



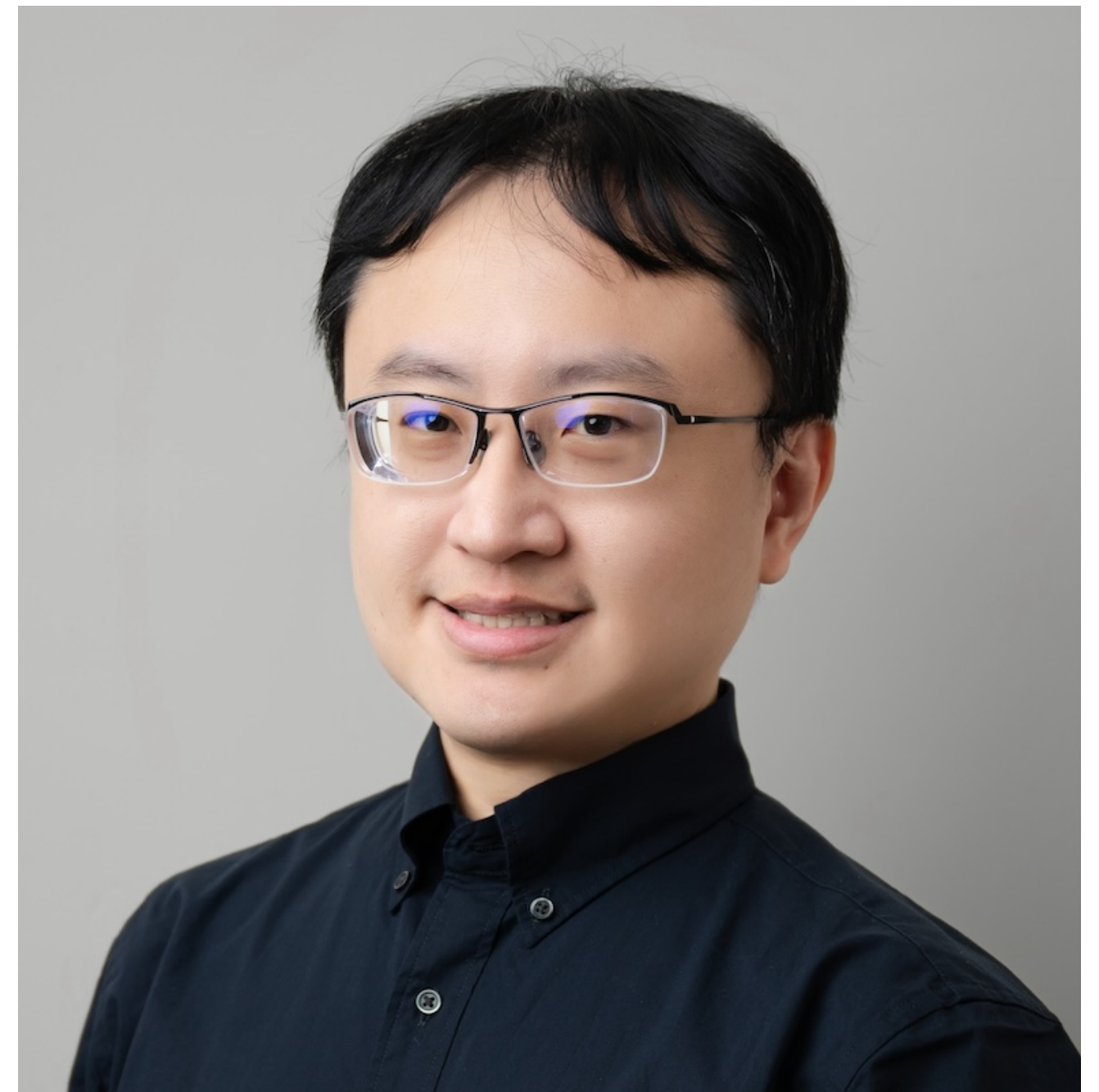
淺談模型上下文協定 MCP 應用開發

ihower@生成式AI年會 2025/5/23

投影片已在 <https://ihower.tw>

我是誰

- 張文鈺，網路暱稱 ihower
- 2002 年開始從事 Web 軟體開發
- 擅長 Ruby on Rails 技術的 Full-Stack Web Developer
- 2005 年 個人部落格 <https://ihower.tw>
- 2018 年 自行開業 愛好資訊科技 <https://aihao.tw>
- 2023 年 開始投入 AI 工程領域
 - 🚀 經營 AI Engineer 電子報
 - 🎓 招牌課程: LLM 應用開發工作坊



紐約 AI Engineer Summit 特輯 #26 ➤



i 愛好 AI Engineer 電子報
寄給我 ▾

4月16日 週三 下午5:54 (9 天前) ☆ ☺ ↶ ⋮

AI Engineer SUMMIT

Hello! 各位 AI 開發者大家好 🙌

我是 ihower，這一期電子報跟大家分享二月底在紐約舉辦的 [AI Engineer Summit](#)。這大會的定位就是針對 AI 軟體工程師，今年的大主題就是最酷的 Agents 工程。

主辦單位已經釋出全部錄影在 [Youtube](#) 上了，總共有六十多場演講我都消化了，以下精選了我最有收穫的 16 場分享给大家。

因為全放內容會太多，以下只有我簡單的介紹。有興趣的可以點下去看我用 AI 整理後的中等長度摘要。如果覺得不夠，Facebook 留言中裡面有放完整版逐字稿跟截圖的連結，網頁我也有放一鍵 Copy 全部逐字稿，方便讓你貼去 AI 自己問。

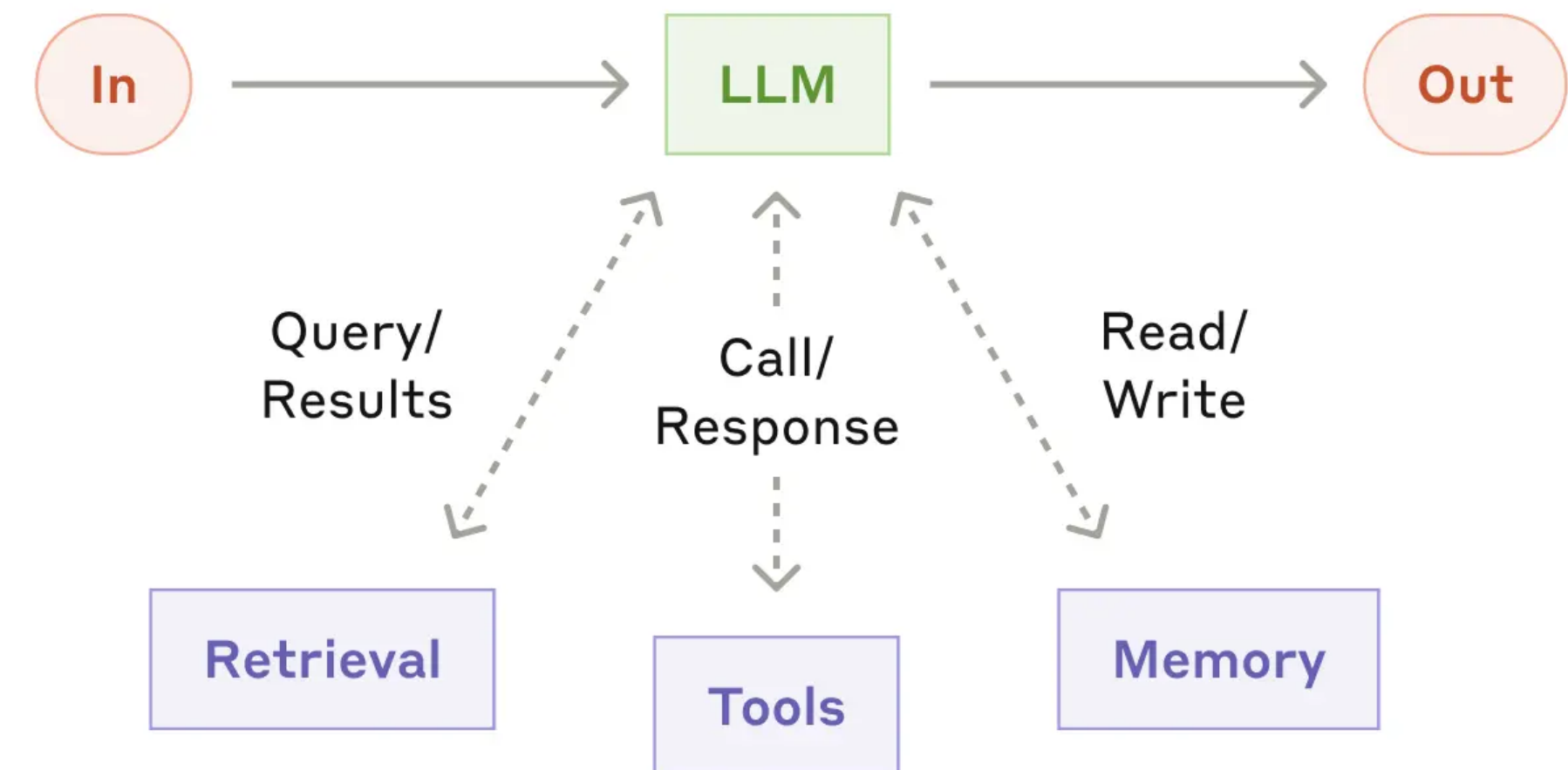
Agenda

-  Why MCP?
-  MCP Server
-  Remote MCP Server
-  MCP Client
-  Roadmap

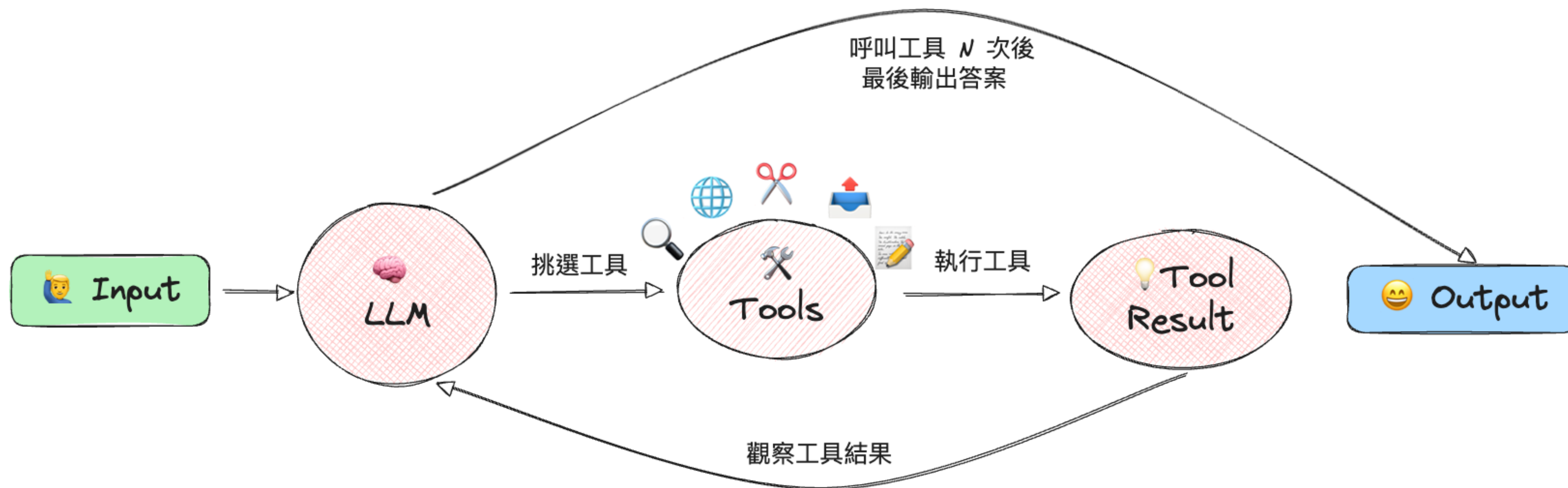
前情提要: LLM-based Agent

一個使用 LLM 大模型的 AI 元件: 它可以根據目標指令，決定步驟、不斷呼叫工具，直到完成任務

- 🧠 LLM 就像一位被關在房間裡的天才
- 🔒 無法主動學習或操作世界，也沒辦法存取即時資訊或使用電腦
- 🧰 需要透過工具，由外部系統幫它執行，或是存取長期記憶
- 🗣️ LLM 會輸出一段“工具執行指令”，由外部系統執行後把結果送回來，讓它不斷決定下一步，直到完成任務



Agent is LLM using tools in a loop



Agent is LLM using tools in a loop

```
instructions = "目標任務描述"
tools = [工具1, 工具2, 工具3....]
messages = [] # 記錄目前 messages 狀態

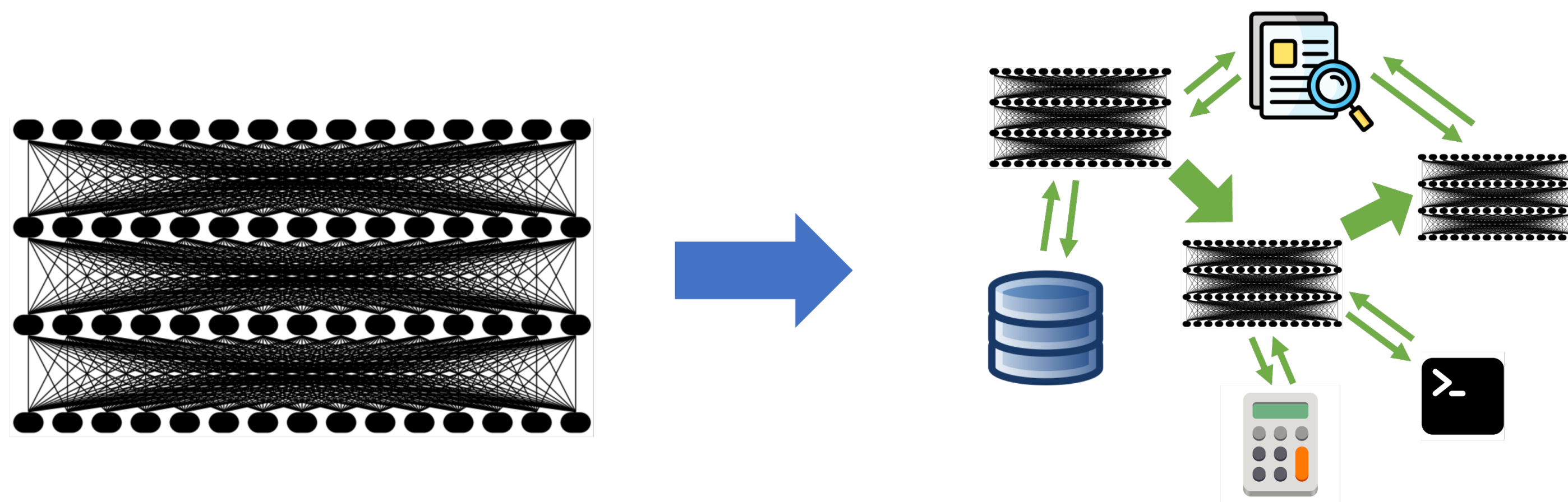
while True:
    response = model.run(instructions, tools, messages) # 呼叫 LLM 大模型，請問下一步

    if response.tool_calls:
        tool_result = tools.run(response.tool_calls) # 執行工具
        messages.append(tool_result) # 更新狀態
    else:
        print(response.final_answer) # 不需要執行工具，回傳最後答案
        break
```

可參考我之前的
淺談 LLM-based AI Agents 應用開發 分享
<https://ihower.tw/blog/archives/12586>

Raw intelligence $\neq$ Intelligent software systems

- 🧠 大模型的「原始智力」不等於「智慧軟體系統」
- 大模型 LLM 的「智力」只是基石，要把它轉化成真正有效的智慧系統，還需要兩個關鍵：
 - 🔧 工具整合: 讓模型能實際操作、查資料、記憶與回應
 - 📦 正確上下文(context): 依照任務動態提供模型需要知道的資訊
- 這就是所謂的 Context Engineering，動態處理「LLM 在當前任務中所需要的所有資訊和工具」



Model Context Protocol

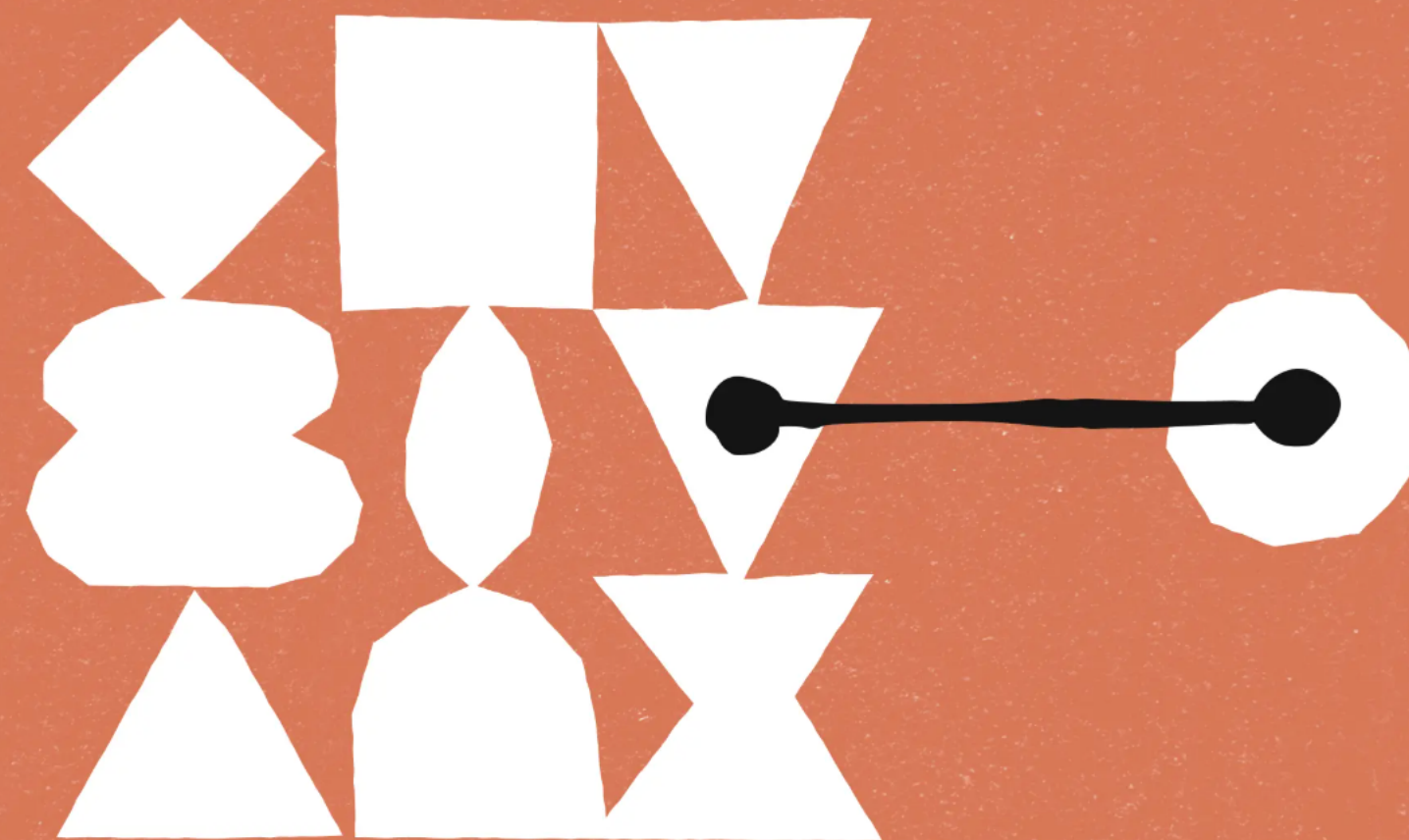
模型如何取得 Context 的標準化協議
包括 Tools、Resources 和 Prompts

