Generative AI
AI Agent、Agentic AI

生成式 AI
AI 代理、代理人 AI

生成式 AI（Generative AI）如 ChatGPT、Midjourney 爆紅，靠著訓練資料學會「創作」文字、圖片甚至程式碼。代理型 AI（Agentic AI）。更進一步，強調的是「自主行動」、「自己決策」，不等你下指令，它就能幫你完成複雜任務。一邊是「被動回應」的內容創作者，一邊是「主動行動」的數位代理人。生成式 AI 和代理型 AI，兩種思維，兩種邏輯，導向完全不同的未來。

What is Generative AI?

Generative AI is a type of artificial intelligence that is designed to create something new. It doesn't just repeat information – it learns patterns from massive amounts of data and then generate fresh content-text, images, music and even video.

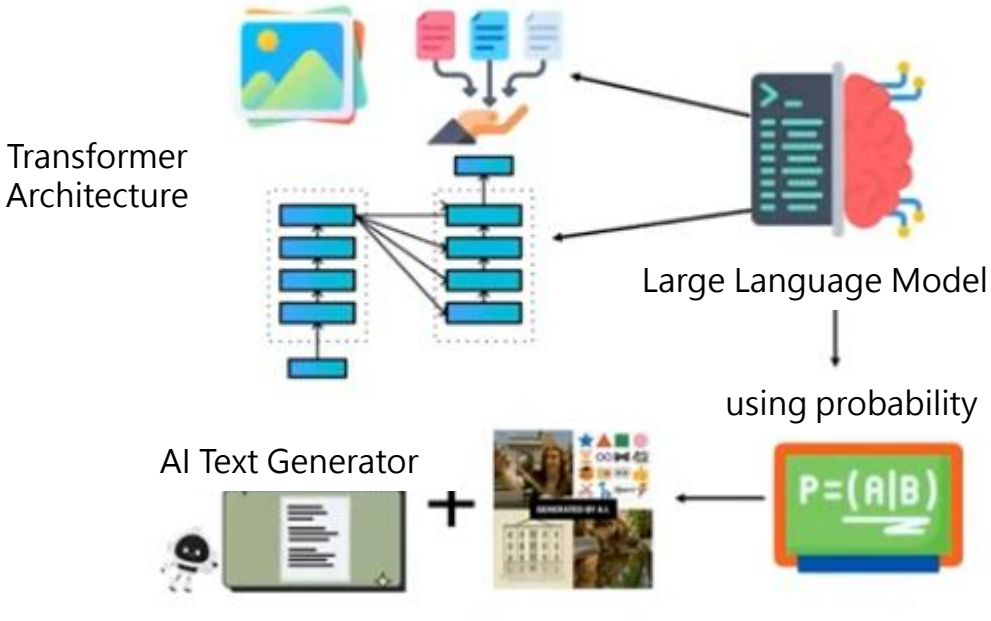
Generative AI

ChatGPT

DALL·E

GitHub Copilot

How Generative AI works?



The model uses something called transformer architecture which basically breaks down text into small units called tokens and learns the relationship between them and using probability it can predict the next word sentence and even image pixel based on patterns it has seen before.

AI Agent

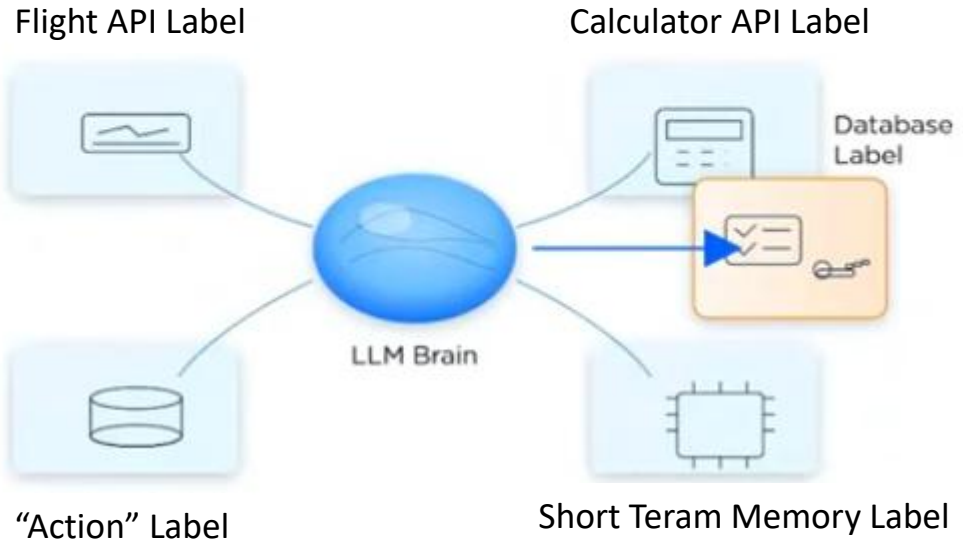
What is AI Agent?

An AI agent is a program that can not only generate answers but also take action. It uses generative AI as its brain, but it's connected to external tools, APUs, or memory (Data Base).

- AI Agents are **goal-oriented** and **action-focused** rather than purely generative.
- AI agents typically follow a **perception-reasoning-action-learning** cycle.

How it Works

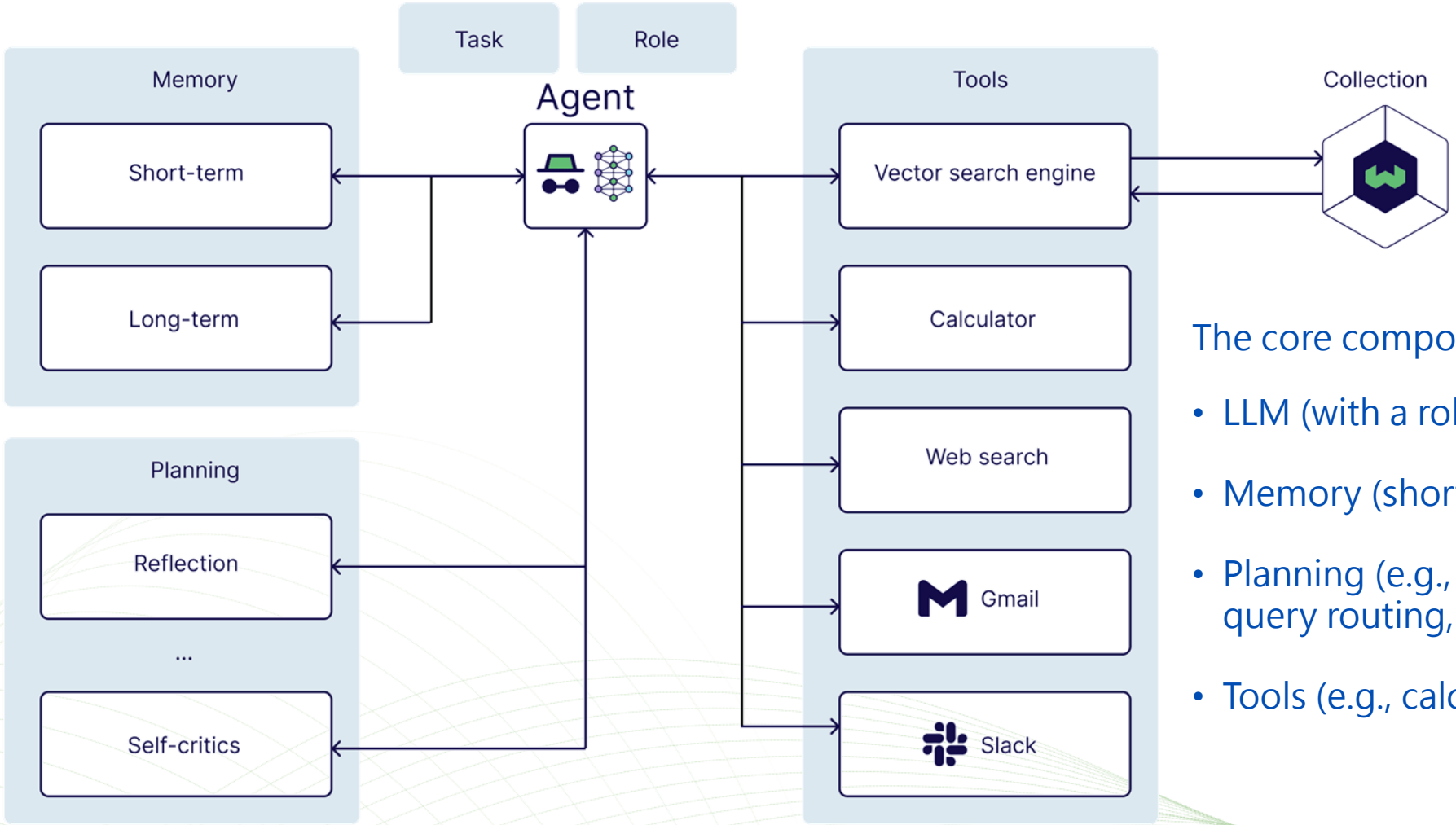
Model Structure



The LLM Brain isn' t working alone. Around it, the AI is connected to tools and APIs that can call whenever needed. The short memory can remember when it just did and what it needs to do next. It can actually take action using that tools.

Jonathan Chen

AI Agent



The core components of an AI agent are:

- LLM (with a role and a task)
- Memory (short-term and long-term)
- Planning (e.g., reflection, self-critics, query routing, etc.)
- Tools (e.g., calculator, web search, etc.)

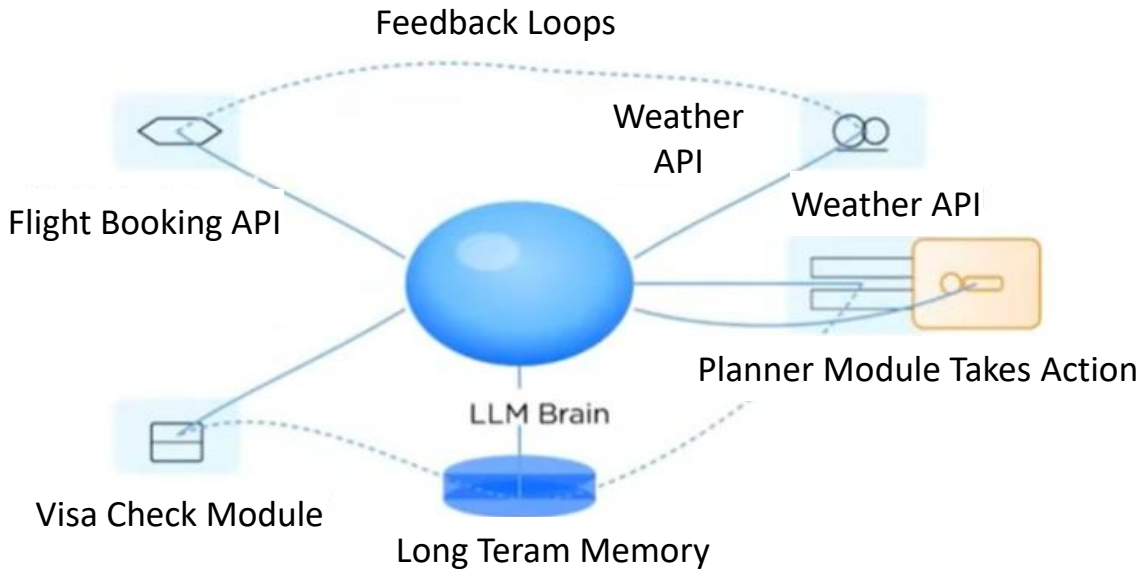
What is Agentic AI?

Agentic AI is an artificial intelligence system made up of independent agents that can take initiative and perform sequences of actions to complete tasks by reasoning, learning, and adapting to changing circumstances.

- Agentic AI is an artificial intelligence system that can accomplish a specific goal with limited supervision. It consists of AI agents - machine learning models that mimic human decision-making to solve problems in real time.

How Agentic AI works

Model Structure



Agent AI has Long Term Memory so it remembers when it left off and keeps track of the progress and if something changes like flight gets cancelled the system uses feedback loops to adjust the plan and find another option instead of starting from the scratch

Agentic AI – AI Agents

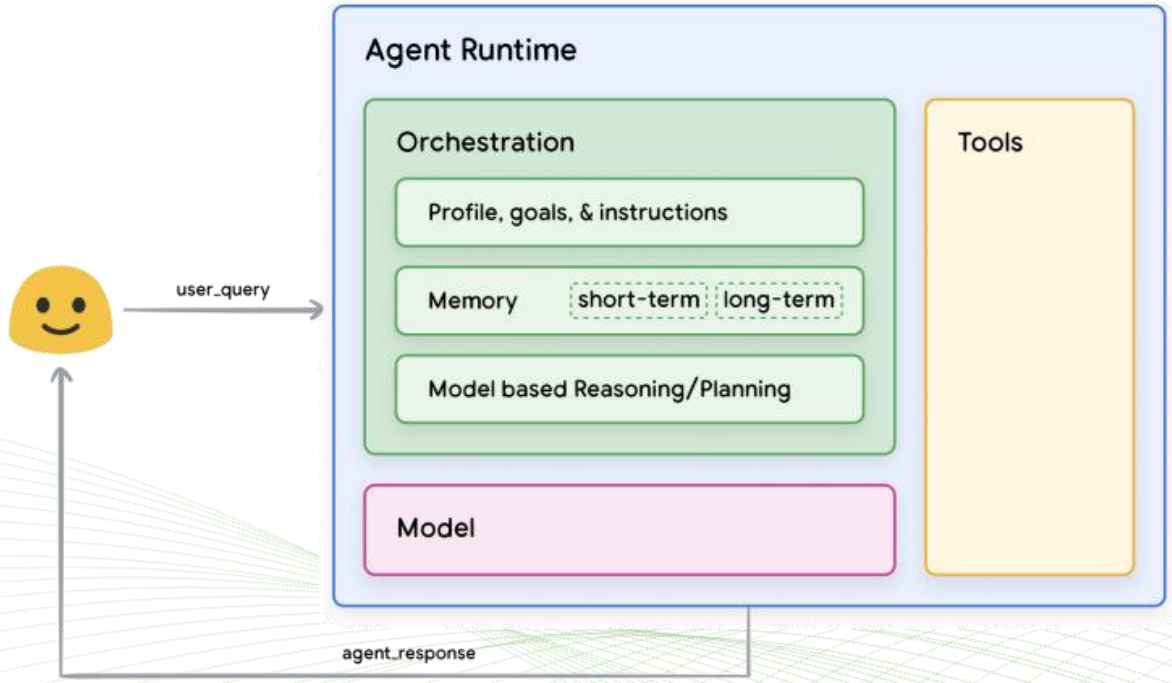
What are AI agents?

An AI agent is a self-directed autonomous application that harnesses large language model (LLM) reasoning, tool usage, and context-awareness from numerous data sources to carry out tasks.

Agents can think and act independently without outside intervention. Agents can think through chain-of-thought, plan, execute (Reason + Act= ReAct), and refine their actions as needed. Businesses are looking into adopting these autonomous agents for applications such as customer service automation, supply-chain optimization, and content generation.

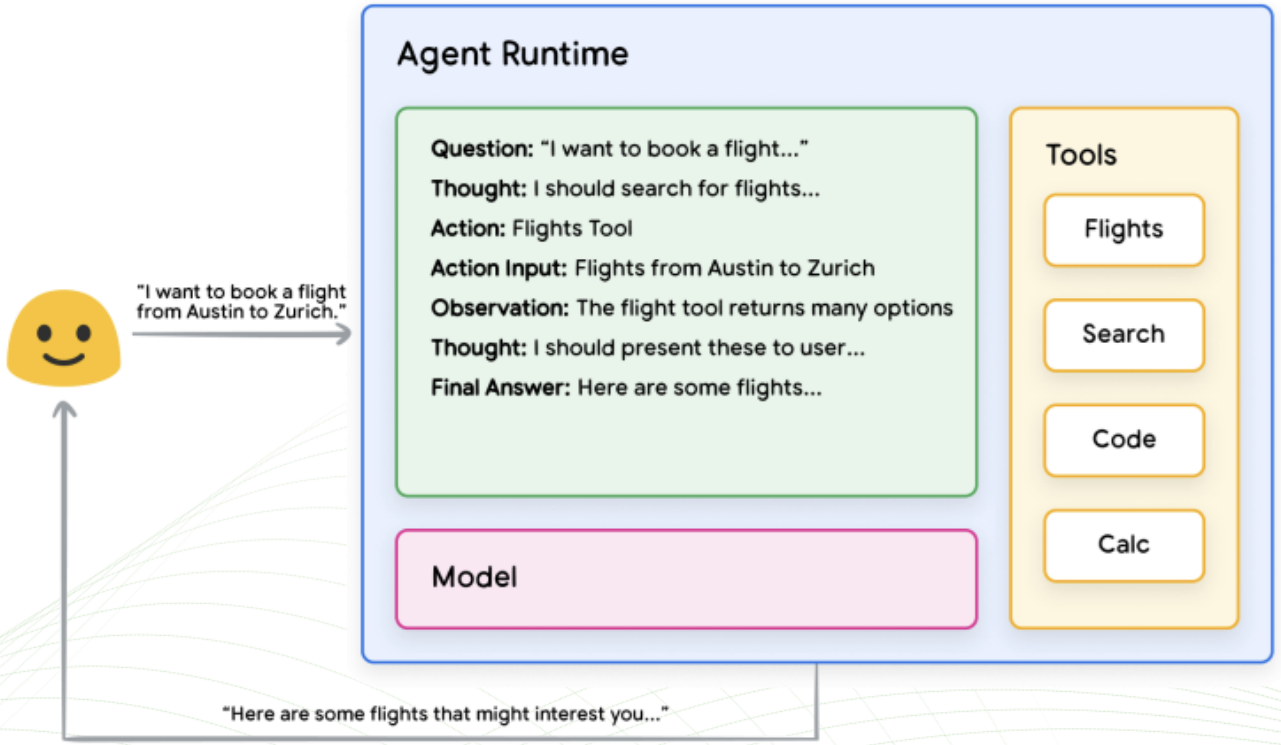
There are three essential components in an agent' s cognitive architecture:

- Model - the centralized decision maker for agent processes
- Toos - tools empowering agents to interact with external data and services
- The Orchestration layer describes a cyclical process that governs how the agent takes in information, performs some internal reasoning, and uses that reasoning to inform its next action or decision.



