And a lot more...

- **Long-lived vs short-lived connections and streaming**
  Smooth transition between stateful and stateless capabilities
- **Additional clients with remote support**
  Expand client ecosystem to support remote MCP servers
- **Namespacing**
  Prevent collisions between entities of multiple servers; create logical groups of servers or tools
- **Elicitation**
  Servers proactively request context from users
- **Authentication and authorization**
  Updates to the specification to solidify OAuth2 for authentication alongside authorization schemes

Models are only as good as the **context** given to them

ANTHROP\C

DeepLearning.AI

What is the Model Context Protocol (MCP)

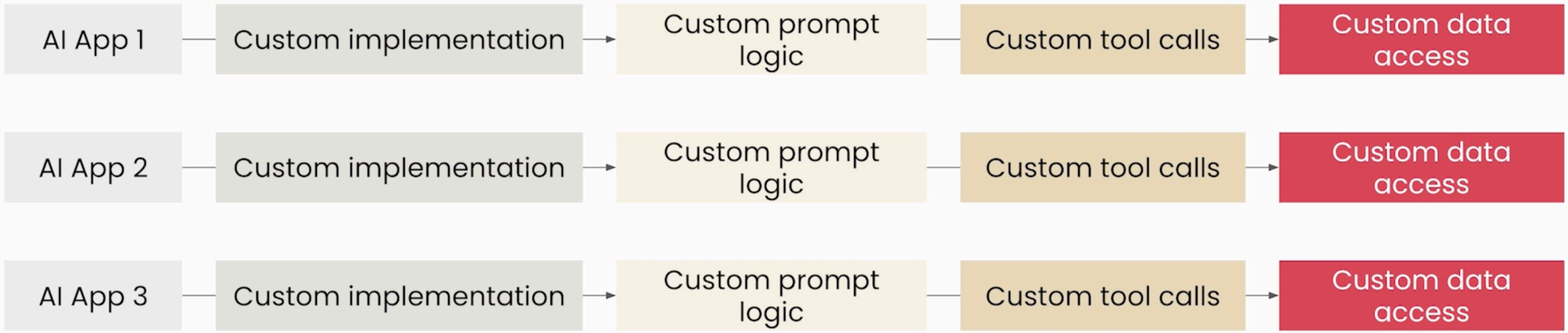
MCP is an open protocol that standardizes how your **LLM applications** connect to and work with your **tools & data sources**.

