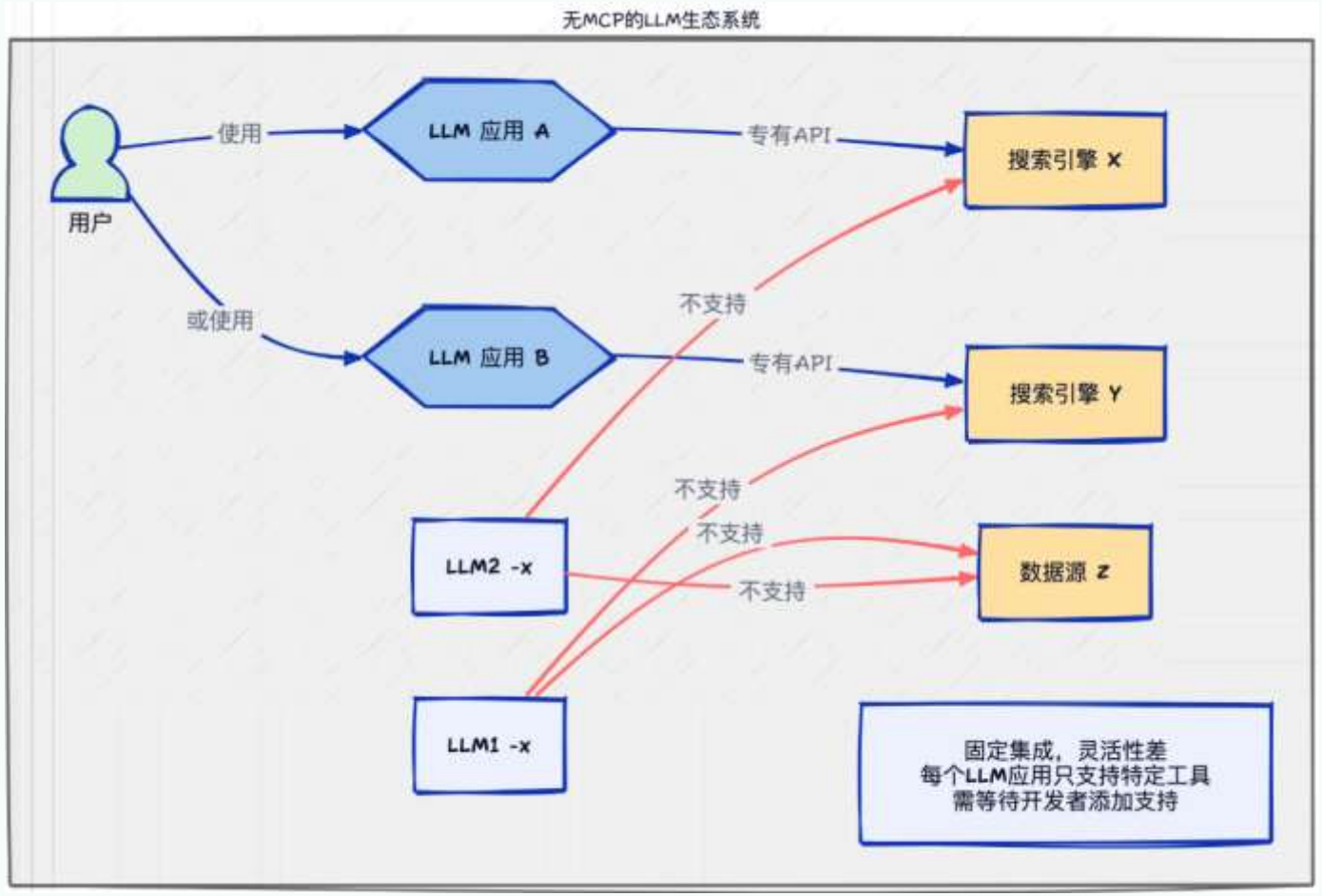
1、并非所有聊天机器人都支持联网功能

2、即使支持联网，也可能不包含你习惯使用的搜索引擎。

对于大众用户来说

举例2：联网搜索

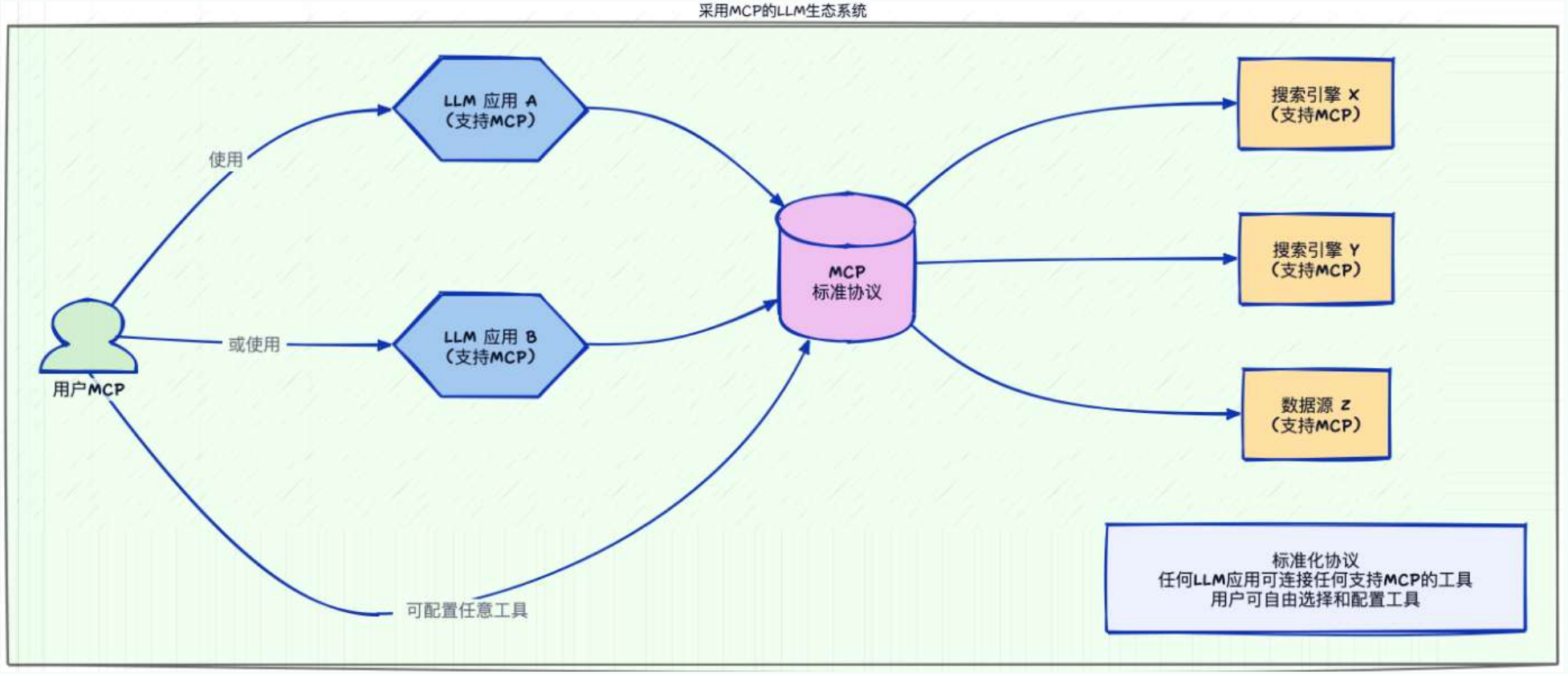
在没有 MCP 的情况下，用户只能等待开发者添加特定搜索引擎的支持。



对于大众用户来说

举例2：联网搜索

有了 MCP 后，只需简单配置，就能将所需服务接入当前使用的聊天机器人。



对于大众用户来说

举例3：业绩查询

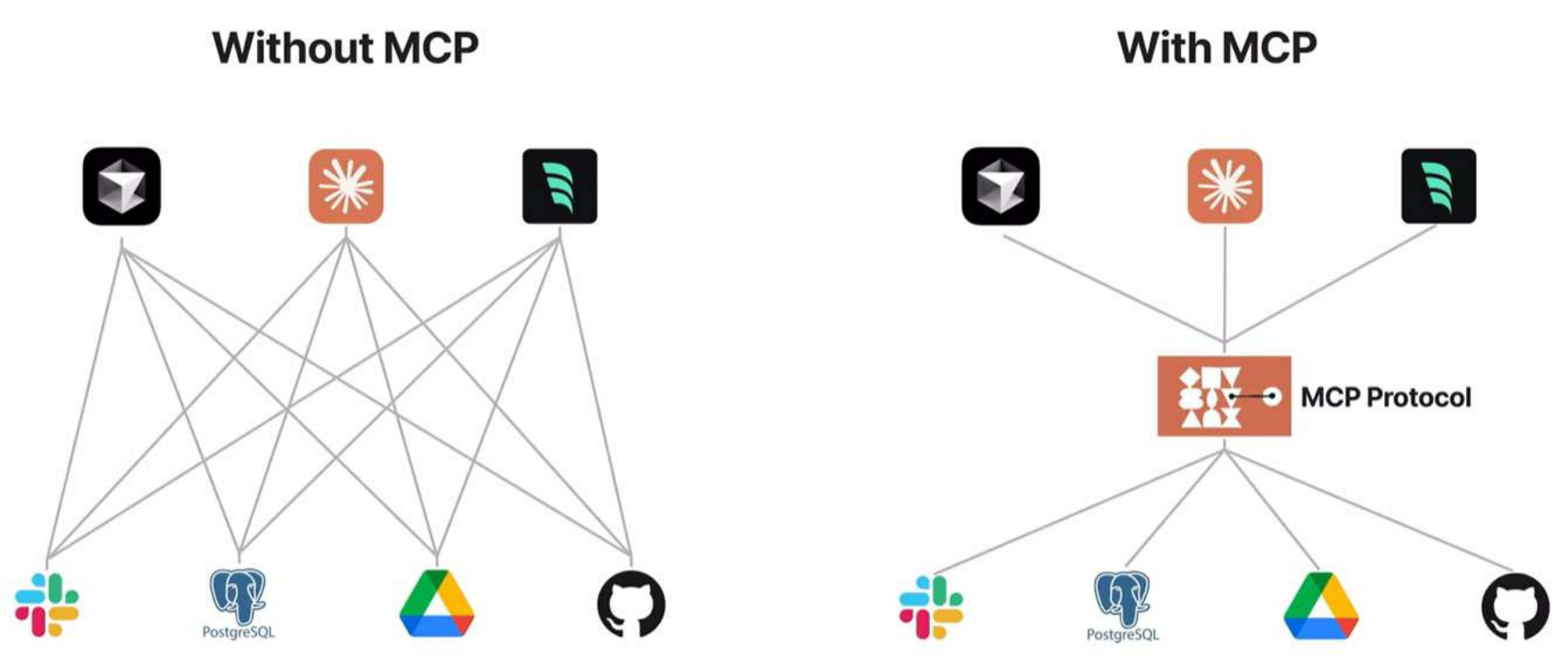
用户询问“查询上季度营业额”，MCP 自动组合调用 CRM 系统 API（获取客户数据）+ 财务系统 API（调取报表）+ 邮件 API（发送总结报告）。

2. MCP是什么?

2.1 MCP的理解

MCP（Model Context Protocol，模型上下文协议），2024年11月底，由Anthropic 推出的一种开放标准。旨在为大语言模型（LLM）提供统一的、标准化方式与外部数据源和工具之间进行通信。