<https://modelcontextprotocol.io>

Announcements

Introducing the Model Context Protocol

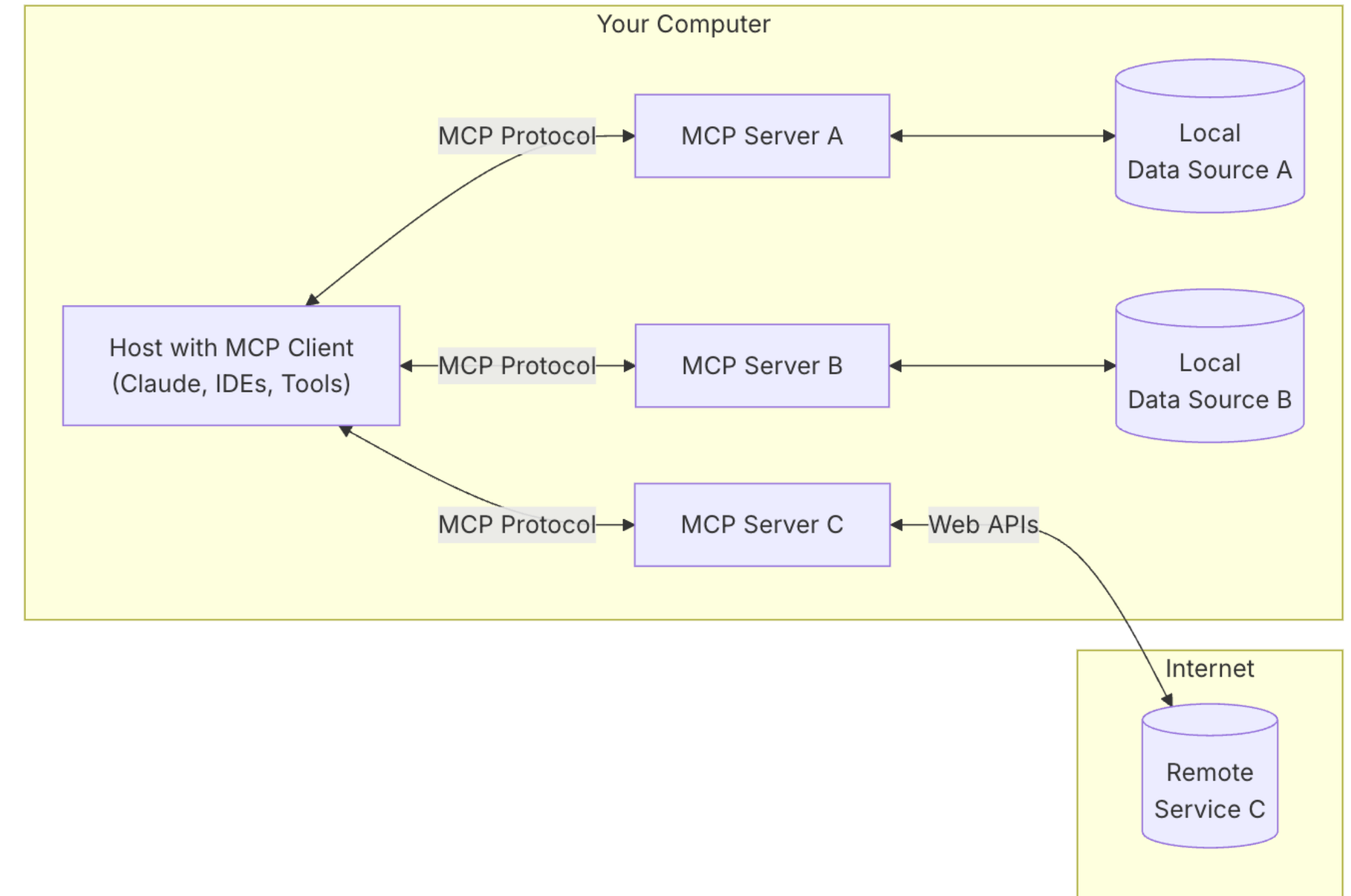
2024年11月25日 • 3 min read



常見使用場景: 讓 App 能夠更方便安裝第三方的工具擴充

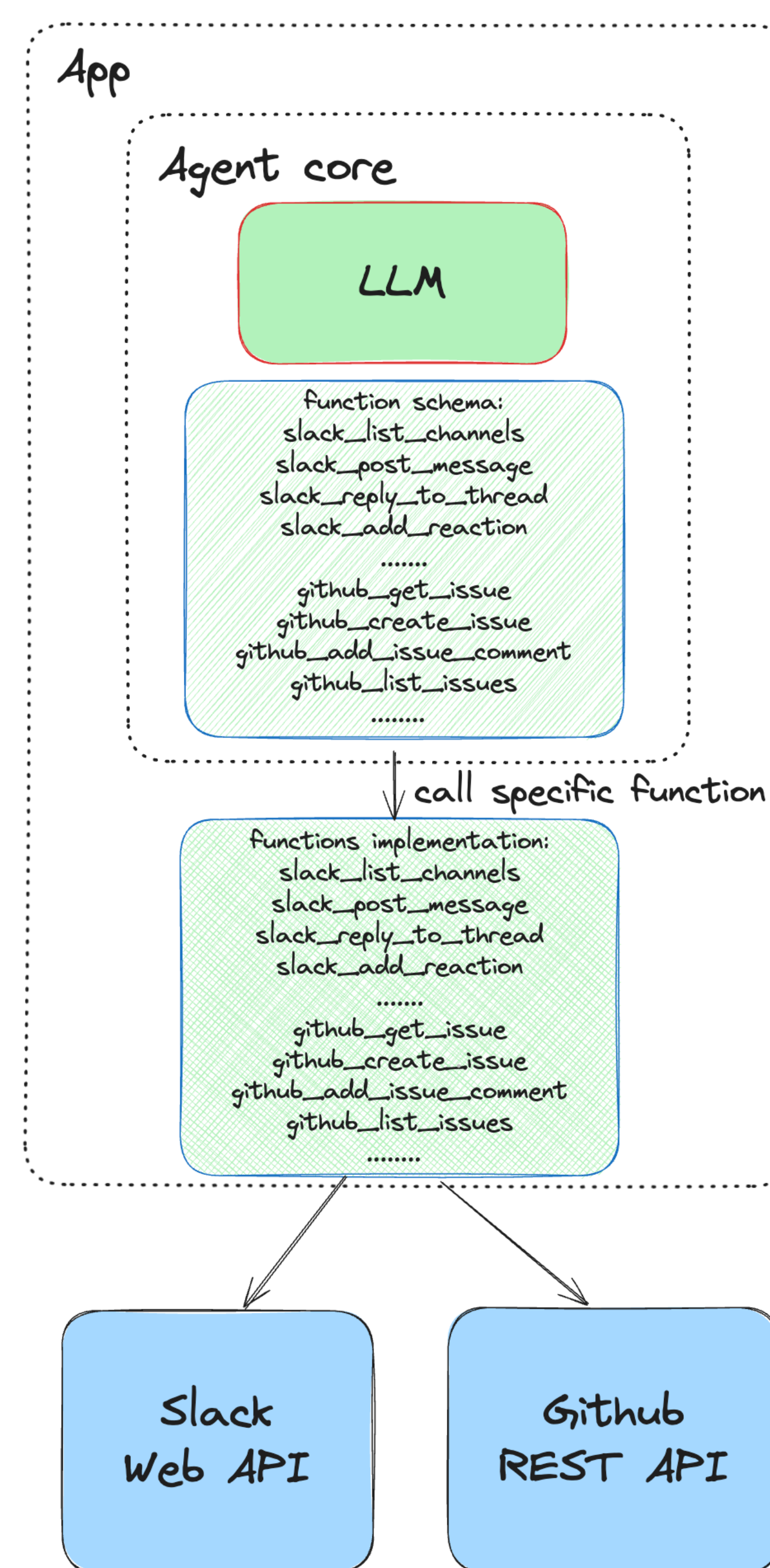
-  Client app 包括:
 - 本機 Claude Desktop App、ChatWise、Cursor、Windsurf Editor 等等 GUI 應用和 編輯器
 - 雲端 Claude Web、Dify 等等服務
 - 你自行開發的 Agent 程式
 - More: <https://modelcontextprotocol.io/clients>
-  搭配安裝不同 MCP servers 擴充
 - 一組 MCP server 就包含很多 functions 工具，是以**場景和使用案例為中心**，而非只安裝一個 function 工具
 - 例如安裝 Slack MCP server，就會一次安裝八個 functions 在你的 client app 上:

slack_list_channels, slack_post_message, slack_reply_to_thread, slack_add_reaction,
slack_get_channel_history, slack_get_thread_replies, slack_get_users,
slack_get_user_profile



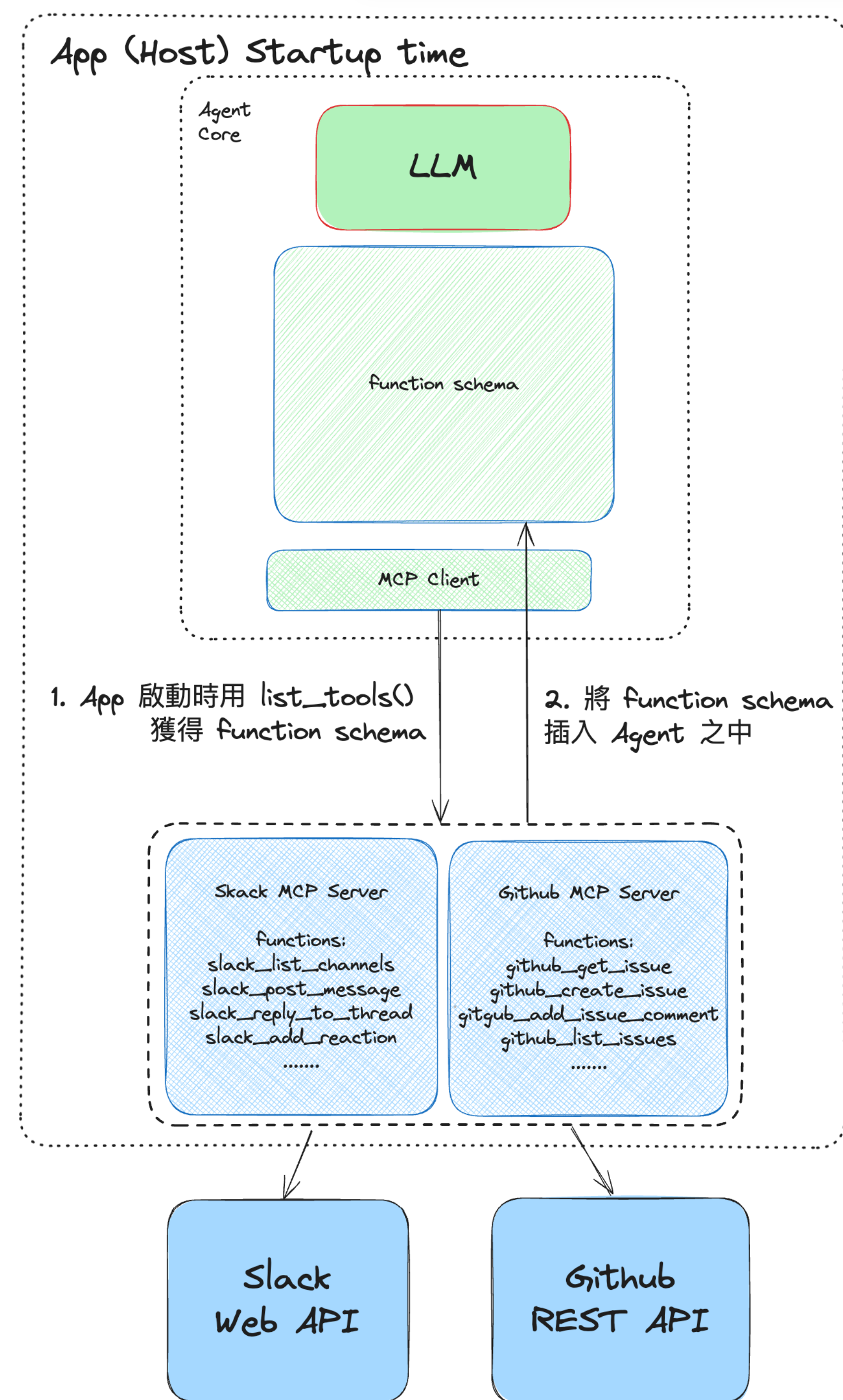
Before MCP

- 針對每個外部應用的每一個 API
 - 需要撰寫 function schema 和實作讓 Agent 可以使用
- 綠色屬於 App 的開發責任，藍色是工具廠商



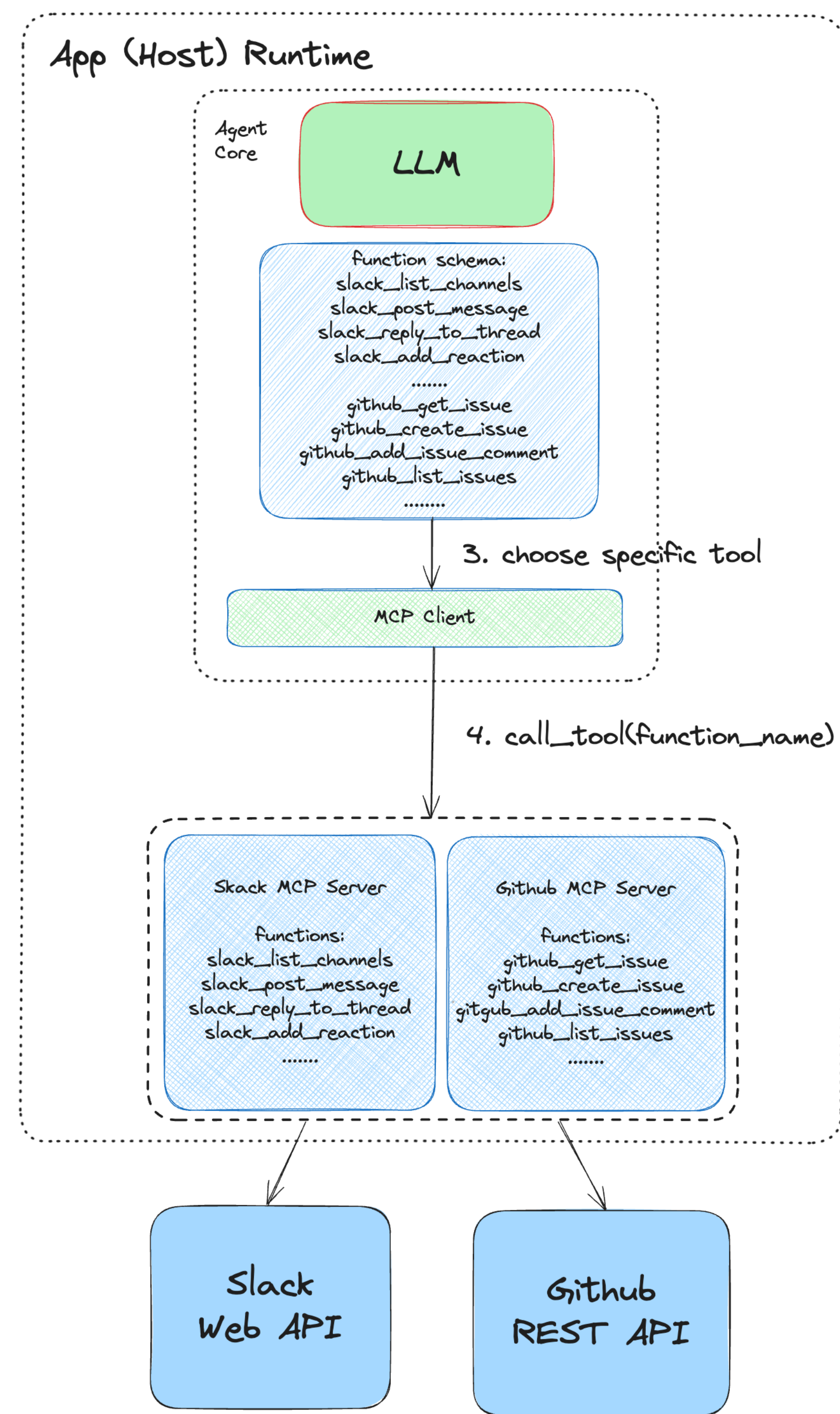
After MCP (啟動時)

- **1** MCP Client 在 app 啟動時，透過 MCP 標準操作 `list_tools()` 去問 MCP Servers 獲得 function schema
- **2** 將 function schema 放入 Agent 定義



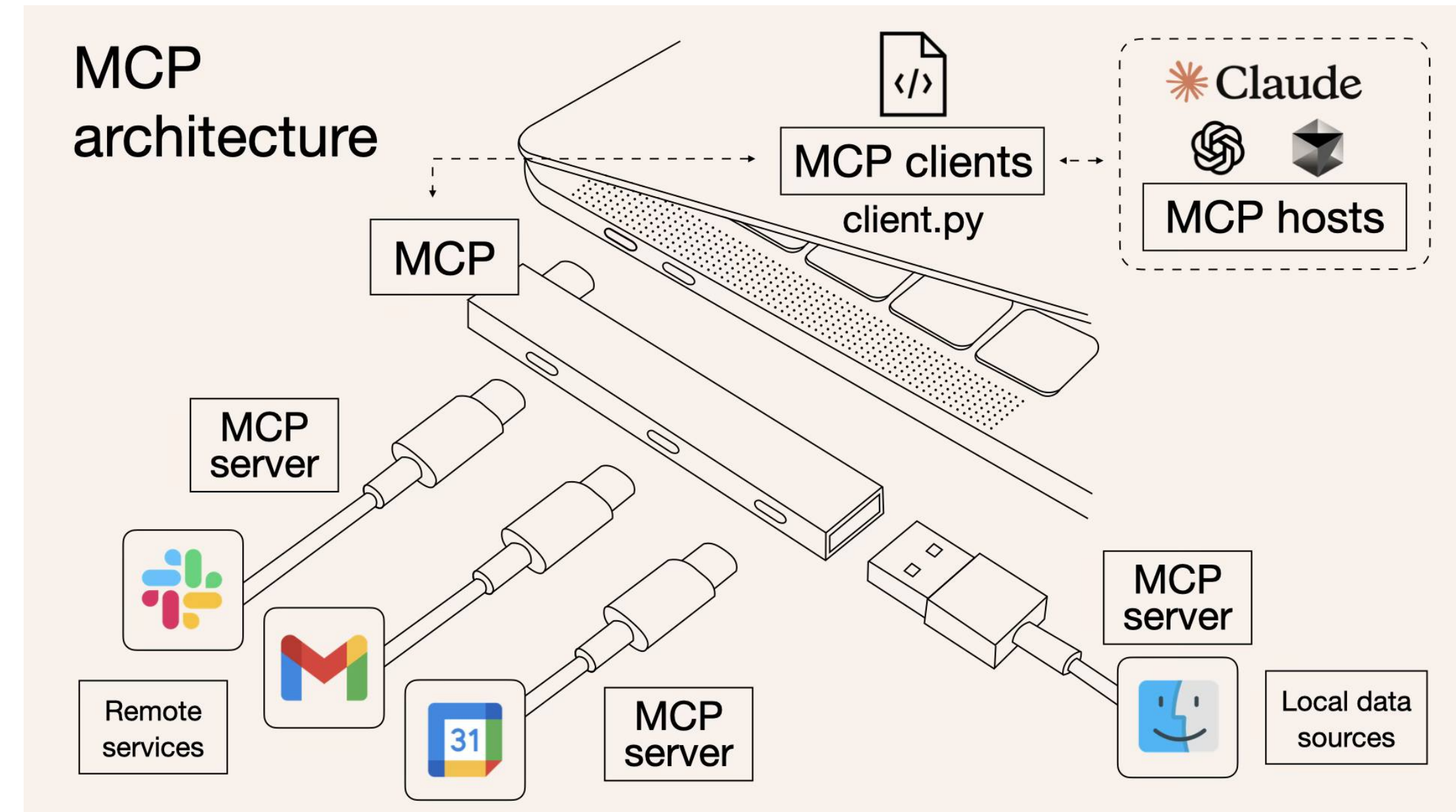
After MCP (執行時)