REST APIs
Standardize how **web applications** interact with the **backend**

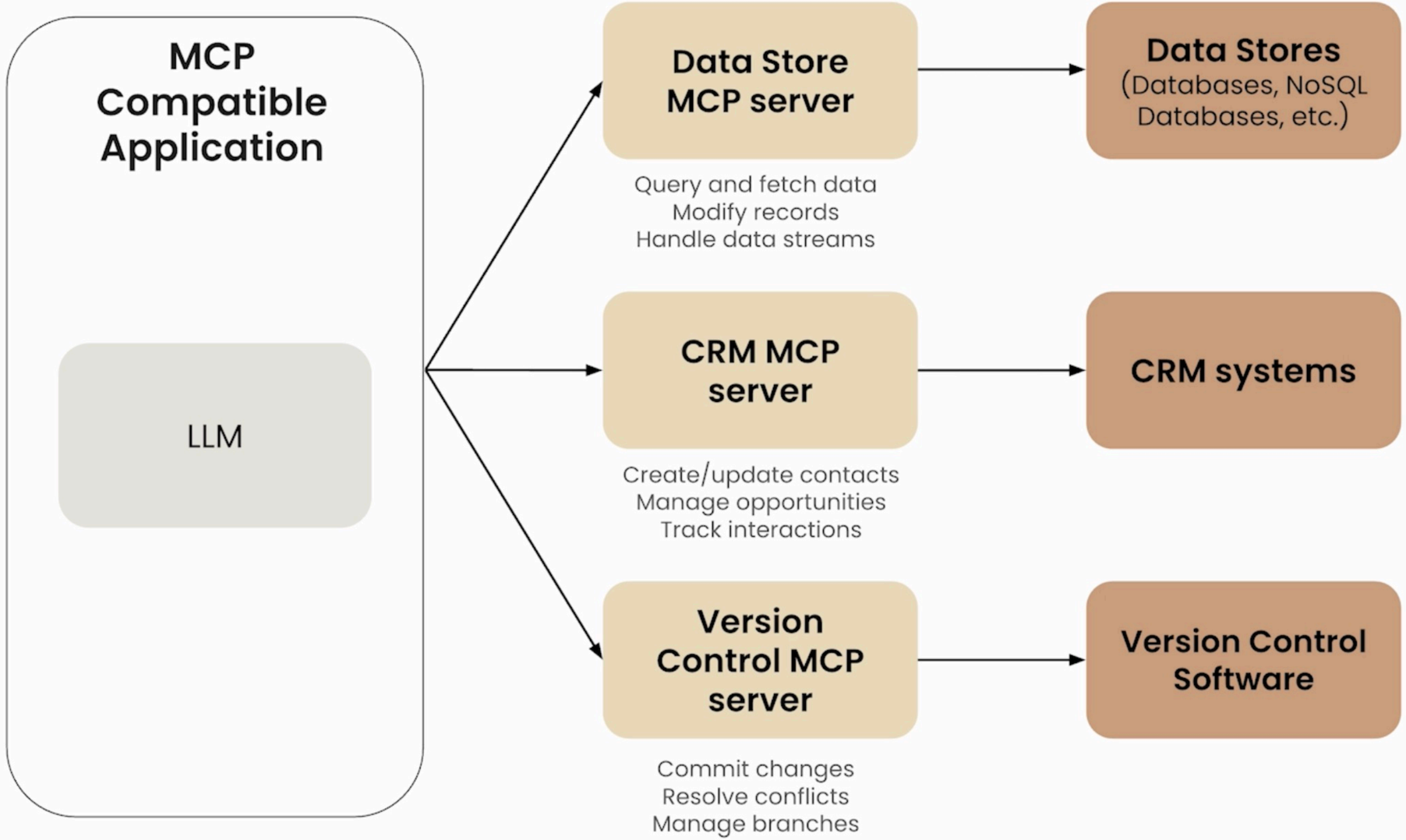
LSP
Standardizes how **IDEs** interact with **language-specific tools**