Agentic AI – AI Agents

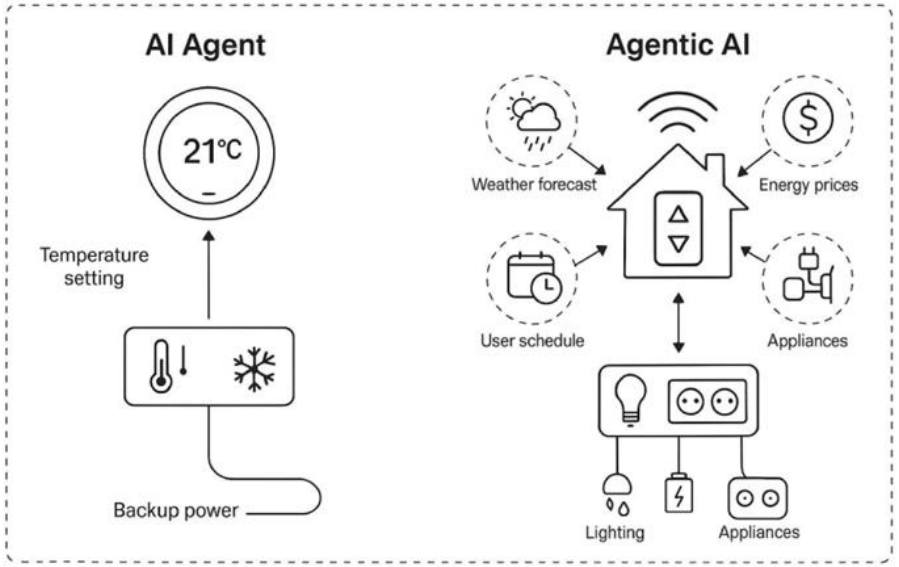
1. User sends query to the agent.
2. Agent begins the ReAct sequence.

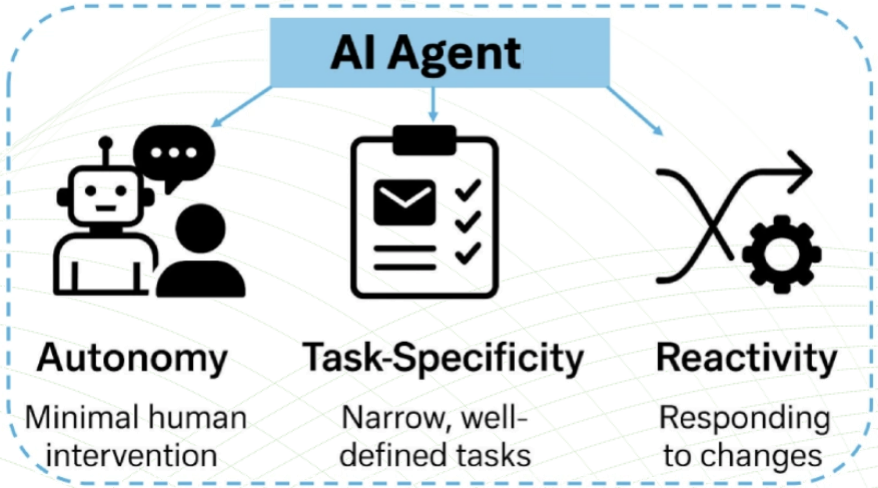


4. The ReAct loop concludes and a final answer is provided back to the user.

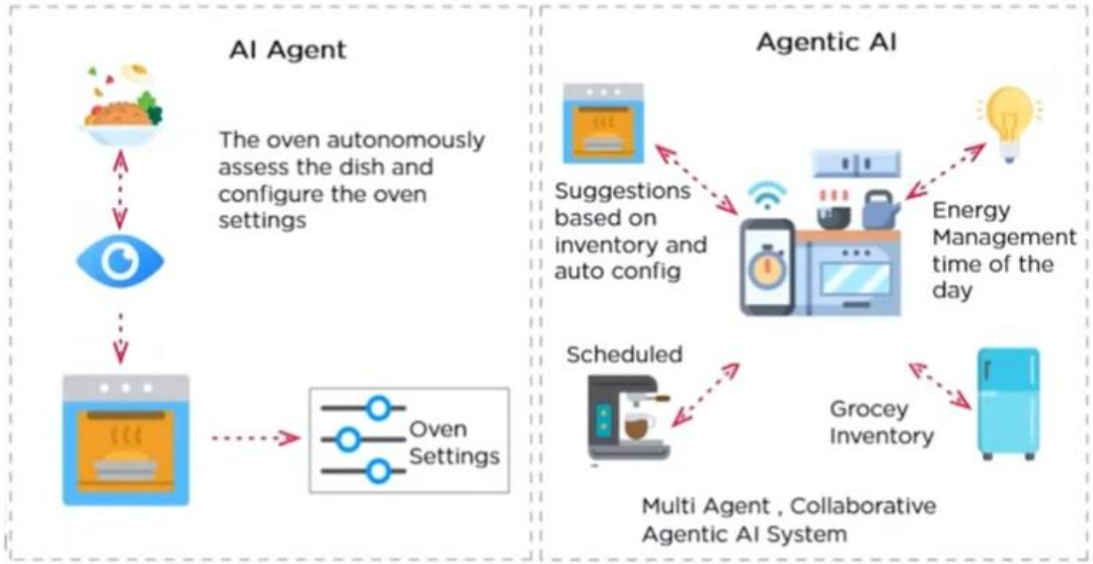
3. The agent provides a "prompt" to the model, asking it to generate one of the next ReAct steps and its corresponding output:
   a. **Question**: The input question from the user query, provided with the prompt.
   b. **Thought**: The model's thoughts about what it should do next
   c. **Action**: The model's decision on what action to take next
      i. This is where tool choice can occur
      ii. For example, an action could be one of [Flights, Search, Code, None], where the first 3 represent a known tool that the model can choose, and the last represents "no tool choice.
   d. **Action input**: The model's decision on what inputs to provide to the tool (if any)
   e. **Observation**: The result of the action / action input sequence
      i. This thought / action / action input / observation could repeat N-times as needed.
   f. **Final answer**: The model's final answer to provide to the original user query.

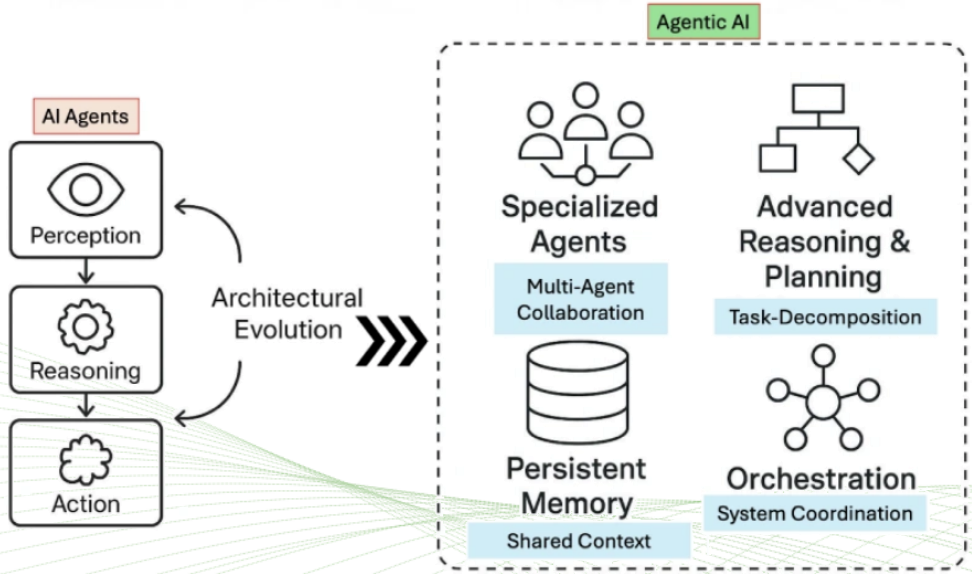
AI Agent vs. Agentic AI





Smart Kitchen System





The distinction between **Models** and **Agents**

Models	Agents
Knowledge is limited to what is available in their training data.	Knowledge is extended through the connection with external systems via tools.
Single inference / prediction based on the user query. Unless explicitly implemented for the model, there is no management of session history or continuous context. (i.e. chat history).	Managed session history (i.e. chat history) to allow for multi turn inference / prediction based on user queries and decisions made in the orchestration layer. In this context, a ‘turn’ is defined as an interaction between the interacting system and the agent. (i.e. 1 incoming event/ query and 1 agent response).
No native tool implementation.	Tools are natively implemented in agent architecture.
No native logic layer implemented. Users can form prompts as simple questions or use reasoning frameworks (CoT - Chain-of-Thought 逐步思考鏈, ReAct, etc.) to form complex prompts to guide the model in prediction.	Native cognitive architecture that uses reasoning frameworks like CoT, ReAct, or other pre-built agent frameworks like LangChain.

Side-by-side Model Structure Comparison

Feature	Generative AI	AI Agent	Agentic AI
Core Brain	LLM	LLM + Tools & APIs	LLM + Multiple Agents + Planner + Memory
Action	Generates content only	Performs narrow tasks using Tools	Plans & executes multi-step workflows
Memory	None	Short-Term	Long Term + Contextual
Autonomy 自主	Low	Medium	High
Example	Write a story	Books a flight	Plan a full holiday including visa

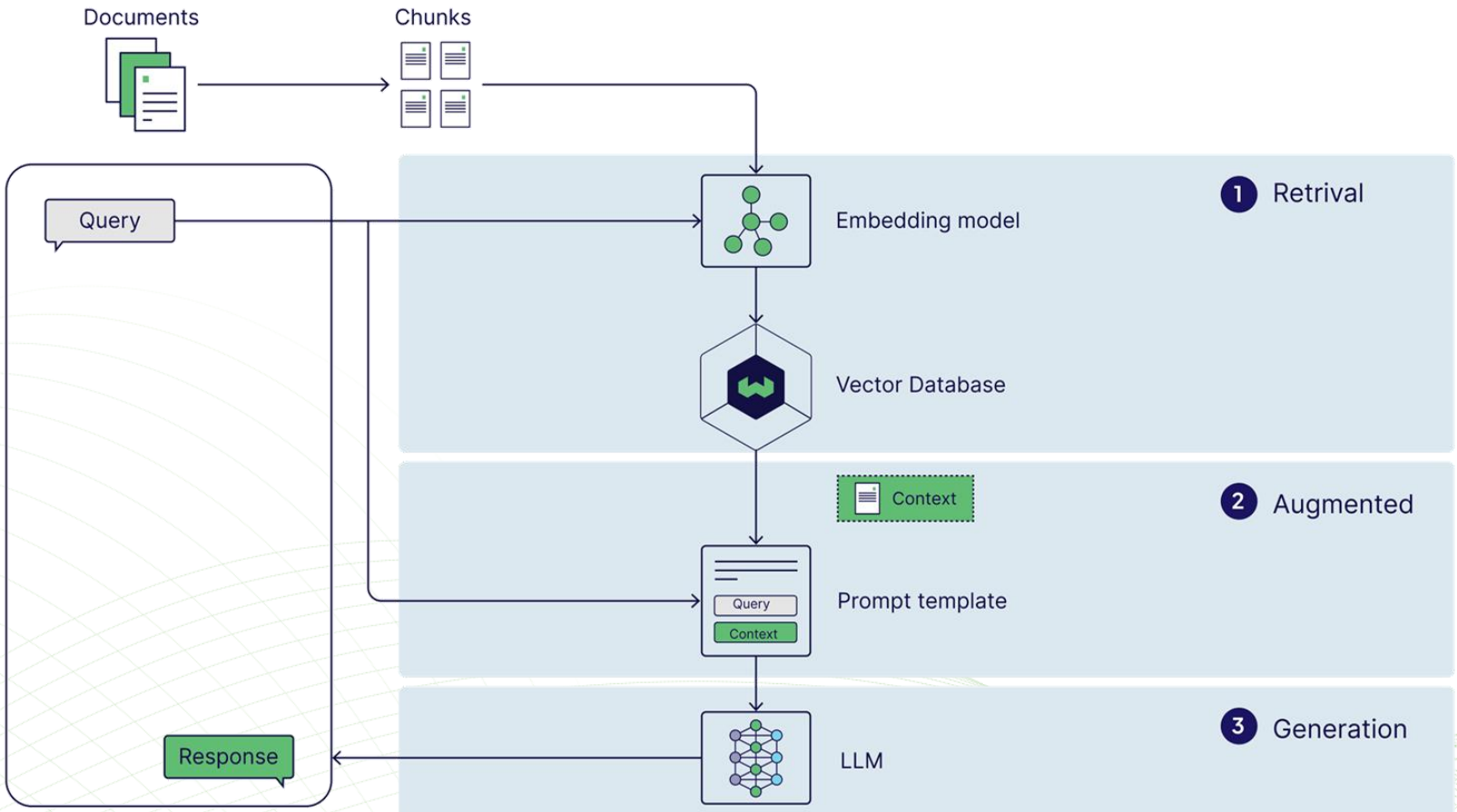
Generative AI

AI Agent

Agentic AI

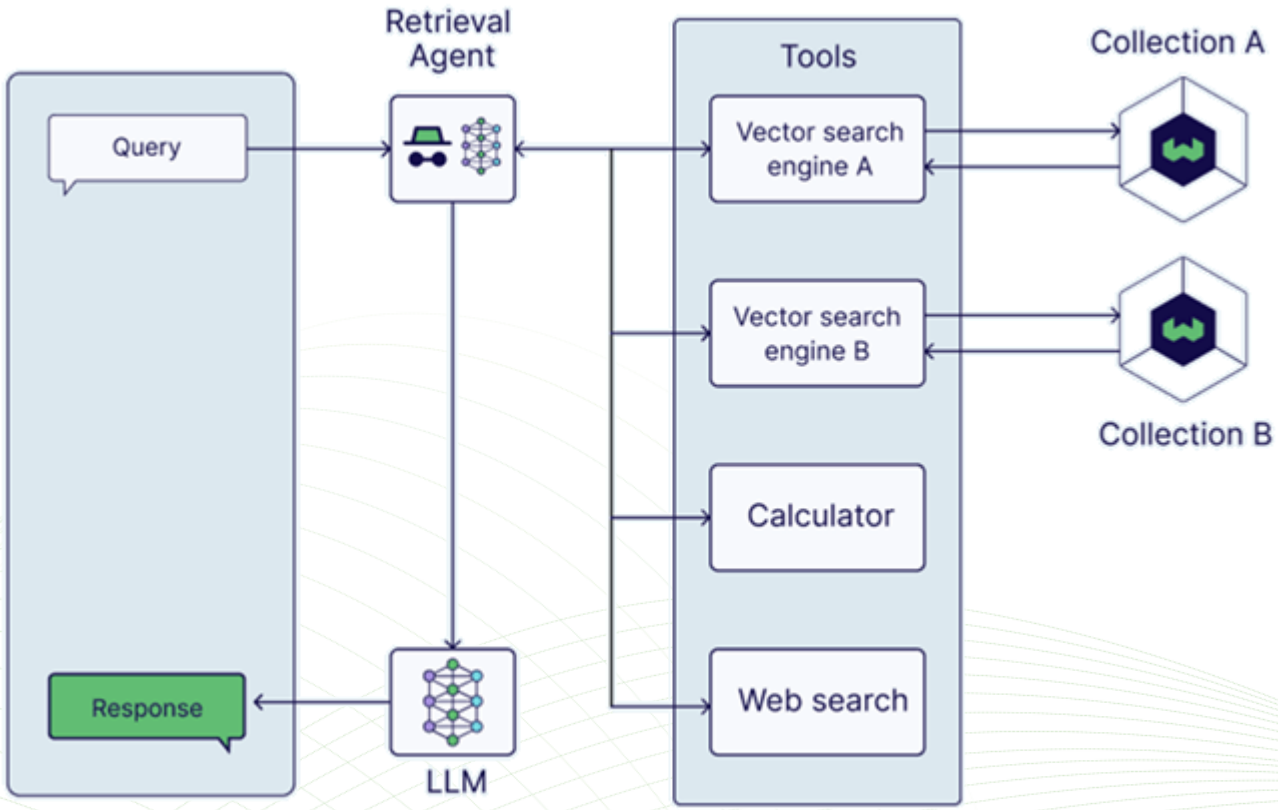
Retrieval-Augmented Generation (RAG)

A naive RAG pipeline consists of a retrieval component (typically composed of an embedding model and a vector database) and a generative component (an LLM). At inference time, the user query is used to run a similarity search over the indexed documents to retrieve the most similar documents to the query and provide the LLM with additional context.



Single Agentic RAG

Agentic RAG describes an AI agent-based implementation of RAG. Specifically, it incorporates AI agents into the RAG pipeline to orchestrate its components and perform additional actions beyond simple information retrieval and generation to overcome the limitations of the non-agentic pipeline.



Single-Agent RAG (Router)

The retrieval component becomes agentic through the use of retrieval agents with access to different retriever tools, such as:

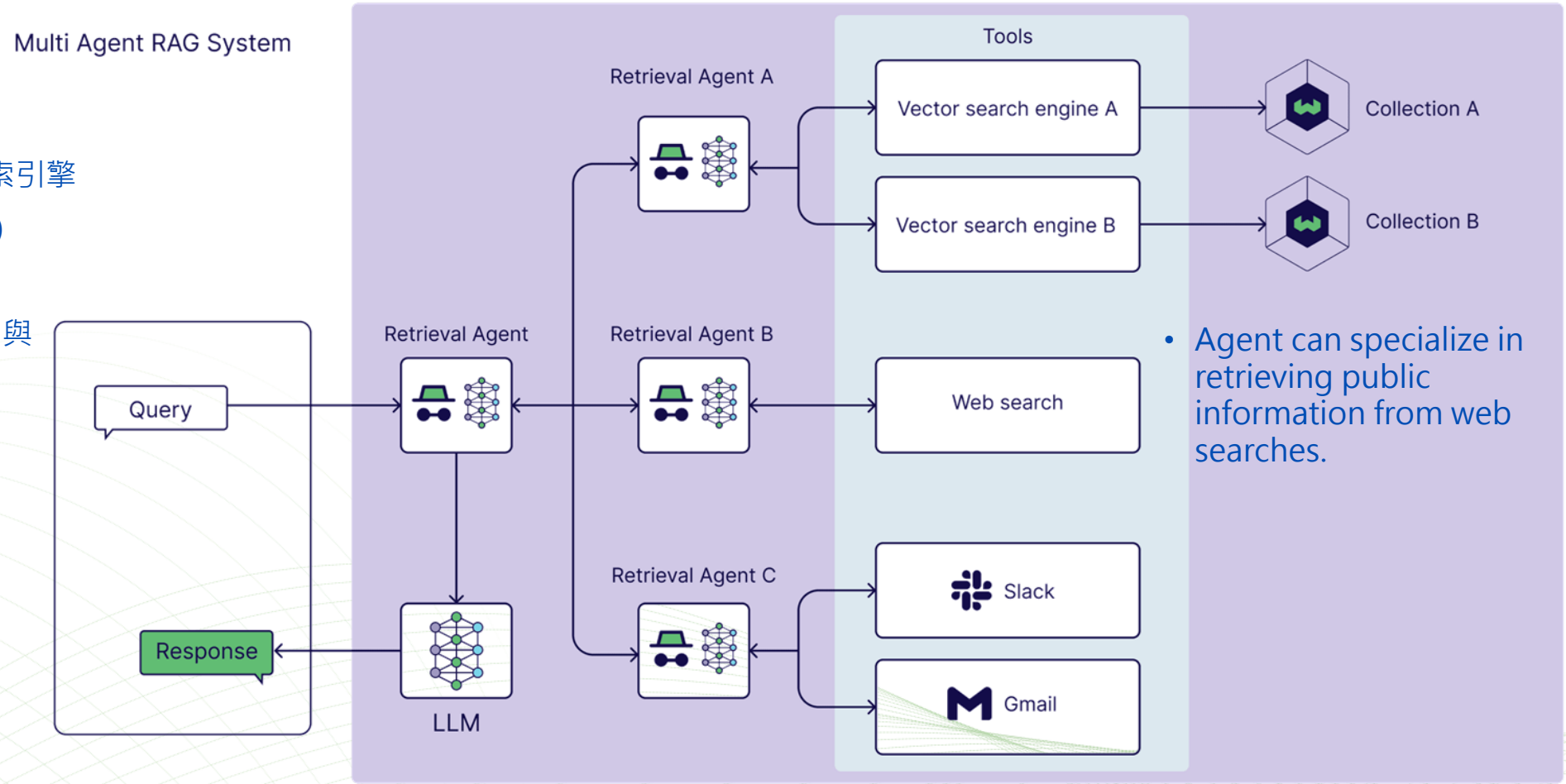
- Vector search engine (also called a query engine) that performs vector search over a vector index (like in typical RAG pipelines)
- Web search
- Calculator
- Any API to access software programmatically, such as email or chat programs
- and many more.