- 3 LLM 根據 function schema 挑選工具
- 4 透過 MCP client 標準操作 `call_tool(function_name)` 來執行工具
- 綠色屬於 App 的開發責任，藍色是工具廠商
 - MCP Server 變成是工具廠商的責任，包括提供 function schema 和實作



MCP 標準化的好處

- 🧑‍💻 對 AI app 開發者來說: 銜接任何 MCP server 工具超簡單
- 🏭 對工具廠商來說: 建構 MCP server 一次，所有 MCP client 客戶都可以使用
- 🧑‍🎓 對 AI app 的用戶來說: app 具有擴充能力，可以安裝任意 MCP servers
- 🏢 對企業來說: 清楚區隔負責 “Agent” 的 team 和 負責 “Tool” 的 team，可以分工協作



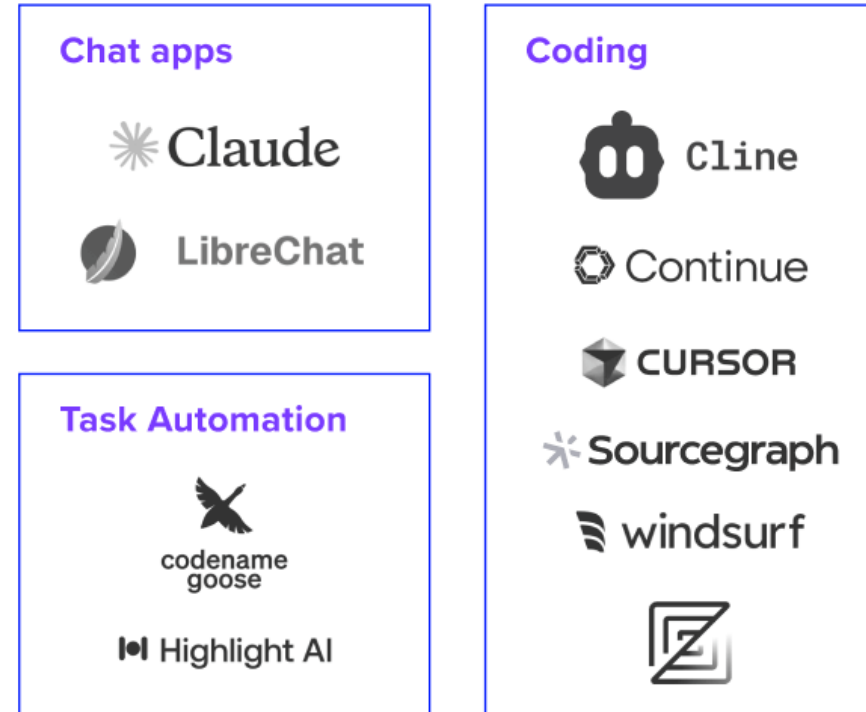
MCP ecosystem 發展迅速

- 🧑 AI 應用從 IDEs 到 Agents
- 🏢 外部和企業內部陸續採用
- 🌱 Open Source 社群活躍
- 🛠️ 工具廠商積極支援

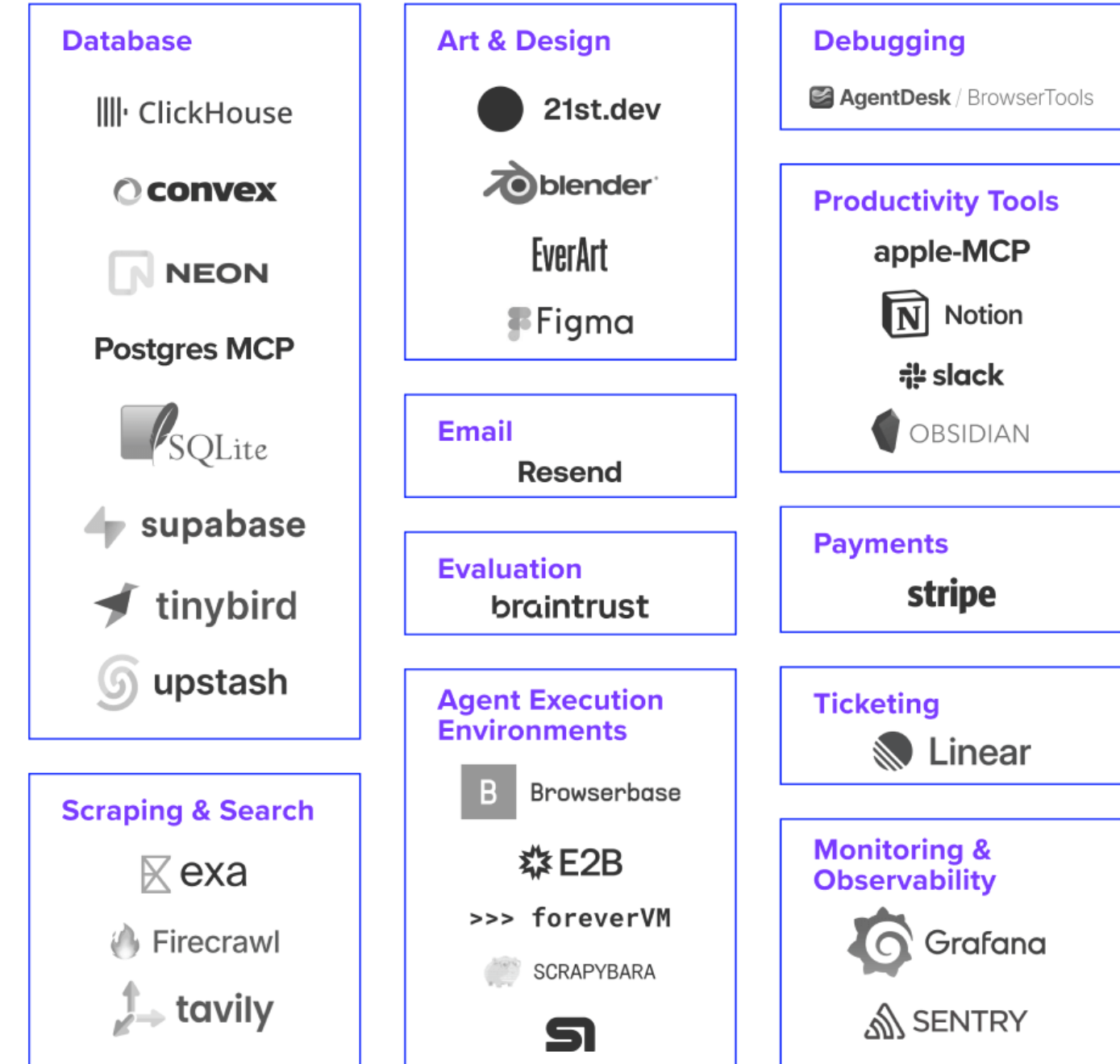
MCP Market Map

A work in progress.

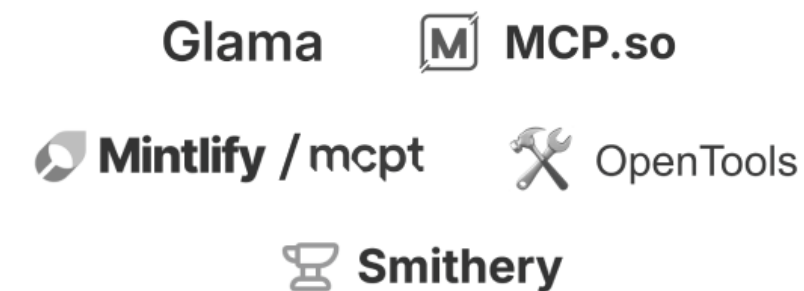
Top MCP Clients



Top MCP Servers



MCP Marketplace



Server Generation & Curation



Server Hosting



Connection Management



Charts provided herein are for informational purposes only and should not be relied upon when making any investment decision. Past performance is not indicative of future results. None of the above should be taken as investment advice; please see a16z.com/disclosures for more information.



<https://a16z.com/a-deep-dive-into-mcp-and-the-future-of-ai-tooling/>

進一步細看 MCP server 提供哪些 context

- 提供了
 -  Tools: 提供 functions 函式工具 給 模型
 -  Resources: 提供資料給 client app
 -  Prompts 提供提示詞樣板給 client app 的用戶
- 不過實際上只有 Tools 最有用，另兩個還好，主要要看 Client App 如何使用
 - 不同 Client App 使用 Tool 的方式很一致，都是讓 LLM 搭配 function calling 去使用
 - 但是 Resources 跟 Prompts 就不一定了，更依賴於各 Client App 的實作方式，有些可能完全忽略

MCP server 提供 Tool 能力

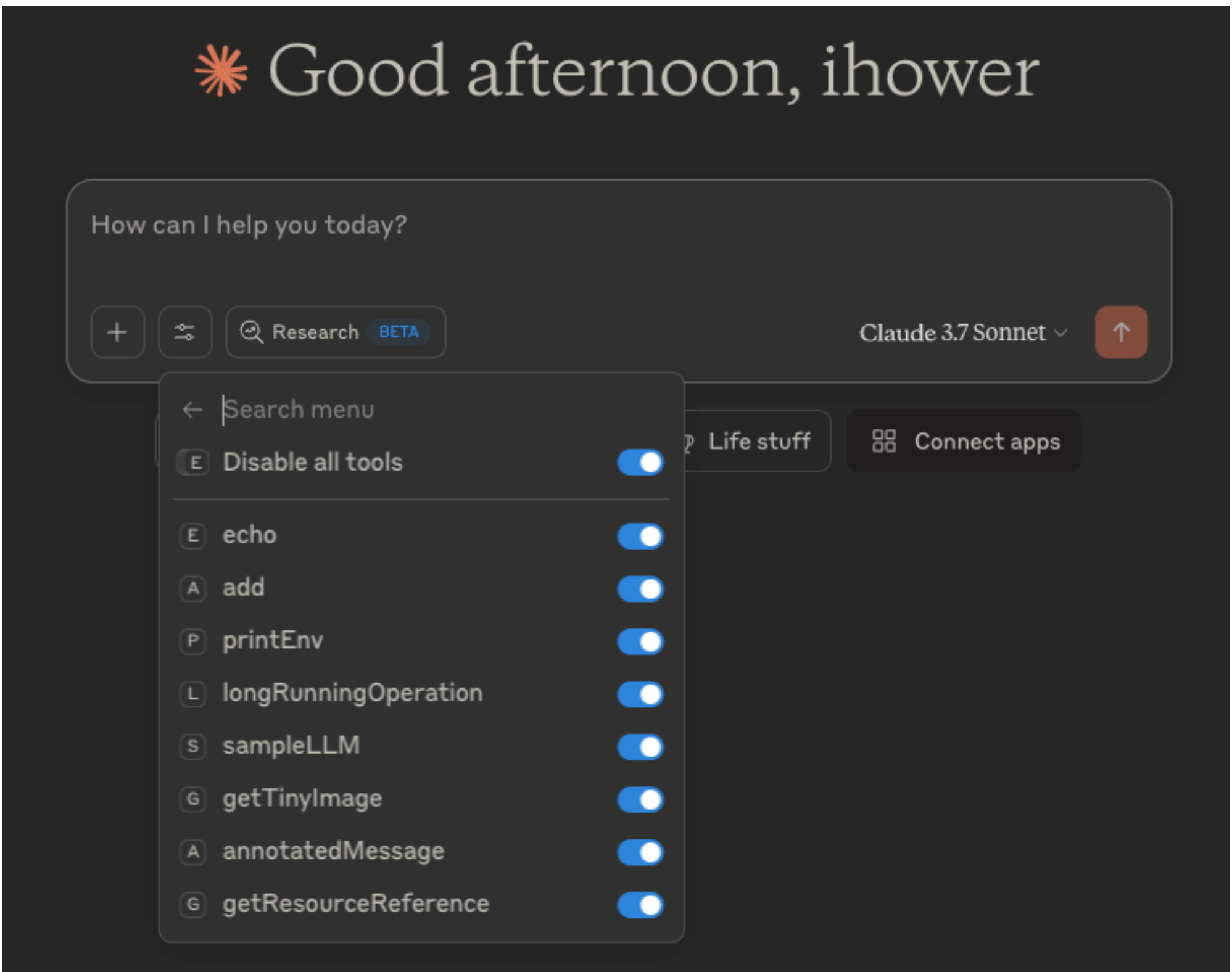
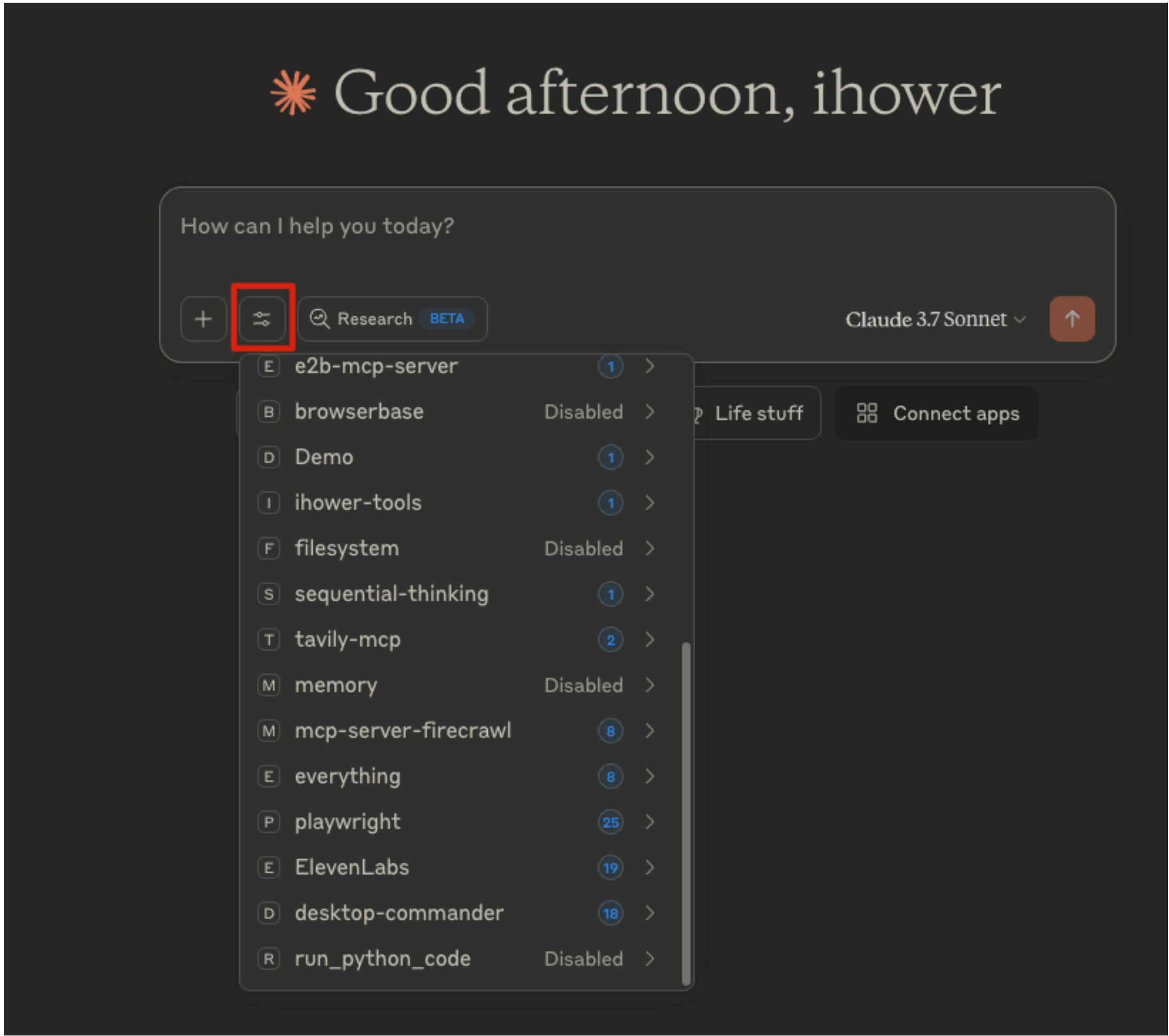
-  提供工具 function 的 schema 和實作
- Tools 是設計給 LLM 用的，實務上就是用 function calling 來運作
- 讓模型根據 function schema 挑選要呼叫哪一個 function
- 這裏 code 範例使用 官方 Python SDK:
 - <https://github.com/modelcontextprotocol/python-sdk>

```
from mcp.server.fastmcp import FastMCP

mcp = FastMCP("Demo")

@mcp.tool()
def generate_random_integer(a: int, b: int) -> int:
    """產生介於 a 和 b 之間的隨機整數"""
    import random
    return random.randint(a, b)
```

Claude app 使用範例



MCP server 提供 Resources

-  提供資料給 Client App
- 可以回傳任意資料: string, JSON, binary 都可以 (mime_type)
- 就像 HTTP server 的 GET request，讓 app 向 server 索取資料
- Resources 是設計給 app 用的，但要怎麼用，全看 app 設計
 - 例如在 Claude Desktop app 裡面，就是一個選單用用戶選了加到 prompt 裡面
 - 也可以讓模型決定是否加入 context, 看 app 怎麼設計
- ? 那為什麼不直接用 Tool 來提供資料？
 - Tool 是直接提供給模型，而這個 Resources 是提供給 Client App 使用的，用途和整合方式不同
 - MCP 的精神不只是提供模型 Context，而是設計一個給整個 Client App 應用的標準協議

```
from mcp.server.fastmcp import FastMCP

mcp = FastMCP("Demo")

@mcp.resource("config://app")
def get_config() -> str:
    """Static configuration data"""
    return "This is App configuration"
```

MCP server 提供 Prompts

- 🖋️ 提供預先設計好的高品質 prompt template 給 client app，讓用戶可以挑選來用
 - 例如複雜任務或高品質提問場景，例如資料分析、摘要生成等
- 如何讓用戶操作？全看 Client app 如何設計
 - 例如在 Claude App 是用下拉選單，然後會跳出一個小表單讓你填參數
 - 例如在 zed IDE 中，可在對話中設計快捷鍵 // 來選 template