2.1 MCP的理解

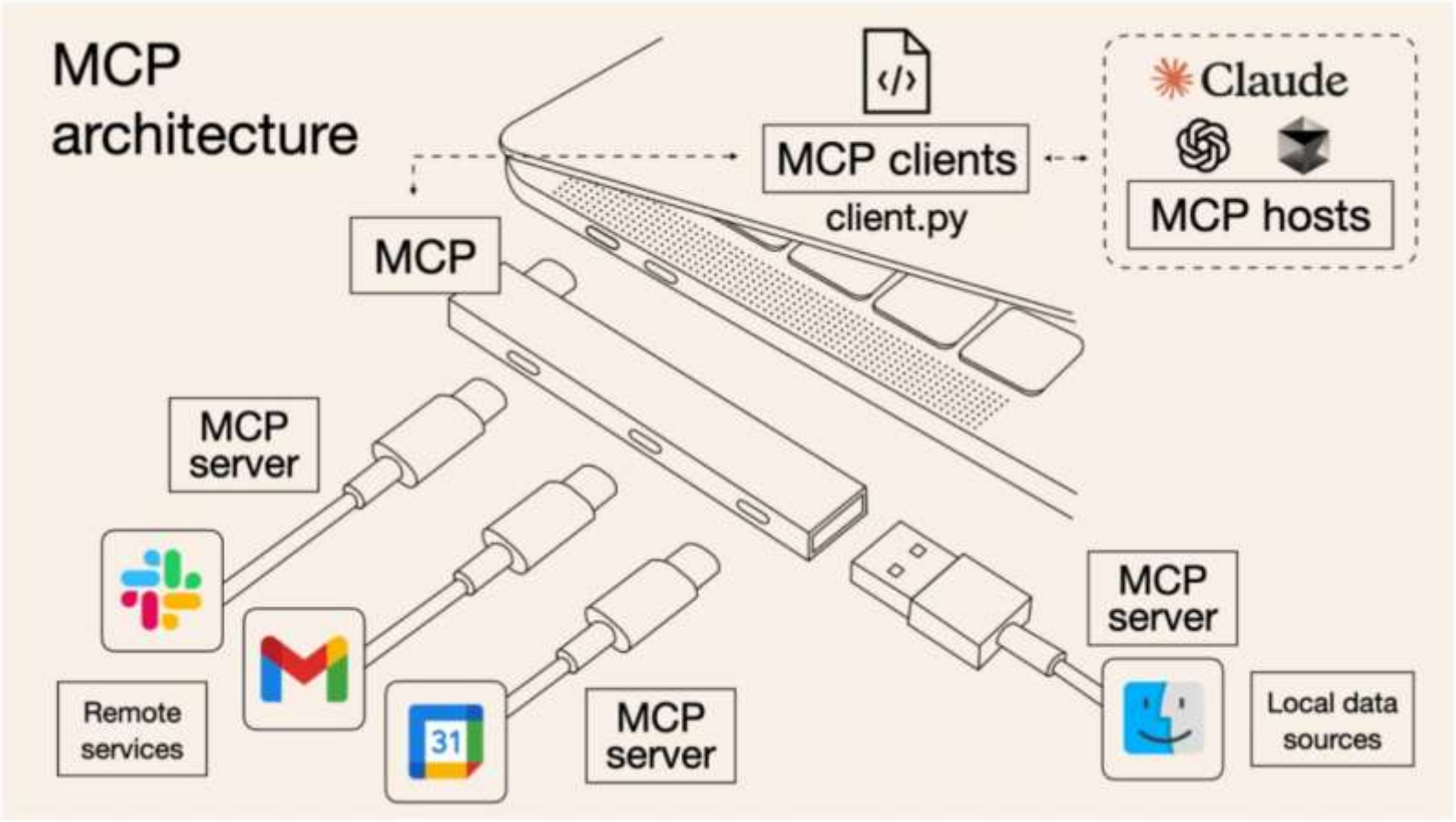


传统AI集成的问题：这种为每个数据源构建独立连接的方式，可以被视为一个M*N问题。

问题：架构碎片化，难以扩展，限制了AI获取必要上下文信息的能力

MCP解决方案：提供统一且可靠的方式来访问所需数据，克服了以往集成方法的局限性。

2.1 MCP的理解



MCP 作为一种标准化协议，极大地简化了大语言模型与外部世界的交互方式，使开发者能够以统一的方式为 AI 应用添加各种能力。

2.1 MCP的理解

官方文档：https://modelcontextprotocol.io/introduction

特征	MCP	TCP/IP、HTTPS
本质	协议（Protocol）	协议（Protocol）
作用	标准化 AI模型与上下文来源/工具之间的数据交互方式	标准化 设备之间 的网络通信方式
目标	让不同模型应用可以用统一方式访问资源/工具	让不同设备、系统可以互通数据
好处	消除碎片化集成、形成生态闭环	解决设备互联、实现互联网基础