MCP
Standardizes how **AI applications** interact with **external systems**

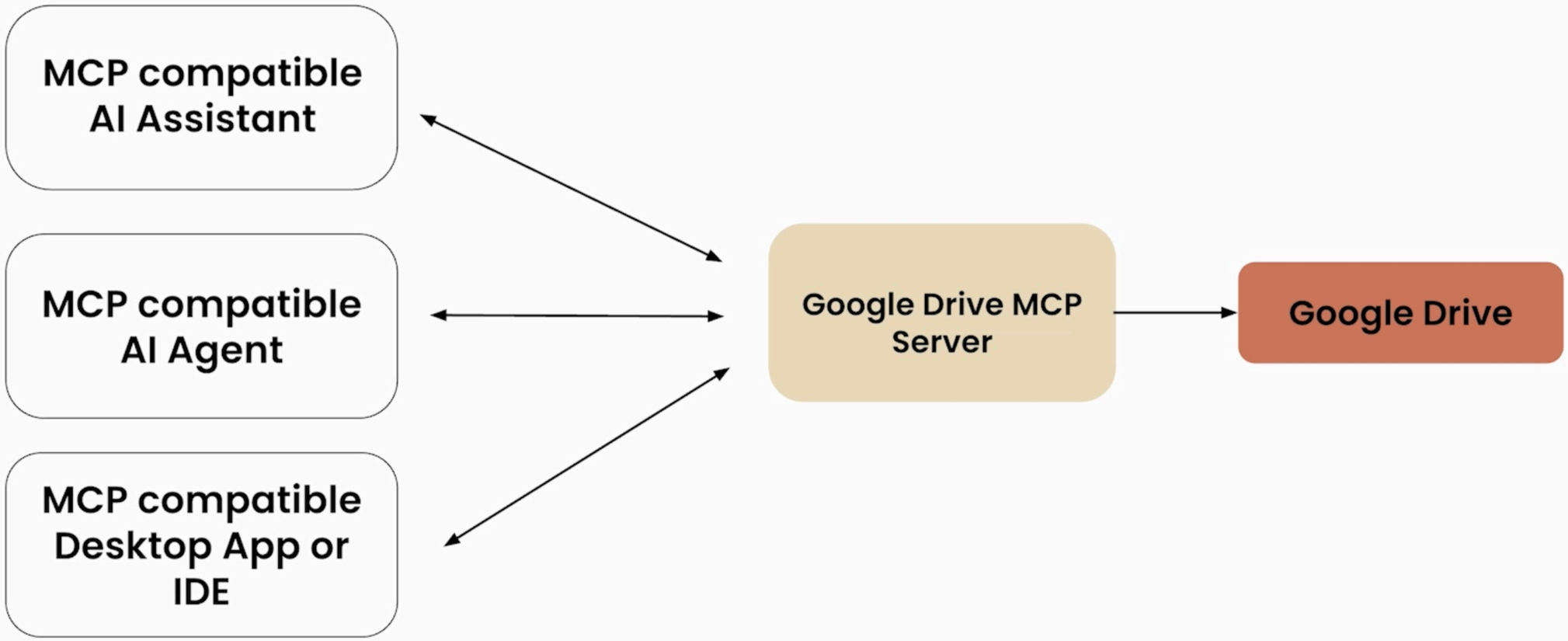
Without MCP: Fragmented AI Development



With MCP: Standardized AI Development



With MCP: MCP servers are reusable by various AI applications



With MCP: Standardized AI Development

- **For AI application developers**
  Connect your app to any MCP server with 0 additional work
- **For tool or API developers**
  Build an MCP server once which can be adopted everywhere
- **For AI application users**
  AI applications have extensive capabilities
- **For enterprises**
  Clear separation of concerns between AI product teams

The MCP Ecosystem is Growing Fast

AI Applications, from IDEs to Agents

Server Builders (3,000+ and counting)