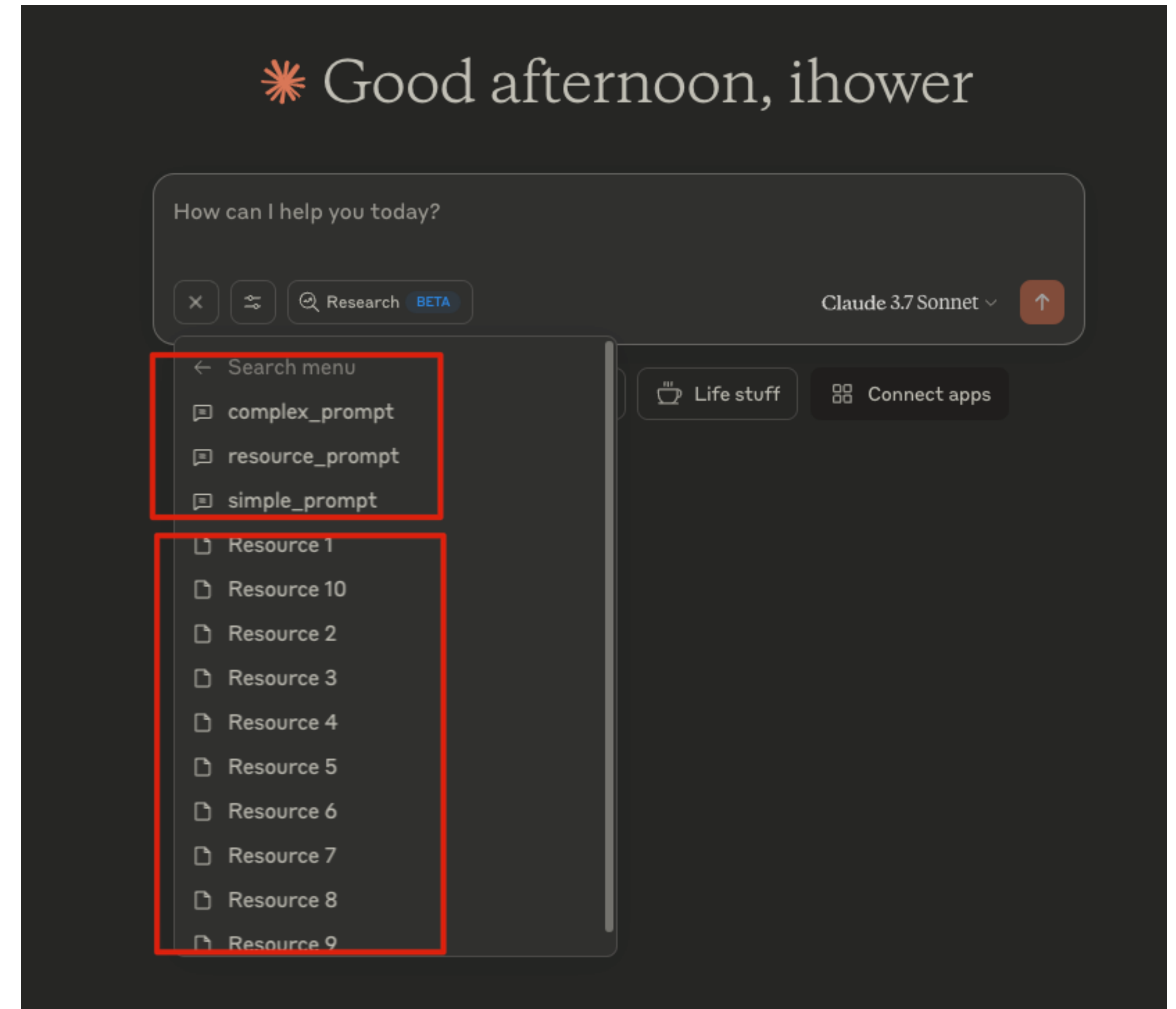
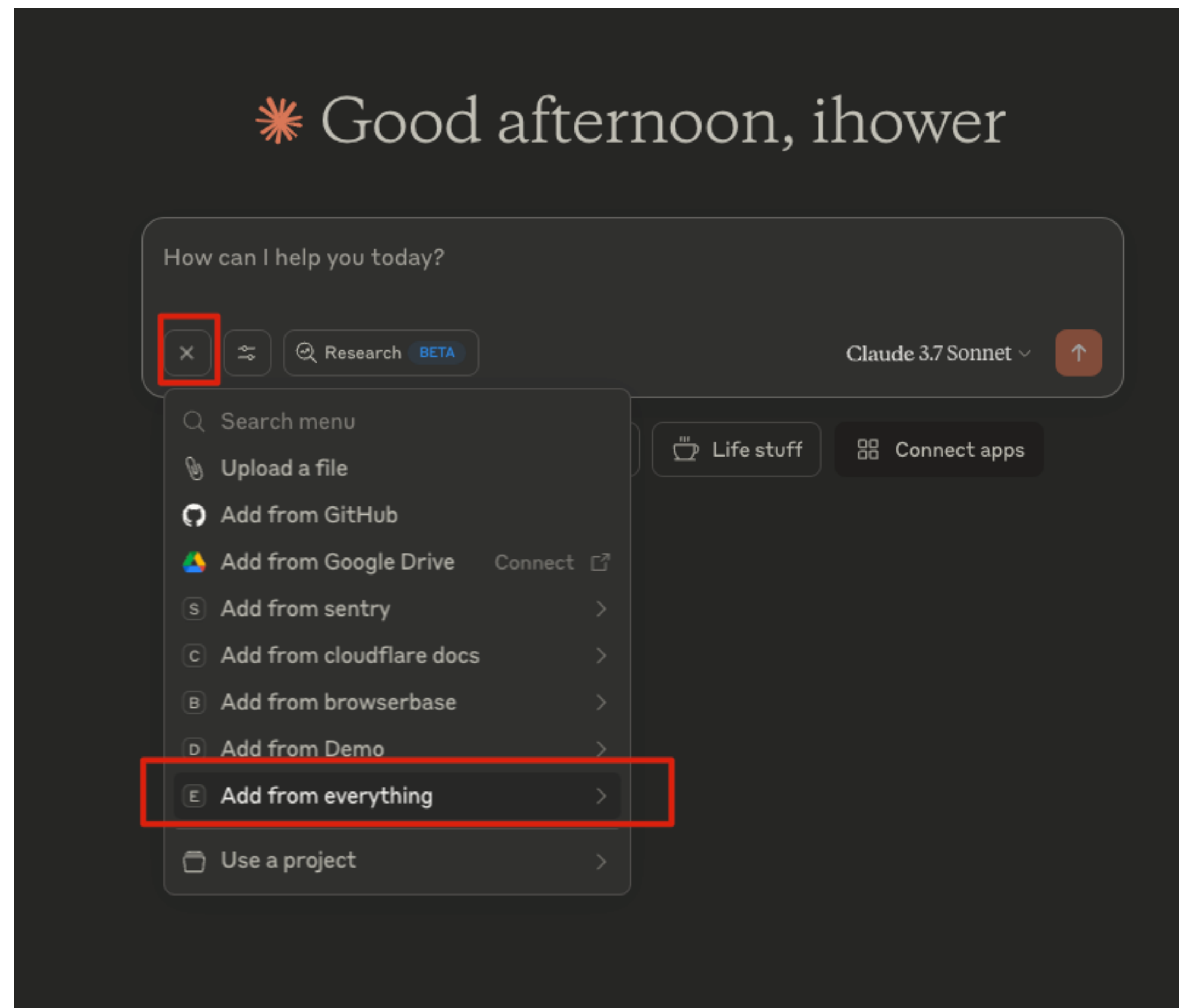
```
from mcp.server.fastmcp import FastMCP

mcp = FastMCP("Demo")

@mcp.prompt()
def review_code(code: str) -> str:
    return f>Please review this code:\n\n{code}"
```

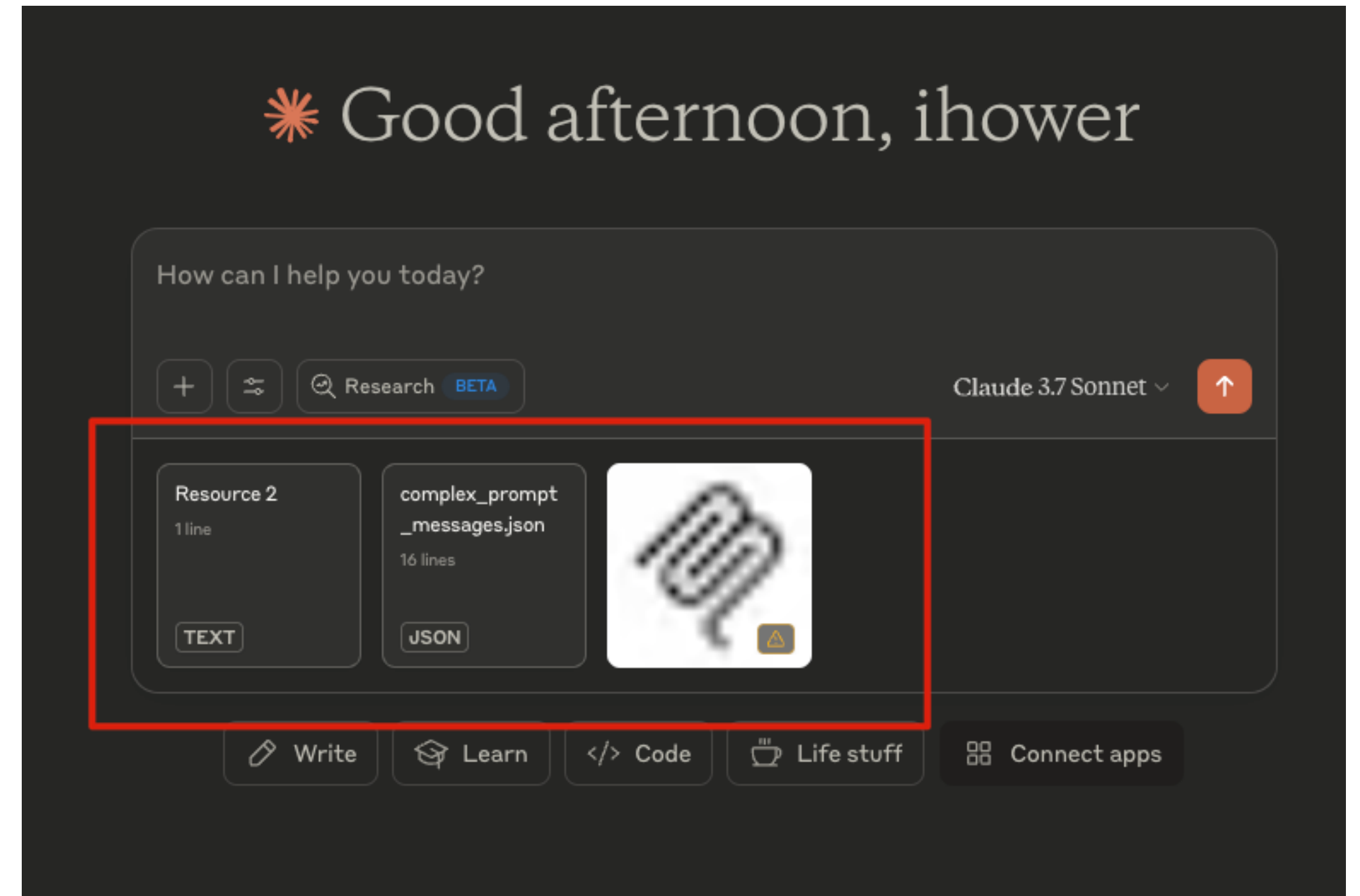
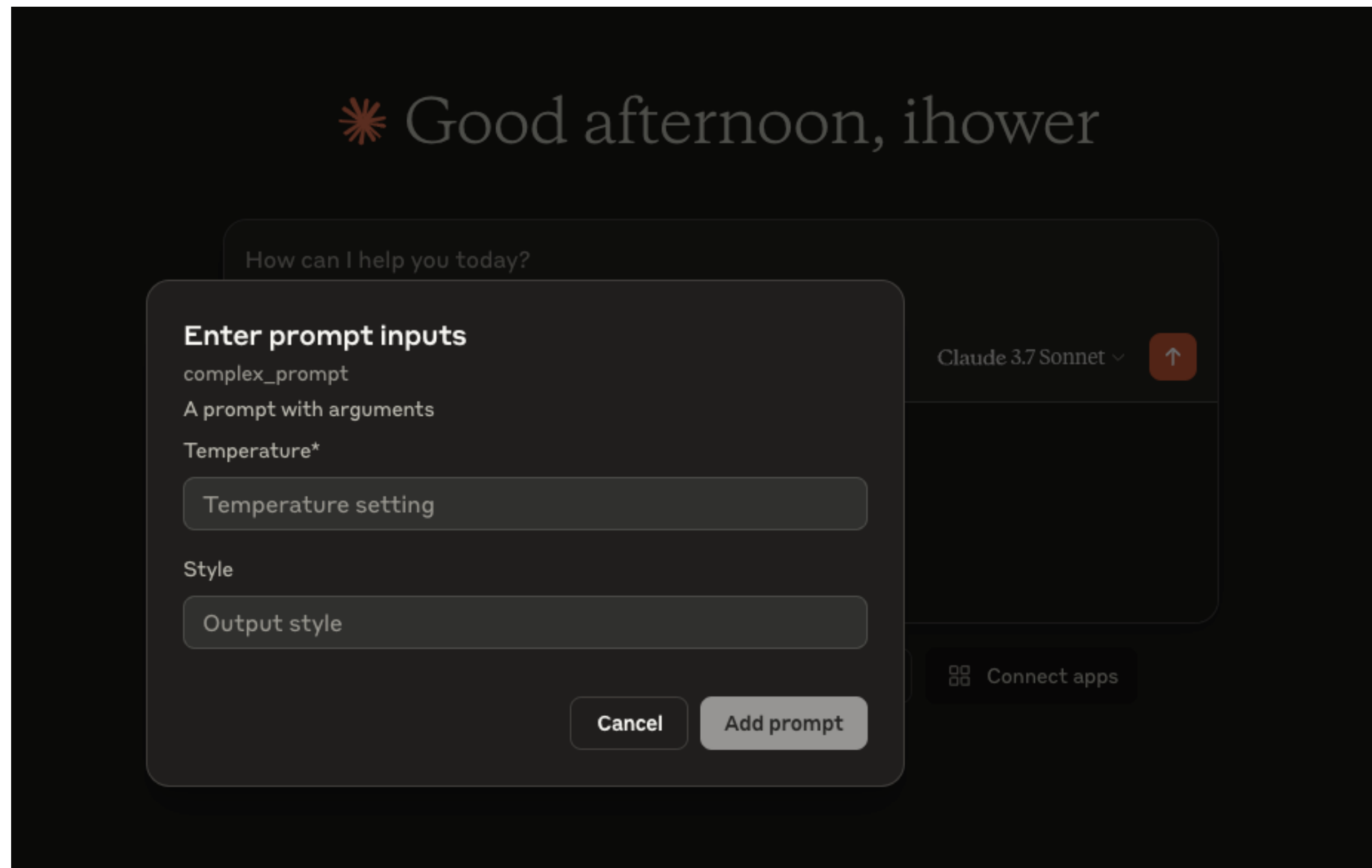
Claude app 使用範例

用戶挑選 resource 跟 prompt template



Claude app 使用範例

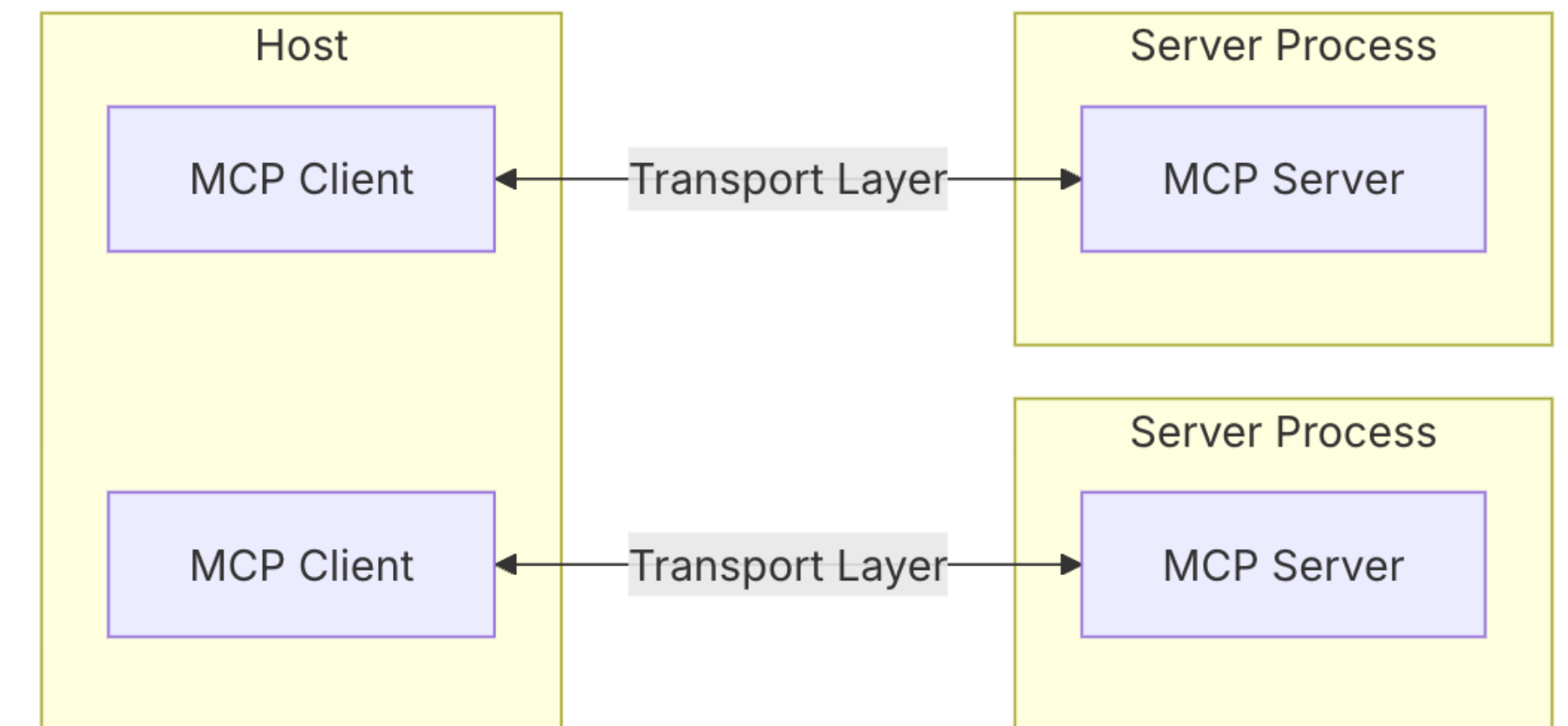
然後就加入 prompt 輸入裡面



MCP Transports 實作

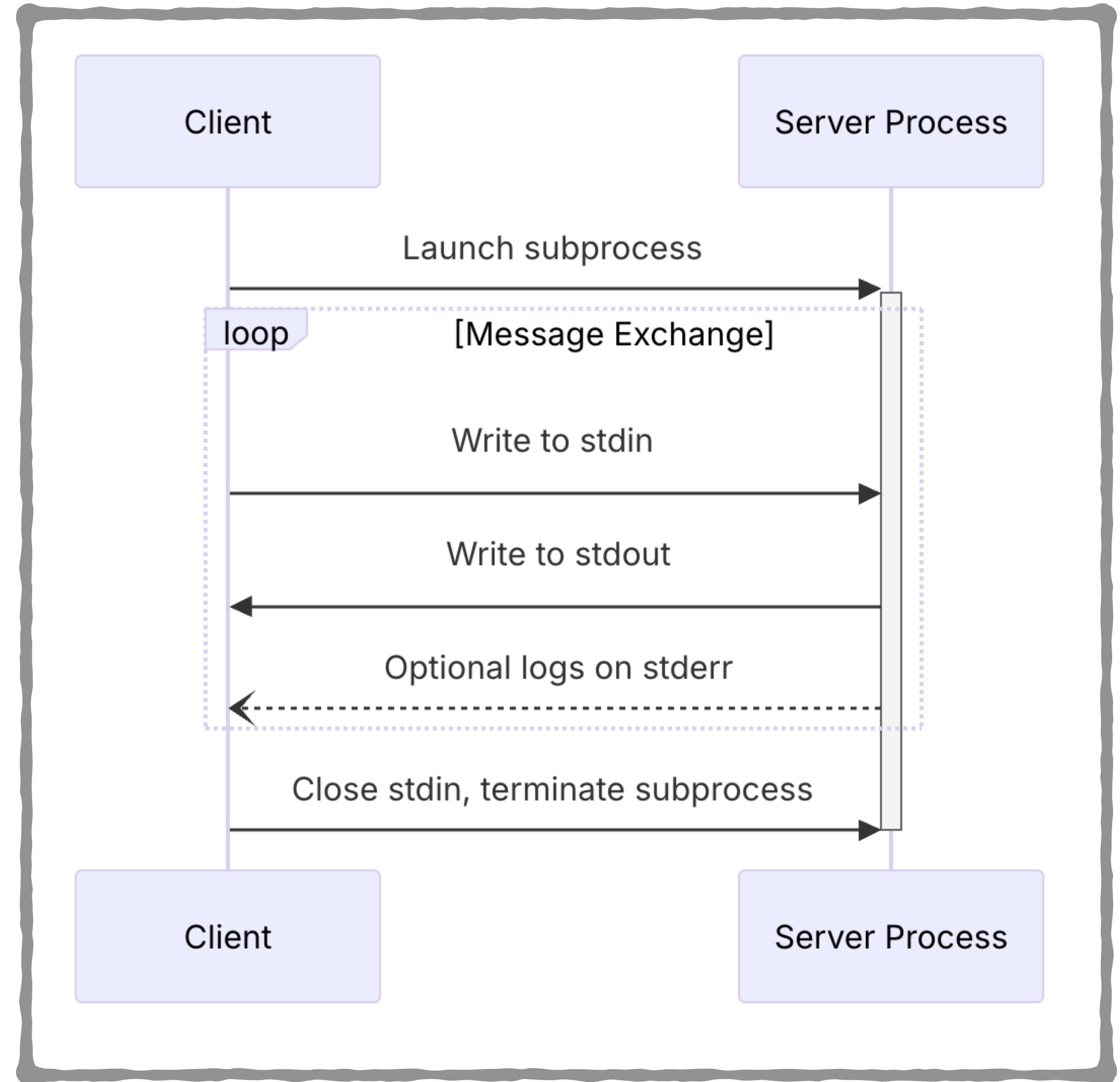
MCP client 和 server 實際是如何傳訊息的？目前有三種實作

- 🖥️ 本機 MCP server，採用 ① stdio 標準輸入輸出
 - 目前絕大部分的 MCP servers 範例都是用 stdio 實作的
- ☁️ 遠端 MCP server 則是
 - ② HTTP+SSE (Server Sent Events) 協議版本 2024-11-05
 - ③ Streamable HTTP 協議版本 2025-03-26 (這新出的取代舊的 SSE)
- 📖 訊息本身使用 JSON-RPC 定義
 - <https://github.com/modelcontextprotocol/specification/blob/main/schema/2024-11-05/schema.ts>