2.2 MCP推广时间线

2024年11月底，Anthropic推出了MCP。

目标就是能在Agent 的开发过程中，让大模型更加便捷地调用外部工具。

2.2 MCP推广时间线

今年2月份，

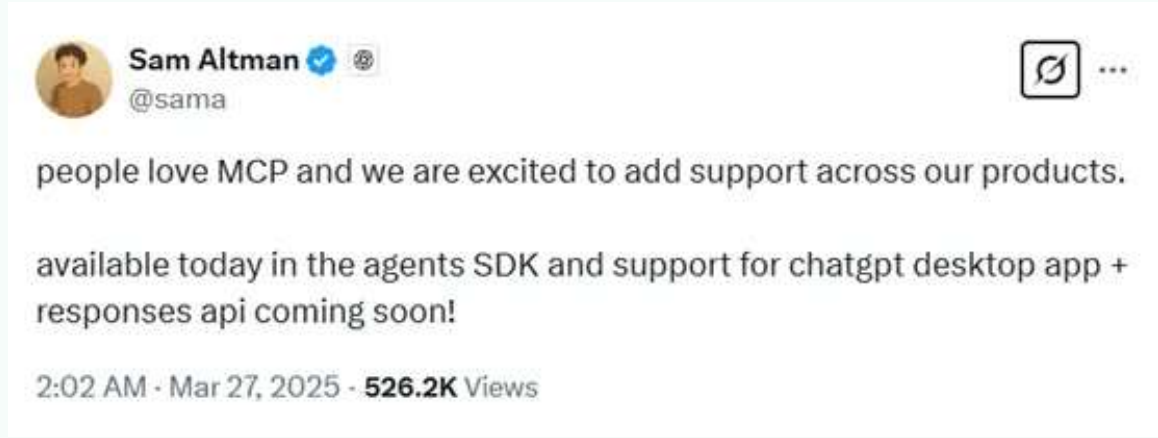
Cursor正式宣布加入MCP功能支持，

一举将MCP推到了全体开发人员面前！

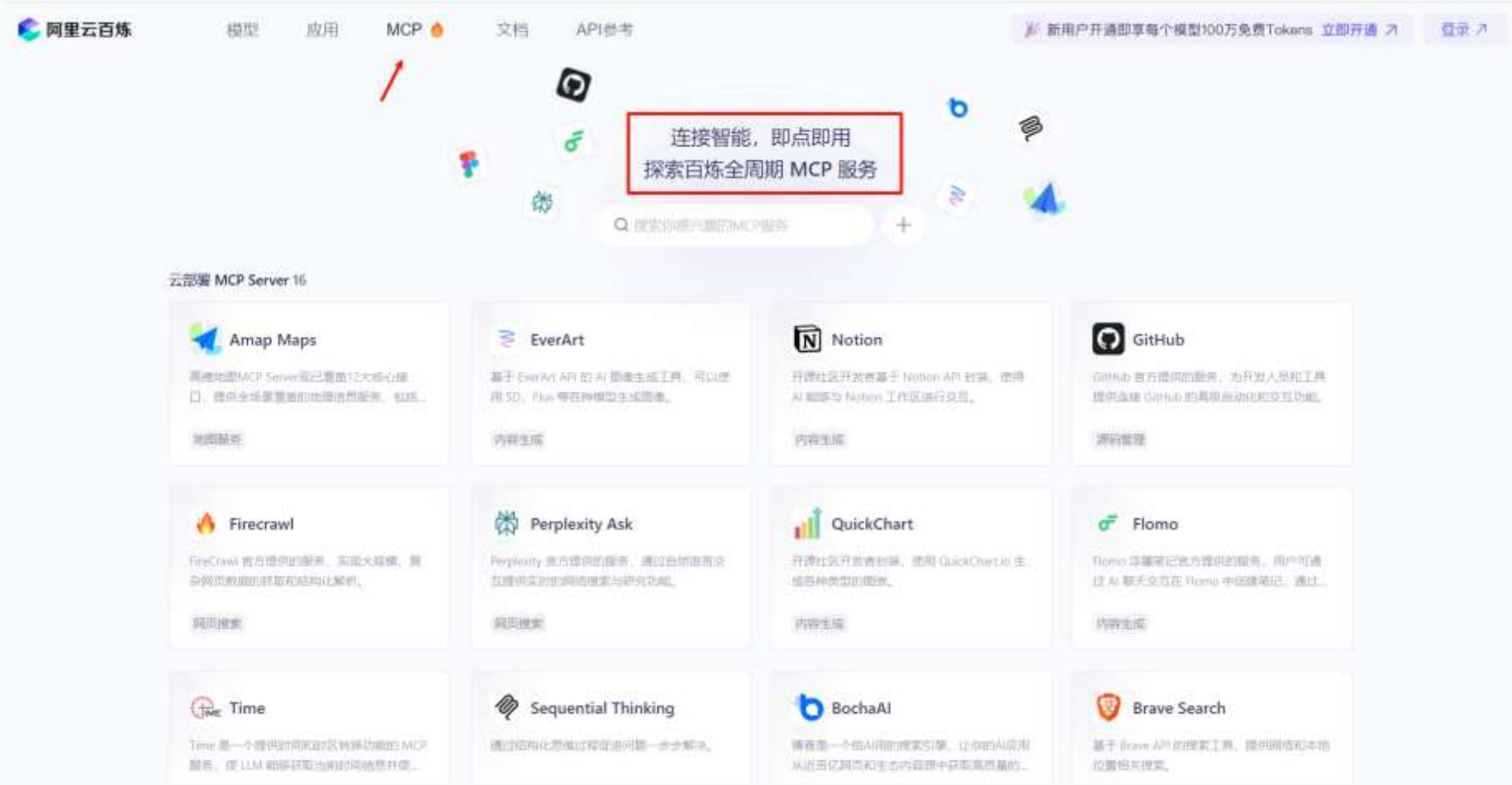
2.2 MCP推广时间线

2025年3月27日，OpenAI智能体支持MCP。

OpenAI联合创始人兼首席执行官Sam Altman也特意发文大赞MCP，可见其对Agent的重要性。

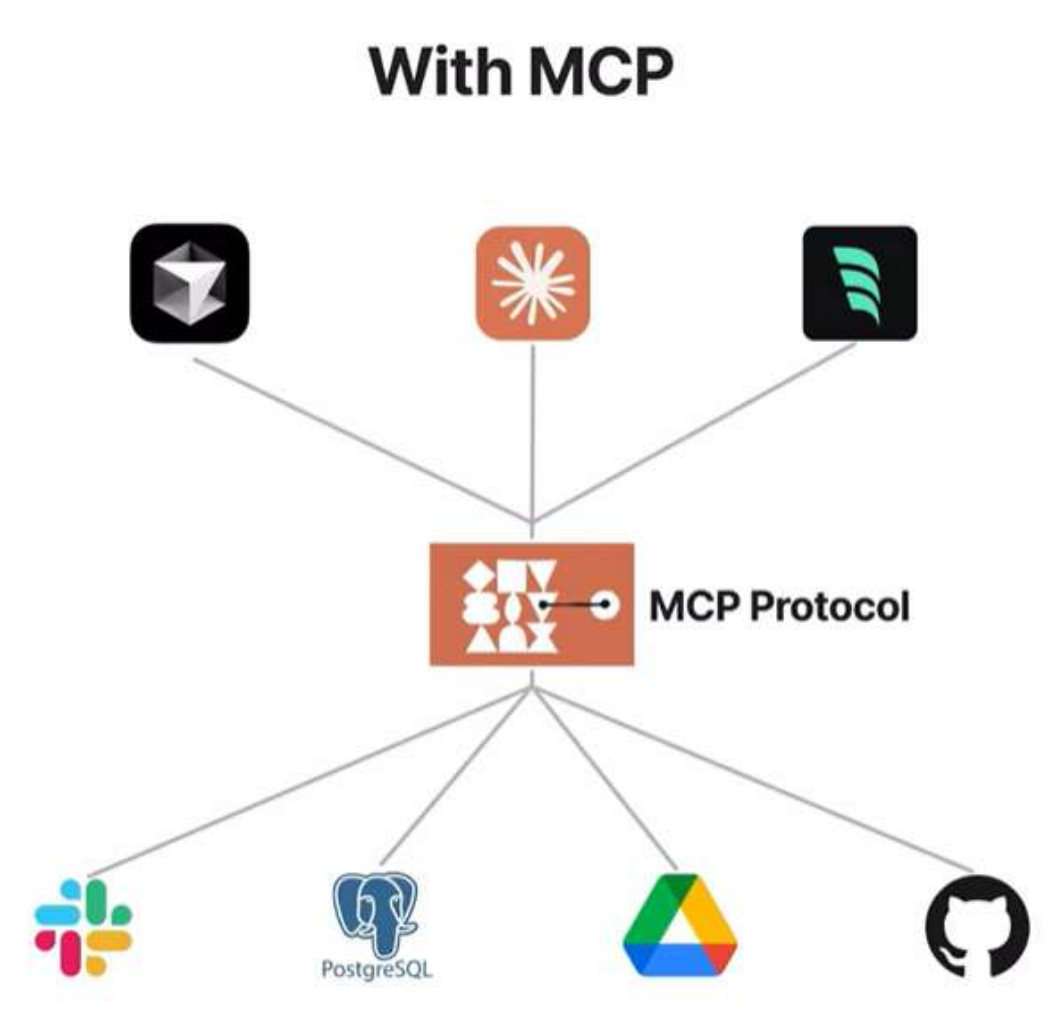


2.2 MCP推广时间线



https://bailian.console.aliyun.com/?tab=mcp#/mcp-market

2.3 哪些平台支持MCP查询



2.3 哪些平台支持MCP查询

github查看：

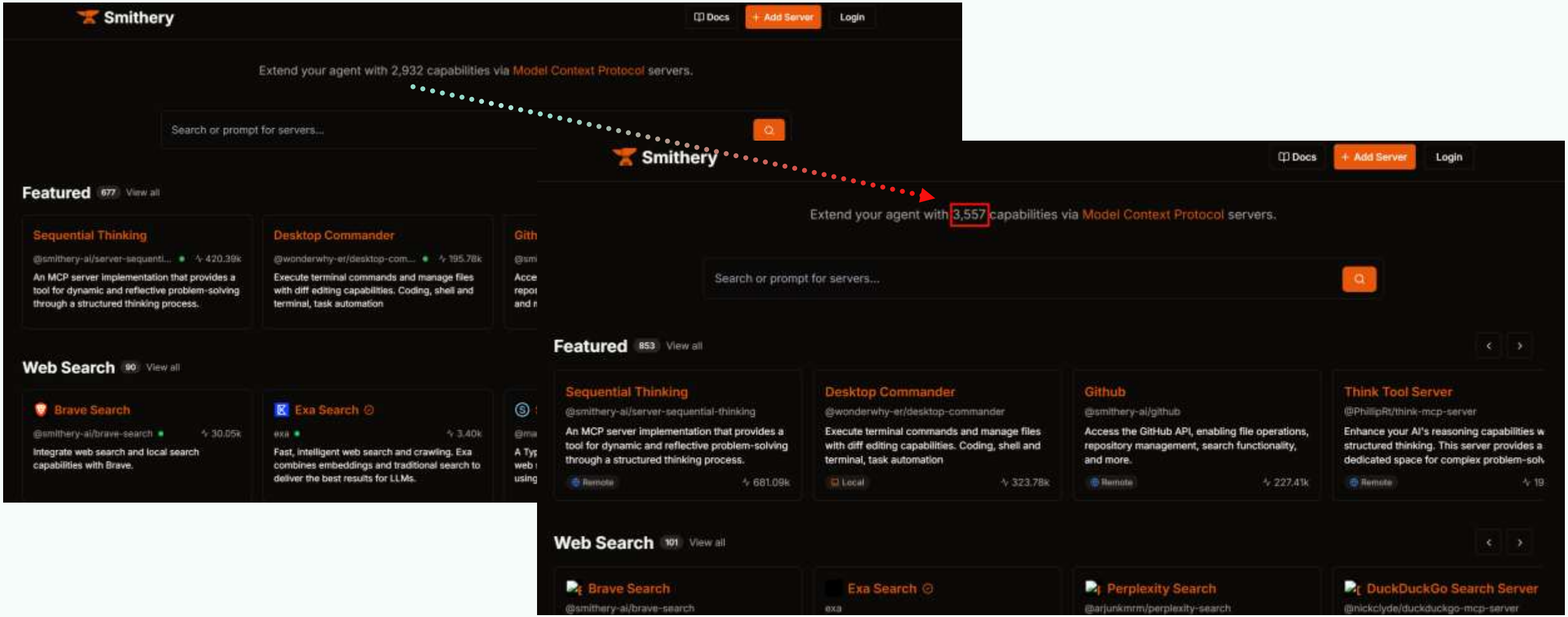
- MCP官方资源：https://github.com/modelcontextprotocol/servers
- MCP热门资源：https://github.com/punkpeye/awesome-mcp-servers

其它平台：

- Glama：https://glama.ai/mcp/servers
- Smithery：https://smithery.ai
- cursor：https://cursor.directory
- MCP.so：https://mcp.so/zh
- 阿里云百炼：https://bailian.console.aliyun.com/?tab=mcp#/mcp-market

2.3 哪些平台支持MCP查询

在Smithery平台上你可以轻松查找不同功能对应的工具及服务。



2.3 哪些平台支持MCP查询

这里有两点要说的：

第1，随着越来越多的Server接入MCP协议，未来AI能够直接调用的工具将呈现指数级增长，这能从根源上打开Agent能力的天花板。未来 AI 生态系统将变得更加开放和强大。

第2，目前社区的 MCP Server 还是比较混乱，有很多缺少教程和文档，很多的代码功能也有问题，大家只能凭经验和参考官方文档了。

3. 程序员如何使用MCP

3.1 MCP应用场景

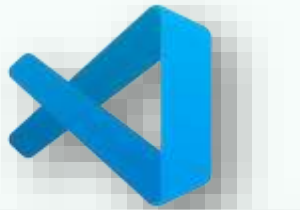
应用领域	典型场景	MCP价值	代表实现
智能编程助手	代码生成、Bug修复、API集成	安全访问本地代码库、CI/CD系统	Cursor、VS Code插件
数据分析工具	自然语言查询数据库、可视化生成	安全查询内部数据库、连接BI工具	XiYanSQL-MCP、数据库MCP服务器
企业知识管理	知识库查询、文档生成、邮件撰写	安全访问内部文档、保护隐私数据	文件系统MCP、Email-MCP
创意设计工具	3D建模、图形生成、UI设计	与专业软件无缝集成	Blender MCP、浏览器自动化
工作流自动化	多系统协调、事件驱动流程	跨系统安全协作	Cloudflare MCP、AWS自动化套件

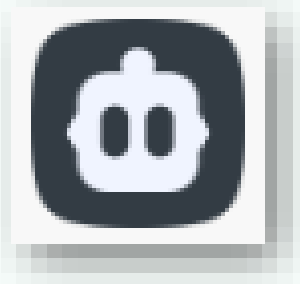
3.1 MCP应用场景

应用领域	典型场景	MCP价值	代表实现
智能编程助手	代码生成、Bug修复、API集成	安全访问本地代码库、CI/CD系统	Cursor、VS Code插件
数据分析工具	自然语言查询数据库、可视化生成	安全查询内部数据库、连接BI工具	XiYanSQL-MCP、数据库MCP服务器
企业知识管理	知识库查询、文档生成、邮件撰写	安全访问内部文档、保护隐私数据	文件系统MCP、Email-MCP
创意设计工具	3D建模、图形生成、UI设计	与专业软件无缝集成	Blender MCP、浏览器自动化
工作流自动化	多系统协调、事件驱动流程	跨系统安全协作	Cloudflare MCP、AWS自动化套件

3.1 MCP应用场景

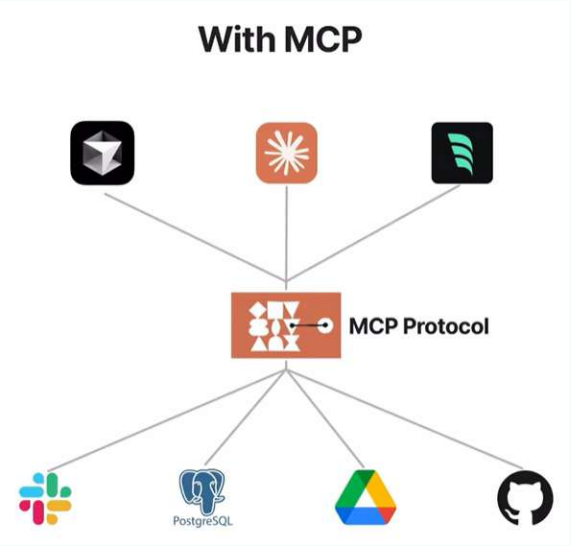
应用领域	典型场景	MCP价值	代表实现
智能编程助手	代码生成、Bug修复、API集成	安全访问本地代码库、CI/CD系统	Cursor、VS Code插件
数据分析工具	自然语言查询数据库、可视化生成	安全查询内部数据库、连接BI工具	XiYanSQL-MCP、数据库MCP服务器
企业知识管理	知识库查询、文档生成、邮件撰写	安全访问内部文档、保护隐私数据	文件系统MCP、Email-MCP
创意设计工具	3D建模、图形生成、UI设计	与专业软件无缝集成	Blender MCP、浏览器自动化
工作流自动化	多系统协调、事件驱动流程	跨系统安全协作	Cloudflare MCP、AWS自动化套件





3.2 使用前的准备工作

① MCP的通信机制



根据 MCP 的规范，当前支持两种通信机制（传输方式）：

- **stdio(标准输入输出)**：主要用在本地服务上，操作你本地的软件或者本地的文件。比如 Blender 这种就只能用 Stdio 因为他没有在线服务。 MCP默认通信方式

- **SSE(Server-Sent Events)**：主要用在远程通信服务上，这个服务本身就有在线的 API，比如访问你的谷歌邮件，天气情况等。

3.2 使用前的准备工作

① MCP的通信机制：stdio方式

优点

- 这种方式适用于客户端和服务器在同一台机器上运行的场景，简单。
- stdio模式无需外部网络依赖，通信速度快，适合快速响应的本地应用。
- 可靠性高，且易于调试

缺点

- Stdio 的配置比较复杂，我们需要做些准备工作，你需要提前安装需要的命令行工具。
- stdio模式为单进程通信，无法并行处理多个客户端请求，同时由于进程资源开销较大，不适合在本地运行大量服务。（限制了其在更复杂分布式场景中的使用）

3.2 使用前的准备工作

① MCP的通信机制：SSE方式

场景

- SSE方式适用于客户端和服务器位于不同物理位置的场景。
- 适用于实时数据更新、消息推送、轻量级监控和实时日志流等场景
- 对于分布式或远程部署的场景，基于 HTTP 和 SSE 的传输方式则更为合适。

优点

- 配置方式非常简单，基本上就一个链接就行，直接复制他的链接填上就行

3.2 使用前的准备工作

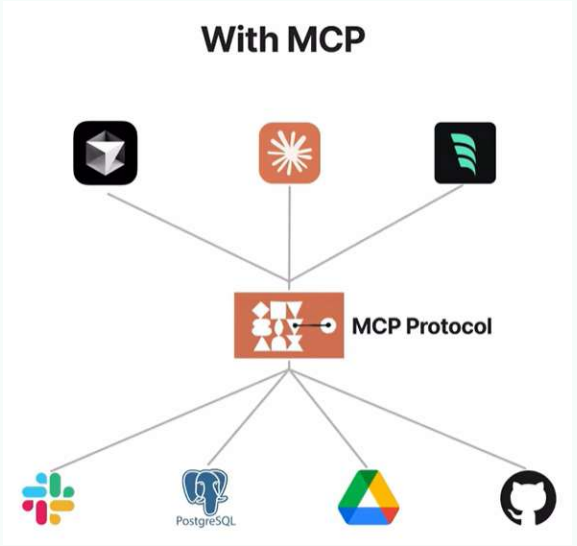
② stdio的本地环境安装

stdio的本地环境有两种：

一种是Python 编写的服务，

一种用TypeScript 编写的服务。

分别对应着uvx 和 npx 两种指令。



3.2 使用前的准备工作

② stdio的本地环境安装： uvx

两种安装方式：

第1种：若已配置Python环境，可使用以下命令安装：

```
pip install uv
```

3.2 使用前的准备工作

② stdio的本地环境安装： uvx

两种安装方式：

第2种：在Windows下可以通过PowerShell运行命令来安装uv。

powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex”

```
PS C:\WINDOWS\system32> powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
Downloading uv 0.6.14 (x86_64-pc-windows-msvc)
Installing to C:\Users\shkst\.local\bin
  uv.exe
  uvx.exe
everything's installed!

To add C:\Users\shkst\.local\bin to your PATH, either restart your shell or run:

    set Path=C:\Users\shkst\.local\bin;%Path%   (cmd)
    $env:Path = "C:\Users\shkst\.local\bin;$env:Path"   (powershell)
```