Enterprises, for internal and external use cases

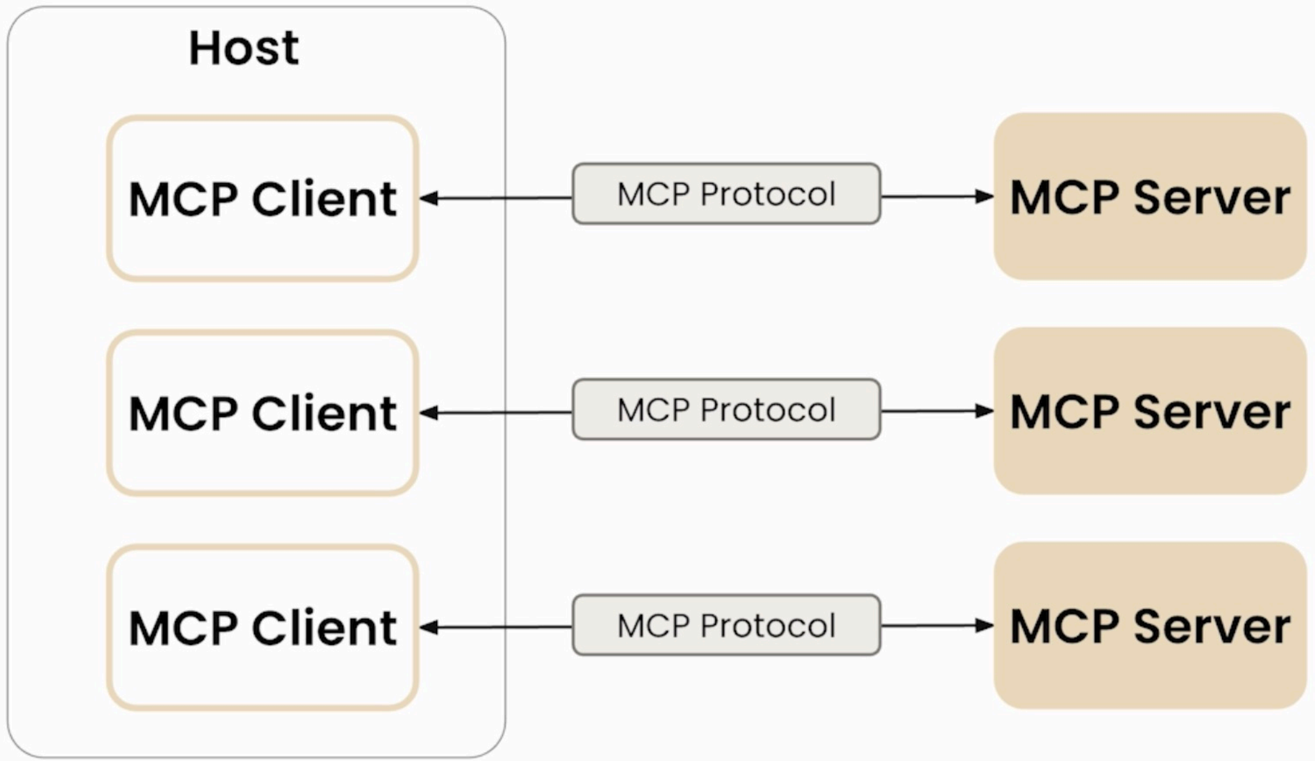
Open Source Contributors

And a lot more...

Common Questions

Who authors the MCP Server?	Anyone! Often the service provider itself will make their own MCP implementation. You can make a MCP server to wrap up access to some service.
How is using an MCP Server different from just calling a service's API directly?	MCP Servers provide tool schemas + functions. If you want to directly call an API directly, you'll be authoring those on your own.
Sounds like MCP Servers and tool use are the same thing.	MCP Servers provide tool schemas + functions already defined for you.

Client-Server Architecture

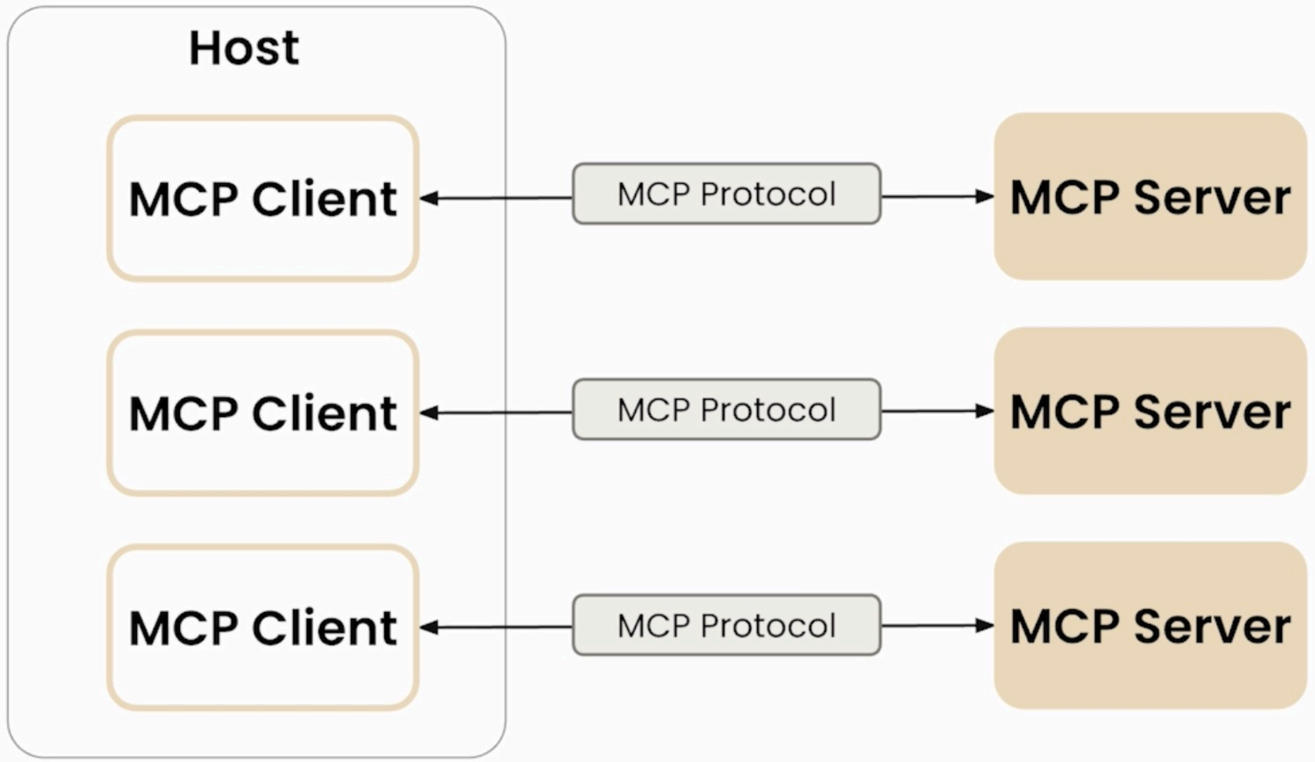


Client-Server Architecture

Host are LLM applications that want to access data through MCP (ex: Claude Desktop, IDEs, AI agents).