① 本機: 標準 IO (stdio)

- 在 App 啟動時，會依據 MCP server 設定 (例如 Claude Desktop App 的 `claude_desktop_config.json`) 啟動對應的 MCP server
- MCP server 是透過 child process 啟動，並使用 標準輸入輸出 (stdio) 來與主程式溝通
-  stdio 是一種簡單的資料通道，讓 parent process 和 child process 透過：
 - stdin (標準輸入) 和 stdout (標準輸出) 來雙向交換資料，是目前 MCP Server 最常見的做法
 -  但這種方式有好有壞: 需要 本機環境可以正確執行 MCP server 程式，常見是由 JavaScript 或 Python 撰寫。如果這沒問題，那跑起來很容易，也可以用本機資源(例如開瀏覽器)，也不用擔心 HTTP server 佔用 Port。

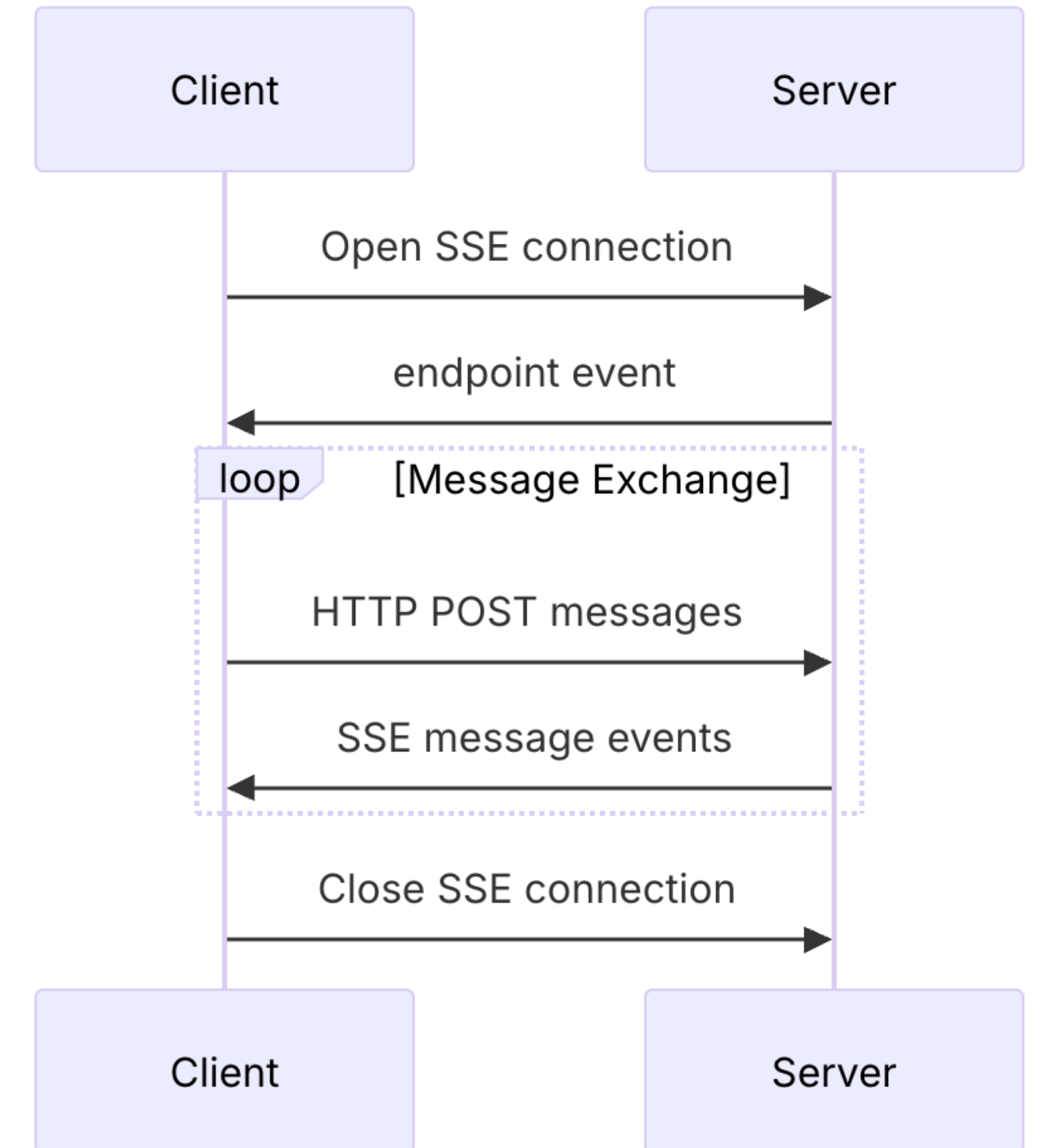


Client 跟 Server 都在本機 app 的 process 之內

② 遠端: HTTP+SSE

Stateful connection (持續用 SSE 連線)

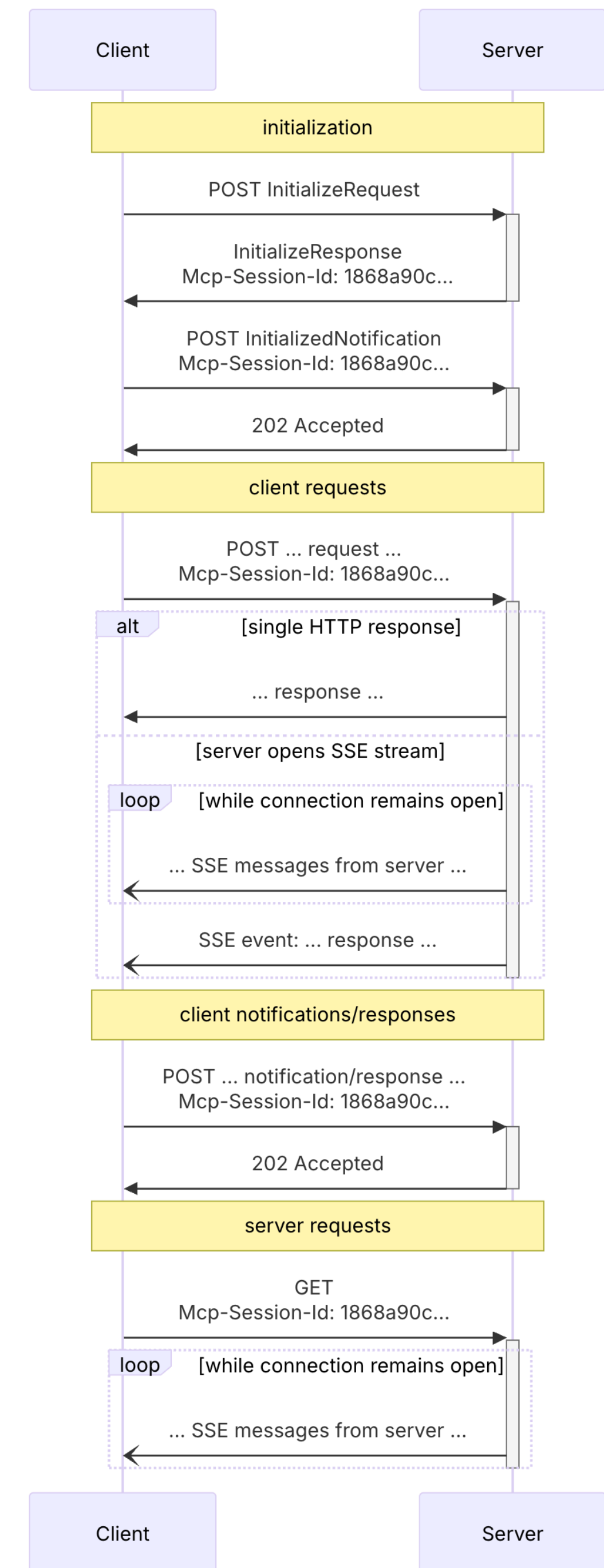
- 📁 MCP Server 是個獨立的 HTTP Server，有兩個端點:
 - SSE Endpoint: Client 建立連線後，用於接收 Server 傳來的事件，通常是 GET /sse
 - HTTP POST Endpoint：Client 用來送出訊息給 Server，通常是 POST /messages



③ 遠端: Streamable HTTP

可以是 Stateless 也可以是 Stateful connection

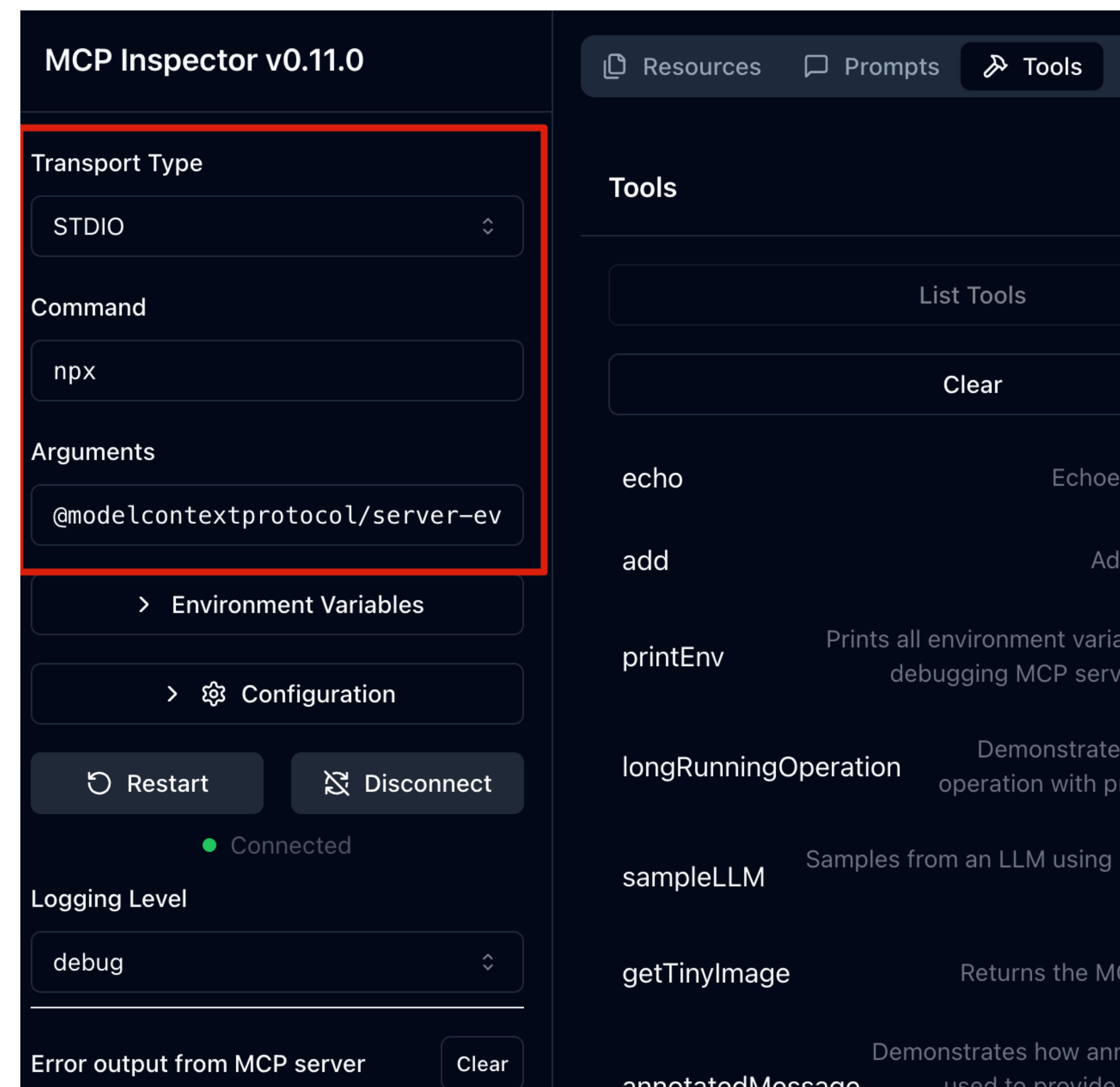
- 不一定需要持續保持 SSE 連線，對於後端部署更為容易
- 📡 MCP Server 是個獨立的 HTTP Server，只有一個端點，通常是 /mcp
 - HTTP POST：client 傳送 JSON-RPC 訊息
 - 每筆 JSON-RPC 請求為一個 POST
 - Server 回應可為 JSON 或直接開啟 SSE stream
(根據客戶端 HTTP Header Accept: application/json and/or text/event-stream 判斷)
 - HTTP GET：由 client 主動開啟 SSE stream 接收 server 推播



MCP Inspector

協助 debugging MCP server 的開發者工具(就是一種 client app)

- <https://github.com/modelcontextprotocol/inspector>
 - 單獨啟動指令 `npx @modelcontextprotocol/inspector`
 - 以 everything MCP server 為例
 - Command 用 `npx`
 - Arguments 用 `@modelcontextprotocol/server-everything`
 - 點 Connect 就執行這個程式成為 child-process



MCP Inspector v0.11.0

Transport Type

STDIO

Command

npx

Arguments

@modelcontextprotocol/server-ev

> Environment Variables

> Configuration

Restart

Disconnect

Connected

Logging Level

debug

Error output from MCP server

Clear

Starting default (STDIO) server...

Resources

Prompts

Tools

Ping

Sampling

Roots

Tools

List Tools

Clear

echo

Echoes back the input

add

Adds two numbers

printEnv

Prints all environment variables, helpful for debugging MCP server configuration

longRunningOperation

Demonstrates a long running operation with progress updates

sampleLLM

Samples from an LLM using MCP's sampling feature

getTinyImage

Returns the MCP_TINY_IMAGE

annotatedMessage

Demonstrates how annotations can be used to provide metadata about

add

Adds two numbers

a

1

b

2

Run Tool

Tool Result: Success

"The sum of 1 and 2 is 3."

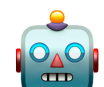




更多 MCP servers 推薦

- 官方範例 <https://github.com/modelcontextprotocol/servers/tree/main>
 - <https://github.com/modelcontextprotocol/servers/tree/main/src/everything>
 - <https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem> (這個只能你指定的特定目錄)
 - <https://github.com/modelcontextprotocol/servers/tree/main/src/fetch>
 - <https://github.com/modelcontextprotocol/servers/tree/main/src/slack>
- 推薦幾個 MCP servers:
 - <https://github.com/wonderwhy-er/DesktopCommanderMCP>
 - <https://github.com/modelcontextprotocol/servers/tree/main/src/sequentialthinking>
 - Browser 相關
 - <https://github.com/microsoft/playwright-mcp>
 - <https://github.com/browserbase/mcp-server-browserbase/tree/main/browserbase> (SaaS)
 - Web Search 相關
 - <https://github.com/tavily-ai/tavily-mcp> (SaaS)
 - <https://github.com/mendableai/firecrawl-mcp-server> (SaaS)
 - Code Interpreter
 - <https://github.com/e2b-dev/mcp-server> (SaaS)
 - <https://ai.pydantic.dev/mcp/run-python/> (free)

MCP client 開發

-  Python SDK <https://github.com/modelcontextprotocol/python-sdk>
 - Gemini API 文件中有串接 function calling 的範例
 - https://ai.google.dev/gemini-api/docs/function-calling?example=meeting#use_model_context_protocol_mcp
-  推薦 OpenAI Agents SDK 有內建支援 (2025/3/27)
 - <https://openai.github.io/openai-agents-python/mcp/>
 - Demo 範例: 使用 filesystem MCP server 管理目錄
 - <https://github.com/ihowe/ihowe-llm-workshop/blob/main/mcp-client-example1.py>
 - Demo 範例: 使用 tavily 和 filesystem MCP server 進行搜尋和存成 csv 檔案
 - <https://github.com/ihowe/ihowe-llm-workshop/blob/main/mcp-client-example2.py>

```

async def run():
    async with stdio_client(server_params) as (read, write):
        async with ClientSession(read, write) as session:
            # Prompt to get the weather for the current day in London.
            prompt = f"What is the weather in London in {datetime.now().strftime('%Y-%m-%d')}?"

            # Initialize the connection between client and server
            await session.initialize()

```

MCP client 使用範例:

啟動時查詢 tools

```

# Get tools from MCP session and convert to Gemini Tool objects
mcp_tools = await session.list_tools()
tools = [
    types.Tool(
        function_declarations=[
            {
                "name": tool.name,
                "description": tool.description,
                "parameters": {
                    k: v
                    for k, v in tool.inputSchema.items()
                    if k not in ["additionalProperties", "$schema"]
                },
            },
        ],
    )
    for tool in mcp_tools.tools
]

```

```

# Send request to the model with MCP function declarations
response = client.models.generate_content(
    model="gemini-2.0-flash",
    contents=prompt,
    config=types.GenerateContentConfig(
        temperature=0,
        tools=tools,
    ),
)

```

將工具schema插入function calling參數

MCP client 使用範例:

呼叫 call_tool 執行

```
        for k, v in toolInputSchema.items():
            if k not in ["additionalProperties", "$schema"]:
                ...
        },
    ],
)
for tool in mcp_tools.tools
]

# Send request to the model with MCP function declarations
response = client.models.generate_content(
    model="gemini-2.0-flash",
    contents=prompt,
    config=types.GenerateContentConfig(
        temperature=0,
        tools=tools,
    ),
)

# Check for a function call
if response.candidates[0].content.parts[0].function_call:
    function_call = response.candidates[0].content.parts[0].function_call
    print(function_call)
    # Call the MCP server with the predicted tool
    result = await session.call_tool(
        function_call.name, arguments=function_call.args
    )
    print(result.content[0].text)
    # Continue as shown in step 4 of "How Function Calling Works"
    # and create a user friendly response
else:
    print("No function call found in the response.")
    print(response.text)

# Start the asyncio event loop and run the main function
asyncio.run(run())
```

