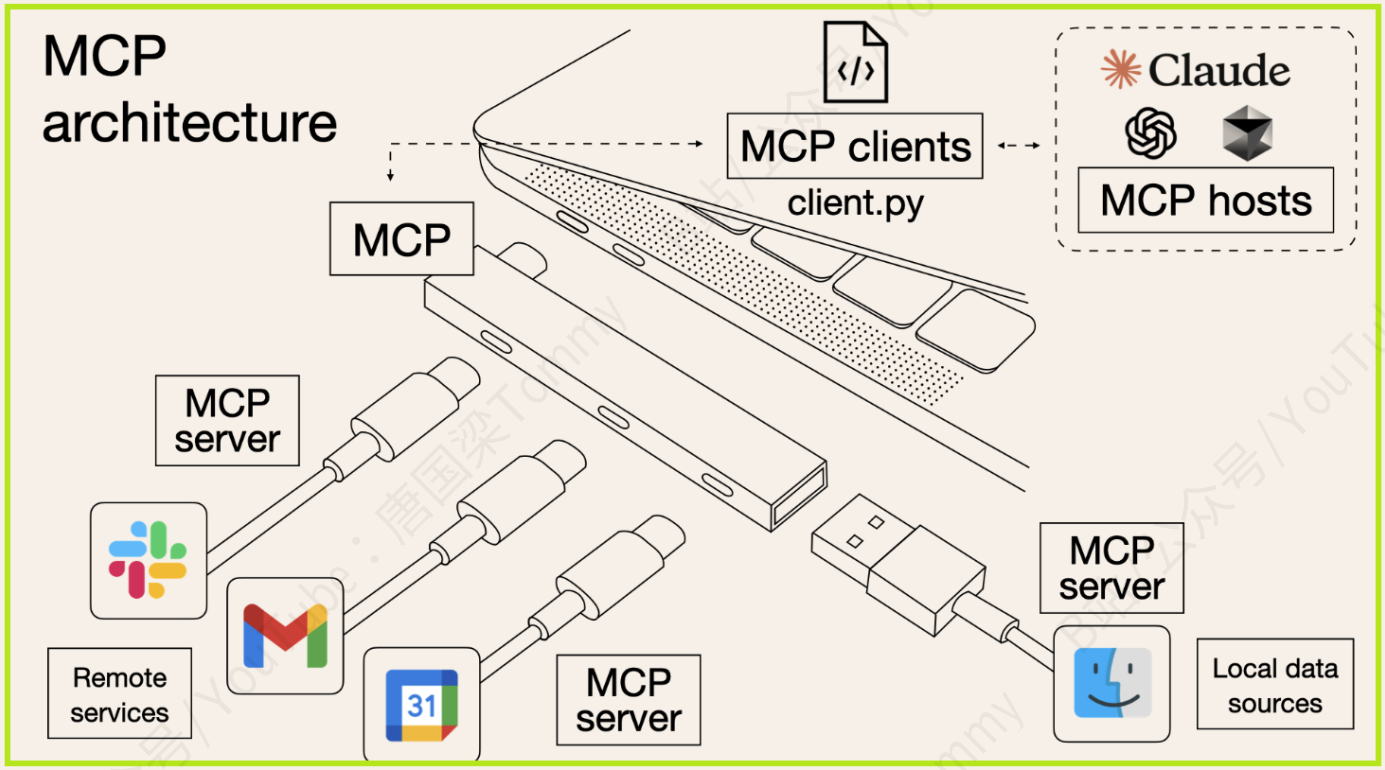


✅ MCP Servers可以提供三种类型的标准能力:

- **资源 (Resources)**: 是可以被引用和检索的数据对象，包括文档、图像、数据库模式和其他结构化数据。客户端可以读取这些文件的数据。例如，一个 MCP 服务器可以暴露一个包含产品信息的数据库模式作为资源。
- **提示词 (Prompts)**: 提示词，为用户预先定义好的完成特定任务的模板。提示有助于确保常见任务的一致性、高质量 AI 输出。它们可以包含动态内容的占位符，并且可以链接在一起以创建复杂的工作流程。例如，一个 MCP 服务器可以提供一个用于生成产品描述的提示模板。
- **工具 (Tools)**: 是可以被LLM调用的函数或第三方服务，例如查询数据库、调用 API 或处理数据。每个工具都有名称、描述和输入/输出模式定义。LLM 根据工具的描述决定应该使用哪个工具。例如，一个 MCP 服务器可以提供一个用于查询天气预报的工具。

✅ **通信协议**采用 JSON-RPC 2.0，支持两种类型的通讯机制：

- **本地通讯 - Stdio（标准输入输出）** ： 适用于本地进程间通信。
- **远程通讯 - 基于SSE的HTTP通讯** ： 支持流式传输与远程服务交互。