Multi Agentic RAG System

The single-agent system also has its limitations because it's limited to only one agent with reasoning, retrieval, and answer generation in one. Therefore, it is beneficial to chain multiple agents into a multi-agent RAG application.

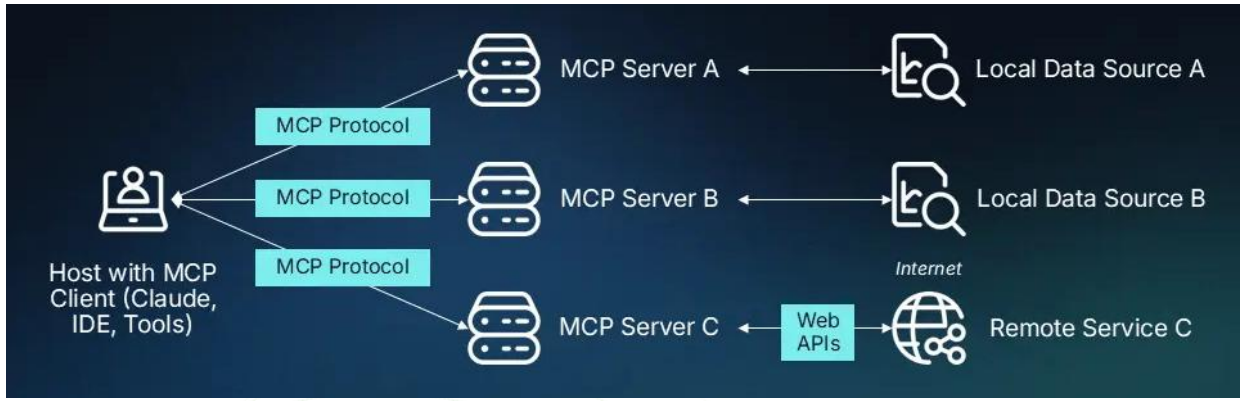
關鍵技術的支援

- 向量資料庫與檢索引擎
- 語言模型（LLM）
- 工具調用框架
- 反思 (Reflection)與規畫模組



- Agent can specialize in retrieving public information from web searches.

An open protocol that standardizes how your LLM applications connect to and work with your tools and data sources.



Model Context Protocol
MCP - 模型上下文協定

在人工智慧 (AI) 領域，MCP 是一個開放標準協定，旨在標準化 AI 應用程式與外部工具、資料庫和服務之間的整合。它允許可代理的 AI（AI agents）存取即時資訊並執行實際動作，而不僅僅是進行對話。

https://www.youtube.com/watch?v=kOhLoixrJXo&t=732s

MCP In 26 Minutes (Model Context Protocol)

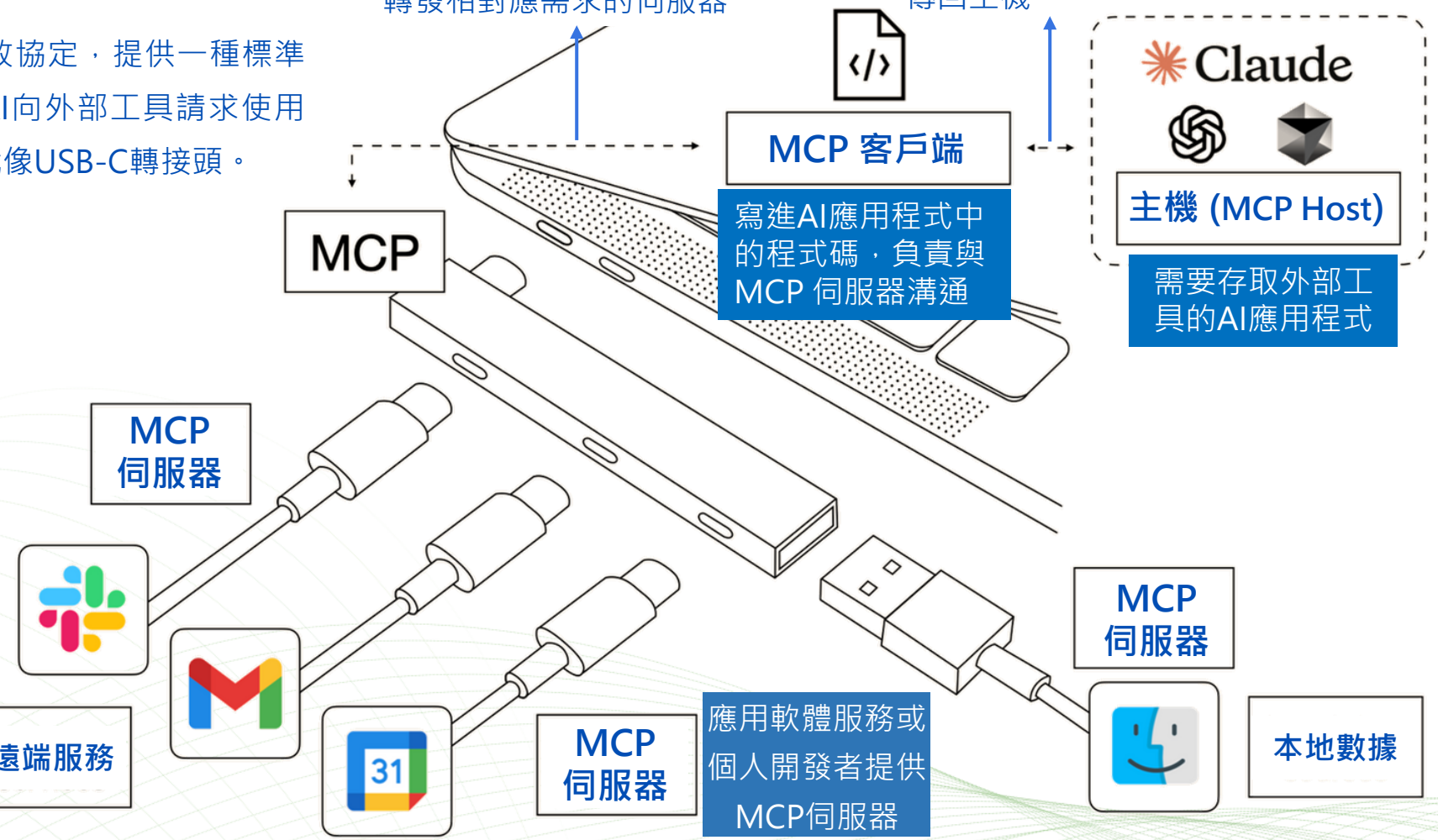
Model Context Protocol (MCP)

MCP示意圖

MCP是一種開放協定，提供一種標準化的方式，讓AI向外部工具請求使用服務和數據，就像USB-C轉接頭。

與MCP伺服端採專用1對1串接；MCP在收到資訊請求後，轉發相對應需求的伺服器

主機向客戶端發出資訊請求，客戶端則將回應傳回主機



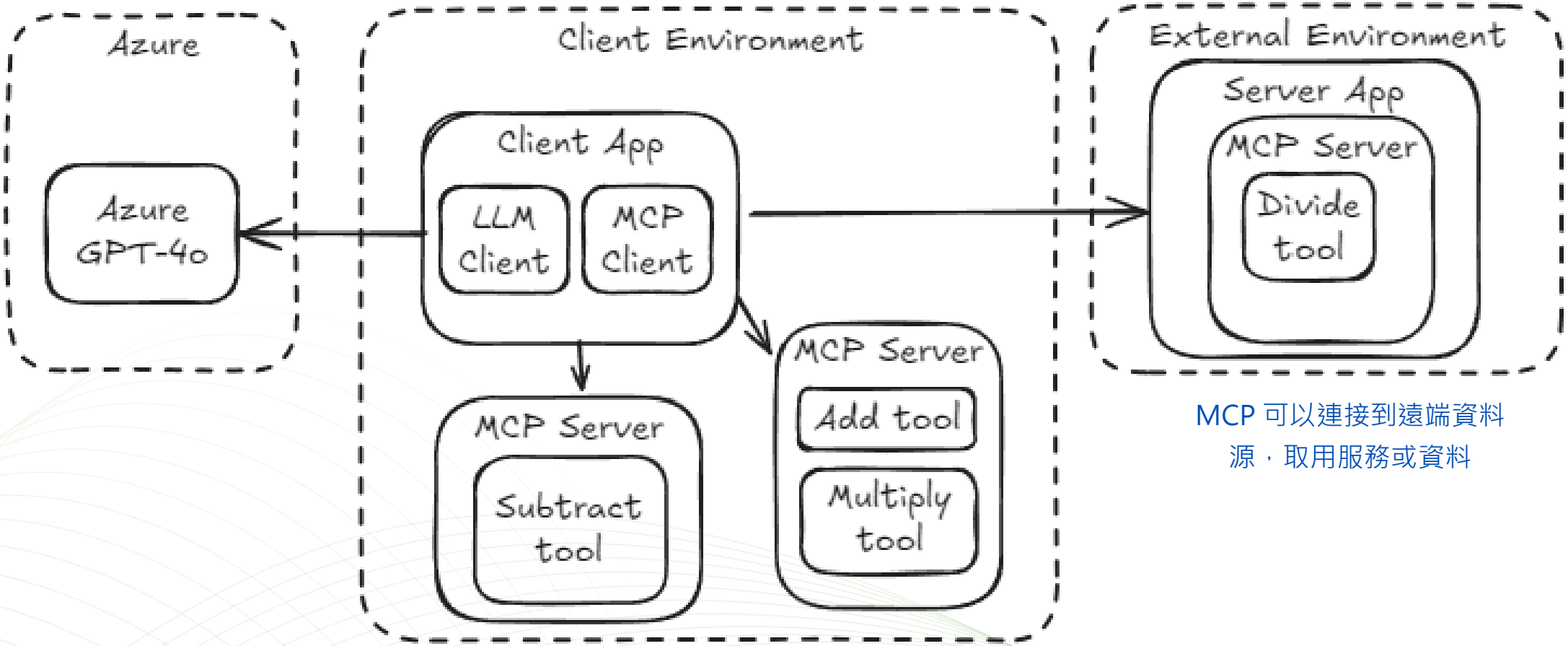
MCP 不只是適用於企業內部的AI工作流程，也有助於AI串接外部應用工具。

MCP 可以連接到遠端資料源，取用服務或資料

MCP 可以連接到地端，取用本地資料

Model Context Protocol (MCP)

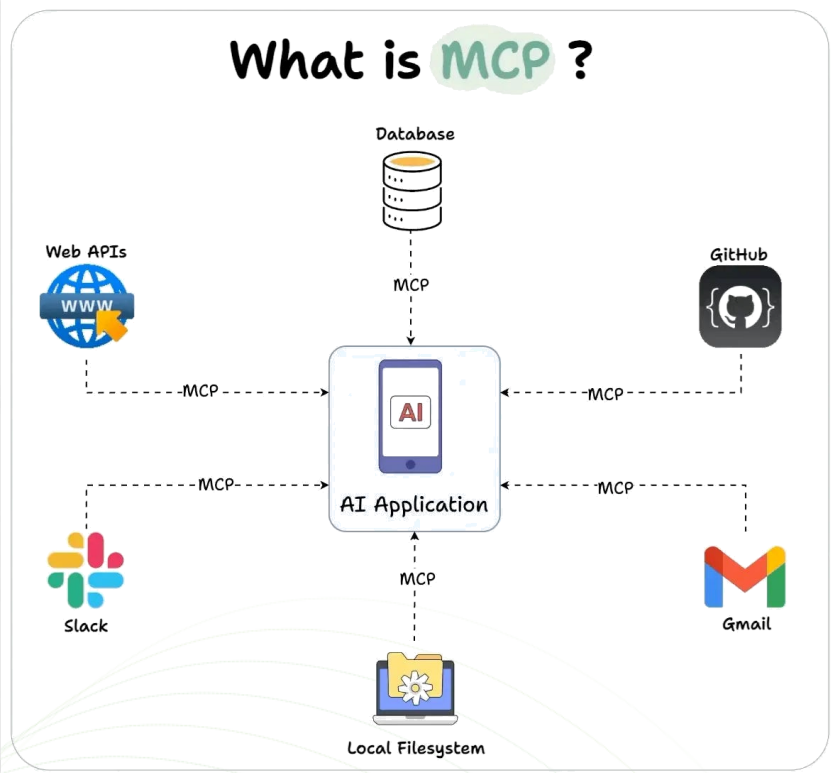
You can see that the client application manages to connect to the 2x local MCP servers and the remote server, and is able to obtain the list of tools provided by each.

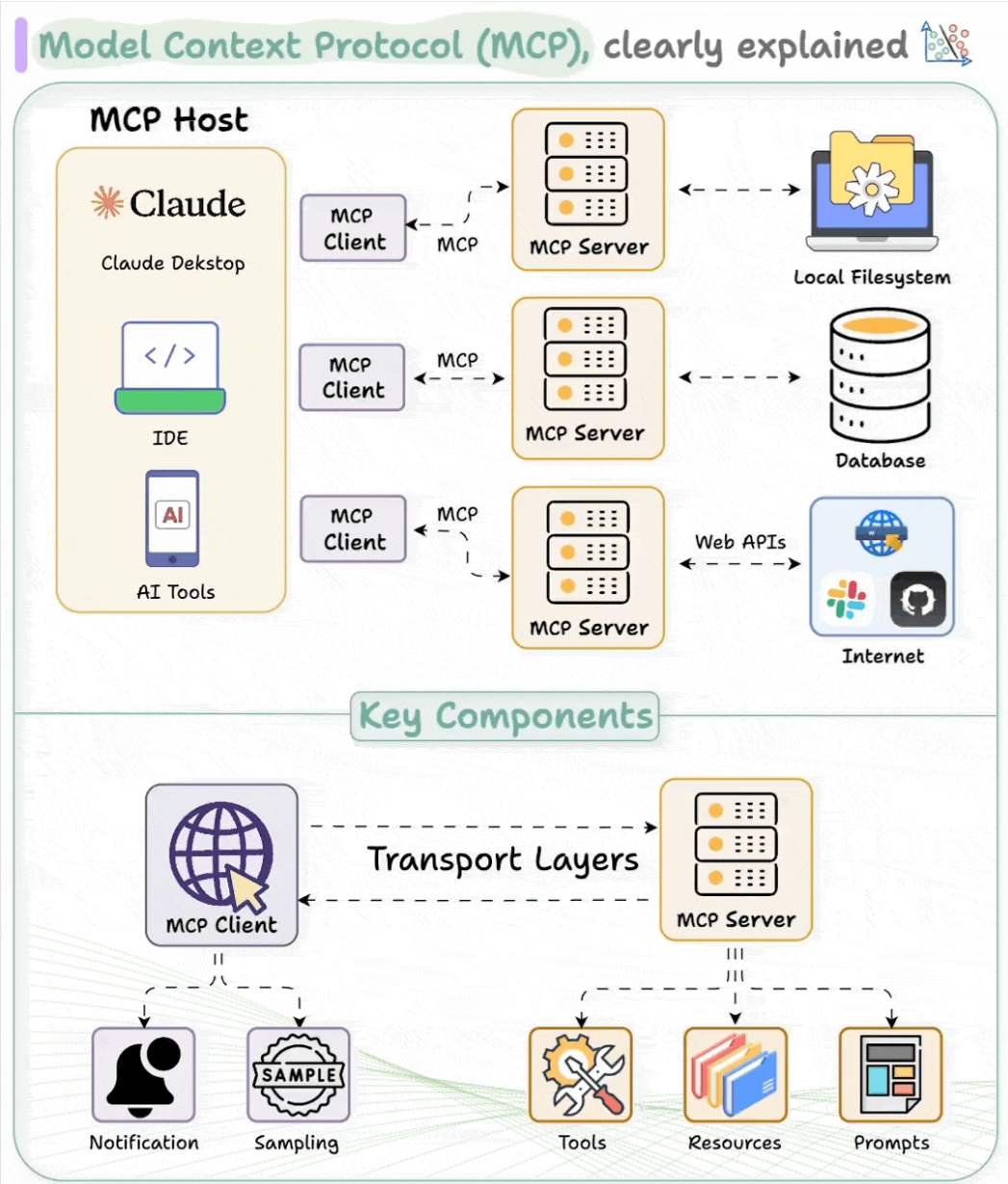


MCP 可以連接到遠端資料源，取用服務或資料

MCP 可以連接到地端，取用本地資料

Model Context Protocol (MCP)