3.2 使用前的准备工作

② stdio的本地环境安装： uvx

验证：重启终端并运行以下命令检查是否正常：

```
uv --version
uvx --help
```

3.2 使用前的准备工作

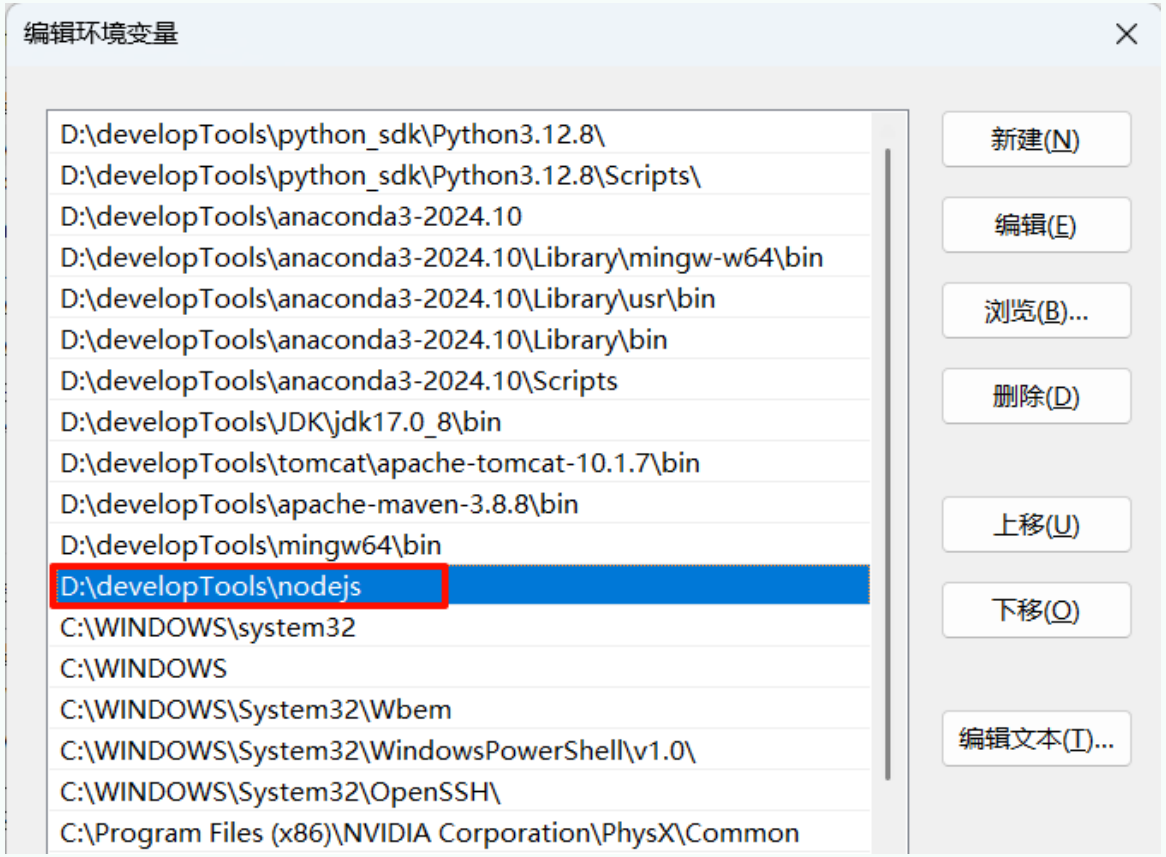
② stdio的本地环境安装： npx

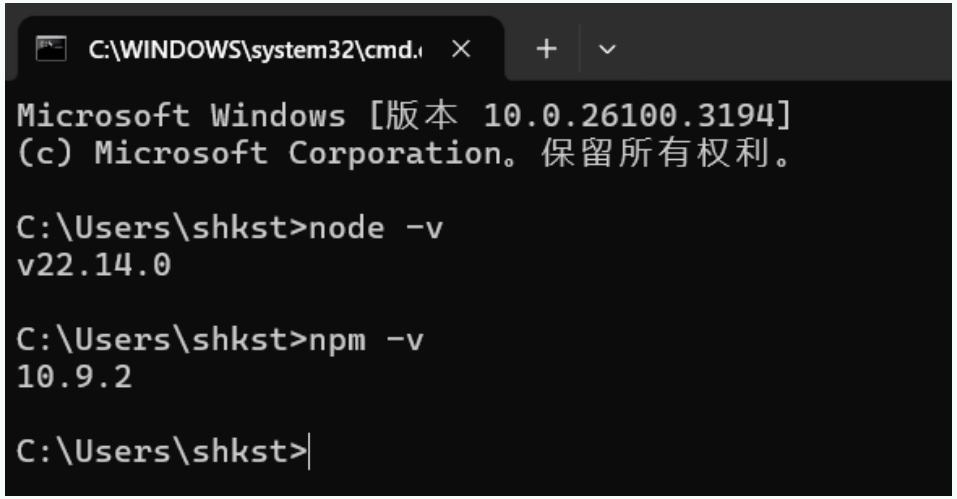
Node.js下载的官网：https://nodejs.org/zh-cn

3.2 使用前的准备工作

② stdio的本地环境安装： npx

配置环境变量，并测试





3.3 Cursor中使用MCP

3.3 Cursor中使用MCP

cursor中国区官网：https://www.cursor.com/cn

具体操作见《Cursor中使用mcp.md》

3.4 Cline中使用MCP

具体操作见《Cline中使用mcp.md》

4. MCP的工作原理

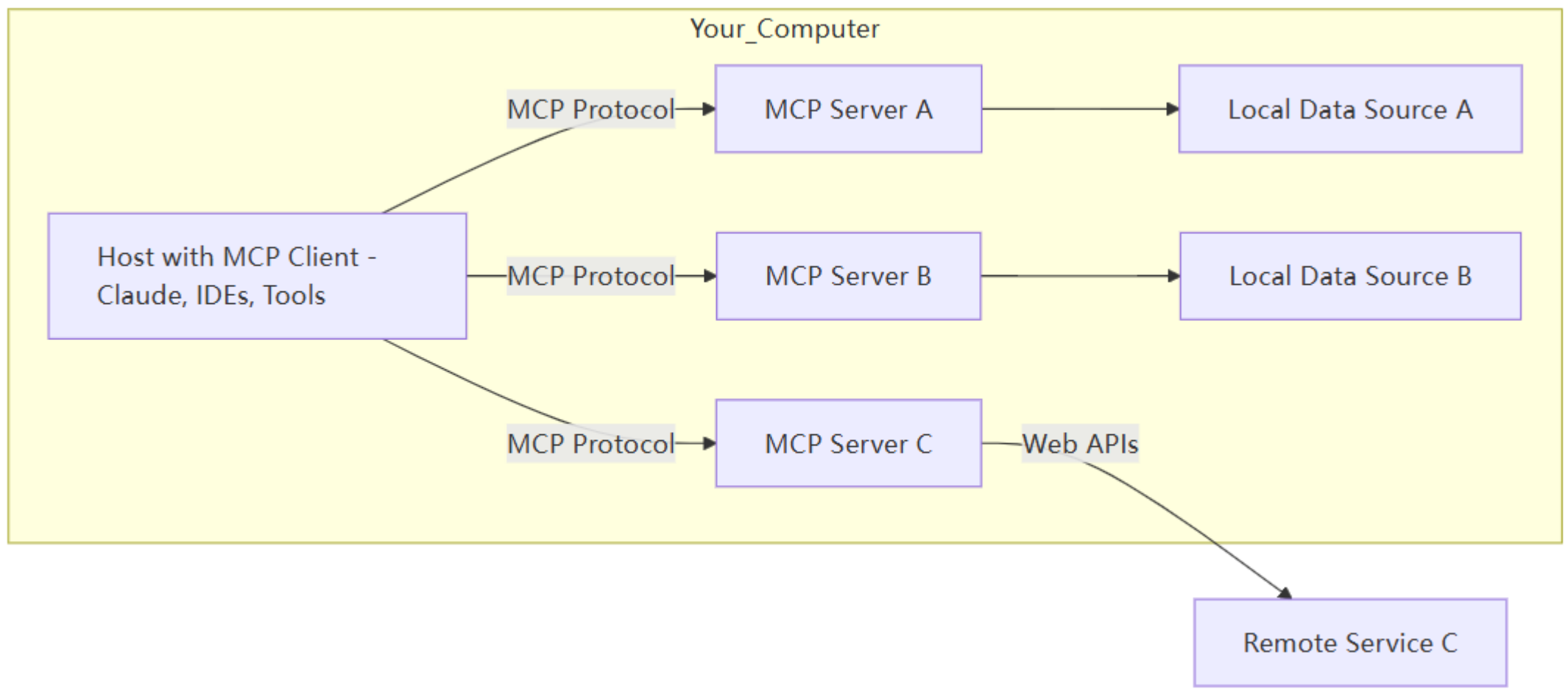
4.1 MCP的C/S架构

5个核心概念

MCP 遵循客户端-服务器架构（client-server），其中包含以下几个核心概念：

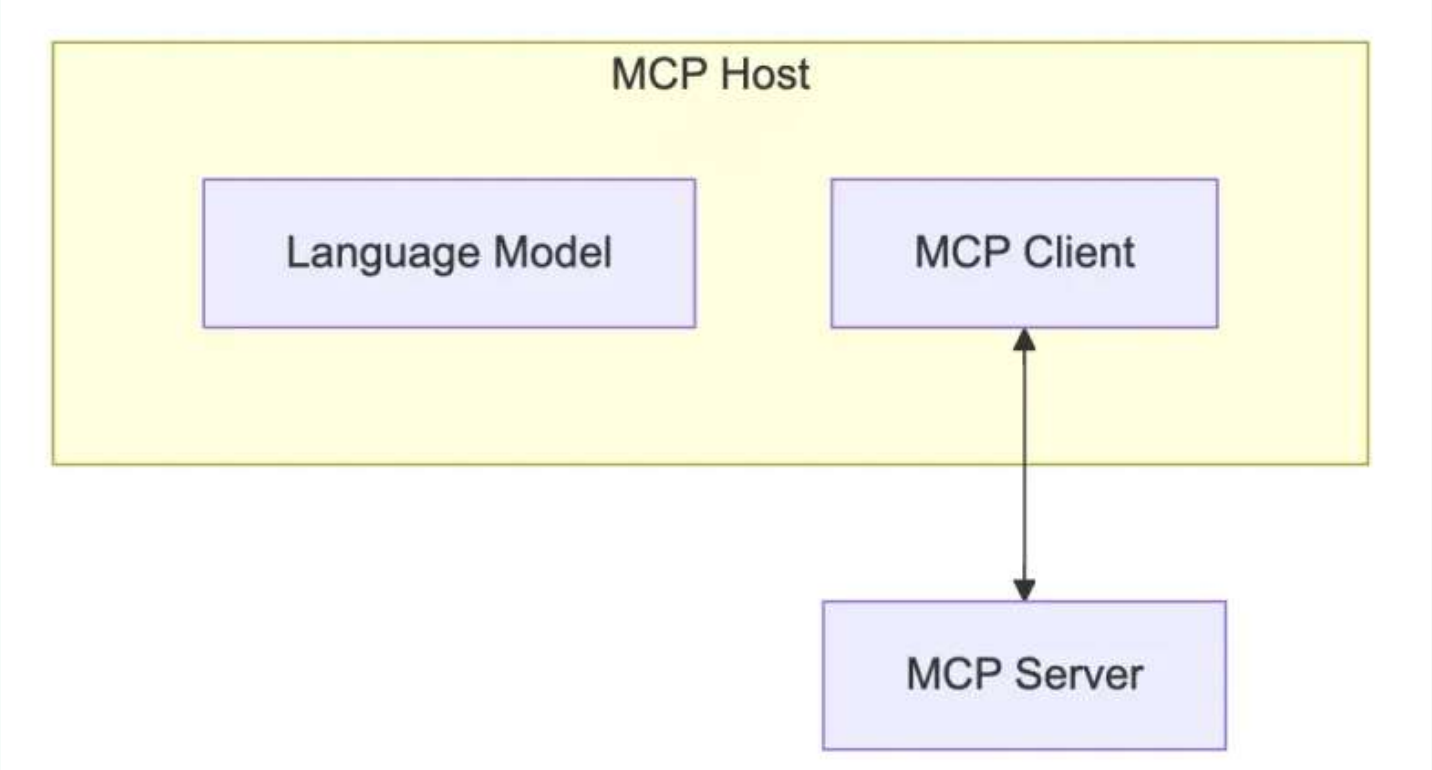
- **MCP 主机(MCP Hosts)**
- **MCP 客户端(MCP Clients)**
- **MCP 服务器(MCP Servers)**
- **本地资源(Local Resources)**
- **远程资源(Remote Resources)**

4.1 MCP的C/S架构


Your_Computer
Host with MCP Client -
Claude, IDEs, Tools
MCP Protocol
MCP Server A
Local Data Source A
MCP Protocol
MCP Server B
Local Data Source B
MCP Protocol
MCP Server C
Web APIs
Remote Service C