4. MCP 的工作流程

MCP（模型上下文协议）的工作流程主要围绕着**客户端-服务器架构**展开，旨在标准化 AI 模型与外部工具和数据源之间的交互。以下是其主要的工作步骤：

第1步：能力发现：当一个 AI 应用（作为 MCP 客户端）需要扩展其能力时，它首先会连接到一个或多个 **MCP 服务器**。MCP 客户端会向 MCP 服务器询问其提供的功能，即**工具 (Tools)**、**资源 (Resources)** 和 **提示 (Prompts)** 的列表和描述。这就像客户端从服务器获取一个“菜单”，了解服务器可以做什么以及如何使用。

- **工具 (Tools)** 是 AI 模型可以执行的操作或函数，例如查询数据库、发送邮件或生成图像。
- **资源 (Resources)** 是 AI 模型可以访问和检索的数据对象，例如文档、API 响应或数据库模式。
- **提示 (Prompts)** 是预定义的模板，用于指导 AI 的交互，确保一致性和效率。

第2步：增强提示：当用户提出需要外部数据或操作的查询时，该查询以及服务器提供的工具/资源/提示的描述会被发送给 AI 模型（通过其宿主应用，即 MCP 主机）。模型现在“知道”它可以利用哪些服务器功能来完成任务。例如，如果用户询问“明天的天气怎么样？”，发送给模型的提示会包含一个描述“天气 API 工具”的信息，该工具由某个 MCP 服务器提供。

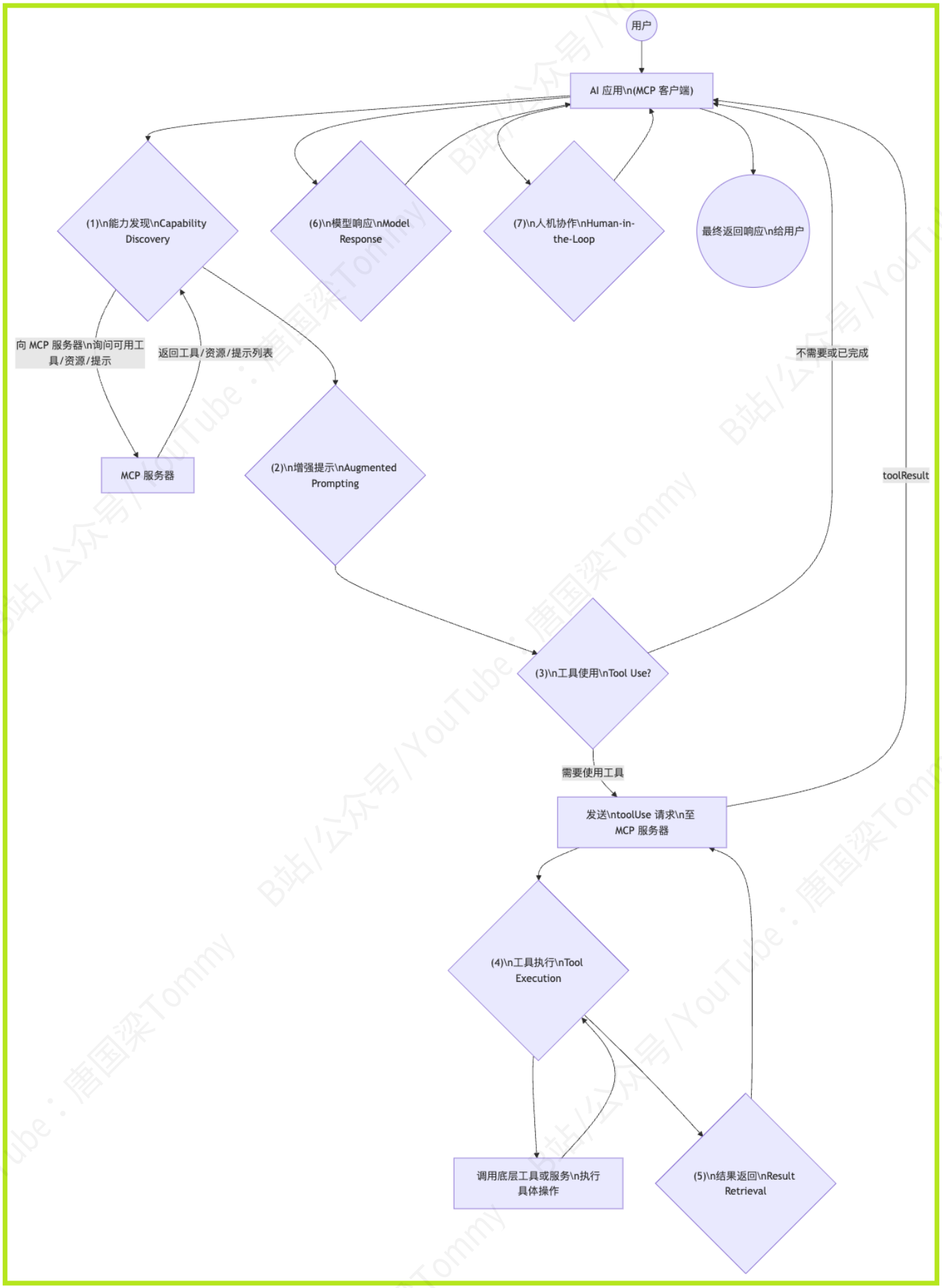
第3步：工具使用：基于用户的查询和可用的工具/资源/提示的描述，AI 模型（在 MCP 客户端中）会决定是否需要使用某个工具来完成任务。如果需要，客户端会向相应的 **MCP 服务器** 发送一个 `toolUse` 消息，指定要使用的工具及其所需的参数。