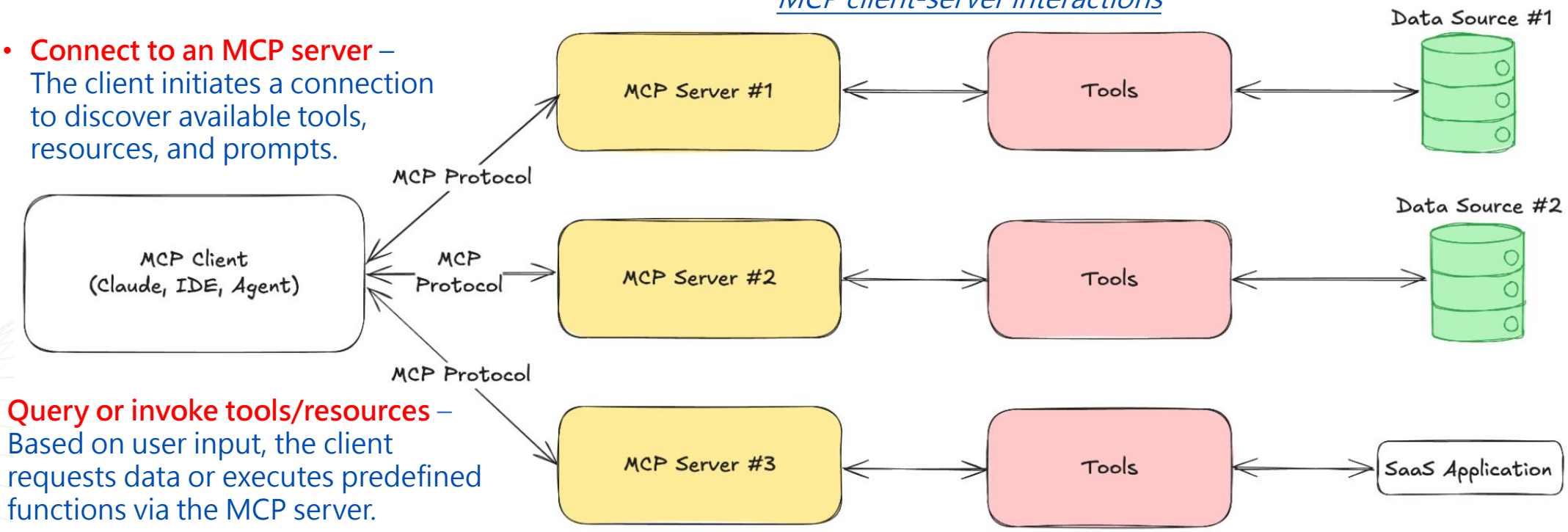
模型上下文協定（Model Context Protocol, MCP）是 AI Agent (AI助理) 技術發展的重要突破，解決了 AI 無法存取外部數據的瓶頸，讓 AI 能夠透過標準化協定與本地電腦、資料庫及網路服務互動。MCP 透過伺服器、用戶端與主機的協作機制，使 AI 助理能執行更複雜的任務，如資料庫查詢、網頁偵錯與檔案管理，進而從被動回應者轉變為主動任務執行者。

Model Context Protocol (MCP)

MCP standardizes how agents retrieve and interact with external data. It provides a structured way to facilitate these interactions for AI systems:

- Pull customer records from a database
- Retrieve documents from cloud storage
- Execute workflows based on real-time inputs

MCP client-server interactions

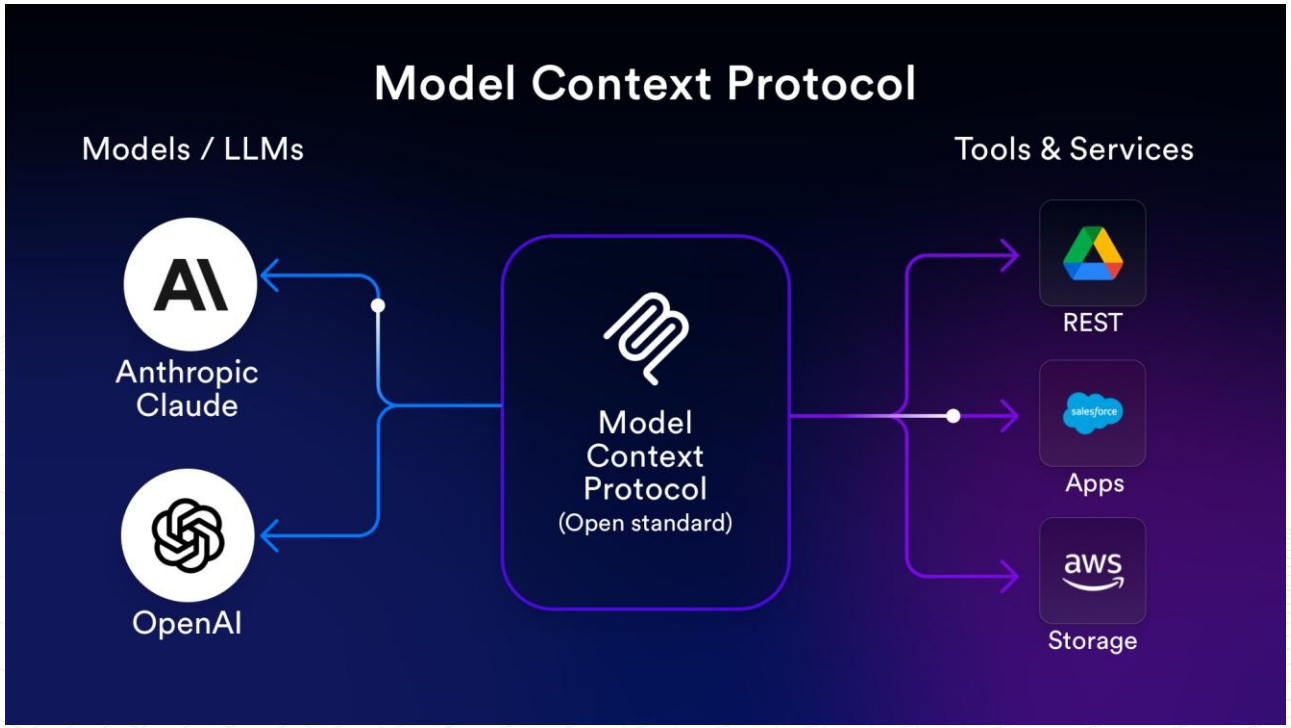


- **Connect to an MCP server** – The client initiates a connection to discover available tools, resources, and prompts.
- **Query or invoke tools/resources** – Based on user input, the client requests data or executes predefined functions via the MCP server.
- **Process and return responses** – The AI agent processes the retrieved data, applies contextual reasoning, and delivers an appropriate response or action.

Model Context Protocol (MCP)

MCP works by defining three key components within its servers:

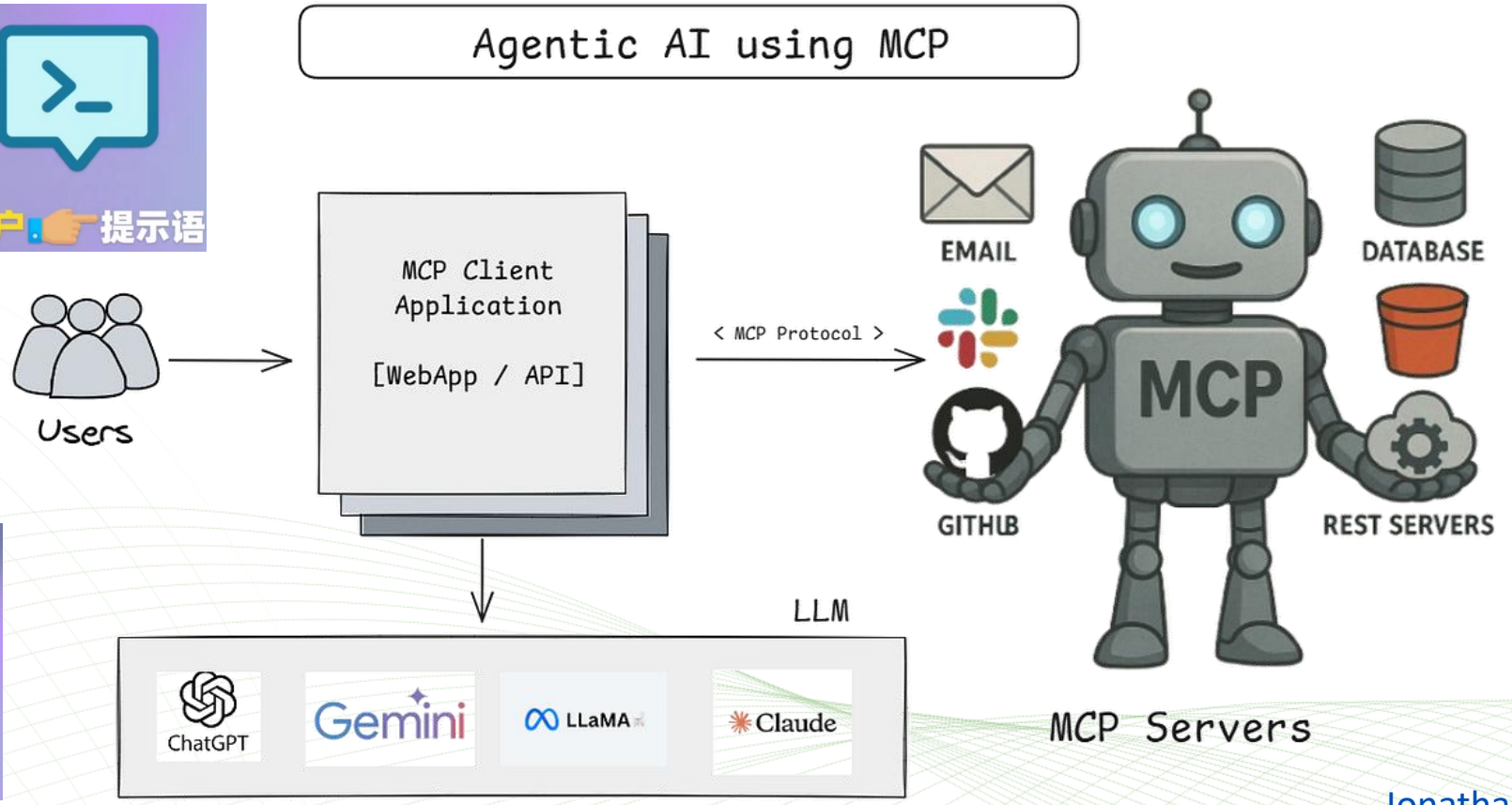
- **Tools** – Functions that AI agents can call to perform specific actions, such as making API requests or executing commands (e.g., querying a weather API).
- **Resources** – Data sources that AI agents can access, similar to REST API endpoints. These provide structured data without performing additional computation.
- **Prompts** – Predefined templates that guide AI models in optimally using tools and resources.



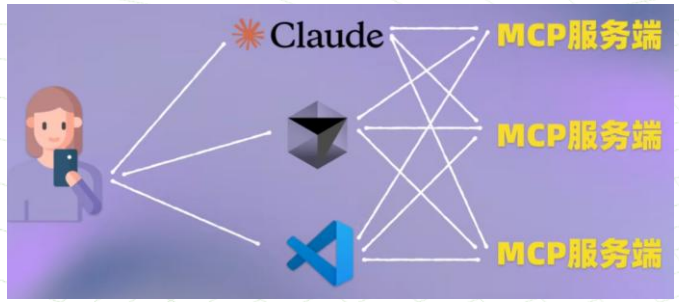
- The Model Context Protocol (MCP) is essential for the future of AI, enabling more flexible and secure interactions between AI models and external resources.
- As AI continues to evolve, MCP' s role as a universal connector will be crucial in building intelligent systems that are truly adaptable, reliable, and scalable.

Agentic AI using MCP

- **Tools**: Functions that the model can call (for example, lookup_customer_by_email).
- **Resources**: Structured data a model can reference (for example, product catalog, user records).
- **Prompt templates**: Pre-written prompts the system can use to guide model behavior (for example, Summarize this customer’ s sentiment history).



- 不同的 MCP Host/Client 可以連結到同一個 MCP Server.



Model Context Protocol (MCP)

The user asks the MCP Client to do something

The MCP Client checks which tools are available from connected MCP Servers

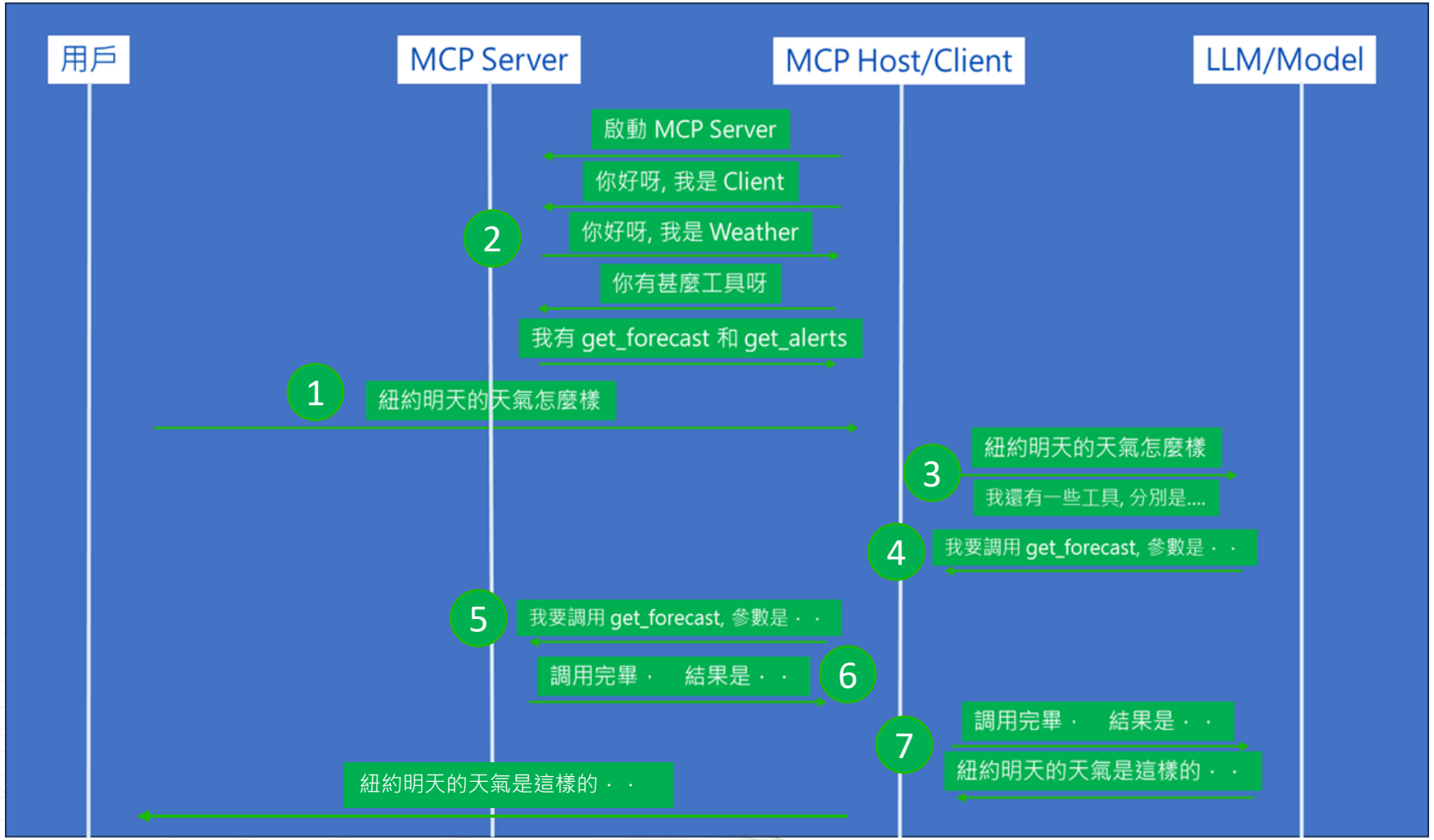
It sends the user’ s request and tool info to the LLM

The LLM picks the right tool and fills in the needed details

The MCP Client sends this to the MCP Server

The MCP Server runs the tool and sends back the result

The LLM used that result to write a reply for the user

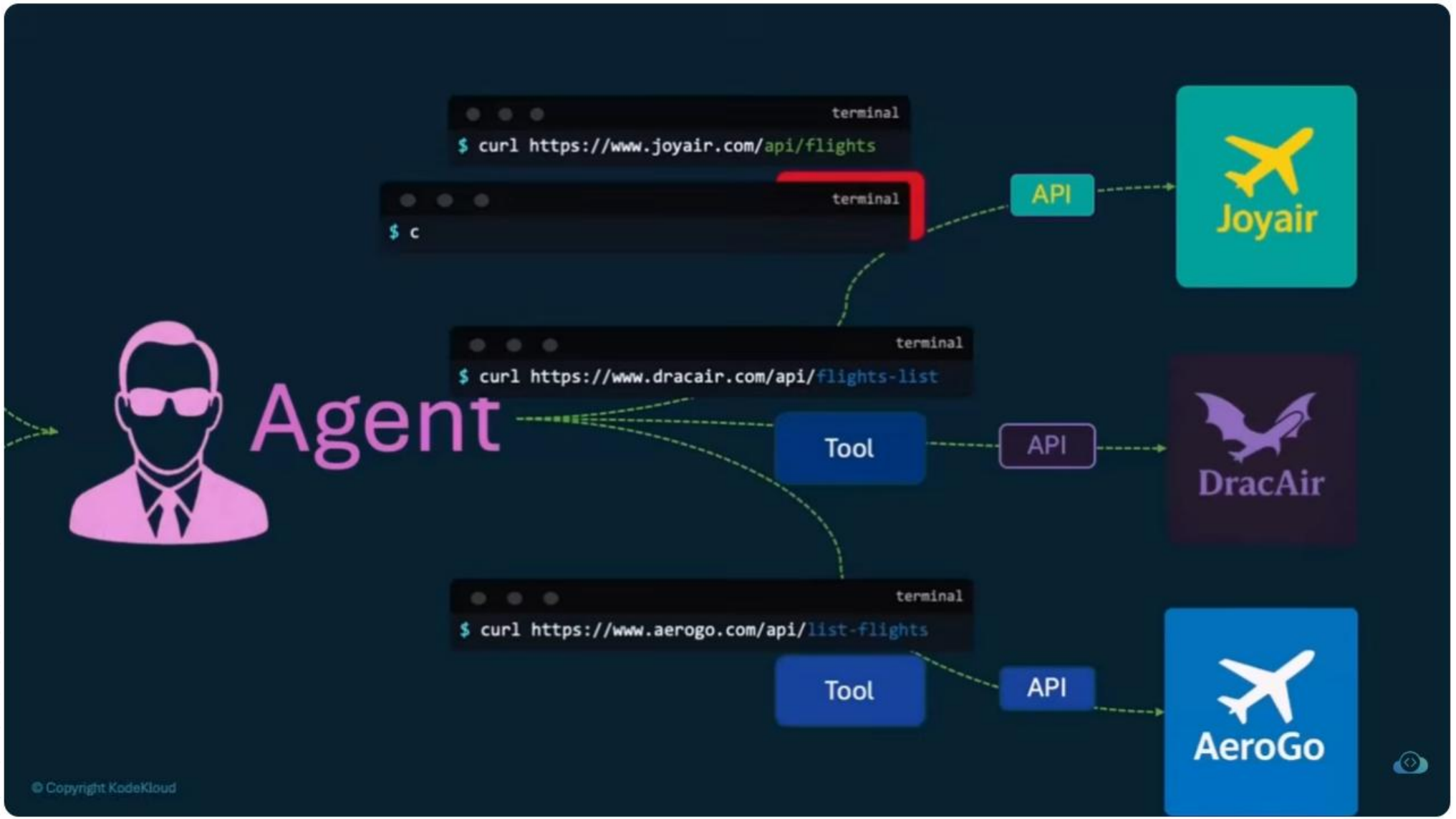


MCP client and server exchange.