MCP Servers are lightweight programs that each expose specific capabilities through MCP.

MCP Clients maintain 1:1 connections with servers, inside the host application.



How does it work?

MCP Client

Invokes **Tools**
Queries for **Resources**
Interpolates **Prompts**

MCP Server

Exposes **Tools**
Exposes **Resources**
Exposes **Prompt Templates**

Tools

Functions and tools that can be invoked by the client

- Retrieve / search
- Send a message
- Update DB records

Resources

Read-only data or exposed by the server

- Files
- Database Records
- API Responses

Prompt Templates

Pre-defined templates for AI interactions

- Document Q&A
- Transcript Summary
- Output as JSON

Defining a Tool

- MCP provides SDK's for building servers and clients in a variety of languages
- You will be using the Python MCP SDK
- The Python SDK makes it very easy to declare tools

MCP Server

```python
@mcp.tool()
def add(a: int, b: int) -> int:
    """
    Args:
        a: First number to add
        b: Second number to add

    Returns:
        The sum of the two numbers
    """
    return a + b
```

Resources

- Allow the MCP Server to expose data to the client
- Similar to GET request handlers in a typical HTTP server
- Can return any type of data - strings, JSON, binary, etc.
    - You set the 'mime_type' to give the client a hint as to what data you are returning
- Two types: direct and templated

MCP Server

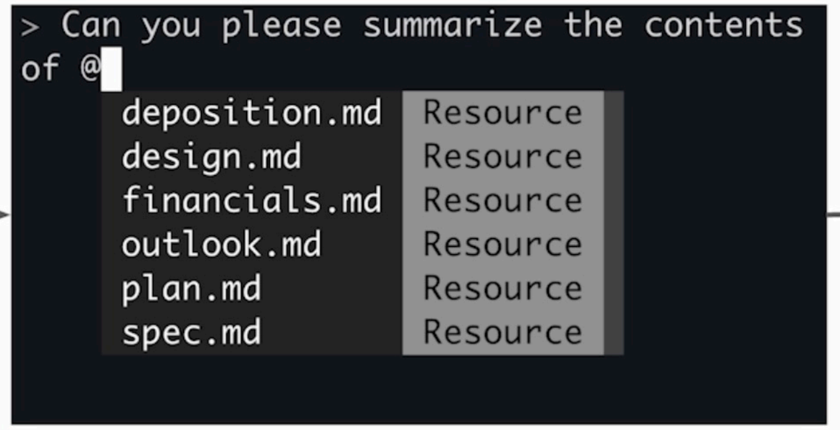
Direct

```
@mcp.resource(
    "docs://documents",
    mime_type="application/json"
)
def list_docs():
    # Return a list of document names
```

Templated

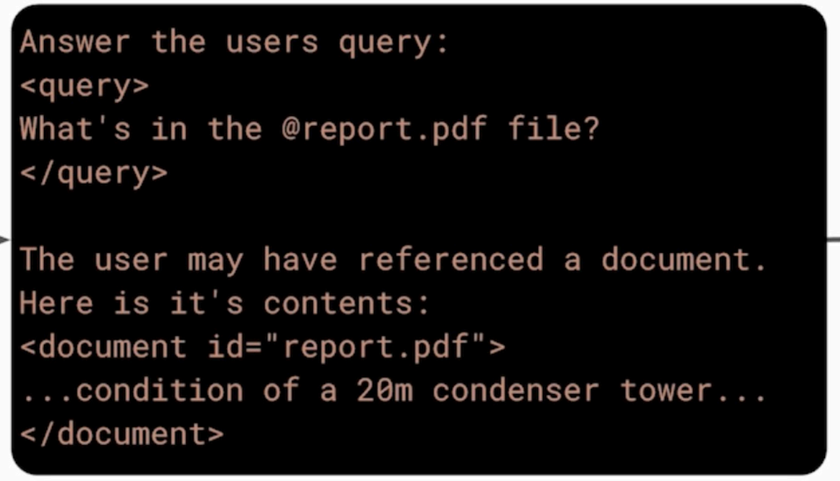
```
@mcp.resource(
    "docs://documents/{doc_id}",
    mime_type="text/plain"
)
def fetch_doc(doc_id: str):
    # Return the contents of a doc
```

Direct resources



...You need the MCP Server to give us a *list* of documents

Templated resources



You need the MCP Server to give us the contents of a *single* document

Prompts

- Defines a set of User and Assistant messages that can be used by the client
- These prompts should be high quality, and well-tested.