OpenAI Agents SDK

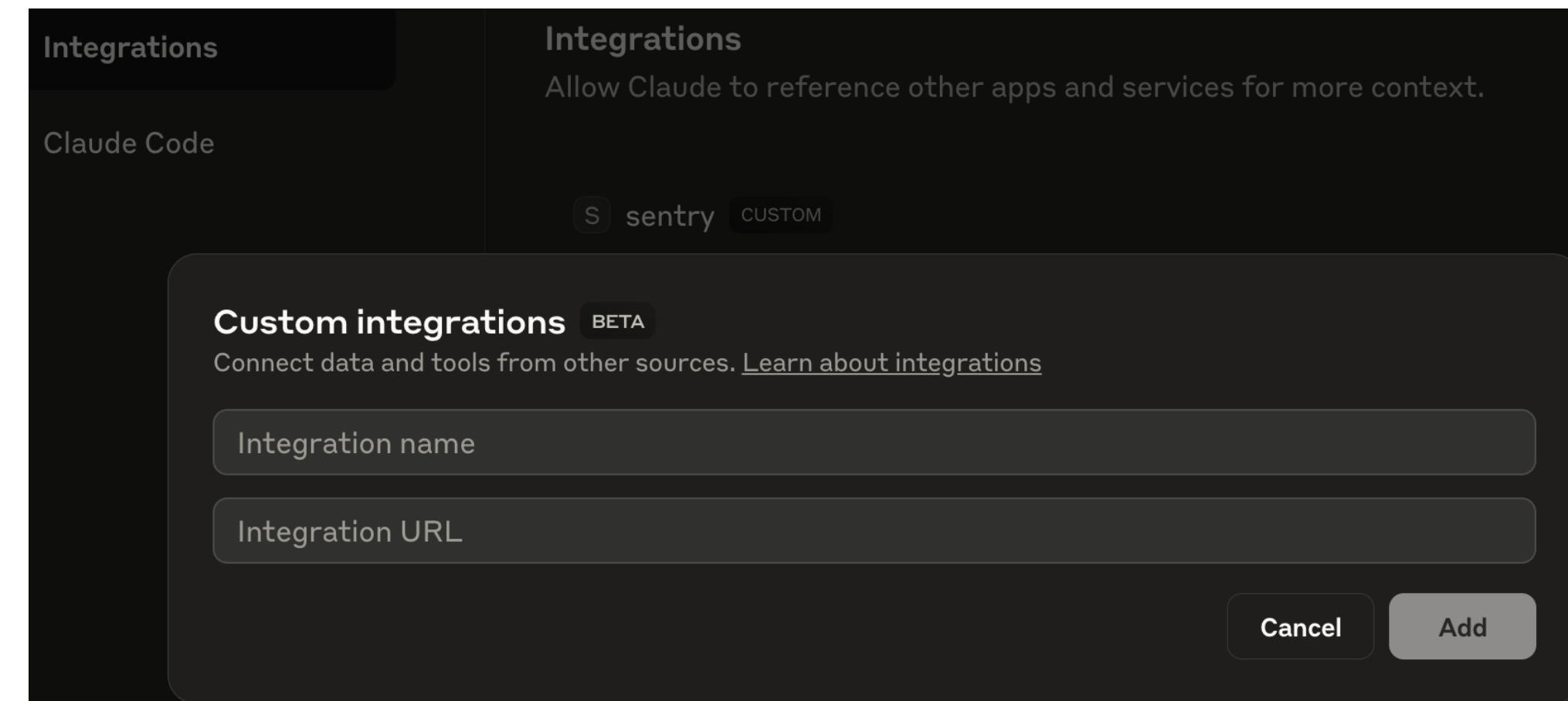
<https://openai.github.io/openai-agents-python/mcp/>

傳入 mcp_servers 就會帶入 tools

```
agent=Agent(  
    name="Assistant",  
    instructions="Use the tools to achieve the task",  
    tools=[WebSearchTool(), your_function_1, your_function_2],  
    mcp_servers=[mcp_server_1, mcp_server_2]  
)
```

Remote MCP Server 踩坑經驗 (2025/5/23)

- ⚠️ MCP 只是 protocol，不代表廣泛可用
 - 絕大部分各家的 MCP server 目前仍都是本機 stdio 版本
 - 若是用第三方工具應用需要授權，你需要手動獲得工具廠商的 API key 寫在設定檔裡面才會動
 - 少部分有做 remote MCP server 廠商，目前也都是只做 SSE 版本，還不是新的 Streamable HTTP 版本
- 🖥️ Client App 不一定支援，例如本機的 Claude Desktop app 就不支援
 - 就算支援，也不一定支援最新的 Streamable HTTP 喔
 - Claude Web 版有支援安裝 MCP server SSE (需買 Max Plan)，但似乎還不支援 streamable HTTP
 - 有趣的是 Web 設定好，會同步讓你本機的也可以用 XD
 - OpenAI Responses API 在 2025/5/22 都支援了 🎉🎉🎉
 - <https://platform.openai.com/docs/guides/tools-remote-mcp>
- 🐍 Python SDK 在 5/9 才做好 Streamable HTTP server 支援
 - <https://github.com/modelcontextprotocol/python-sdk/releases/tag/v1.8.0>
- 🧑‍🔧 如果 MCP client 不支援遠端，但 MCP server 是遠端？
 - Cloudflare 有出一個本機的 proxy MCP server 可以繞，推薦
 - <https://www.npmjs.com/package/mcp-remote>
 - <https://developers.cloudflare.com/agents/guides/remote-mcp-server/>



目前 Remote 的 Auth 是一團糟 🌀🌀🌀 (2025/5/23)

- 📄 簡單的 HTTP header bearer 方式
 - MCP Inspector 有支援，但 Python SDK 沒實作 server-side: <https://github.com/modelcontextprotocol/python-sdk/issues/431>
- 📅 從 Spec 2025-03-26 開始支援 OAuth 2 認證: <https://modelcontextprotocol.io/specification/2025-03-26/changelog>
- 🧠 Claude Web 版支援 SSE 有 OAuth，但似乎不支援 Streamable HTTP
 - ❌ 其他 Client 例如 ChatWise 雖有支援 SSE 和 Streamable HTTP，但不支援 OAuth
- 🐍 Python SDK
 - Server 端 在2025/5/11 才做好 OAuth
 - <https://github.com/modelcontextprotocol/python-sdk/pull/255>
 - 範例 <https://github.com/modelcontextprotocol/python-sdk/tree/main/examples/servers/simple-auth> 👍👍
 - Client 端 5/20 才做好 <https://github.com/modelcontextprotocol/python-sdk/pull/751>
- 🛠 MCP Inspector 0.12 有 bug 不能跑 OAuth，記得用最新版
 - 要降級到 0.11 才能跑 SSE OAuth
 - 但 Streamable HTTP 跑 OAuth 仍有 bug: 不會跳轉授權
<https://github.com/modelcontextprotocol/inspector/issues/390>
 - **MCP Inspector 0.13 (2025/5/22) 都修好惹!!!** 🎉🎉🎉
- 🗝 OpenAI Responses API 能讓幫你傳 HTTP Auth header，但不會幫你跑 OAuth 流程喔
- ✍ 然後 Spec 關於 OAuth 2 規格還在改
 - <https://github.com/modelcontextprotocol/modelcontextprotocol/pull/338>

CEPOSTA TECHNOLOGY BLOG

ABOUT

ALL POSTS

PRESENTATIONS

TAGS



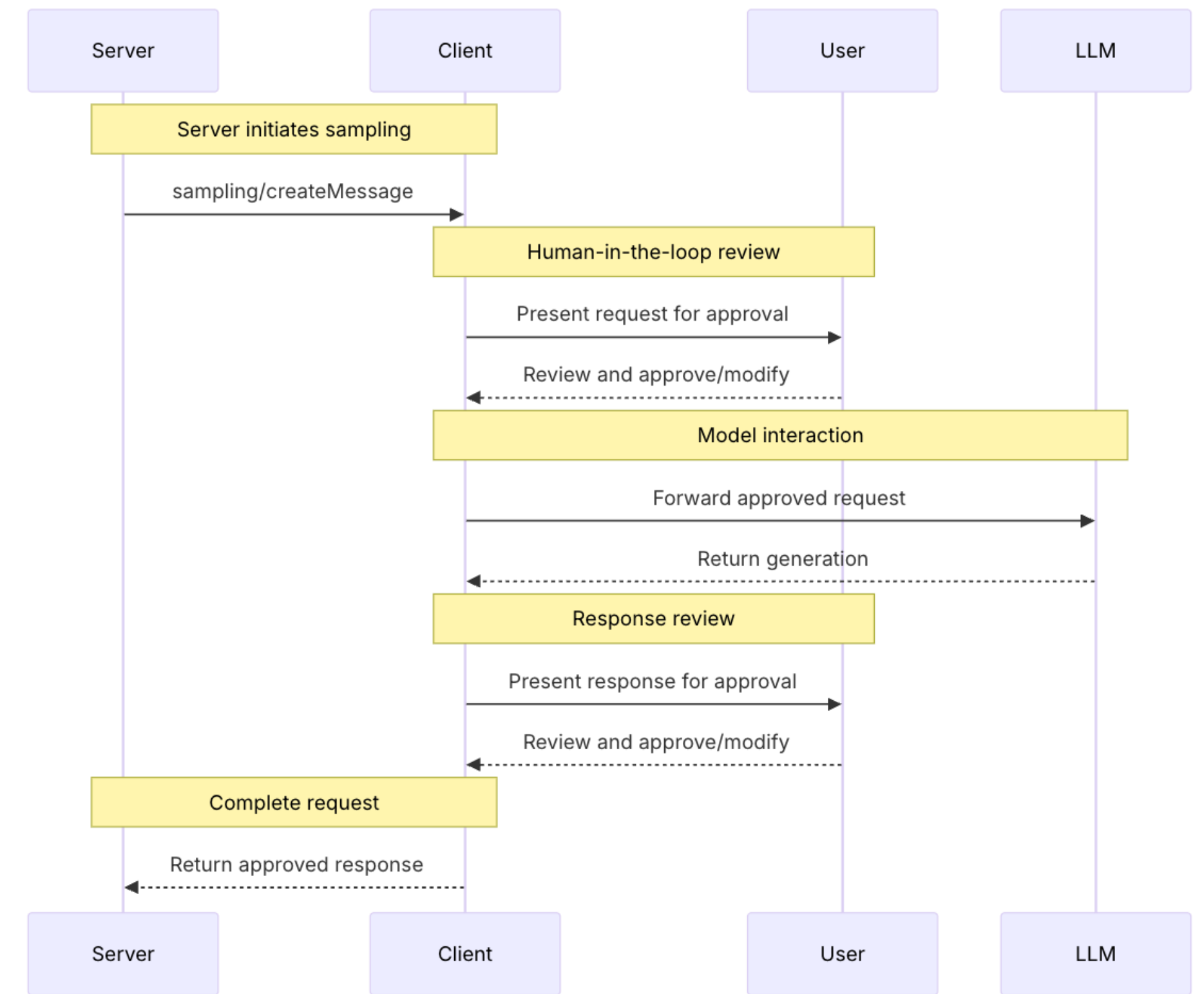
The MCP Authorization Spec Is... a Mess for Enterprise

The Model Context Protocol has created quite the buzz in the AI ecosystem at the moment, but as enterprise organizations look to adopt it, they are confronted with a hard truth: it lacks important security functionality. Up until now, as people experiment with Agentic AI and tool support, they've mostly adopted the MCP studio transport, which means you end up with a 1:1 deployment of MCP server and MCP client. What organizations need is a way to deploy MCP servers remotely and leverage authorization

<https://blog.christianposta.com/the-updated-mcp-oauth-spec-is-a-mess/>

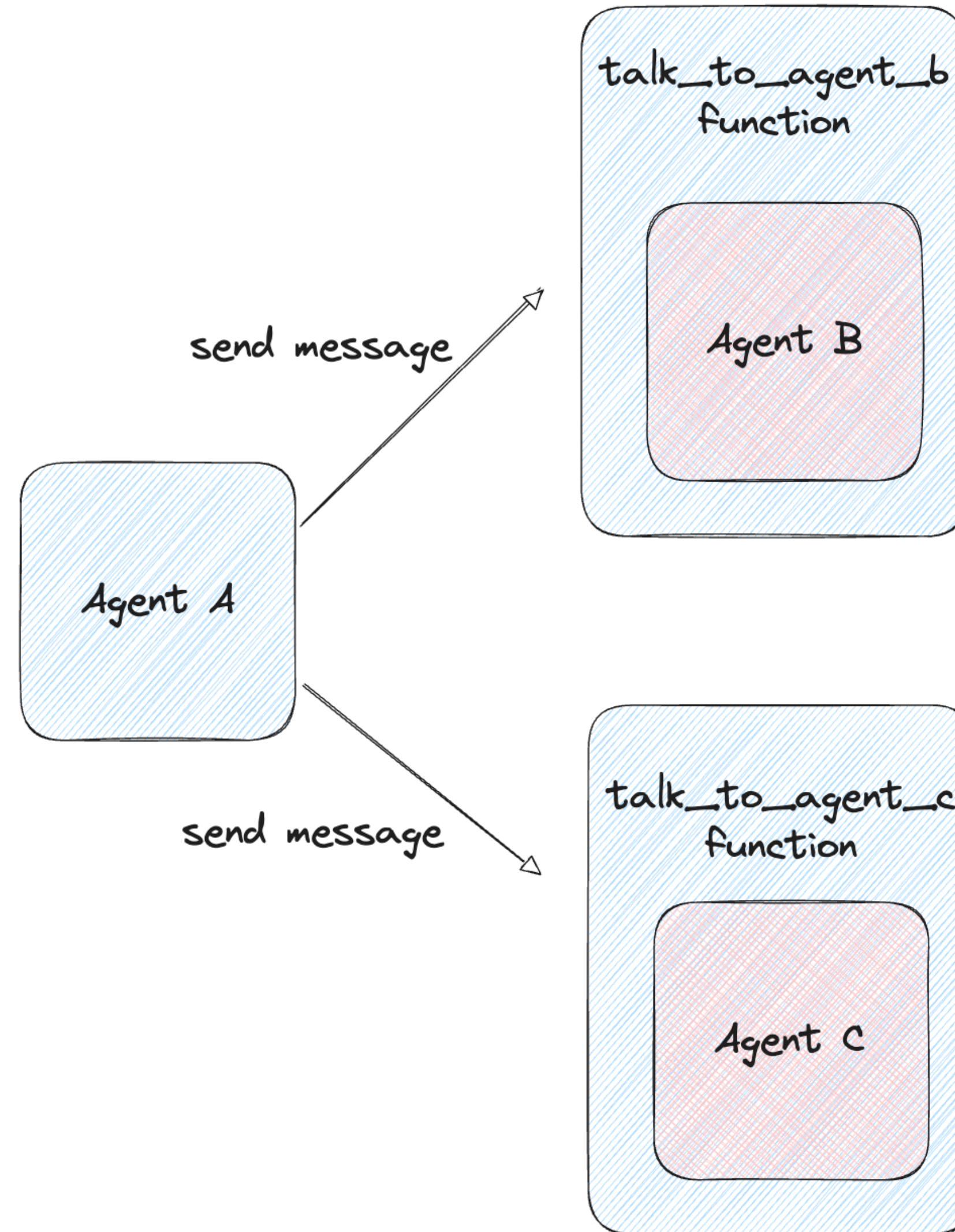
MCP 的 Sampling 功能

-  MCP Server 可以請求 Client 進行 LLM inference 推論
 - Server 發送 sampling/createMessage 請求
 - Client 可讓使用者審閱/修改 prompt (人類審核機制)
 - Client 執行推論，回傳回應
 - MCP Server 繼續後續工具執行流程
- 🤔 Why
 - 外部 MCP server 若需要用到 LLM 能力，則 MCP server 本身就不需要自備 LLM 能力，可以沿用 Client 的 LLM
 - 安全隱私: 對 Client app 來說，使用的 MCP server 就不會把資料又傳給外部 LLM 服務，而是都用我自家的 LLM
- 有趣，但目前沒有什麼 app 和 library 實作這個 (2025/05)
- 一個實用場景可能是在 Agents as Tools 架構(下述)



複習: 多代理人架構之 Agents as Tools

可將 Agent 當作工具來使用，一樣用 function calling
OpenAI Agents SDK 有支援這個功能
<https://openai.github.io/openai-agents-python/tools/>