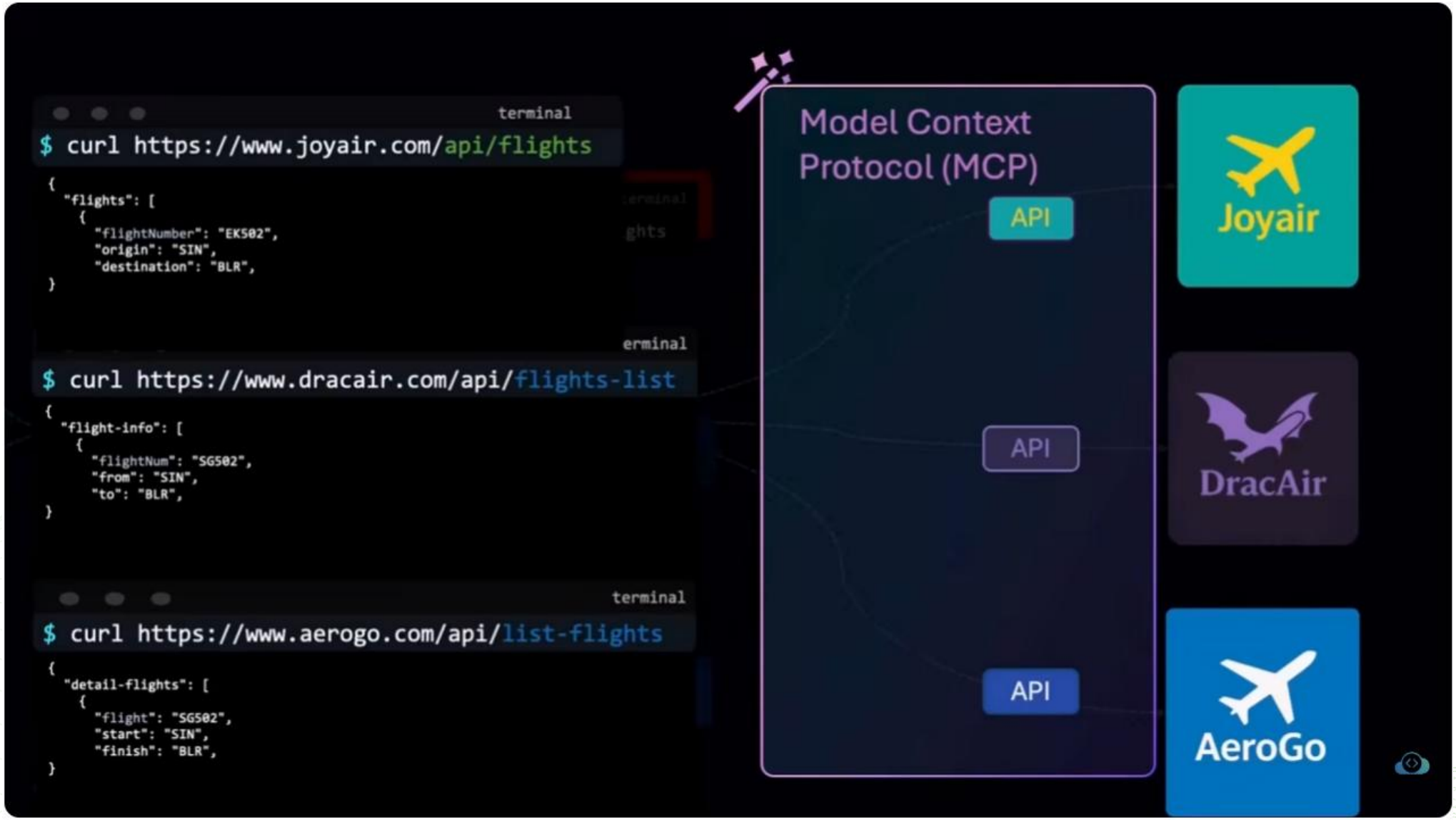
Model Context Protocol (MCP)

- LLM 統一透過 MCP 來取得數據

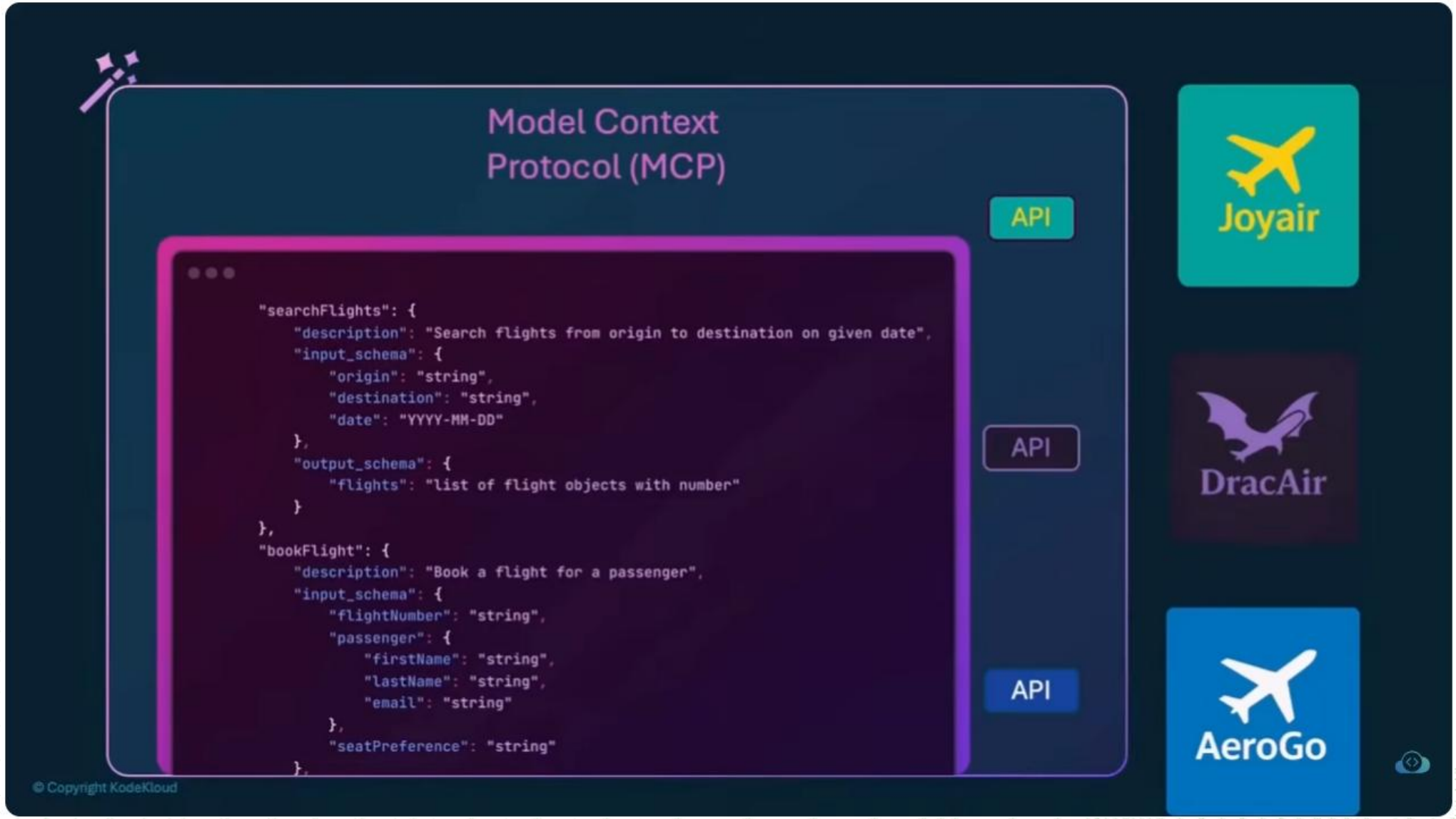
Model Context Protocol (MCP)



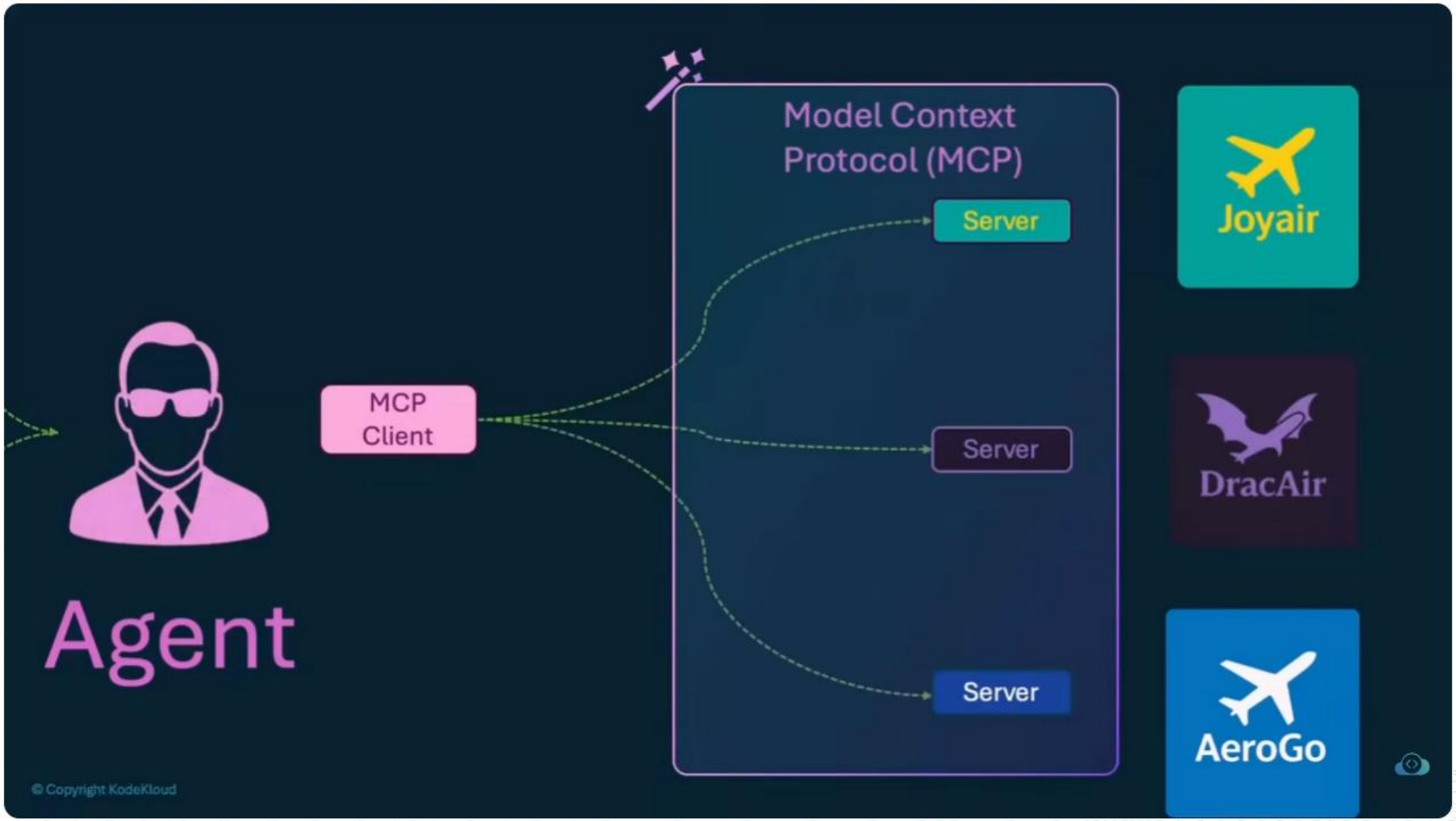
Model Context Protocol (MCP)



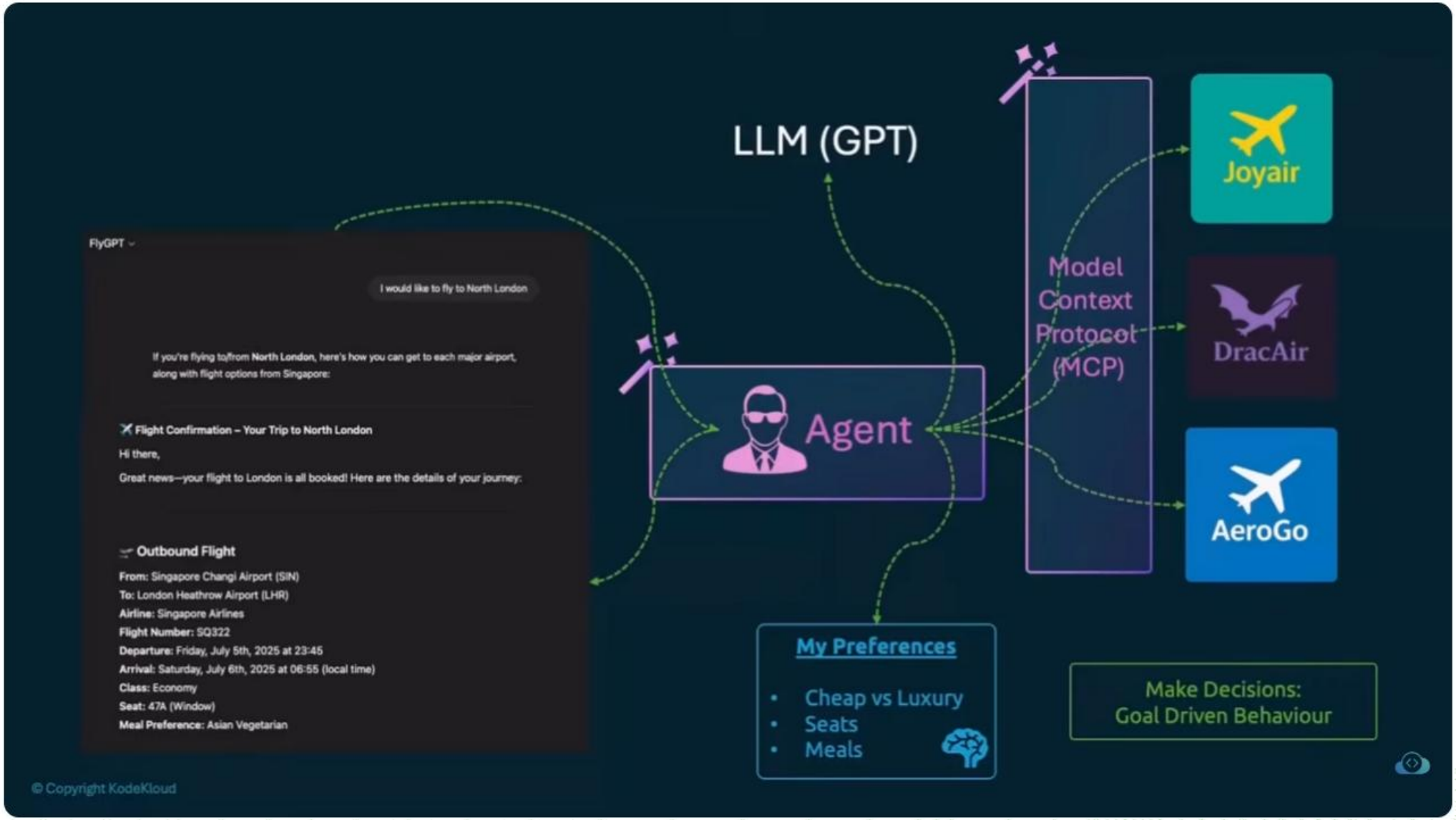
Model Context Protocol (MCP)



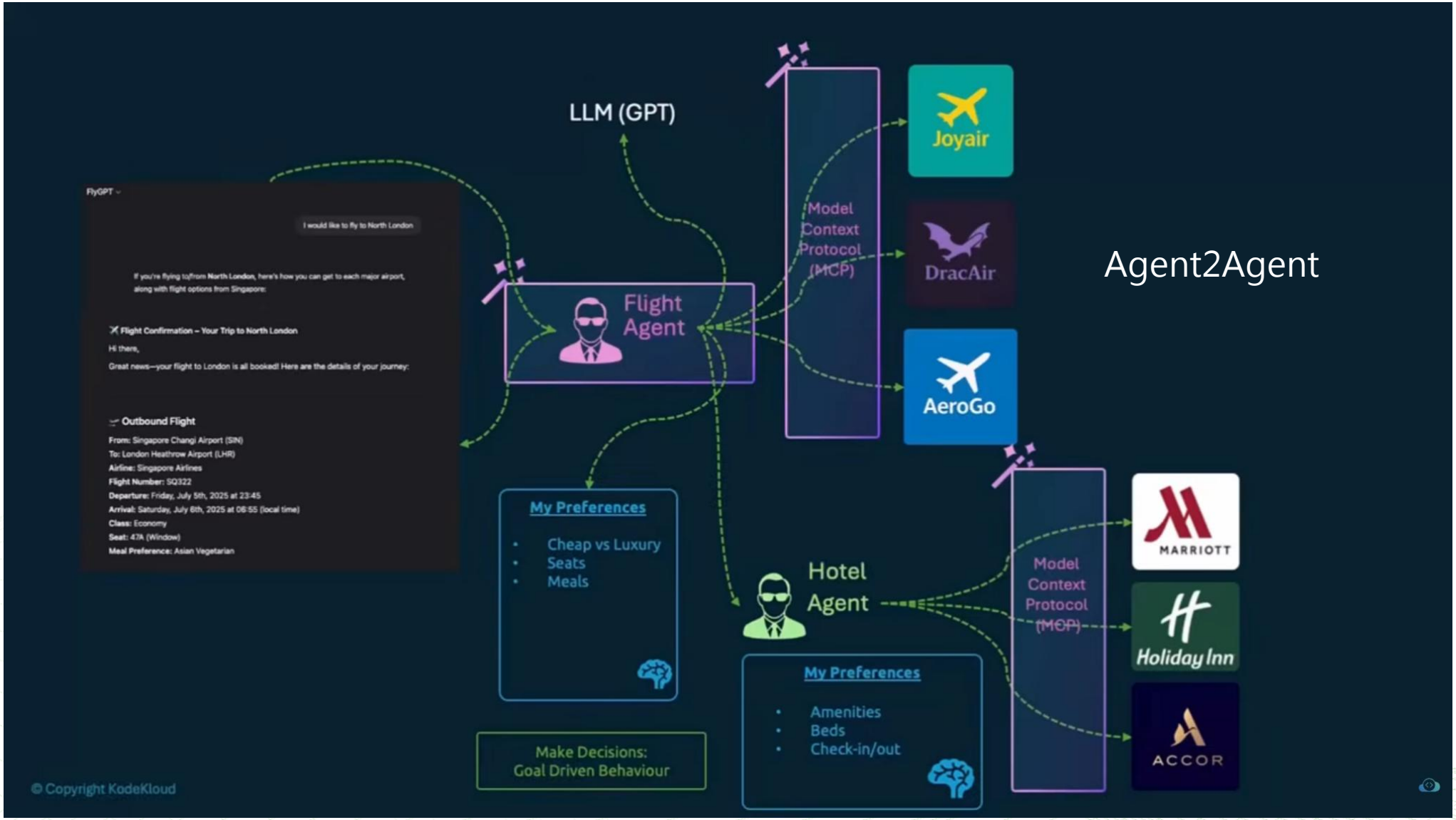
Model Context Protocol (MCP)



