<u>Model Context Protocol:</u>

Idea behind MCP:

Model Context Protocol (MCP) is essentially about giving AI models a **standardized, secure, and structured way** to access external tools, data, and systems — without hardcoding integrations for every use case.

Motivations for MCP :

1. MCP was created because Agents / LLMs needed to access real time information from other systems or sources or services
2. Earlier what was happening is , whenever the third party tools make changes to their API , the Agent needs to change their code in application layer as well.

   After MCP → Now if other tools make change , they need to change their MCP server side code ; client side is unimpacted. So the application layer is decoupled from this

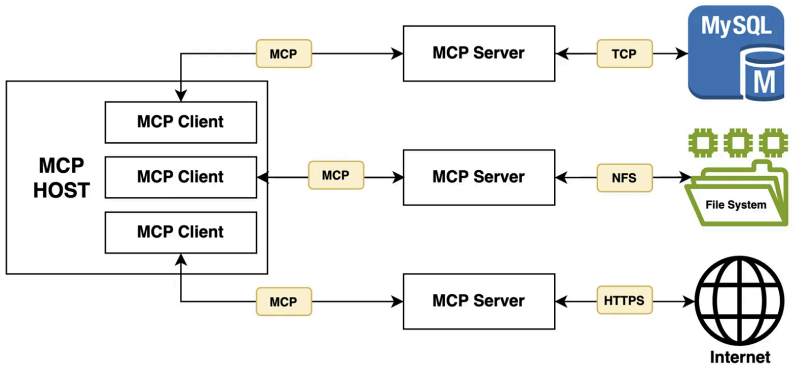
3. There was risk that different tools / vendors can come up with their own format
4. MCP limits the scope (like API) on what data it can access limiting unwanted access of data
5. **<u>If everyone follows MCP</u>**, irrespective of model from OpenAI, Anthropic, or your in-house LLM can talk to the same set of MCP-enabled services without rewriting integration code.

   (U have USB port in Laptop → u can connect Mouse, Keyboard, Digital Pen , Harddisk)

<u>Think of MCP as REST API for AI System</u>

1. Rest -> Standard for apps to communicate over HTTP ; MCP ->Standard for Agents to communicate with tools, data, and actions
2. Rest -> API endpoints expose functionality ; In MCP -> “resources” expose tools, and database etc or could be other things as well
3. Rest ->Any HTTP client can call any REST API ; MCP > Any MCP-compliant client can call any MCP-servers

MCP Components:



The key participants in the MCP architecture are:

> **MCP Host:** The AI application that coordinates and manages one or multiple MCP clients (Example : Claude desktop , Cursor IDE , Chat Application etc)
> **MCP Client:** A component that maintains a connection to an MCP server and obtains context from an MCP server for the MCP host to use
> **MCP Server:** A program that provides context to MCP clients

For example: **Claude Desktop** acts as an MCP host. When **Claude Desktop** establishes a connection to an MCP server, such as the Sentry MCP server, the **Claude Desktop** runtime instantiates an MCP client object that maintains the connection to the Sentry MCP server. When **Claude Desktop** subsequently connects to another MCP server, such as the local filesystem server, the **Claude Desktop** runtime instantiates an additional MCP client object to maintain this connection, hence maintaining a one-to-one relationship of MCP clients to MCP servers.

MCP consists of two layers:

1. **Data layer:** Defines the JSON-RPC based protocol for client-server communication, including lifecycle management, and core primitives, such as tools, resources, prompts and notifications. This is the a core part of MCP is defining the schema and semantics between MCP clients and MCP servers.

2. **Transport layer:** Defines the communication mechanisms and channels that enable data exchange between clients and servers, including transport-specific connection establishment, message framing, and authorization. **Two transport protocol supported** : Stdio transport: Uses standard input/output streams for direct process communication between local processes on the same machine, providing optimal performance with no network overhead.Streamable HTTP

transport: Uses HTTP POST for client-to-server messages with optional Server-Sent Events for streaming capabilities. This transport enables remote server communication and supports standard HTTP authentication methods including bearer tokens, API keys, and custom headers. MCP recommends using OAuth to obtain authentication tokens.

How MCP Works:

When a user interacts with a host application (an AI app) that supports MCP, several processes occur behind the scenes to enable quick and seamless communication between the AI and external systems. Let's take a closer look at what happens when a user asks Claude Desktop to perform a task that invokes tools outside the chat window.

Pre Conversation Protocol Handshake:

1. **Initial connection:** When an MCP client (like Claude Desktop) starts up, it connects to the configured MCP servers on your device.

2. **Capability discovery:** The MCP client asks each server "What capabilities do you offer?" Each server responds with its available tools, resources, and prompts.