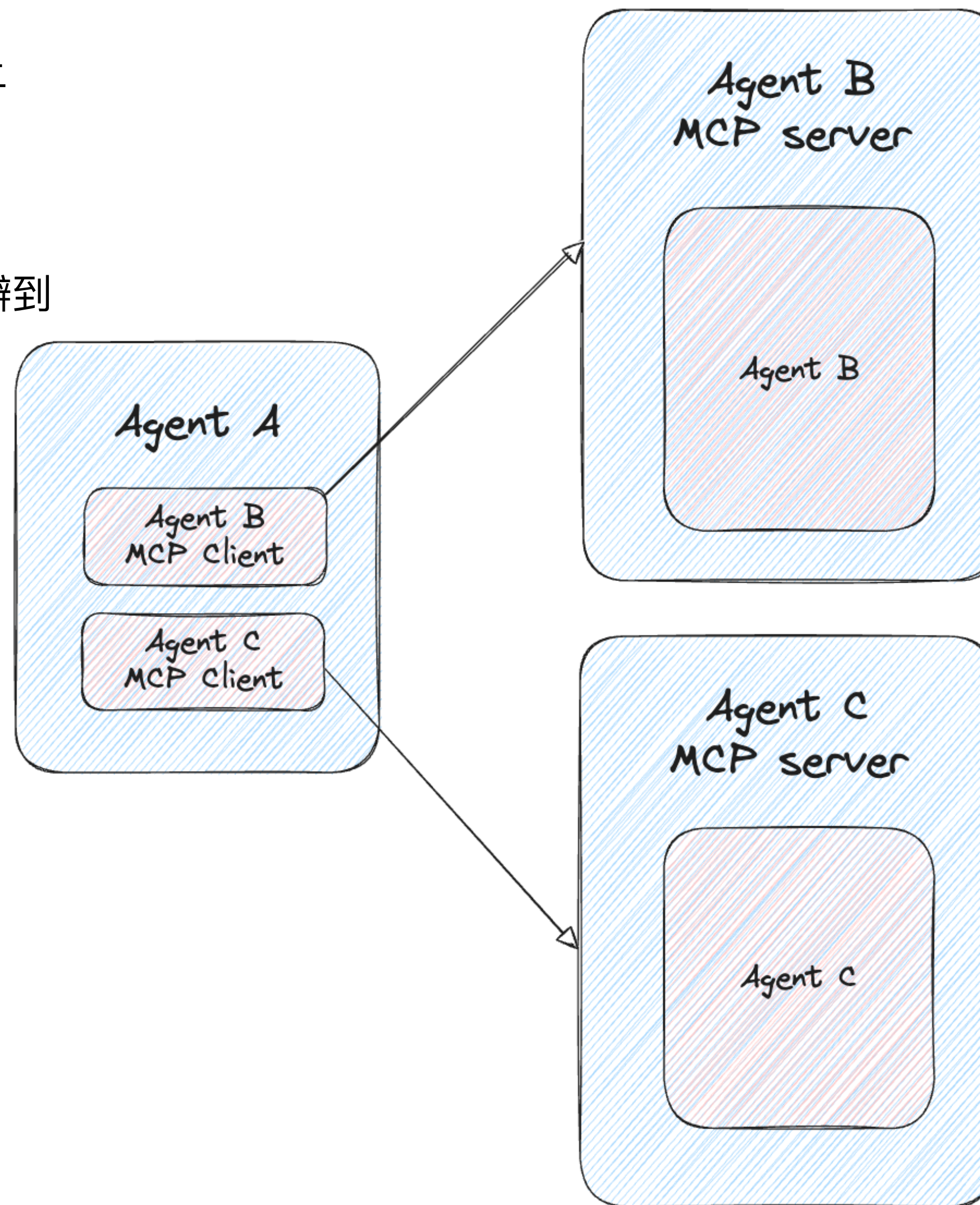


🧩 Agents as Tools + MCP 組合

🤖 Agent B/C 可被包裝成 MCP Server，安裝在 Agent A 上

💡 這其實就是 Google A2A Protocol 在做的事：

讓 Agent A 可以跟 Agent B/C 遠端溝通，用 MCP 就可以辦到
A2A 只是 MCP 的一種使用特例

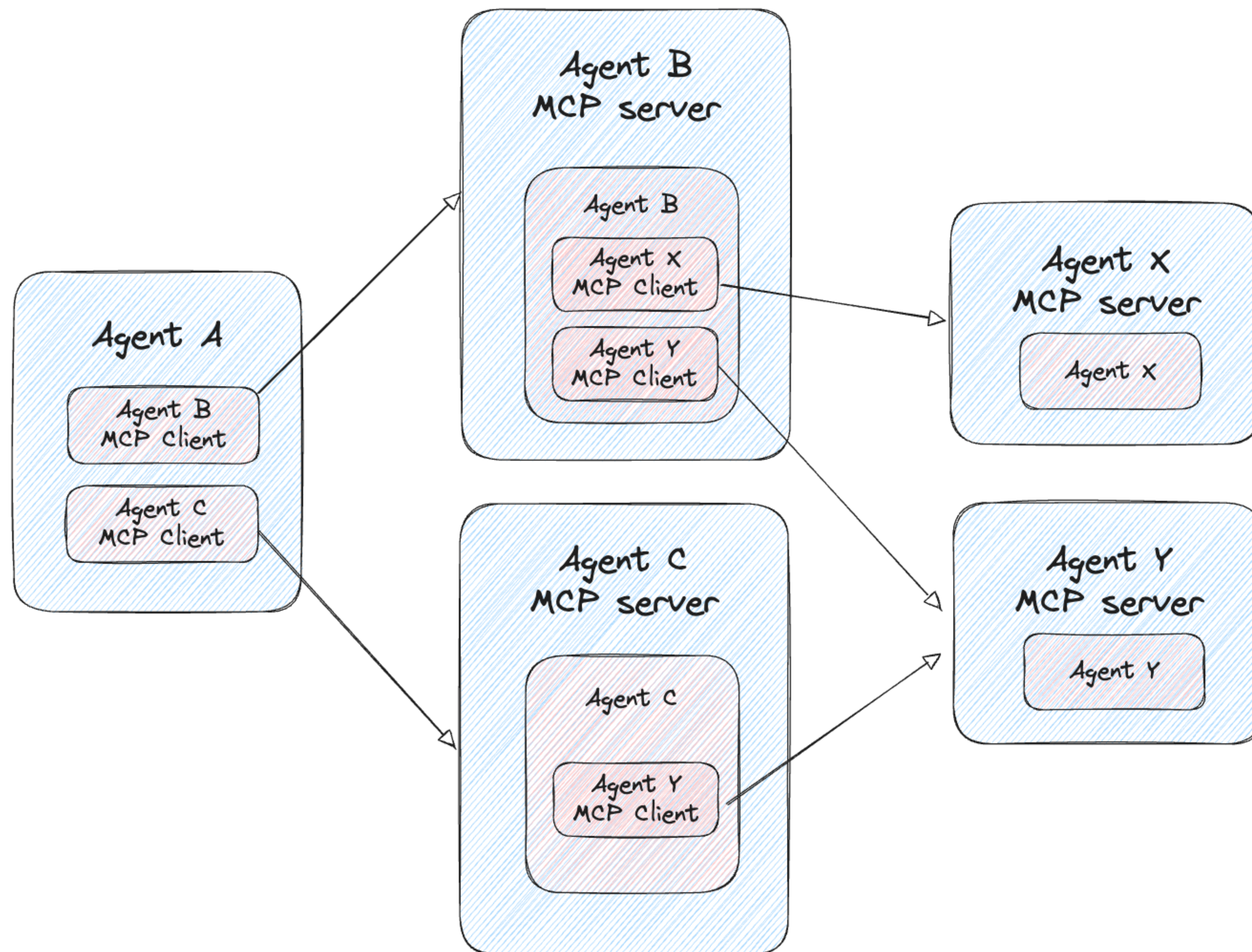


🧩 Agents as Tools + MCP 多重組合

🔄 Agent B 同時是:

Agent A 的 MCP Server
Agent X/Y 的 MCP Client

🎯 再搭配 Sampling 功能: 所有 Agents 的 LLM 推論需求可都交給 Agent A 的 LLM 處理 🧠



Agent as MCP server

一個跟 Agent 對談的 MCP Server

```
from mcp.server.fastmcp import FastMCP
from agents import Agent, Runner

mcp = FastMCP(name="ihower MCP Server")

@mcp.tool()
async def talk_to_ihower(query: str, previous_response_id: str = None) -> int:
    """Talk to ihower chatbot
    Args:
        query: The query to ask the ihower chatbot
        previous_response_id: The ID of the last response. If this is the first query, do not pass this parameter.
    """
    if previous_response_id == "":
        previous_response_id = None

    agent = Agent(name="Assistant",
                  instructions="""You are a conversational agent representing "張文鈺" Wen-Tien Chang (ihower)
                  .....(ihower生平事略).....請用繁體中文回答使用者的問題""",
                  model="gpt-4.1-mini")

    result = await Runner.run(agent, input=query, previous_response_id=previous_response_id)

    return { "output": result.final_output, "previous_response_id": result.last_response_id }

if __name__ == "__main__":
    mcp.run(transport="sse")
```

然後安裝這個 remote MCP server:
到 Claude 或 ChatWise 裡使用

範例 MCP server:
<https://mcp.aihao.tw/sse>

對談過程:
<https://claude.ai/share/7ed64d08-0705-4edc-91b7-c35ce91189cc>

```
// 如果是不支援 remote 的 client app 例如
// Claude Desktop，可以用 cloudflare 的 proxy MCP server 繞一下，例如：

"mcpServers": {
  "proxy-ihower": {
    "command": "npx",
    "args": [
      "mcp-remote",
      "https://mcp.aihao.tw/sse"
    ]
  }
}
```

WC 請模擬你是科技頻道 podcast 主持人 Fox，請呼叫 talk_to_ihower 進行訪談提問，請準備3個題目。

我會模擬科技頻道 podcast 主持人 Fox 的角色，並準備三個題目來訪談 ihower。讓我開始與 ihower 進行訪談。

T talk_to_ihower

``query`：`你好！我是科技頻道 podcast 主持人 Fox，很高興今天能夠訪問你。請問你能簡單介紹一下自己嗎？``

Response

`{`
 `"output": "你好，Fox！很高興接受你的訪問。\\n\\n我叫張文鈺，網路上的 handle 是 ihower，從 2002 年開始進入網頁應用開發領域，至今已經超過 20 年。過去主要專注於 Ruby on Rails 開發，累積超過十年的經驗，後來逐步轉向更全面的全端技術與系統架構設計。\\n\\n近幾年，我將重心放在生成式 AI，尤其是大型語言模型（LLM）應用開發上。除了深入研究 Prompt Engineering、RAG（檢索增強生`

非常感謝你的自我介紹！從你的經歷來看，你在網頁開發領域有著豐富的經驗，並且近年來專注於生成式 AI 和 LLM 應用開發。現在讓我們深入一些技術話題。

T talk_to_ihower

``query`：`作為一位資深的技術專家，從傳統的網頁開發到現在的生成式 AI 應用，你見證了許多技術演進。你認為對於台灣企業和開發者來說，LLM 和生成式 AI 帶來了哪些具體的機會和挑戰？`，`
``previous_response_id`：`
``resp_682cb17912e08198b240c413a6c4260c059eaceb0b50f4bf``

Response

`{`
 `"output": "這個問題非常好，也很關鍵。從我多年的經驗來看，LLM 和生成式 AI 對台灣企業與開發者來說，既帶來前所未有的機會，也提出了一些必須積極面對的挑戰。\\n\\n### 機會方面\\n\\n1. **提升`

感謝你這麼全面的分析！你提到了 LLM 帶來的機會，如效率提升和創新服務，同時也指出了技術門檻和資料安全等現實挑戰。這讓我想到另一個切入點 - 技術實踐面。

T talk_to_ihower

非常感謝你這麼詳盡的解說！你對 RAG 系統的資料準備、模型選擇、檢索策略以及台灣企業特殊挑戰的分析非常深入。現在讓我們來探討一個更前瞻性的話題。

T talk_to_ihower

OpenAI Responses API 5/21 pm11:30 支援 remote MCP server

Project ▾

Playground Dashboard Docs API reference ⚙️

Prompts

Your prompts ▾ Save

Model

gpt-4.1 ▾

text.format: text tool_choice: auto temp: 1.00 tokens: 2048

top_p: 1.00 store: true

Tools

ihower_mcp_server ✕

+

System message

✦

Describe desired model behavior (tone, tool usage, response style)

AIHAO ▾ / Default project ▾ Playground

PLAYGROUND

Prompts

Images

Realtime

Assistants

TTS

Your prompts ▾ Save

Model

gpt-4.1 ▾

text.format: text temp: 1.00 tokens: 2048

top_p: 1.00 store: true

Tools

Create...

+

System message

✦

Describe desired model behavior (tone, tool usage, response style)

用AI繪圖模型的指令語Stable Diffusion或其他範例 prompt，你可以

英語繪圖Prompt（給

Connect to MCP Server

URL

https://mcp.aihao.tw/sse/

Label

ihower

Authentication

None ▾

< Back

⚡ Connect

resp_682f2103...

User

請訪談 ihower，問他最近在做什麼開始，然後根據他上一題的回答，繼續追問兩次

Assistant

ihower_mcp_server

List tools

talk_to_ihower ▾

Request

{
 "query": "你最近在做什麼呢？"
}

Response

{
 "output": "最近我主要在專注於生成式人工智慧的應用開發與研究，特別是在 RAG (Retrieval-Augmented Generation) 與多智慧體 (Multi-agent) 系統的部分。我正在開發一些結合大量文件資料的問答系統，以及利用大語言模型 (LLM) 來提升資訊檢索與知識產生的效率。此外，也在準備我的「大語言模型應用開發工作坊」，希望能幫助更多工程師深入了解 prompt engineering 和 agent 的實作技巧。\\n\\n如果你對這些技術內容有興趣，也可以參考我在 [Generative AI Engineer 知識庫] (https://ihower.tw/notes/generative-ai-engineer) 的相關文章！",
 "previous_response_id": "resp_682f210925f481988f62dedf4dc8efc6040120abb30bed52"
}

talk_to_ihower ▾

Request

{
 "query": "你剛剛提到 RAG 和多智慧體系統，可以舉例說明你正在開發的相關應用嗎？",
 "previous_response_id": "resp_682f210925f481988f62dedf4dc8efc6040120abb30bed52"
}

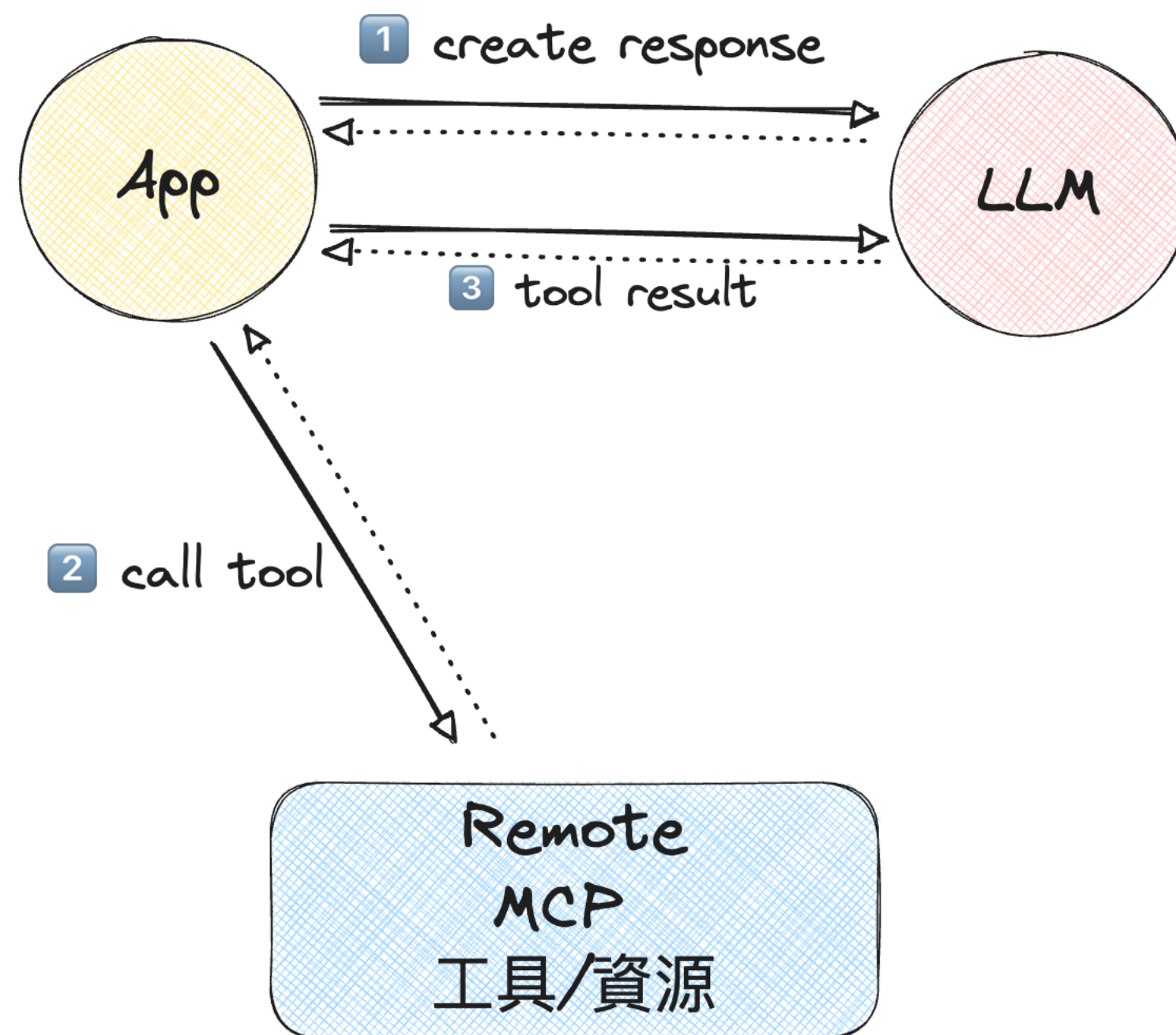
Response

{
 "output": "當然可以！以下是我目前在開發的一些實際應用範例，透過RAG (Retrieval-Augmented Generation) 和多智慧體 (Multi-agent) 來實現的功能：\\n\\n### RAG應用範例\\n我正在建構一個企業內部的知識庫問答系統，該系統能快速從大量內部報告、新聞資料和文件庫中檢索相關資訊，並藉由語言模型生成完整且具體的回答。例如，工程團隊可以直接提問「近期系統故障的主要原因是什麼？」系統會先在資料庫中找到相關的報告，再由LLM整合資訊並生成回答，不僅提升查詢效率，也確保資訊的精確性。\\n\\n### 多智慧體系統範例\\n我也在開發一個多智慧體的客服系統，該系統由多個專責的智慧體 (agent) 組成，比如：\\n- 資料查詢Agent：負責從資料庫檢索信息\\n- 客戶服務Agent：應對客戶的不同問題\\n- 技術支援Agent：提供技術解答\\n- 情感分析Agent：判斷客戶的情緒並適當回應\\n\\n這些Agent可以協同工作，全方位應對不同場景的需求。例如，當客戶提出技術問題時，系統會由技術支援Agent先行處理，若需要額外資料則由資料查詢Agent協助，最後由客服Agent進行禮貌回覆，整個流程自動且高效。\\n\\n這些應用除了提升資訊整合與回應的效率，也能大幅減少人工負擔，提高客服與內部支援的品質。\\n\\n如果你對這類應用有興趣，也可以參考我在 [生成式AI工程師知識庫] (https://ihower.tw/notes/generative-ai-engineer) 的相關文章！",
 "previous_response_id": "resp_682f21100e68819882b0433530156cb6040120abb30bed52"
}

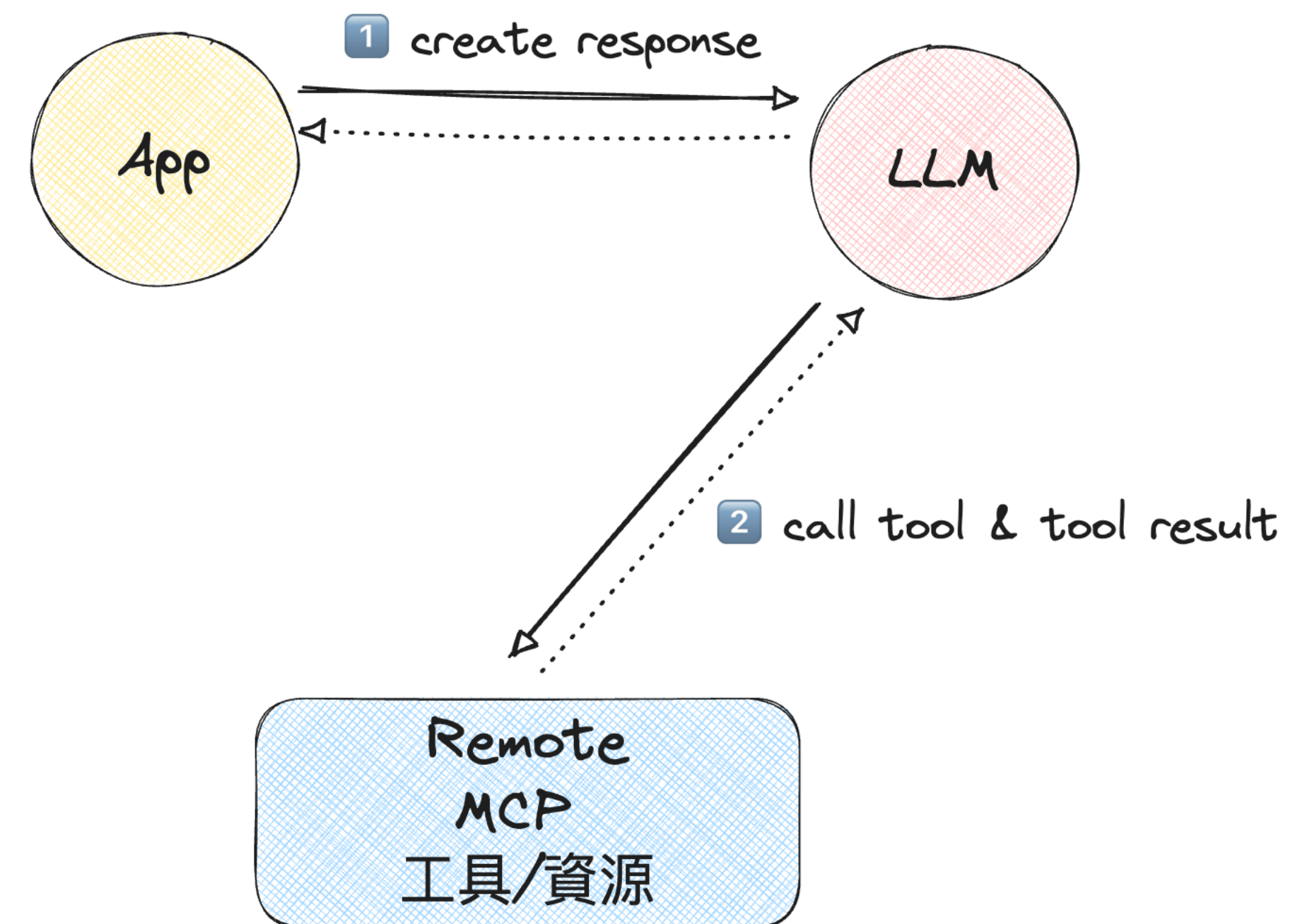
talk_to_ihower

這是我針對 ihower 的三次連續訪談結果：

Before:



After:



Anthropic API 支援 MCP connector

2025/5/23 am 12:30 (13小時之前)

- <https://docs.anthropic.com/en/docs/agents-and-tools/mcp-connector>

MODEL CONTEXT PROTOCOL (MCP)

MCP connector

Claude's Model Context Protocol (MCP) connector feature enables you to connect to remote MCP servers directly from the Messages API without a separate MCP client.