Model Context Protocol (MCP)



Model Context Protocol (MCP)



MCP Clients

https://www.pulsemcp.com/clients

pulse

MCP Servers | MCP Clients | Posts | Partner | THE MCP newsletter

Submit

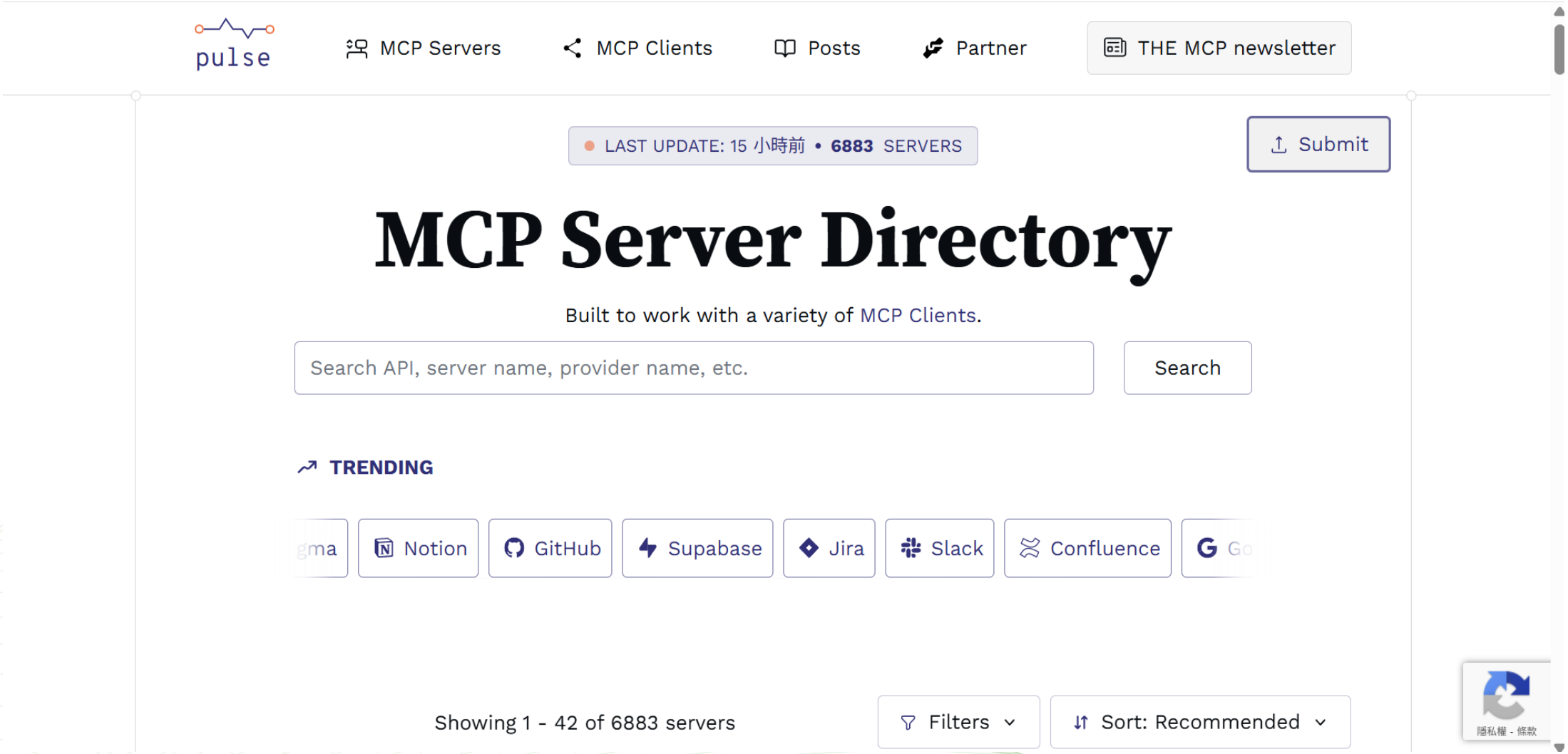
A collection of AI apps and tools that are capable of functioning as MCP clients to interact with the growing list of MCP servers.

We omit solutions like Firebase Genkit, Superinterface, and Langflow from this list because they function as *frameworks* for building new MCP-compatible apps. You also have the option to connect any OpenAI API-compatible apps to MCP via MCP-Bridge.

Featured

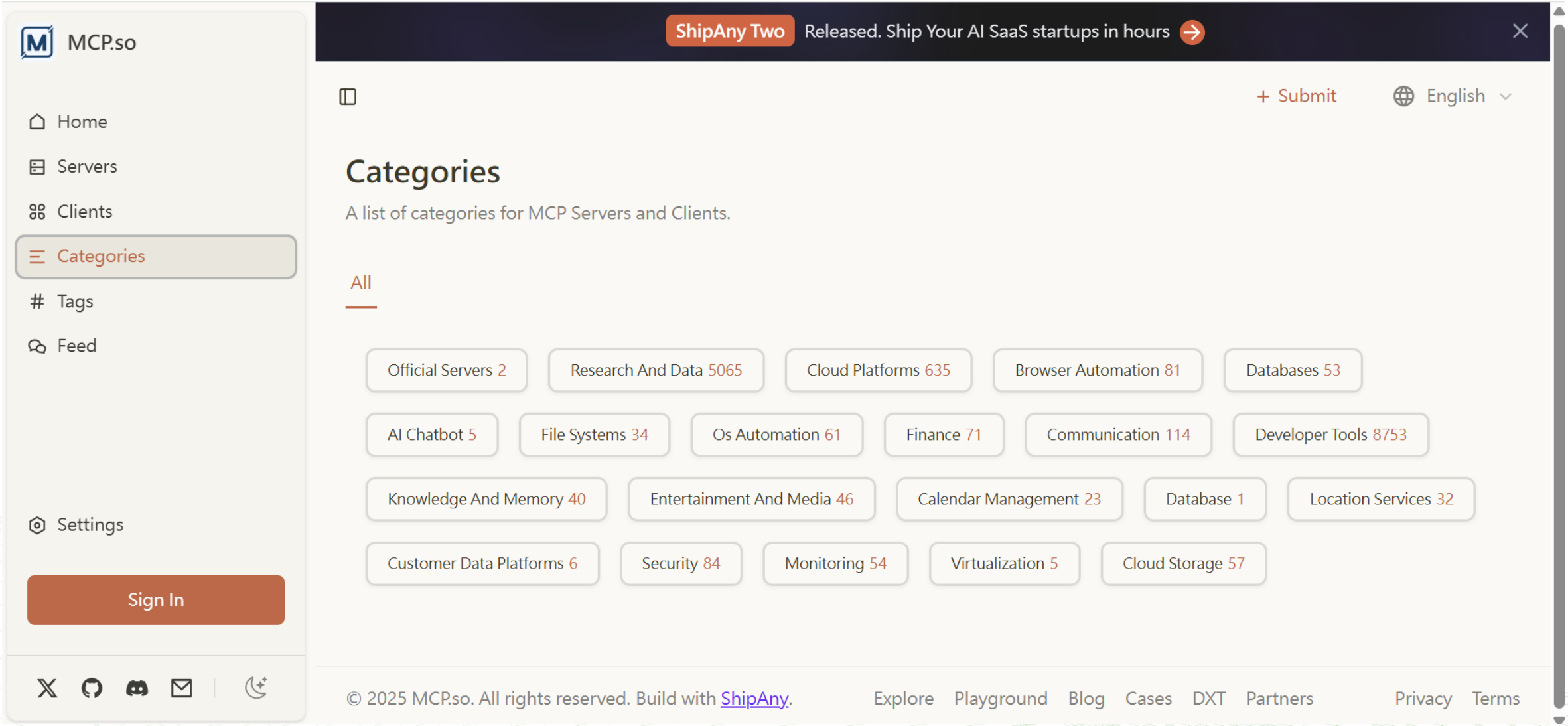
MCP Servers

https://www.pulsemcp.com/servers

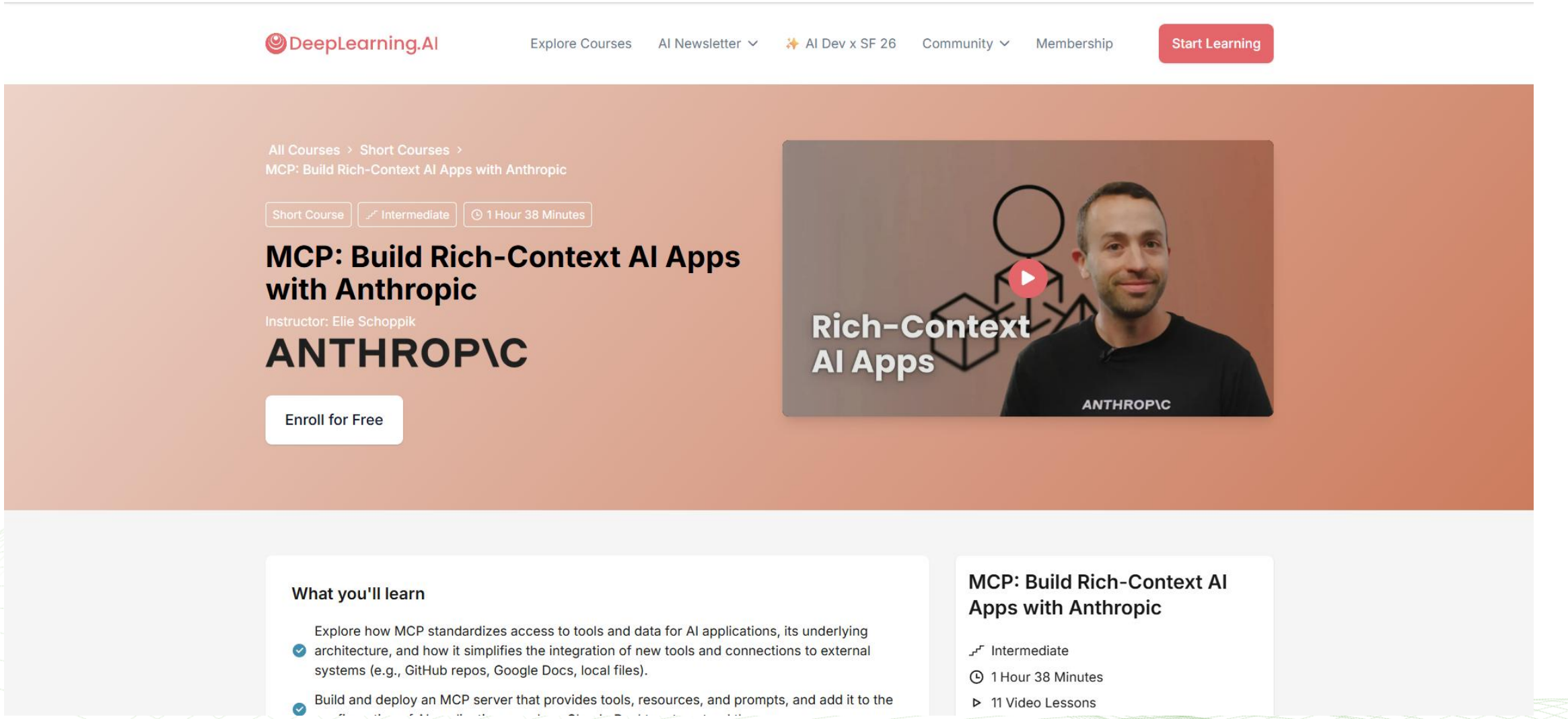


MCP Servers

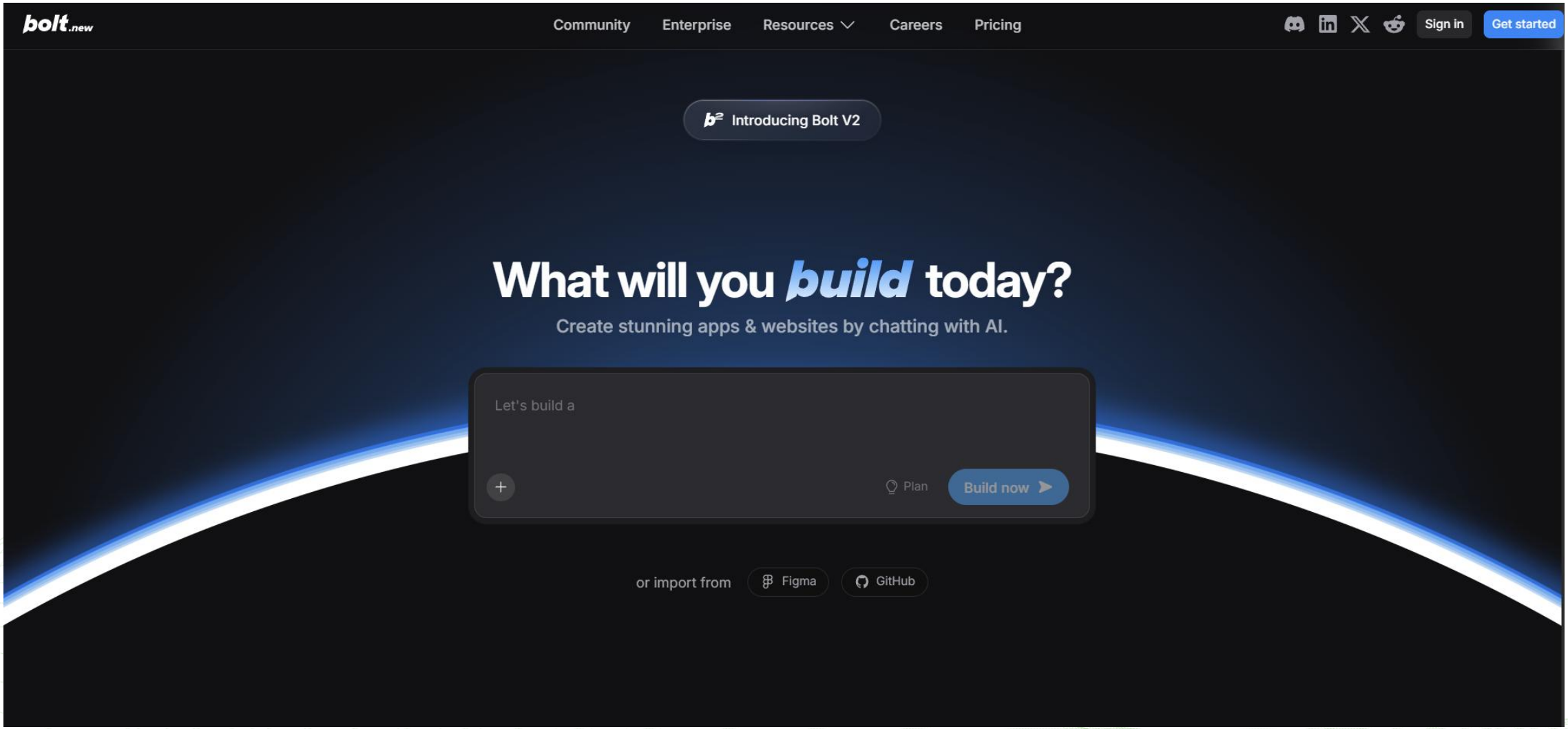
https://mcp.so/servers?tag=featured



https://www.deeplearning.ai/short-courses/mcp-build-rich-context-ai-apps-with-anthropic/



https://bolt.new/



Model Context Protocol (MCP)

https://modelcontextprotocol.info/docs/concepts/transports/

Transports

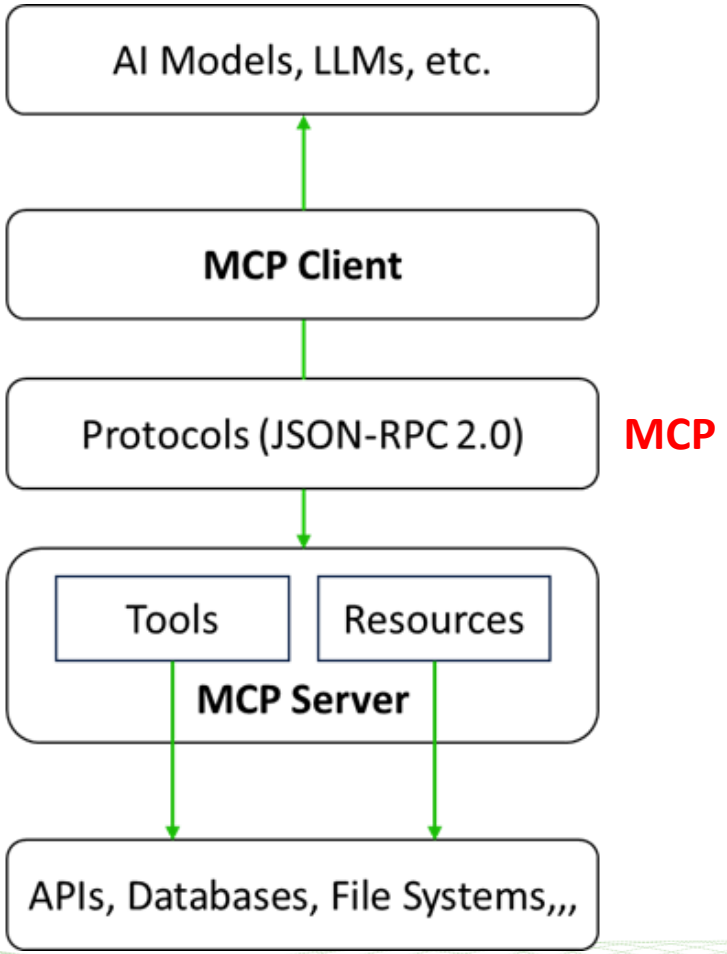
Transports in the Model Context Protocol (MCP) provide the foundation for communication between clients and servers. A transport handles the underlying mechanics of how messages are sent and received.