第4步：工具执行：**MCP 服务器** 接收到 `toolUse` 请求后，会将该请求分发给相应的底层工具或服务。例如，如果请求是使用“天气 API 工具”，服务器会调用实际的天气 API 并获取结果。

第5步：结果返回：**MCP 服务器** 在执行完工具后，会将结果以结构化的格式返回给 **MCP 客户端**，包含一个 `toolResult` 消息以及工具执行的输出。

第6步：模型响应：**MCP 客户端** 接收到 `toolResult` 后，会将工具执行的结果添加到对话历史中，并将其转发回 AI 模型（通常通过其宿主应用）。AI 模型现在可以利用这些信息来生成最终的响应给用户。这个过程可能涉及多次工具调用和数据检索，AI Agent 可以自主决定使用哪些工具、以什么顺序使用以及如何将它们串联起来完成复杂的任务。

第7步：人机协作：MCP 还引入了人机协作的功能，允许人类提供额外的数据并批准某些执行步骤。



总结来说，MCP 的工作流程可以概括为：1️⃣ **AI 客户端发现服务器能力** -> 2️⃣ **AI 模型基于用户查询和可用能力决定使用哪些工具/资源/提示** -> 3️⃣ **客户端向服务器发送请求** -> 4️⃣ **服务器执行操作并返回结果** -> 5️⃣ **客户端将结果提供给 AI 模型以生成最终响应**。

整个过程通过标准化的协议进行通信，使得不同的 AI 模型和各种外部工具及数据源能够以一种通用且灵活的方式进行集成。这种标准化降低了集成复杂性，提高了互操作性，并为构建更强大的 AI 应用奠定了基础。

5. MCP 的关键优势

MCP 为 AI 系统开发带来了多项显著优势:

- **降低集成复杂性** : MCP 提供了一个统一的接口，消除了为每个 LLM 驱动的应用程序集成外部功能而实施其自身方法的必要性。通过标准化的协议，开发者可以更轻松地将 AI 模型连接到各种工具和数据源。它将一个 M×N 问题（M 个 AI 模型与 N 个工具/数据源的集成）转化为一个 M+N 的解决方案，大大减少了集成所需的自定义代码和适配器。
- **加快开发速度**: 开发者可以利用预构建的 MCP 客户端和服务器，从而节省开发自定义集成的时间和精力。工具和 API 开发者只需要构建一个 MCP 服务器，就可以被所有兼容的 MCP 客户端使用。
- **提高可扩展性和可靠性**: 通过标准化的架构，MCP 有助于构建更健壮和可扩展的 AI 系统。维护上下文跨工具也变得更加容易，因为交互共享一个通用的框架。
- **实现供应商无关的开发**: MCP 作为一个开放标准，不限制开发者使用特定的 AI 提供商的生态系统或工具链。任何支持 MCP 的 AI 客户端（例如 Claude 或开源 LLMs）都可以使用任何兼容的 MCP 服务器。
- **增强 AI Agent的上下文感知能力**: MCP 的出现能够帮助上下文层实现最好的效果。由于上下文层通常需要开发者自己进行定义，而 MCP 作为一个开源协议，能够最好地发挥社区的力量来加速这个过程。
- **实现“万能应用”的潜力**: 通过合适的 MCP 服务器，用户可以将每个 MCP 客户端变成一个“万能应用”。

6. MCP 与 API 对比

- **定义**
  - **MCP**：MCP 是一种标准化协议，旨在为大型语言模型（LLM）提供上下文信息，以便它们能够更有效地与外部工具和数据源进行交互。它强调数据的上下文环境和语义背景，使得模型能够理解和处理复杂的结构化数据。
  - **API**：API 是一组规则和工具，允许不同的软件应用程序之间进行通信。它定义了请求和响应的格式，使开发者能够集成不同服务的功能，而无需了解其内部实现细节。

- **主要区别**