① MCP Host

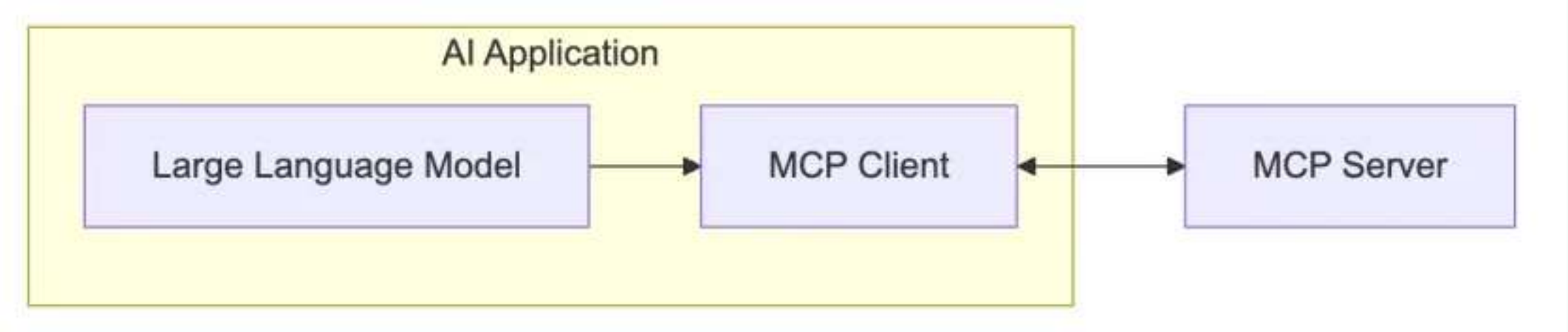
作为运行 MCP 的主应用程序，例如 Claude Desktop、 Cursor、 Cline 或 AI 工具。
为用户提供与LLM交互的接口，同时集成 MCP Client 以连接 MCP Server。



② MCP Client

MCP client 充当 LLM 和 MCP server 之间的桥梁，嵌入在主机程序中，主要负责：

- 接收来自LLM的请求；
- 将请求转发到相应的 MCP server
- 将 MCP server 的结果返回给 LLM



② MCP Client

有哪些常用的Clients

MCP 官网(https://modelcontextprotocol.io/clients) 列出来一些支持 MCP 的 Clients。

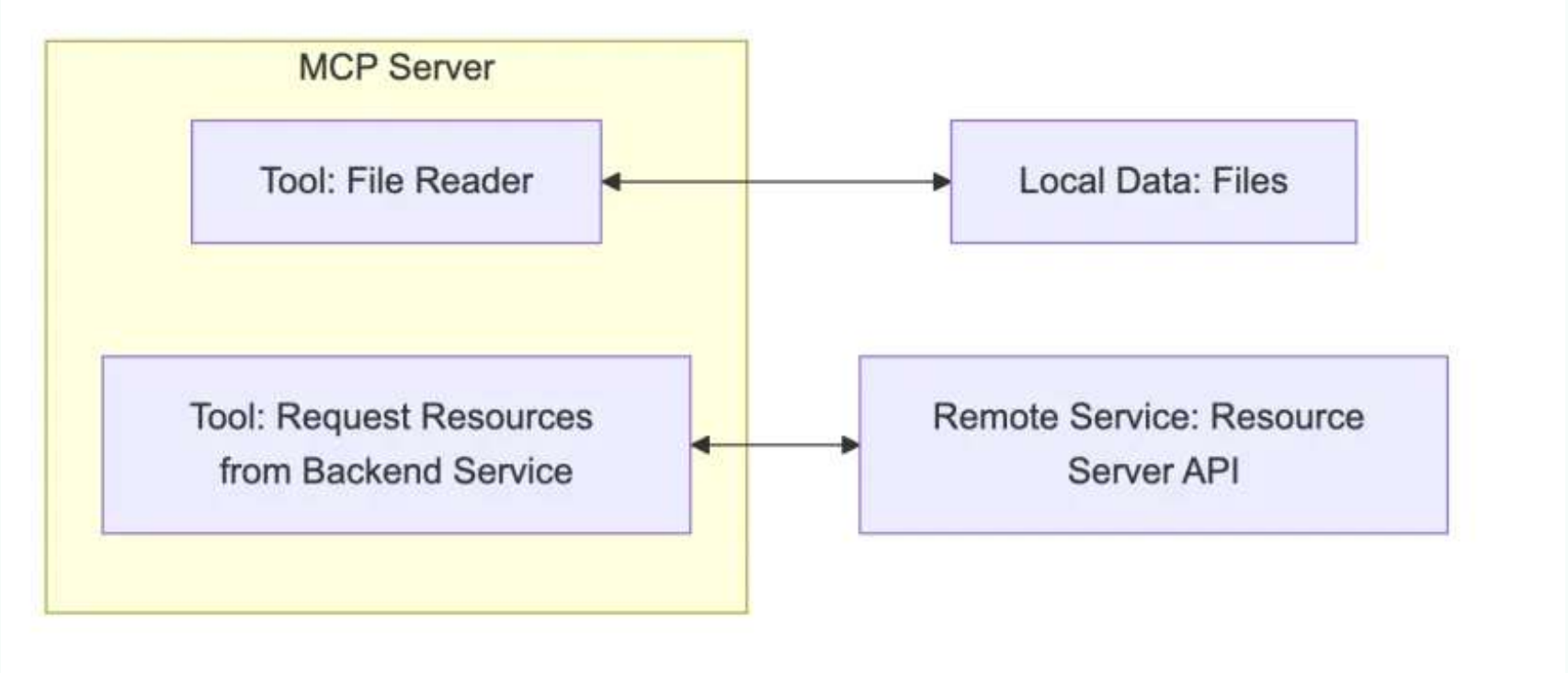
分为两类：

- AI编程IDE：Cursor、Cline、Continue、Sourcegraph、Windsurf 等
- 聊天客户端：Cherry Studio、Claude、Librechat、Chatwise等

更多的Client参考这里：
MCP Clients：https://www.pulsemcp.com/clients
Awesome MCP Clients：https://github.com/punkpeye/awesome-mcp-clients/

② MCP Server

每个 MCP 服务器都提供了一组特定的工具，负责从本地数据或远程服务中检索信息。是 MCP 架构中的关键组件。



② MCP Server

与传统的远程 API 服务器不同，MCP 服务器既可以作为本地应用程序在用户设备上运行，也可部署至远程服务器。

比如你让助手：

- “帮我查航班信息” → 它调用航班查询 API
- “算一下 37% 折扣后多少钱” → 它运行计算器函数

作用：让 LLM 不仅能“说”，还能“做”（执行代码、查询数据等）。

② MCP Server

stdio的方式

与传统的远程 API 服务器不同，MCP 服务器既可以作为本地应用程序在用户设备上运行，也可部署至远程服务器。

SSE的方式

比如你让助手：

- “帮我查航班信息” → 它调用航班查询 API
- “算一下 37% 折扣后多少钱” → 它运行计算器函数

作用：让 LLM 不仅能“说”，还能“做”（执行代码、查询数据等）。

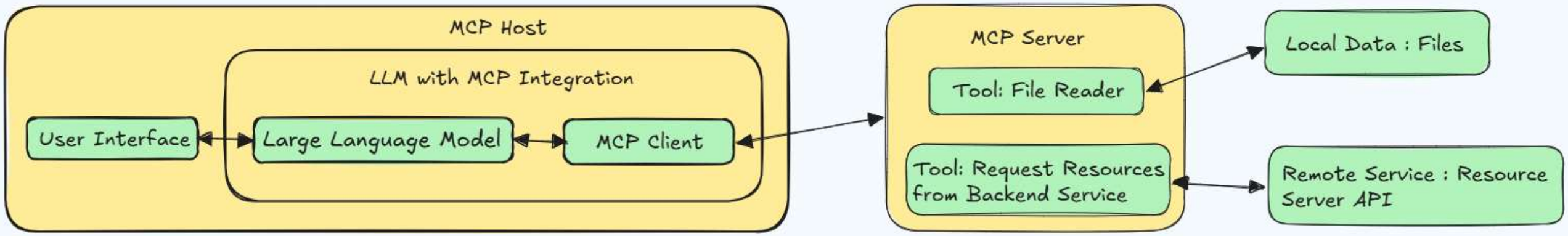
② MCP Server

MCP Server 的本质

本质是运行在电脑上的一个nodejs或python程序。可以理解为客户端用命令行调用了电脑上的nodejs或python程序。

- 使用 TypeScript 编写的 MCP server 可以通过 npx 命令来运行
- 使用 Python 编写的 MCP server 可以通过 uvx 命令来运行。

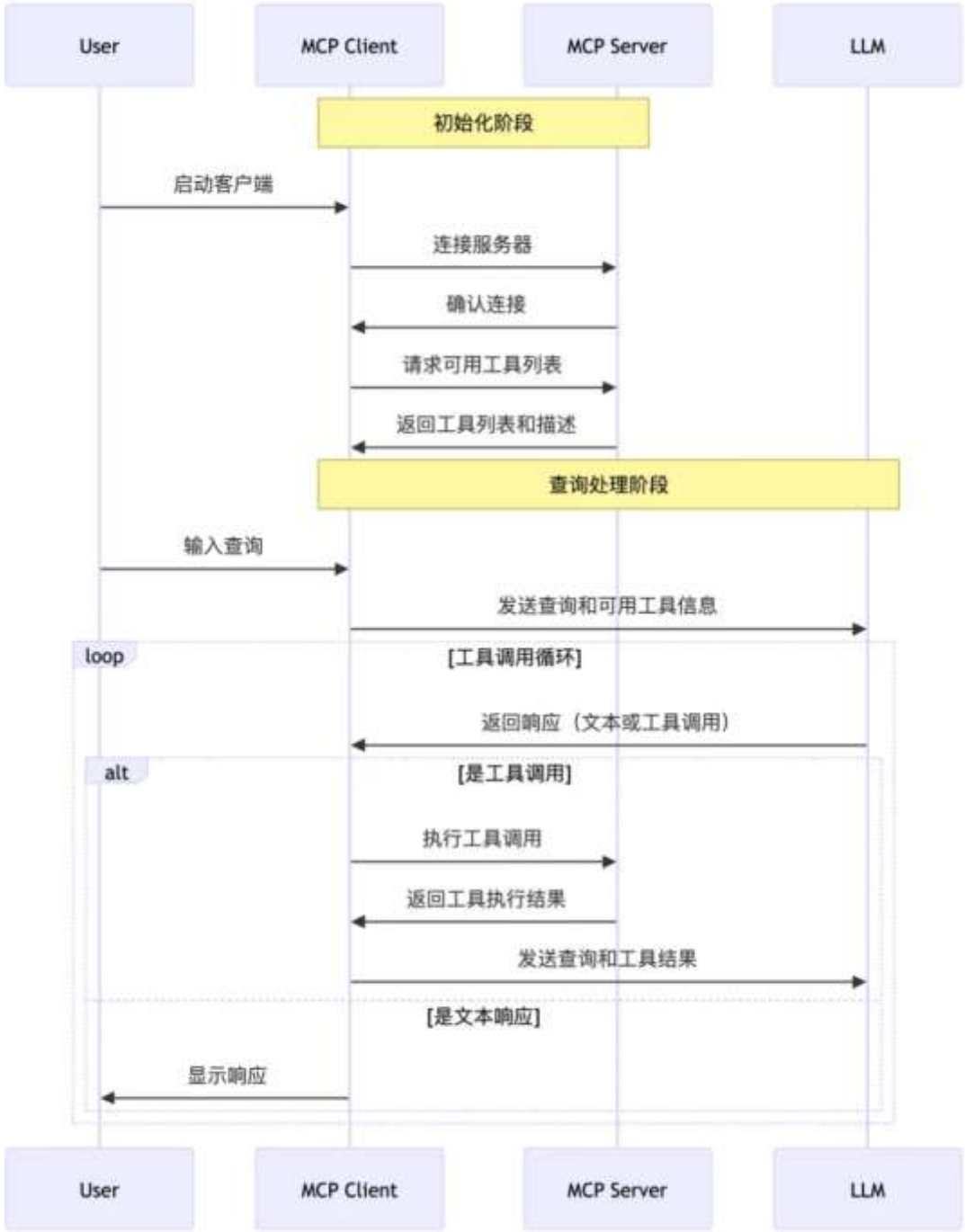
4.1 MCP的C/S架构


MCP Host
LLM with MCP Integration
User Interface
Large Language Model
MCP Client
MCP Server
Tool: File Reader
Tool: Request Resources from Backend Service
Local Data : Files
Remote Service : Resource Server API