MCP Server

Prompt

```
@mcp.prompt(
    name="format",
    description="Rewrites the contents of
a document in Markdown format",
)
def format_document(
    doc_id: str,
) -> list[base.Message]:
    # Return a list of messages
```

ANTHROP\C

If you left this process up to a user, here's what they might write:

```
Convert report.pdf to markdown
```

Yes, it'd work, but the user might get a better result with some strong prompt engineering

DeepLearning.AI

They have more luck if they use a thoroughly evaluated prompt instead!

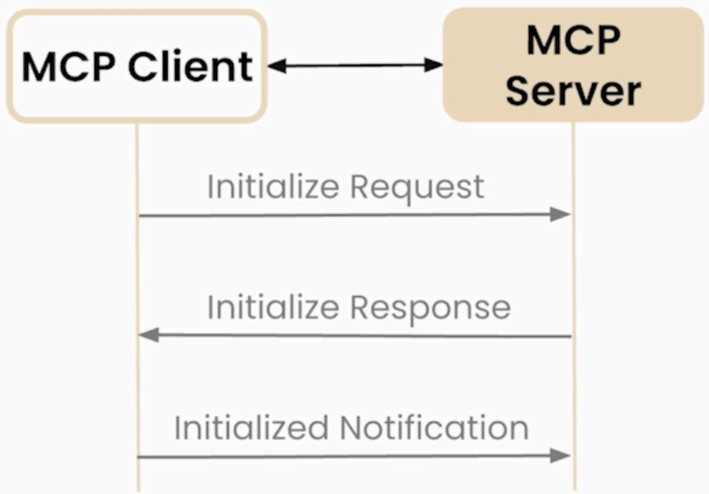
```
You are a document conversion specialist tasked with rewriting documents in Markdown
format. Your goal is to take the content of a given document and convert it into
well-structured Markdown, preserving the original meaning and enhancing readability.
Here is the identifier of the document you need to convert:
<document_id>
{{doc_id}}
</document_id>
Instructions:
1. Retrieve the content of the document associated with the given document_id.
2. Analyze the structure and content of the document.
3. Convert the document to Markdown format, following these guidelines:
- Use appropriate header levels (# for main titles, ## for subtitles, etc.)
- Properly format lists (both ordered and unordered)
- Use emphasis (*italic* or **bold**) where appropriate
- Add links and images using Markdown syntax if present in the original document
- Preserve any special formatting or structure that's important to the document's
meaning
Before providing the final Markdown output, in <document_analysis> tags:
- Identify the main sections and subsections of the document
- Count the number of sections and subsections to ensure proper nesting of headers
This will help ensure a thorough and well-organized conversion.
After your analysis, present the converted document in Markdown format. Use ``` markers
to denote the beginning and end of the Markdown content.
Example output structure:
<document_analysis>
[Your analysis of the document structure and conversion plan]
</document_analysis>
```markdown
# Document Title
## Section 1
Content of section 1...
## Section 2
Content of section 2...
- List item 1
- List item 2
[Link text](https://example.com)
![Image description](image-url.jpg)
```
Please proceed with your analysis and conversion of the document.
```