Message Format

MCP uses JSON-RPC 2.0 as its wire format. The transport layer is responsible for converting MCP protocol messages into JSON-RPC format for transmission and converting received JSON-RPC messages back into MCP protocol messages.

There are three types of JSON-RPC messages used:

- Request
- Response
- Notification



https://www.youtube.com/watch?v=VChRPFUzJGA

Built-in Transport Types

MCP includes two standard transport implementations:

- **Standard Input/Output (stdio)**

  The stdio transport enables communication through standard input and output streams. This is particularly useful for local integrations and command-line tools.

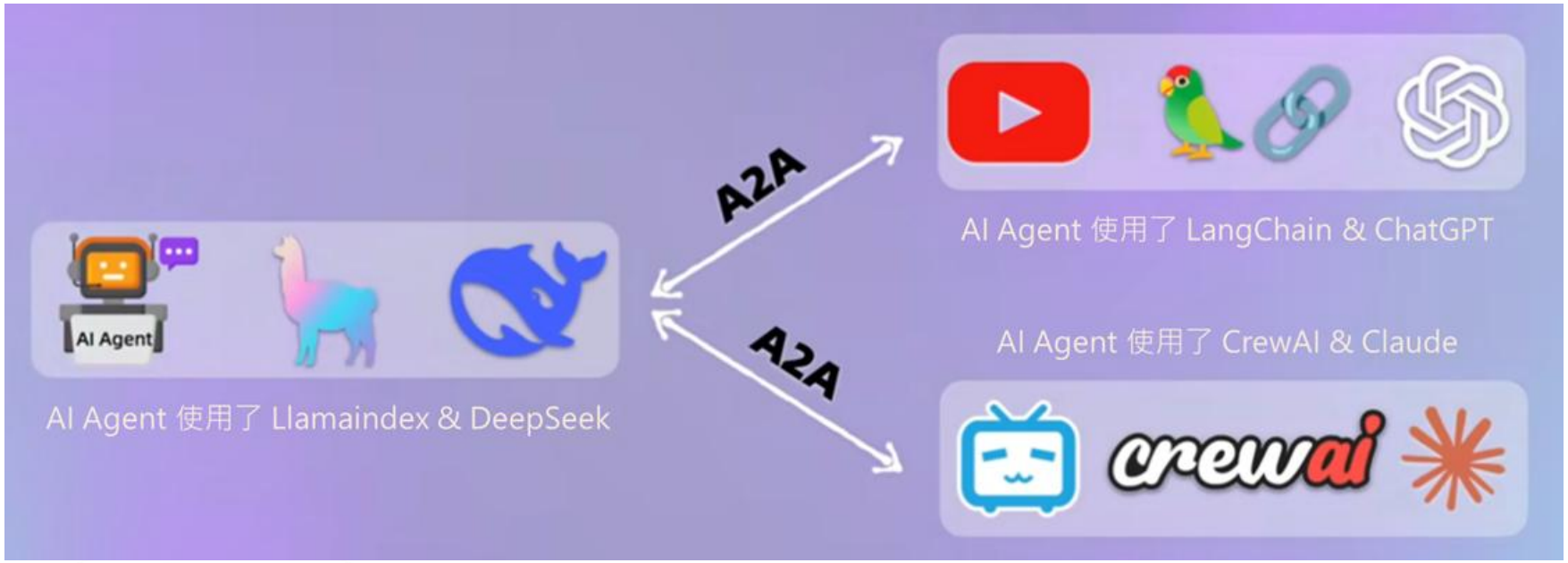
  Use stdio when:
  - Building command-line tools
  - Implementing local integrations
  - Needing simple process communication
  - Working with shell scripts

- **Server-Sent Events (SSE)**

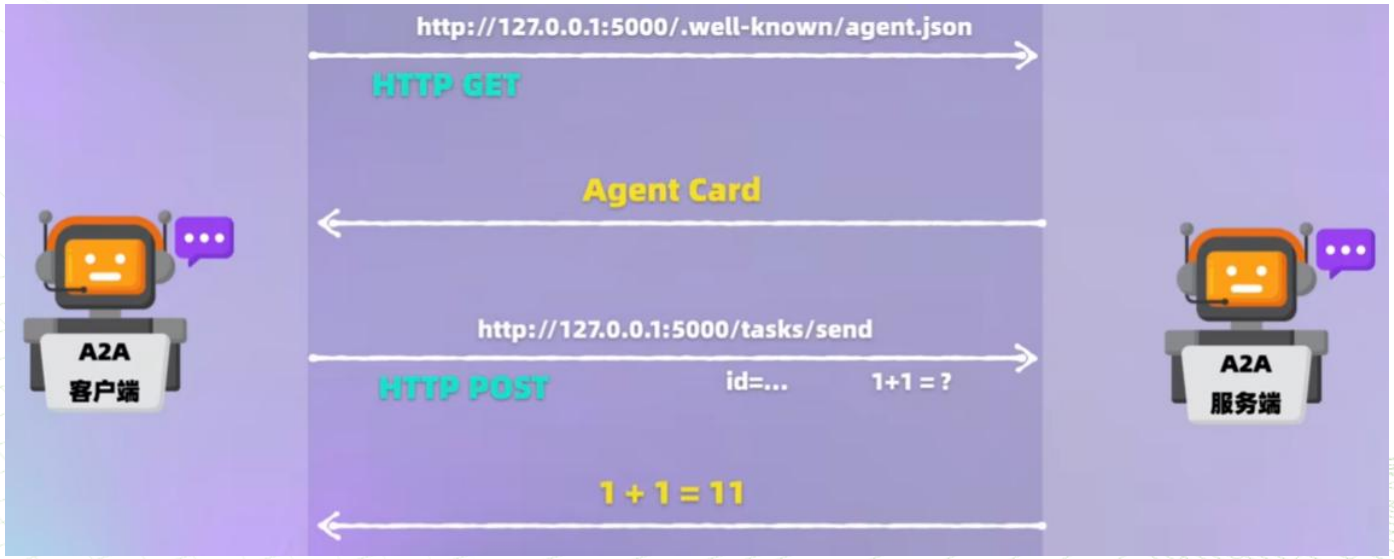
  SSE transport enables server-to-client streaming with HTTP POST requests for client-to-server communication.

  Use SSE when:
  - Only server-to-client streaming is needed
  - Working with restricted networks
  - Implementing simple updates

Google Agent2Agent Protocol (A2A)



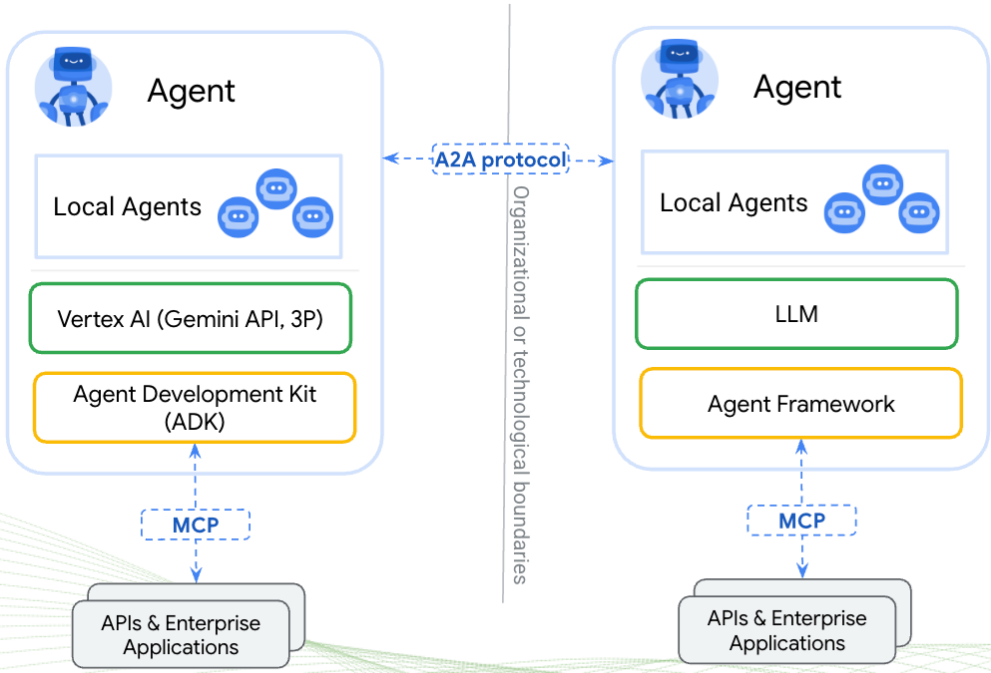
JSON-RPC 2.0



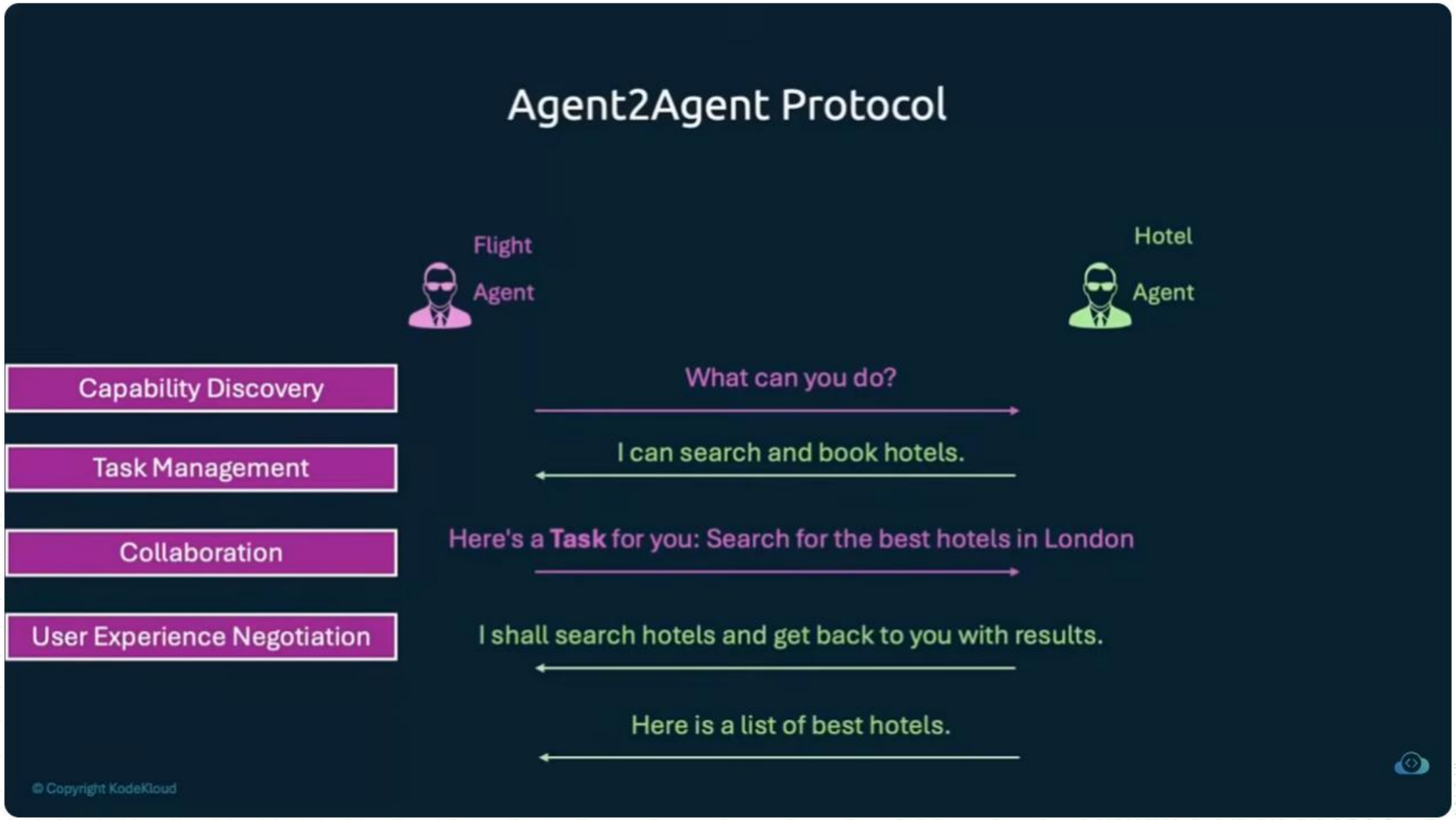
Google Agent2Agent Protocol (A2A)

A2A 是「Agent-to-Agent」的縮寫，意指代理對代理通訊協議。是一種為 AI 代理（Agentic AI）所設計的開放通訊標準協議，使得來自不同平台、模型、供應商的 AI 代理能夠彼此協作、溝通與交換資訊，進而完成跨系統、跨流程的自動化任務。

其中所謂「Agentic AI」，是指具有自主決策、感知、學習與協作能力的人工智慧代理，它們能主動完成任務、與其他代理互動協作，並根據情境調整行為策略。例如，在一個智慧客服系統中，接單代理、物流代理與帳務代理可透過 A2A 協議串接協作，各自處理不同任務，同步狀態並整合結果，實現無縫的自動化流程。

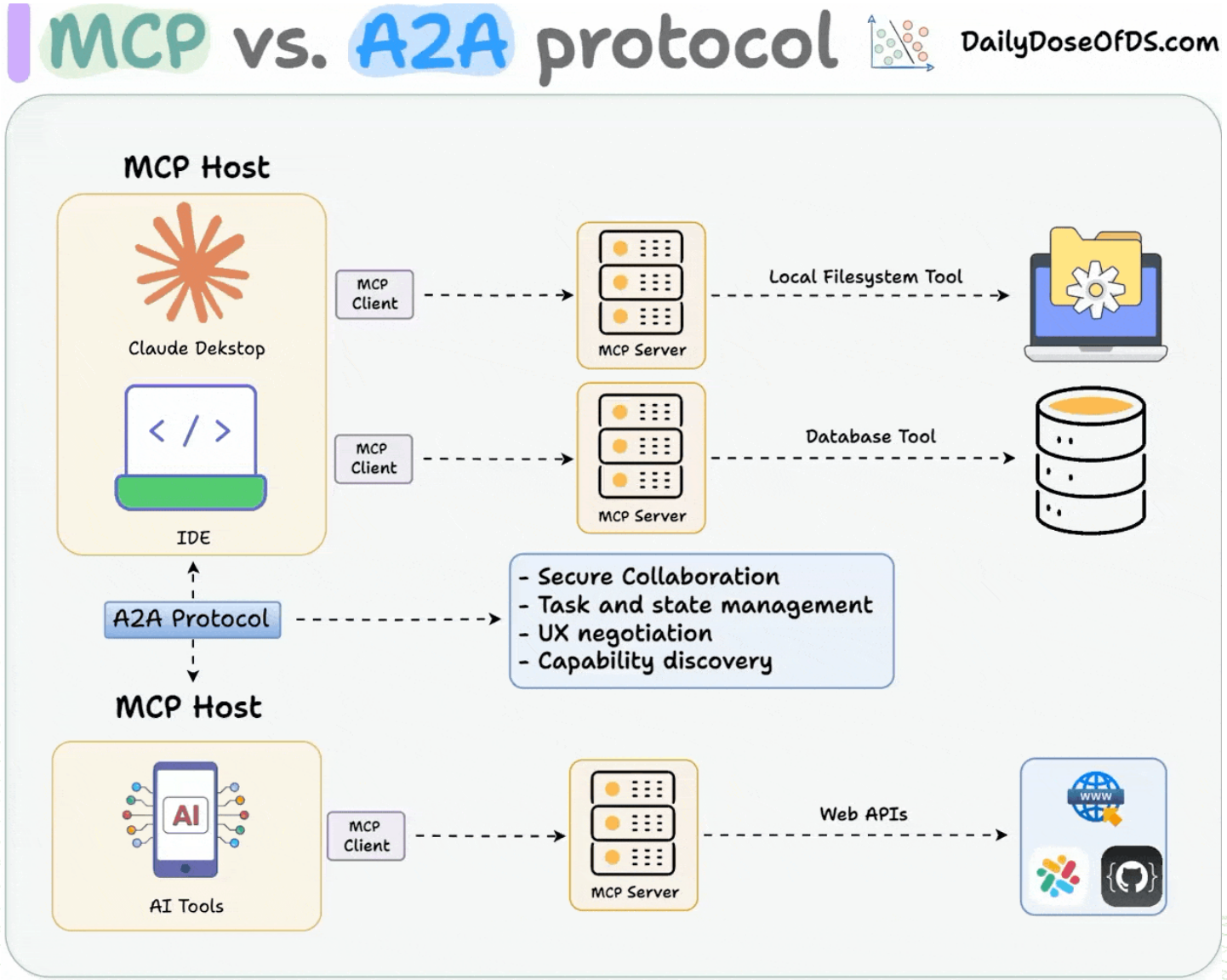


Google Agent2Agent Protocol (A2A)