4.2 MCP工作流程

API 主要有两个

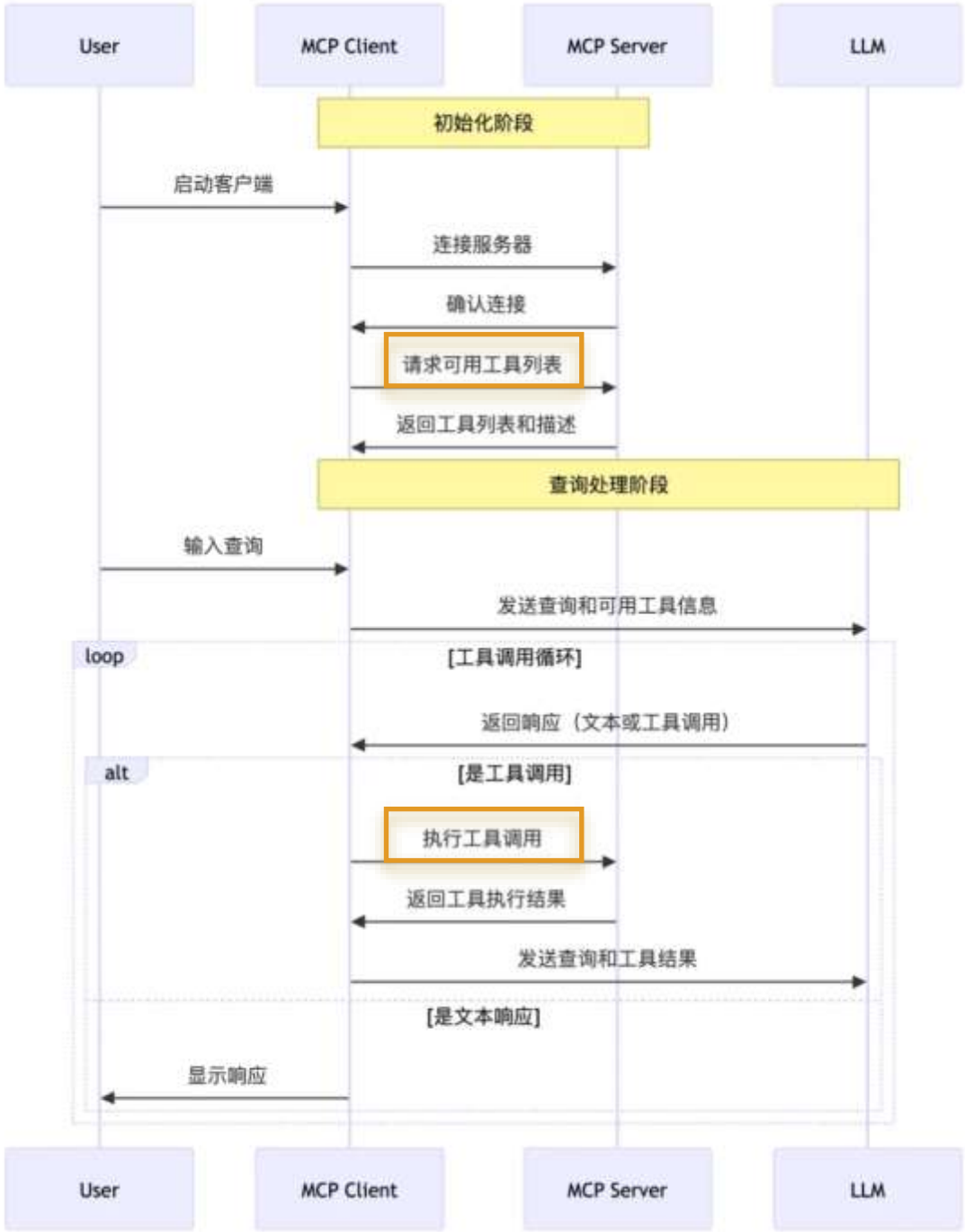
- **tools/list**：列出 Server 支持的所有工具
- **tools/call**：Client 请求 Server 去执行某个工具，并将结果返回



4.2 MCP工作流程

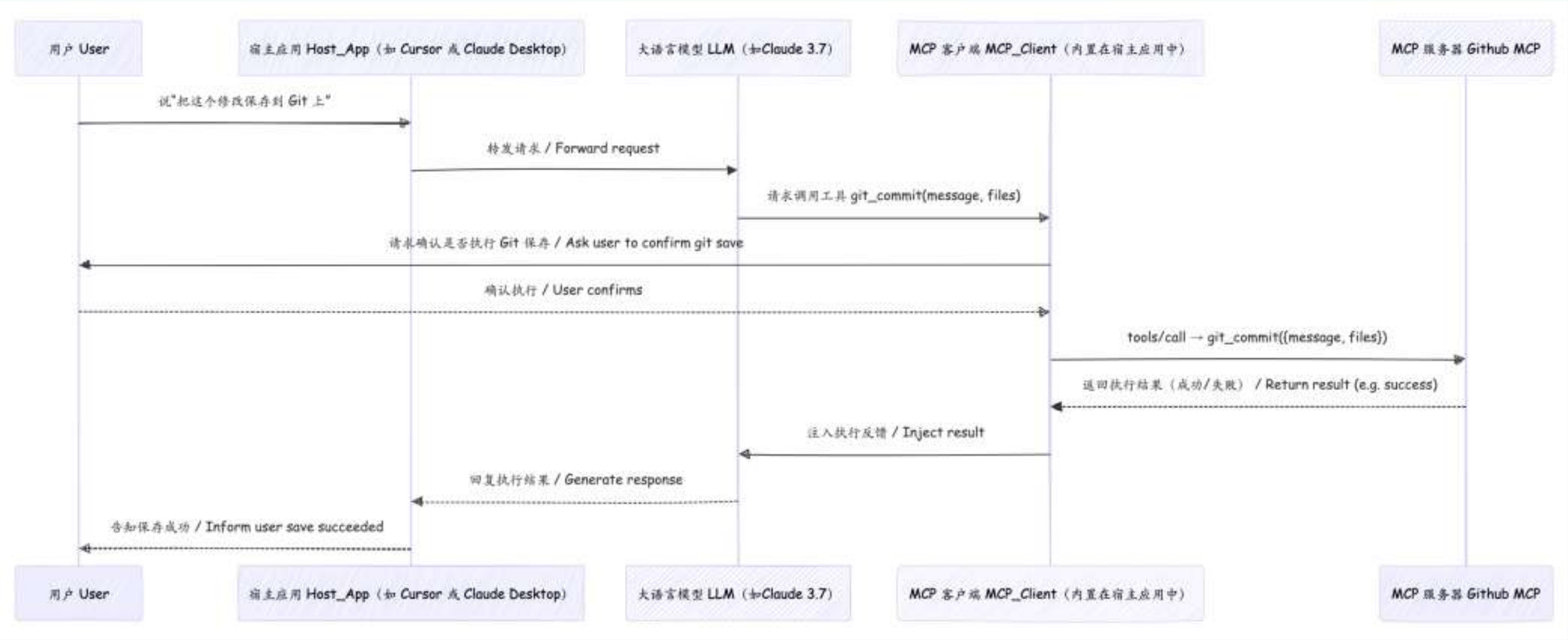
API 主要有两个

- **tools/list**：列出 Server 支持的所有工具
- **tools/call**：Client 请求 Server 去执行某个工具，并将结果返回



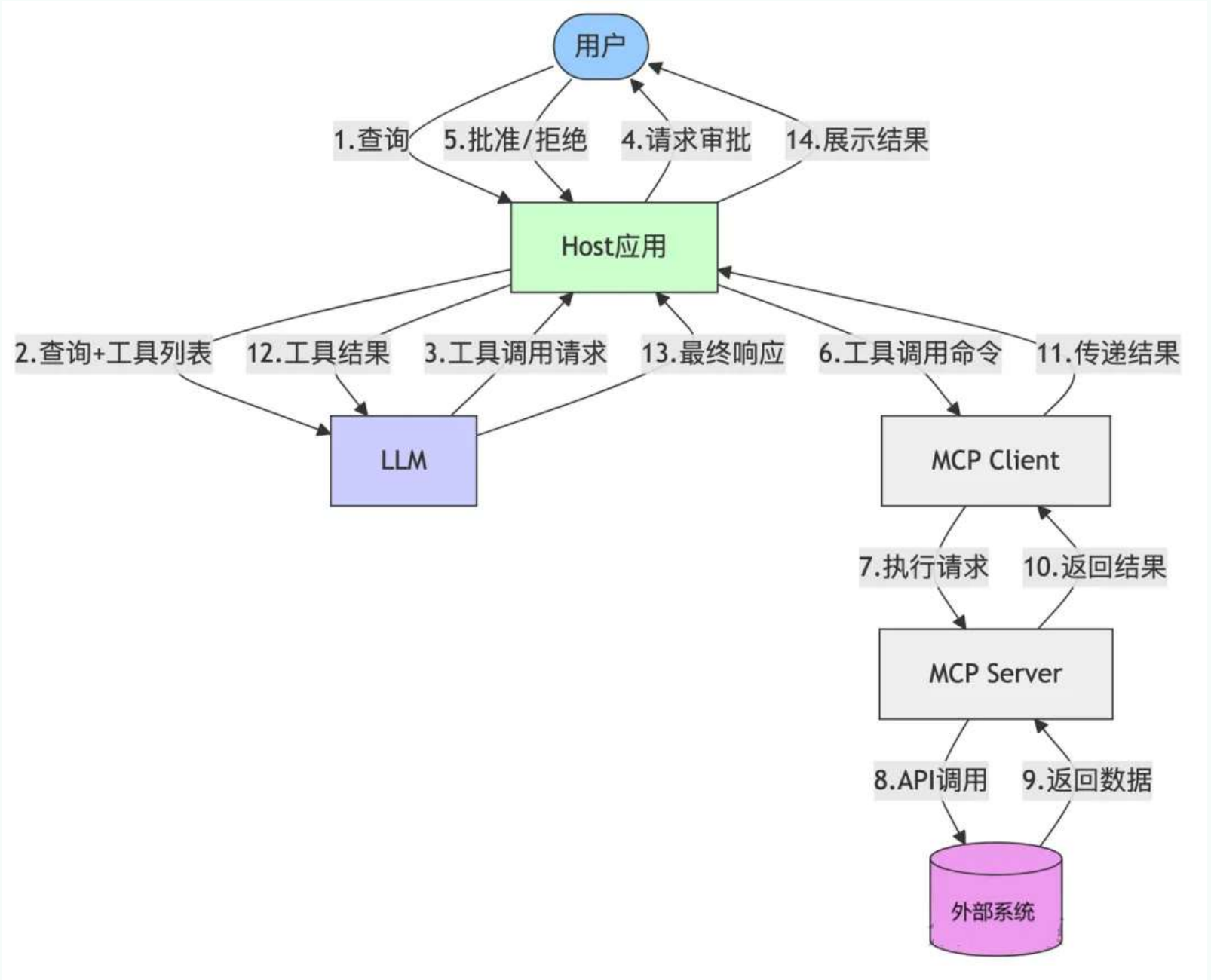
4.2 MCP工作流程

举例：



4.2 MCP工作流程

数据流向图



4.3 回顾：Cursor中使用MCP

在MCP的概念中，Cursor属于一个MCP的宿主应用（Host-app），而Cursor之所以能使用MCP服务，是因为它内置安装了MCP Client。

我们目前在配置Cursor中的MCP时，本质是在配置MCP Server，这些Server是由不同的开发者提供的，他们基于标准化的MCP协议，做了个小的服务，这些服务可能在本地也可能在云端，而我们实际上也完全可以按自己的需要去制作MCP Server。