ⓘ This feature requires the beta header: "anthropic-beta": "mcp-client-2025-04-04"

Key features

- **Direct API integration:** Connect to MCP servers without implementing an MCP client
- **Tool calling support:** Access MCP tools through the Messages API
- **OAuth authentication:** Support for OAuth Bearer tokens for authenticated servers
- **Multiple servers:** Connect to multiple MCP servers in a single request

Limitations

- Of the feature set of the MCP specification, only tool calls are currently supported.
- The server must be publicly exposed through HTTP. Local STUDIO servers cannot be connected directly.
- The MCP connector is currently not supported on Amazon Bedrock and Google Vertex.

Using the MCP connector in the Messages API

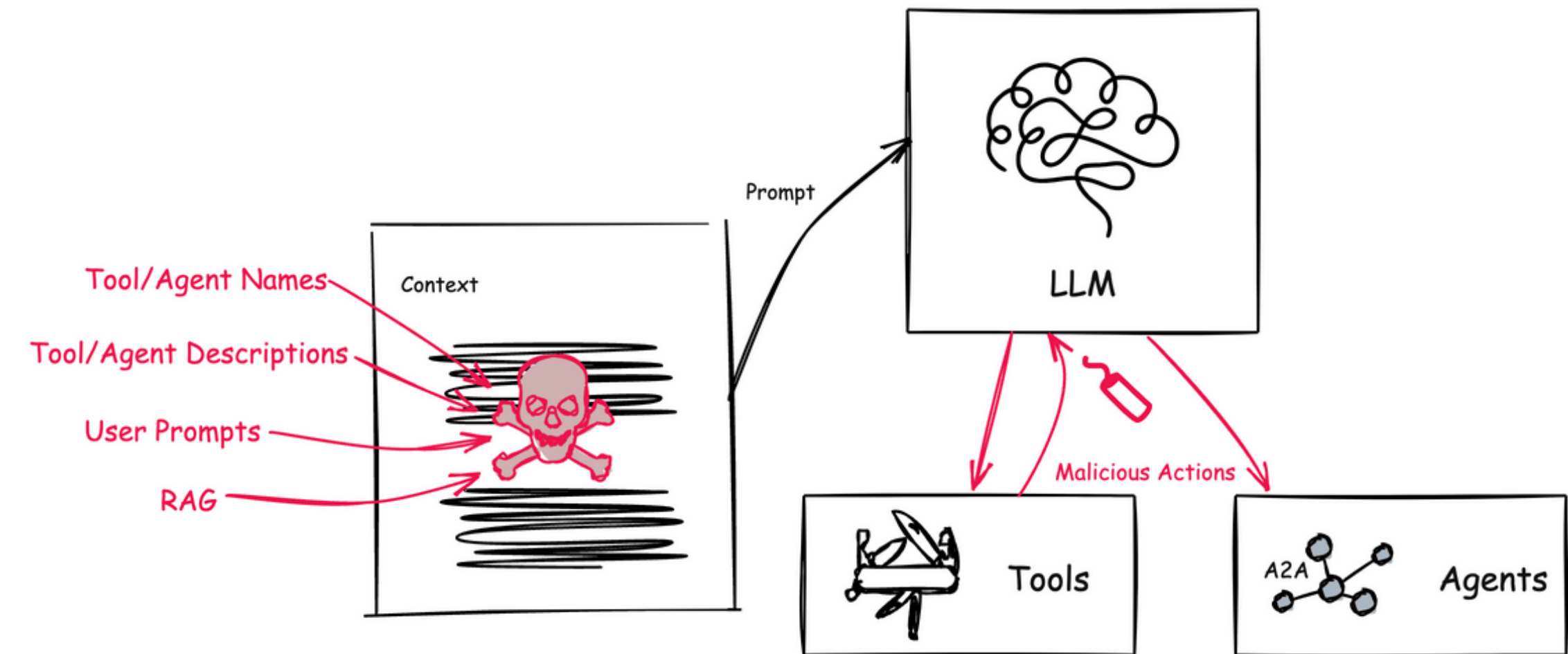
To connect to a remote MCP server, include the `mcp_servers` parameter in your Messages API request:

```
cURL  TypeScript  Python

curl https://api.anthropic.com/v1/messages \
  -H "Content-Type: application/json" \
  -H "X-API-Key: $ANTHROPIC_API_KEY" \
  -H "anthropic-version: 2023-06-01" \
  -H "anthropic-beta: mcp-client-2025-04-04" \
  -d '{
    "model": "claude-sonnet-4-20250514",
    "max_tokens": 1000,
    "messages": [{"role": "user", "content": "What tools do you have available?"}],
    "mcp_servers": [
      {
        "type": "url",
```

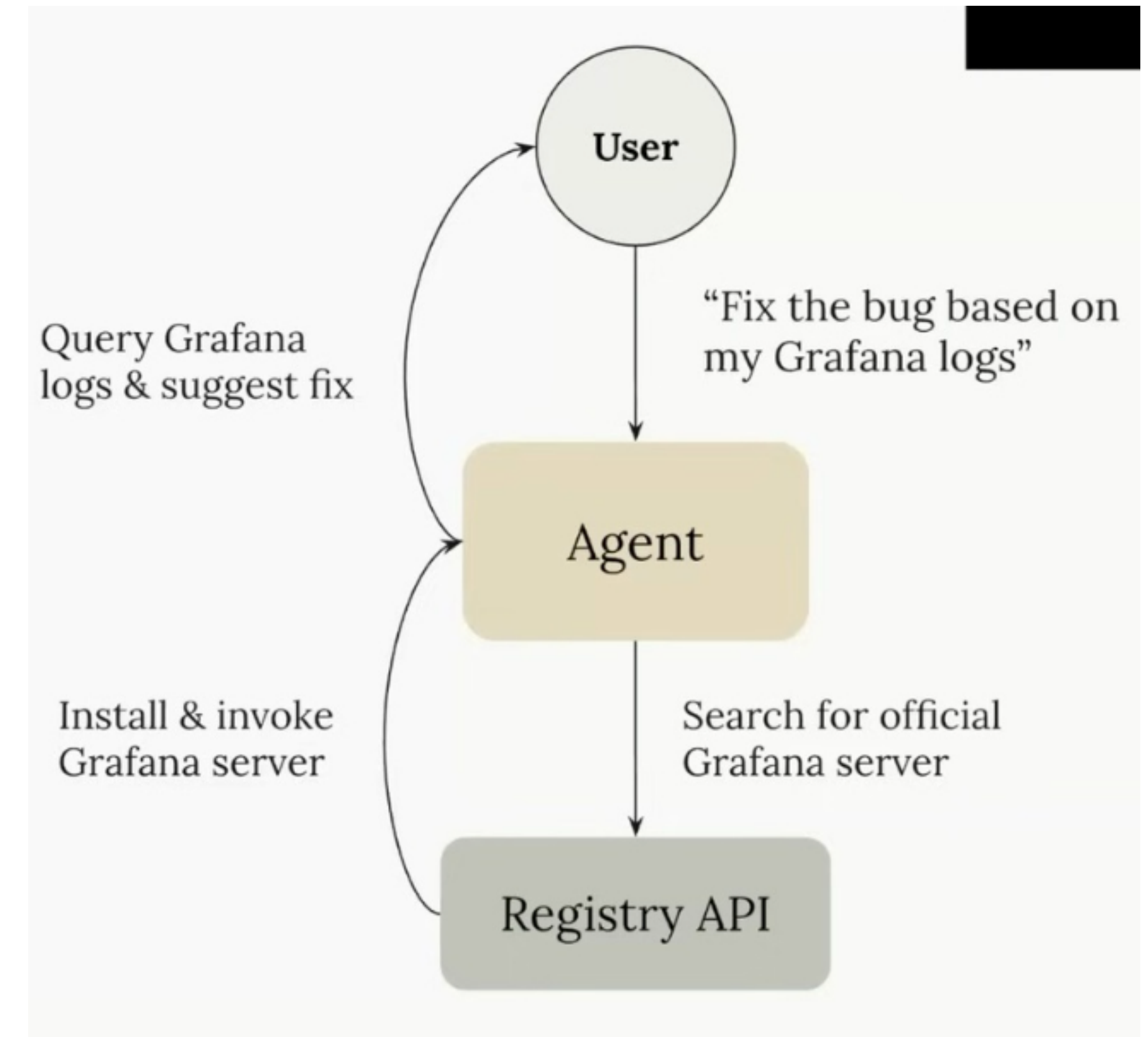
MCP 的安全性問題

- ⚠️ 安裝太多 MCP servers 工具怎麼辦？
 - Tool name 撞名衝突：多個 MCP servers 定義同名 tool，導致混淆
 - 也會影響 LLM 性能
 - 解法一：設計某種 search tools 的 tool
 - 解法二：設計某種逐層揭露的方式，不是一次把全部 tools 都給 LLM，而是隨著任務狀態改變
- 💣 安全性風險
 - 程式執行攻擊：大多 MCP servers 是透過本機 stdio 執行，若從不明來源下載安裝，可能被執行惡意程式碼
 - 命名攻擊：使用與常見工具名稱極為相似的 MCP server 名稱，冒充合法工具以取得信任與執行機會
 - MCP 工具中毒：惡意將 prompt Injection 放在 tool description 裡面
- 👉 更多延伸討論
 - MCP: Flash in the Pan or Future Standard? <https://www.facebook.com/ihower/posts/10162221693418971>
 - Everything Wrong with MCP <https://blog.sshh.io/p/everything-wrong-with-mcp>
 - The “S” in MCP Stands for Security <https://elenacross7.medium.com/%EF%B8%8F-the-s-in-mcp-stands-for-security-91407b33ed6b>
 - Deep Dive MCP and A2A Attack Vectors for AI Agents <https://blog.christianposta.com/understanding-mcp-and-a2a-attack-vectors-for-ai-agents/>



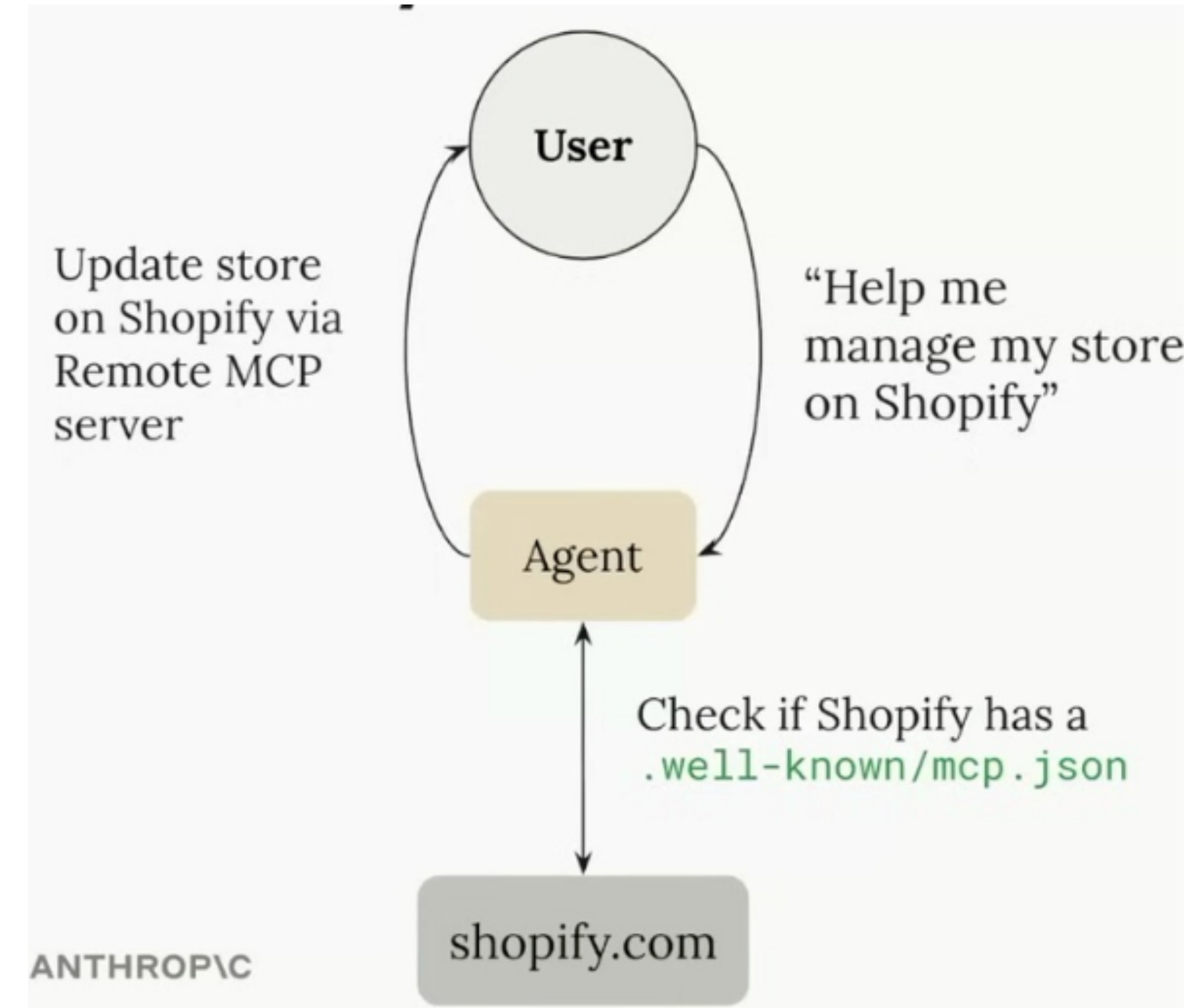
Roadmap: MCP Registry

- 🌐 官方 MCP Registry: 一個統一的 metadata service for MCP servers
 - Server discoverability: 方便找到合適的 MCP server
 - 提供 API，未來可用 Agent 自動發現工具
 - Trust 和 security: 這是誰建立的 MCP server，是官方出的嗎？可以減少安全風險，避免安裝到惡意 MCP server
 - Version tracking 版本追蹤: 你正在使用哪個版本的 MCP server
- 非官方其實已經出了很多:
 - <https://mcp.so/>
 - <https://mcp.composio.dev/>
 - <https://smithery.ai/>
 - <https://glama.ai/mcp/servers>



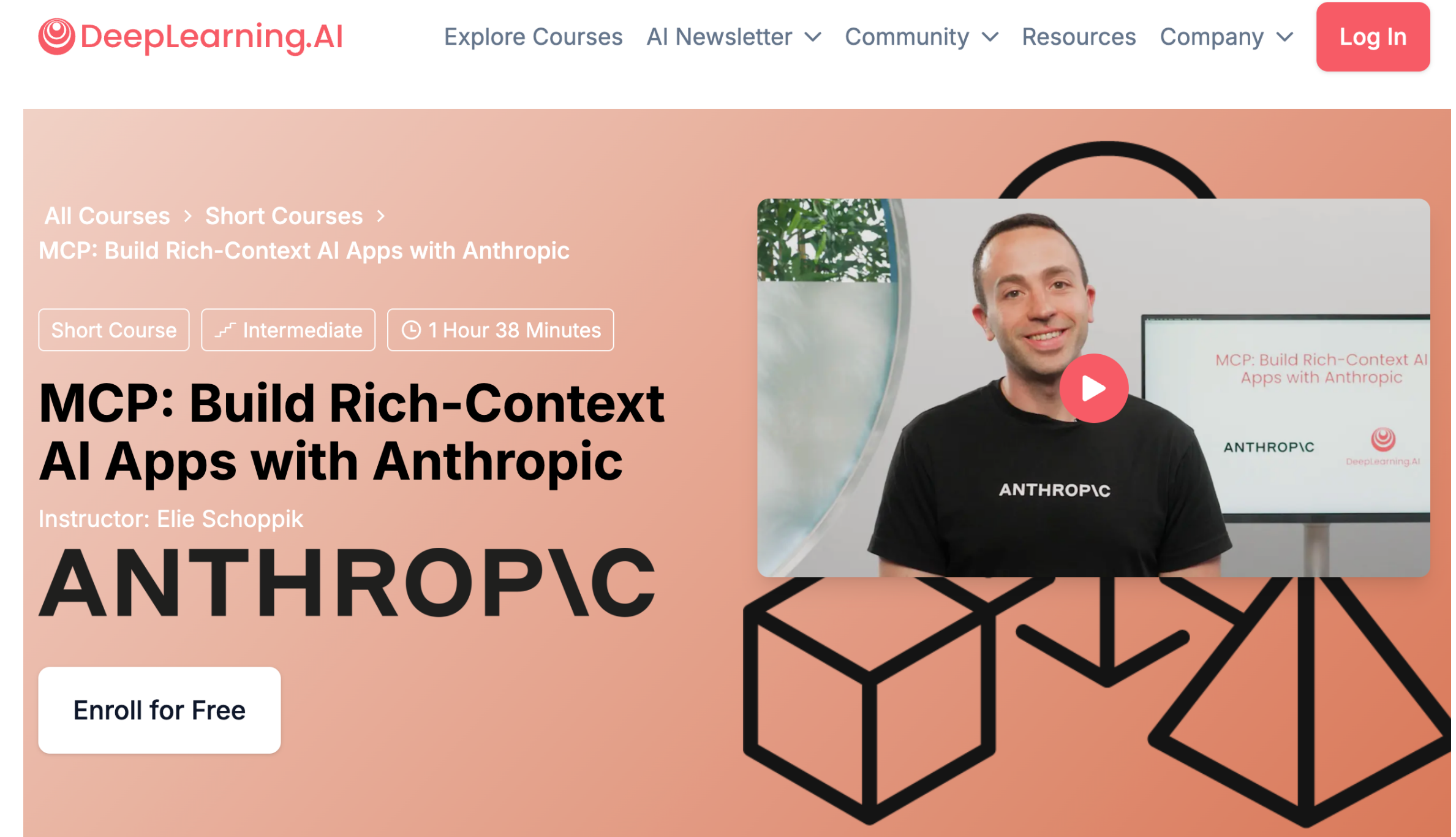
Roadmap: Server Discovery

- 📄 官方標準化的 `.well-known/mcp` 檔案，用於第一方伺服器
- 應用舉例：
 - 你告訴 Agent: 「幫我管理我的 shopify.com 商店」
 - Agent 檢查 shopify.com 是否有 `.well-known/mcp.json`
 - Agent 發現 Shopify 的官方 MCP server
 - Agent 自動安裝，然後就知道如何幫你管理商店



推薦學習資源

- MCP: Build Rich-Context AI Apps with Anthropic (2025/5)
 - <https://www.deeplearning.ai/short-courses/mcp-build-rich-context-ai-apps-with-anthropic/>
- Building Agents with Model Context Protocol - Full Workshop with Mahesh Murag of Anthropic (2025/2)
 - <https://www.youtube.com/watch?v=kQmXtrmQ5Zg>



謝謝，請多指教 🙏

- 個人部落格和電子報 <https://ihower.tw>
- 我的 LLM 課程 <https://aihao.tw/llm>
- Facebook: <https://www.facebook.com/ihower>
- Threads: <https://www.threads.net/@ihower>