3. **Registration**: The client registers these capabilities, making them available for the AI to use during your conversation.

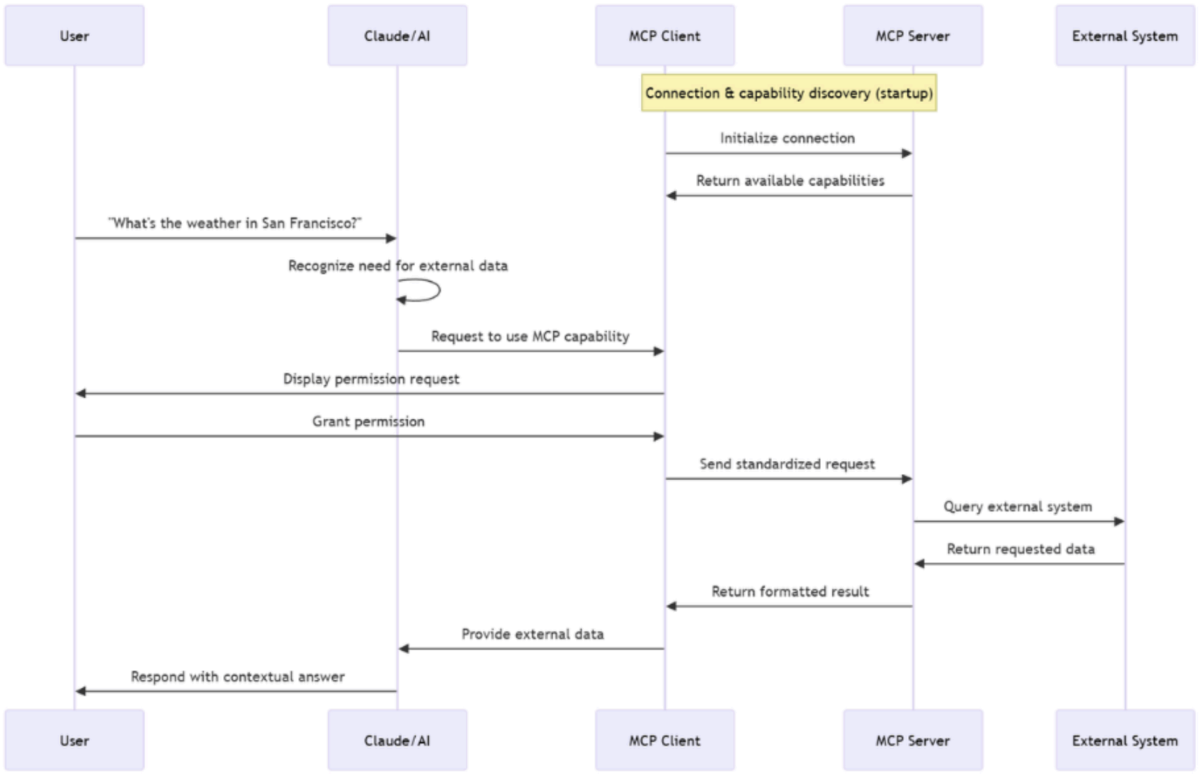
During Conversation :

Let's say you ask Claude, "What's the weather like in San Francisco today?" Here's what happens:

1. **Need recognition:** Claude using its LLM analyzes your question and recognizes it needs external, real-time information that wasn't in its training data.
2. **Tool or resource selection:** Claude identifies that it needs to use an MCP capability to fulfill your request.
3. **Permission request:** The client displays a permission prompt asking if you want to allow access to the external tool or resource.
4. **Information exchange:** Once approved, the client sends a request to the appropriate MCP server using the standardized protocol format.
5. **External processing:** The MCP server processes the request, performing whatever action is needed—querying a weather service, reading a file, or accessing a database.
6. **Result return:** The server returns the requested information to the client in a standardized format.
7. **Context integration:** Claude receives this information and incorporates it into its understanding of the conversation.

8. **Response generation:** Claude generates a response that includes the external information, providing you with an answer based on current data.

Ideally, this entire process happens in seconds, creating an unobtrusive experience where Claude appears to "know" information it couldn't possibly have from its training data alone.


User
Claude/AI
MCP Client
MCP Server
External System
Connection & capability discovery (startup)
Initialize connection
Return available capabilities
"What's the weather in San Francisco?"
Recognize need for external data
Request to use MCP capability
Display permission request
Grant permission
Send standardized request
Query external system
Return requested data
Return formatted result
Provide external data
Respond with contextual answer

Demo:

1. Integrating MCP servers in Claude Desktop
2. Integrating MCP servers in Cursor
3. https://smithery.ai/ repository of all publicly available MCP server
4. Creating custom MCP and connecting with it

5. Connecting with publicly available MCP servers