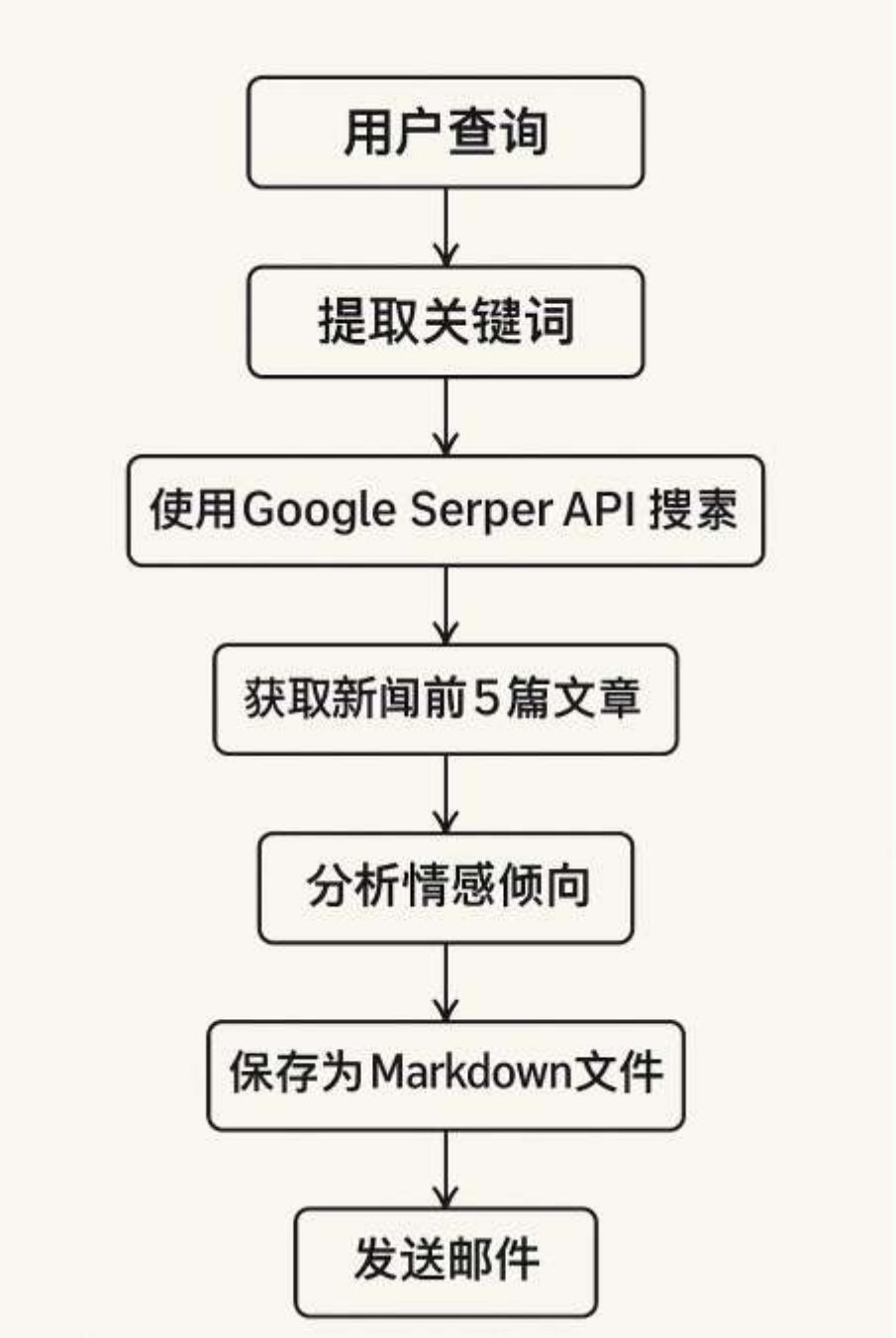
5. 手动开发MCP项目(C/S)

手动开发MCP项目

案例需求

本项目旨在构建一个本地智能舆情分析系统，通过自然语言处理与多工具协作，实现用户查询意图的自动理解、新闻检索、情绪分析、结构化输出与邮件推送。

具体参考《手动开发MCP项目(CS架构）.md》



6. 大众用户如何使用MCP

6.1 Cherry Studio的MCP说明

Cherry Studio 是一款集多模型对话、知识库管理、AI 绘画、翻译等功能于一体的全能 AI 助手平台。支持 Windows，Linux 和 Mac。

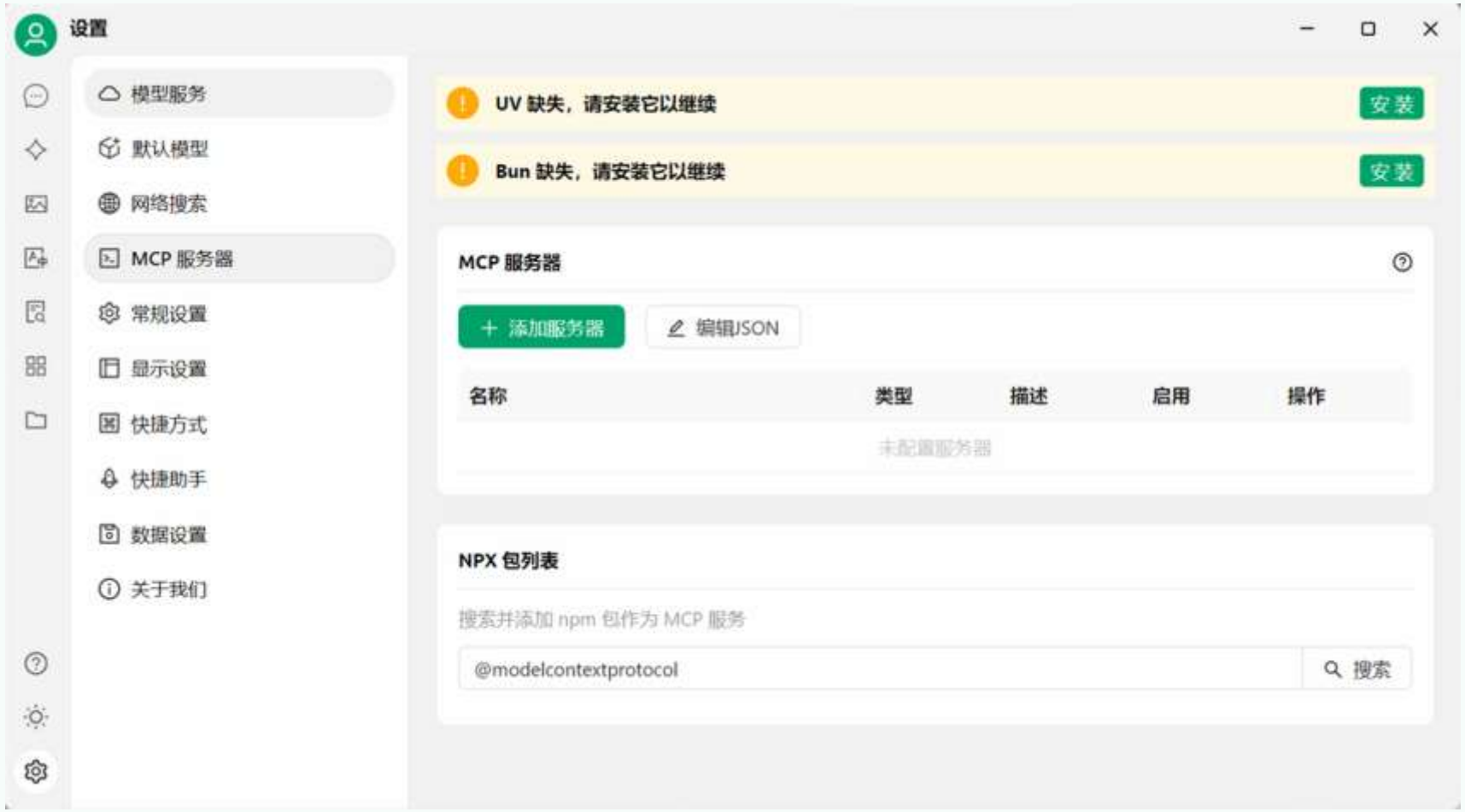
同时，CherryStudio提供了一个简洁便于操作的可视化页面，通过简单的配置即可开启MCP服务。非常适合大众用户用于构建“低代码智能流程”。

Cherry Studio的下载地址：https://cherry-ai.com/

6.2 使用案例

具体详情，见《Cherry Studio中使用MCP案例.md》

6.3 准备工作：安装uv、 bun



Cherry Studio 目前只使用内置的 uv(https://github.com/oven-sh/bun/releases) 和 bun(https://github.com/astral-sh/uv/releases)，不会复用系统中已经安装的 uv 和 bun。

7. 热门MCP Servers推荐

7. 热门MCP Servers推荐

推荐1：文件系统 filesystem

Filesystem MCP 旨在为大型语言模型(LLM)和 AI 助手提供对本地文件系统的安全、受控访问。

主要功能：

- 文件读写：允许读取和写入文件内容，支持创建新文件或覆盖现有文件。

- 目录管理：支持创建、列出和删除目录，以及移动文件或目录。

- 文件搜索：能够在指定路径中搜索匹配特定模式的文件或目录。

- 元数据获取：提供获取文件或目录的详细元数据，包括大小、创建时间、修改时间、访问时间、类 型和权限等信息。

7. 热门MCP Servers推荐

推荐2：数据库 mysqldb-mcp-server

一种模型上下文协议 （MCP） 实现，支持与 MySQL 数据库进行安全交互。此服务器组件可促进 AI 应用程序（主机/客户端）与 MySQL 数据库之间的通信，提供安全的 MySQL 数据库操作，通过受控接口使数据库探索和分析更安全、更有条理。

7. 热门MCP Servers推荐

推荐3：高德地图 amap-maps

高德地图是一个支持任何 MCP 协议客户端的服务器，允许用户轻松地利用高德地图 MCP 服务器进行各种基于位置的服务。

高德地图的主要特点

- 支持多种位置服务，包括地理编码、天气和距离测量
- 提供步行、驾车、公交等多种交通方式的 API
- 允许根据关键字或位置详细搜索兴趣点 （POI）

7. 热门MCP Servers推荐

推荐4：网页数据采集 Firecrawl

Firecrawl MCP 工具是一款基于模型上下文协议（MCP）的企业级网页数据采集服务器。能够为大型语言模型（LLM）提供强大的网页抓取能力。

主要功能：

- JavaScript 渲染：能够处理动态网页内容，突破传统抓取工具的局限，获取更全面的数据。
- 批量处理：支持并行处理和队列管理，提高数据抓取效率。
- 智能限速：根据网络状况和任务需求智能调整抓取速度，避免对目标网站造成过大压力。
- 多种输出格式：支持将抓取的内容转换为 Markdown、HTML 等格式，满足不同场景的需求。

说明：去firecrawl官网注册后即可查看自己的api_key

7. 热门MCP Servers推荐

推荐5：Github

GitHub MCP 服务器是一个模型上下文协议 （MCP） 提供与 GitHub API 无缝集成的服务器，从而实现面向开发人员的高级自动化工具和交互功能。

使用案例：

- 自动化 GitHub 工作流程和流程。
- 从 GitHub 存储库中提取和分析数据。
- 构建与 GitHub 生态系统交互的 AI 驱动的工具和应用程序。