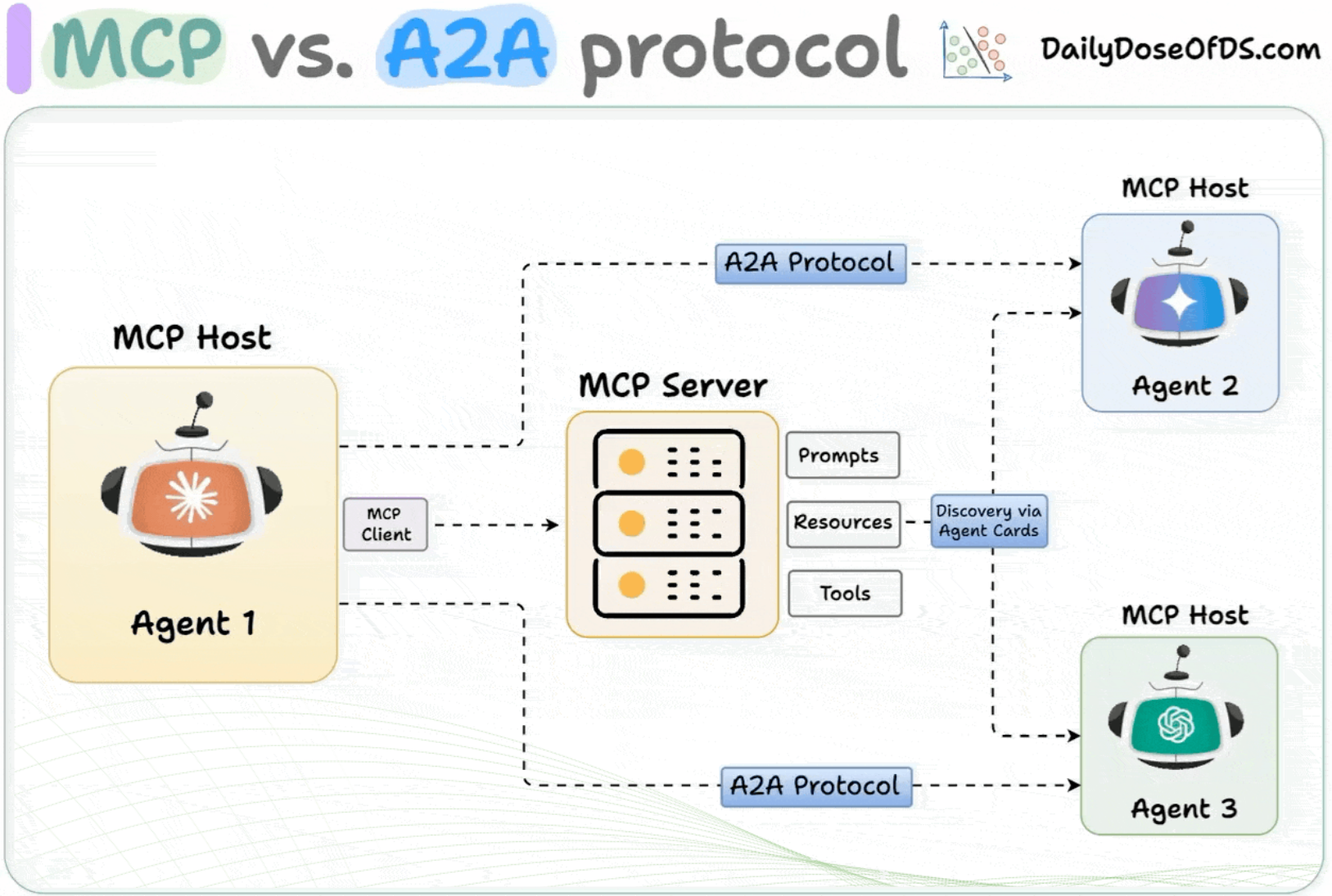
Model Context Protocol (MCP) vs. Agent2Agent Protocol (A2A)

- Agentic applications require both A2A and MCP.
- MCP provides agents with access to tools/APIs, DBs, Files, etc.
- While A2A allows agents to connect with other agents and collaborate in teams.

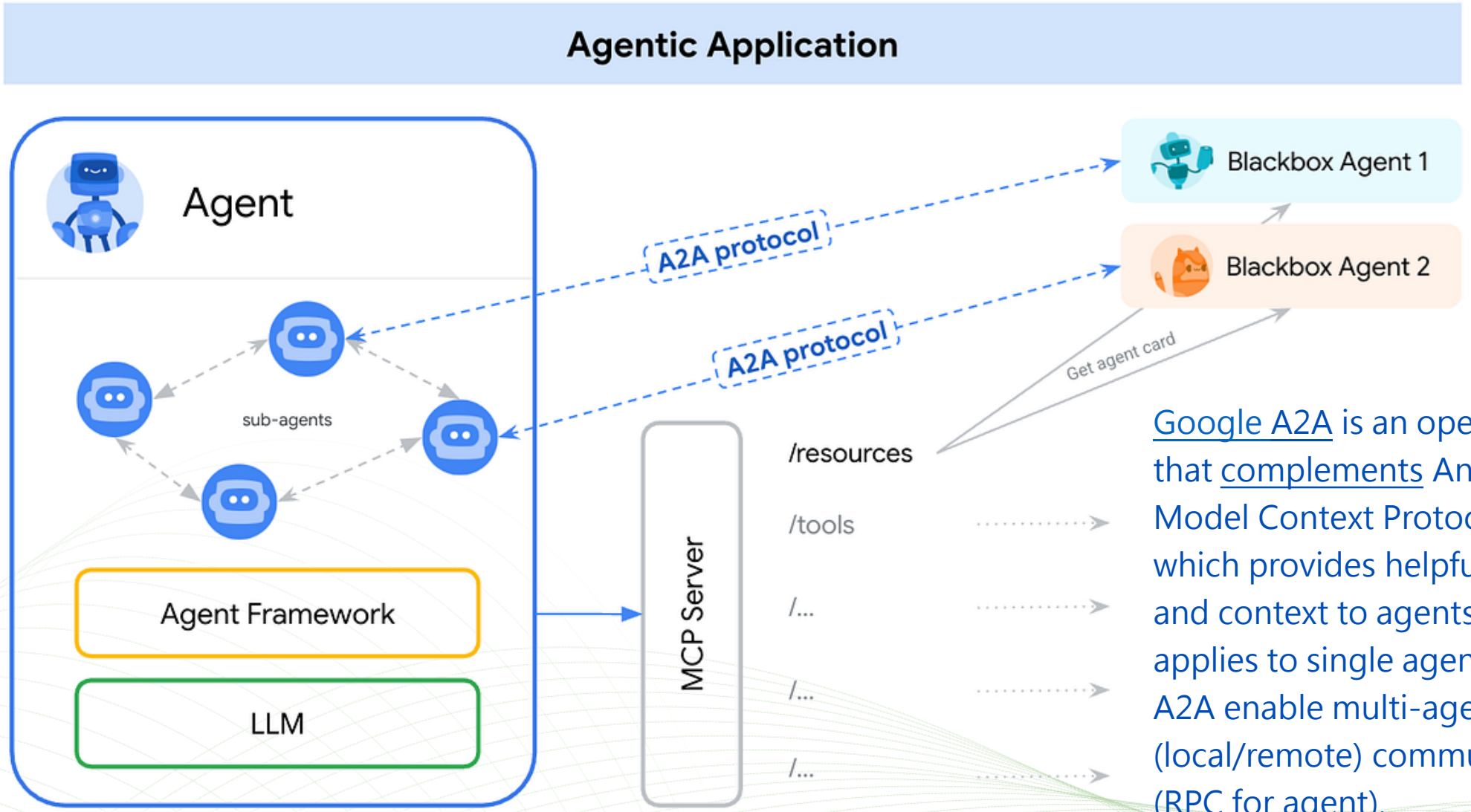


Model Context Protocol (MCP) vs. Agent2Agent Protocol (A2A)

- An agentic application might use A2A to communicate with other agents, while each agent internally uses MCP to interact with its specific tools and resources.



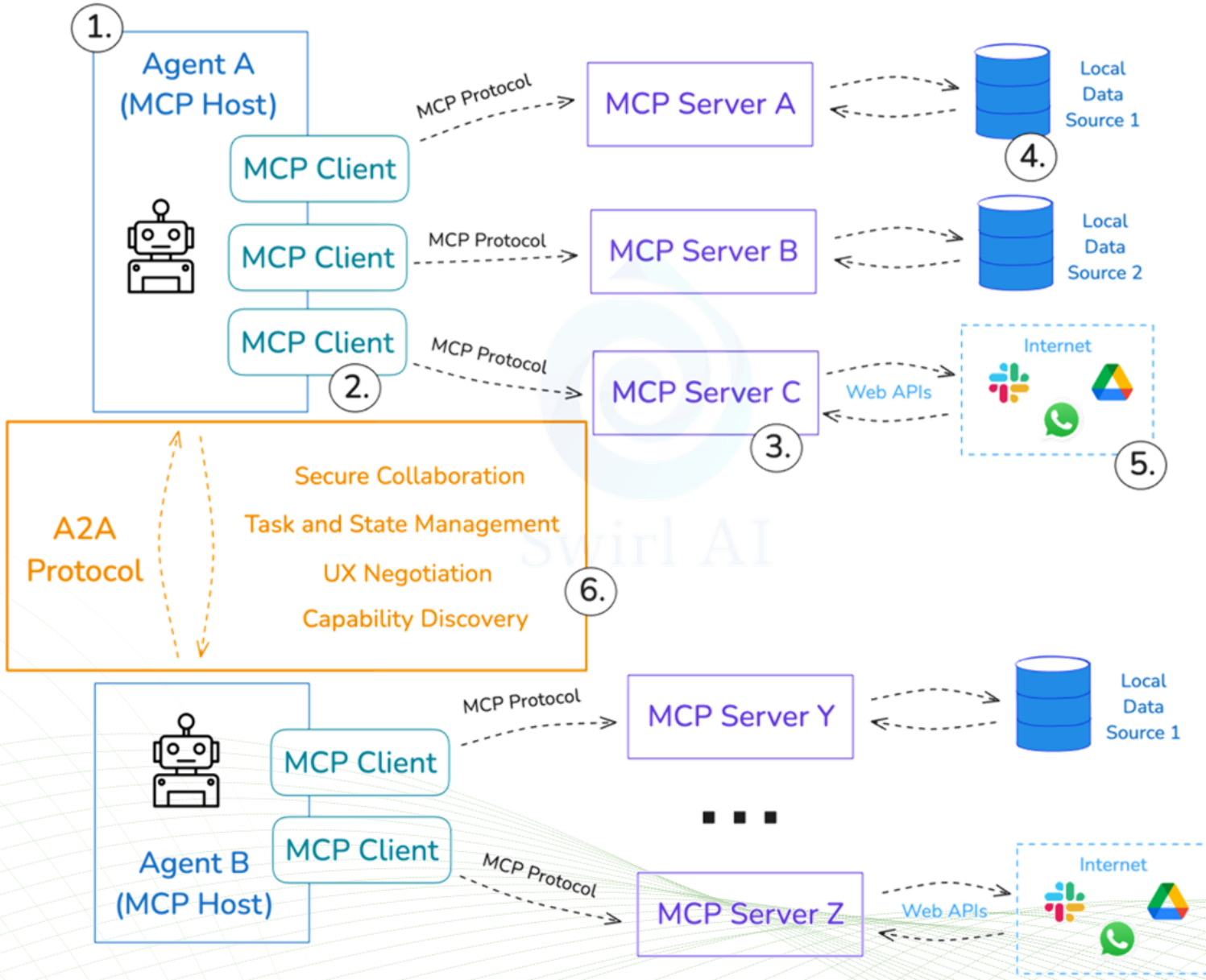
Model Context Protocol (MCP) vs. Agent2Agent Protocol (A2A)



Google A2A is an open protocol that complements Anthropic's Model Context Protocol (MCP), which provides helpful tools and context to agents. MCP applies to single agent, while A2A enable multi-agent (local/remote) communication (RPC for agent).

Model Context Protocol (MCP) vs. Agent2Agent Protocol (A2A)

- An agent application might use A2A to communicate with other agents, while internally, each agent employs MCP to interact with its specific tools and resources.
- A2A communication must occur over HTTP(S), with all request and response payloads formatted using JSON-RPC 2.0. The A2A server provides services at the URL defined in its AgentCard.



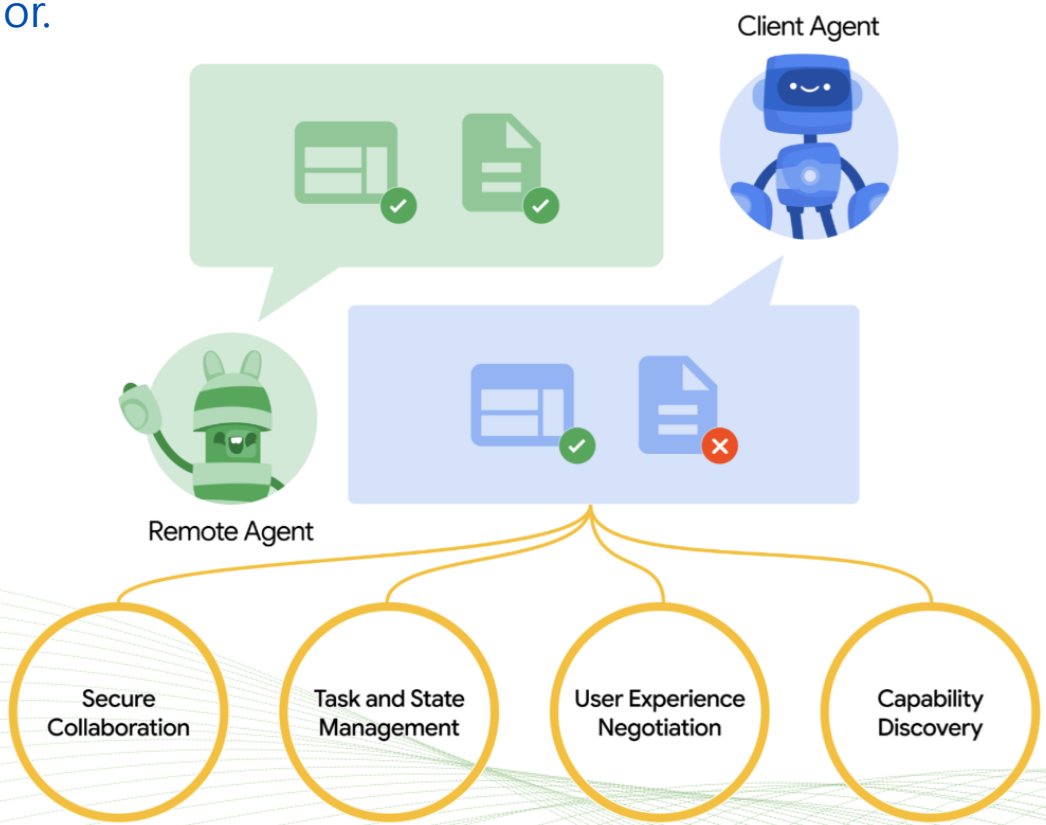
Jonathan Chen

Model Context Protocol (MCP) vs. Agent2Agent Protocol (A2A)

- MCP focuses on how agent interact with tools and data sources.
- Agen2Agent protocol solves a different challenge. How do agents communicate with each other? This is useful for systems where multiple specialized agents collaborate to solve complex problems. Agent2Agent is an open protocol that provides a standard way for agents to collaborate with each other regardless of the underlaying framework or vendor.
- Agent2Agent is complementary to MCP and not competitive with it. The protocol facilitates communication between what it calls a client agent and a remote agent. The client agent is responsible for formulating and communicating tasks while remote agent is responsible for acting those tasks to provide information or take action.

Use MCP for Tools.

Use Agent2Agent for Agents.



Model Context Protocol (MCP) vs. Agent2Agent Protocol (A2A) vs. Agent Communication Protocol (ACP)

Today’ s AI ecosystem faces challenge: how do we enable seamless communication across different AI platforms, models, and architectures? Three groundbreaking protocols - Model Context Protocol (MCP), Agent Communication Protocol (ACP), and Agent-to-Agent Protocol (A2A) - are emerging as the solution to this critical challenge.

MCP	ACP	A2A
anthropic	IBM	Google
Provides context to LLMs	Enables agent communication	Facilitates agent collaboration
Centralized hub-and-spoke	Protocol-driven, often decentralized	Decentralized peer-to-peer
Tool access and data retrieval	Multi-agent negotiation	Cross-platform workflows
Tool access and data retrieval	Best Use Case	Cross-platform workflows

	MCP Model Context Protocol	**ACP** Agent Communication Protocol	**A2A** Agent2Agent Protocol
Primary purpose	LLM communicates with data and tools	Agent to agent communication	Agent to agent communication
Key points	• User-controlled prompts • APP-controlled resources • Model-controlled tools	• Natural language flexibility • Model dependency management	• Capability discovery • Task Management • Collaboration • UX negotiation
Adoption	Wide developer adoption	Pre-alpha	Launched with support from 50+ tech partners

https://www.youtube.com/watch?v=iV3Iesez2s8&t=831s

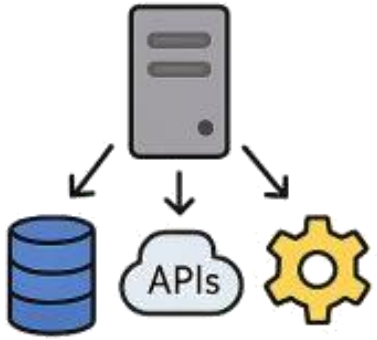
Jonathan Chen

Model Context Protocol (MCP) vs. Agent2Agent Protocol (A2A) vs. Agent Communication Protocol (ACP)

ACP -

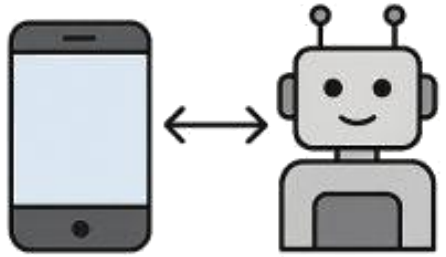
- Decentralized and optimized for local, offline environments like mobile phones or robots.
- Agents talk directly using flexible protocols (gRPC, ZeroMQ, IPC).

MCP
(Model Context Protocol)



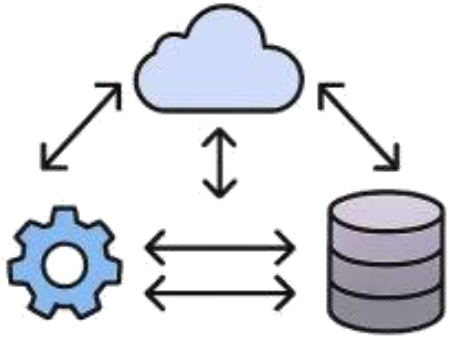
- Focuses on feeding external context (data/tools) into large models like LLMs.
- Useful when your LLM needs to pull information from APIs, databases, tools at runtime.
- Centralized: the server manages all the tool information.

ACP
(Agent Communication Protocol)



- Optimized for local, offline, embedded agent networks (e.g., a phone, a robot).
- Agents talk to each other directly, no external server needed.
- Very flexible with protocols (gRPC, IPC, ZeroMQ)

A2A
(Agent-to-Agent Protocol)



- Built for cross-platform, enterprise-scale collaboration.
- Agents on different devices, clouds, or organizations can communicate securely.
- Designed for distributed workflows where multiple agents (LLMs, databases, planners) collaborate horizontally.