说明：去https://github.com/settings/tokens 申请自己的token

7. 热门MCP Servers推荐

推荐6：Git

用于 Git 存储库交互和自动化的模型上下文协议服务器。

直接的Git仓库操作，包括读取、搜索和分析本地仓库

7. 热门MCP Servers推荐

推荐7：记忆图谱 memory

基于知识图谱的长期记忆系统用于维护上下文

使用本地知识图谱的持久内存的基本实现。这使 Claude 可以在聊天中记住有关用户的信息。

7. 热门MCP Servers推荐

推荐8：控制台 desktop-commander

在计算机上无缝执行终端命令和管理流程。使用强大的命令执行和文件作工具简化您的开发任务。

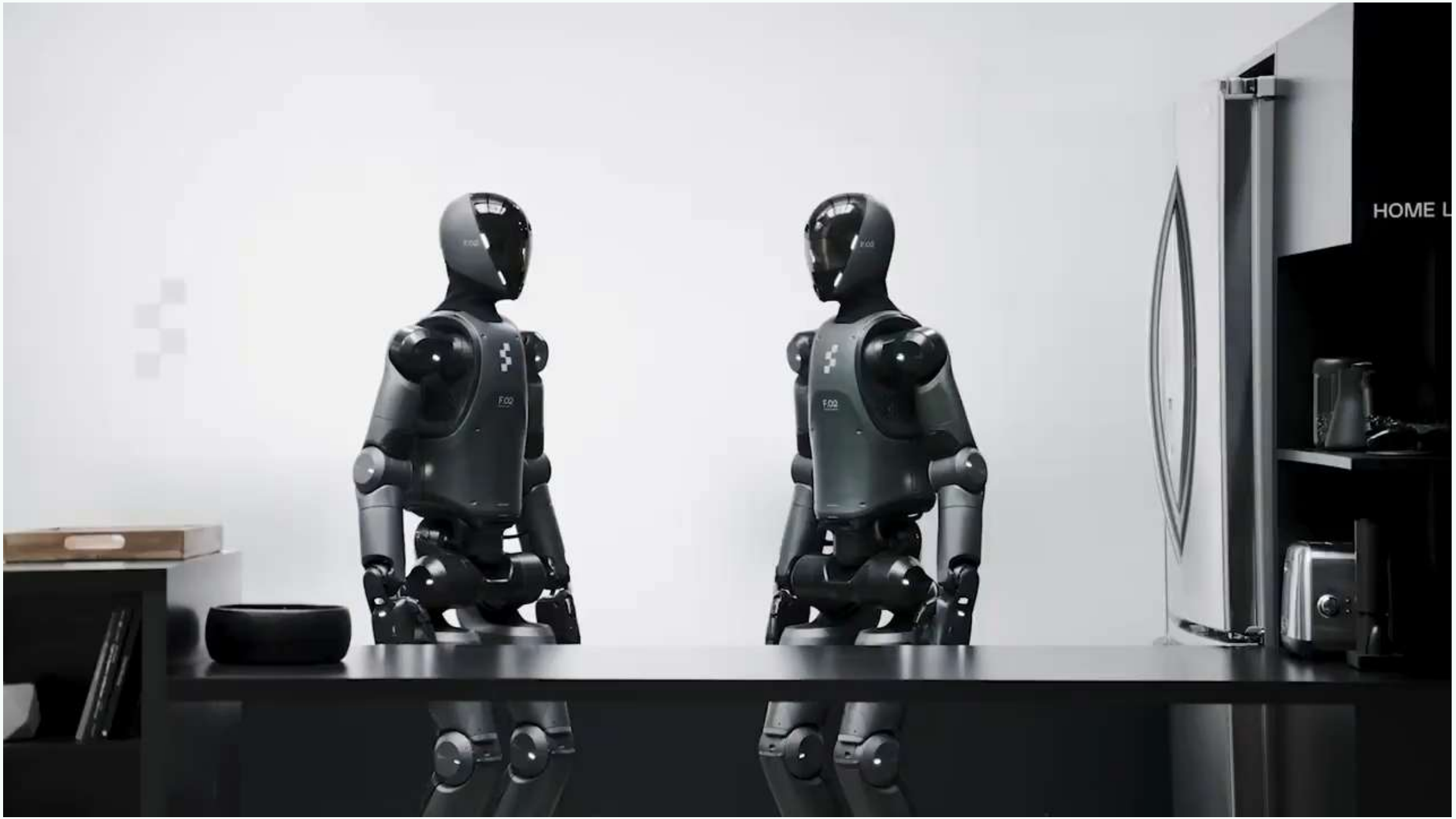
7. 热门MCP Servers推荐

推荐9：社交软件 Slack

用于 Slack API 的 MCP 服务器，使 LLM 能够与 Slack 工作区进行交互，用于频道管理和消息传递。

说明：去https://app.slack.com/intl/zh-cn注册并获取自己的team id

8. A2A协议：开启Agent间自然协作



8、A2A：开启 Agent 间的自然协作

在 AI Agent 的世界里，主要解决两大互联领域的挑战：

第一、Agent 与 Tools（工具）的交互

Agent 需要调用外部 API、访问数据库、执行代码等。

第二、Agent 与 Agent（其他智能体或用户）的交互

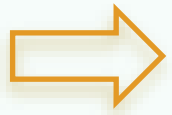
Agent 需要理解其他 Agent 的意图、协同完成任务、与用户进行自然的对话。

8、A2A：开启 Agent 间的自然协作

在 AI Agent 的世界里，主要解决两大互联领域的挑战：

第一、Agent 与 Tools（工具）的交互

Agent 需要调用外部 API、访问数据库、执行代码等。



MCP

第二、Agent 与 Agent（其他智能体或用户）的交互

Agent 需要理解其他 Agent 的意图、协同完成任务、与用户进行自然的对话。

8、A2A：开启 Agent 间的自然协作

在 AI Agent 的世界里，主要解决两大互联领域的挑战：

第一、Agent 与 Tools（工具）的交互

Agent 需要调用外部 API、访问数据库、执行代码等。

⇨ **MCP**

第二、Agent 与 Agent（其他智能体或用户）的交互

Agent 需要理解其他 Agent 的意图、协同完成任务、与用户进行自然的对话。

⇨ **A2A**

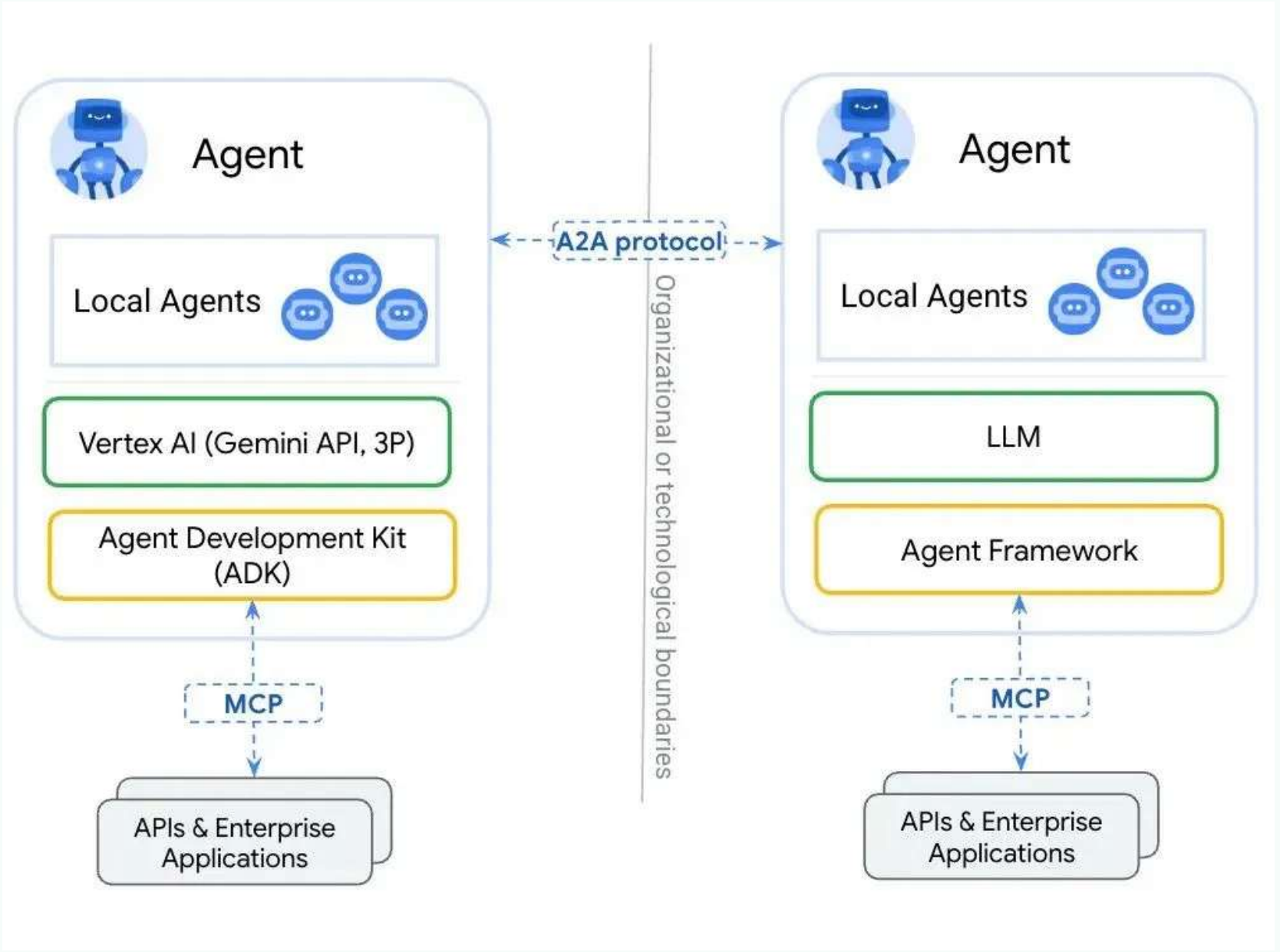
8.1 A2A的发布

谷歌，25年4月10日发布开源的、应用层协议 A2A（Agent-to-Agent 协议），即 Agent-to-Agent。其设计目的是使智能体（Agent）间能够以一种自然的模态进行协作，类似于人与人之间的互动。

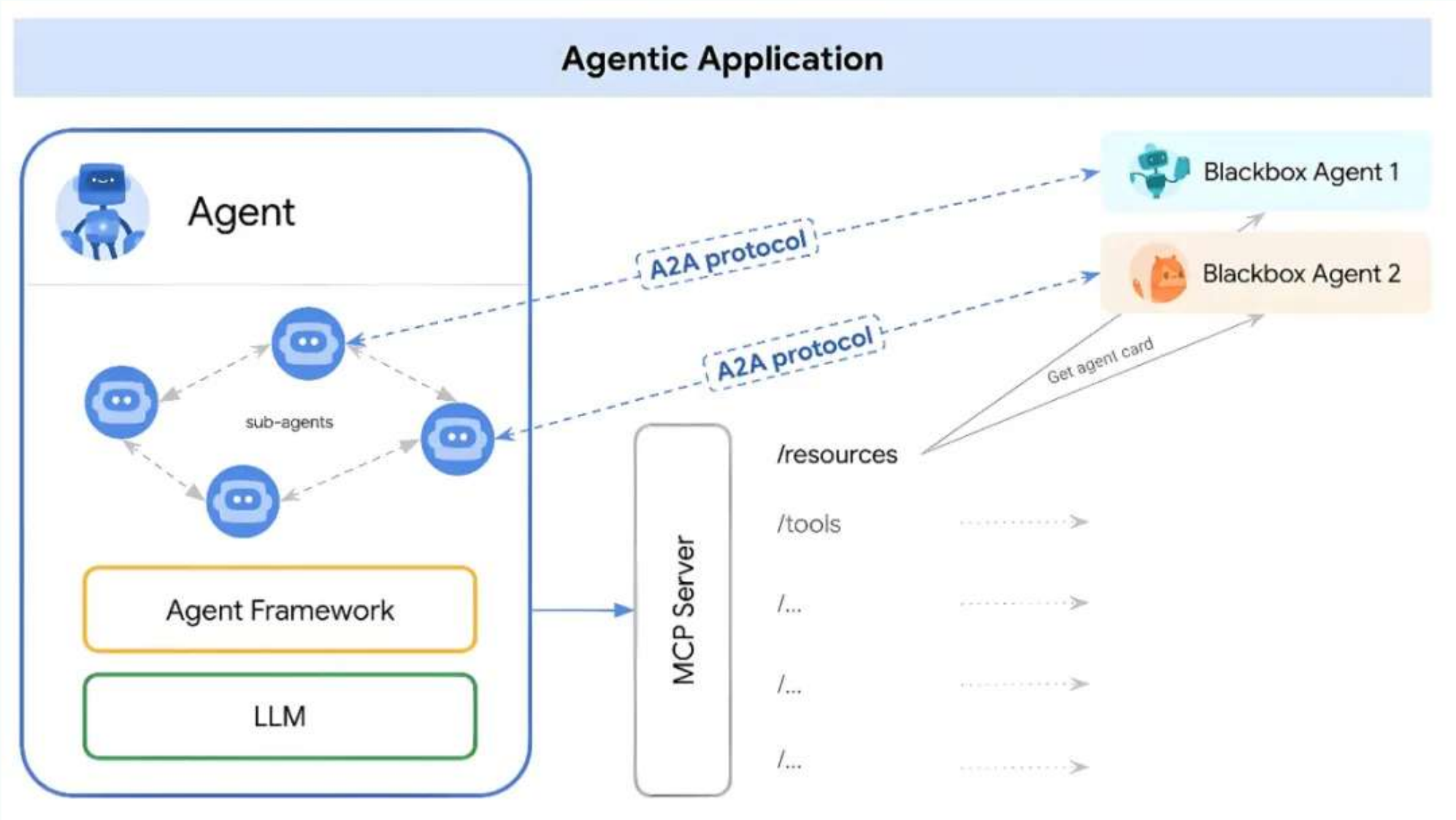
Github 地址：https://github.com/google/A2A

8.2 A2A的设计意义

基于不同底层框架和供应商平台创建的 AI Agent 之间可以实现通信、发现彼此的能力、协商任务并开展合作，企业可以通过专业的智能体团队处理复杂的工作流程。这无疑是其最为突出的贡献。



8.2 A2A的设计意义


Agentic Application
Agent
sub-agents
Agent Framework
LLM
A2A protocol
A2A protocol
Blackbox Agent 1
Blackbox Agent 2
Get agent card
MCP Server
/resources
/tools
/...
/...
/...

8.3 举例

举例1：阿里云 & 火山云

阿里云上创建的 AI Agent，通过A2A协议，可以与火山云上创建的 AI Agent 进行无缝的通信与协作。

8.3 举例

举例2：修理汽车

用户（或代表用户的智能体）对修理店智能体说：“给我看看左前轮的照片，似乎漏液了，这种情况多久了？”

- A2A 协议使得人与智能体之间这种更自然、多轮次的对话式互动成为可能。

修理店智能体在诊断出问题后，可能需要向零件供应商智能体查询某个零件的库存和价格。

- 这种智能体与智能体之间的协作同样需要 A2A 协议来支持。

8.3 举例

举例3：人才招聘

利用 A2A 协议，招聘流程可以如此高效：

在谷歌的 Agentspace 统一界面中，招聘经理可以向自己的智能体下达任务，让其寻找与职位描述、工作地点和技能要求相匹配的候选人。

然后，该智能体立即与其他专业智能体展开互动，寻找潜在候选人。

用户会收到推荐人选，之后可以指示自己的智能体安排进一步的面试，面试环节结束后，还可以启动另一个智能体来协助进行背景调查。

8.4 A2A展望

谷歌已经与超过 50 家技术合作伙伴（例如 Atlassian、 Box、 Salesforce、 SAP 等）和服务提供商建立了合作关系。这表明了行业对这些协议的认可和采用，对于 AI 学习者来说，也意味着这些协议可能会成为未来职业发展中的关键技能。

尚硅谷让天下没有难学的技术