ANTHROP\C

DeepLearning.AI

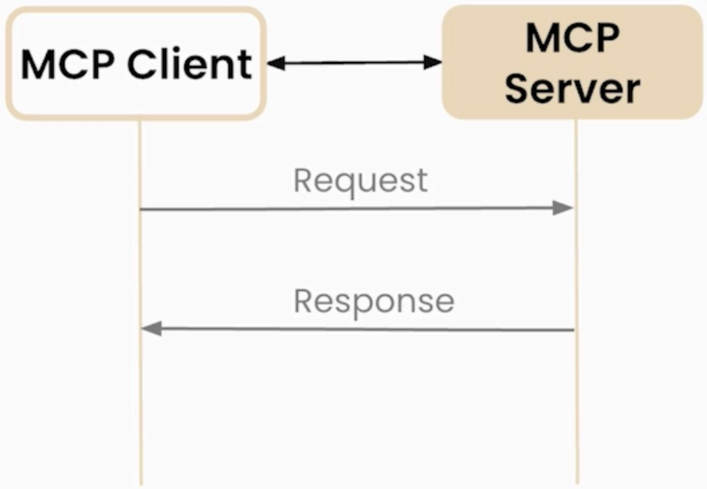
Communication Lifecycle

1. Initialization



Connection is established

2. Message Exchange



3. Termination

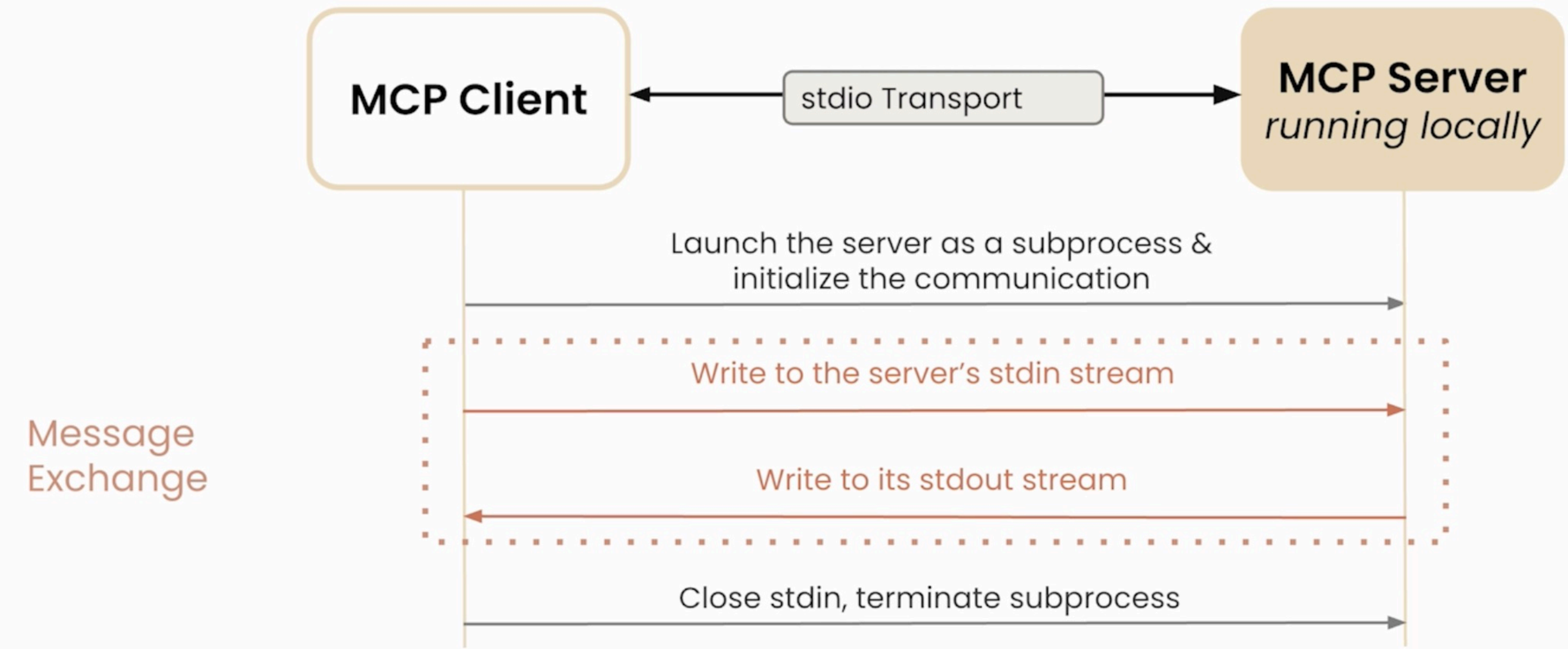
MCP Transports

A transport handles the underlying mechanics of how messages are sent and received between the client and server.

1. For servers running locally: **stdio** (standard input output)
2. For remote servers:
   a. **HTTP+SSE (Server Sent Events)** (from protocol version 2024-11-05)
   b. **Streamable HTTP** (as of protocol version 2025-03-26)

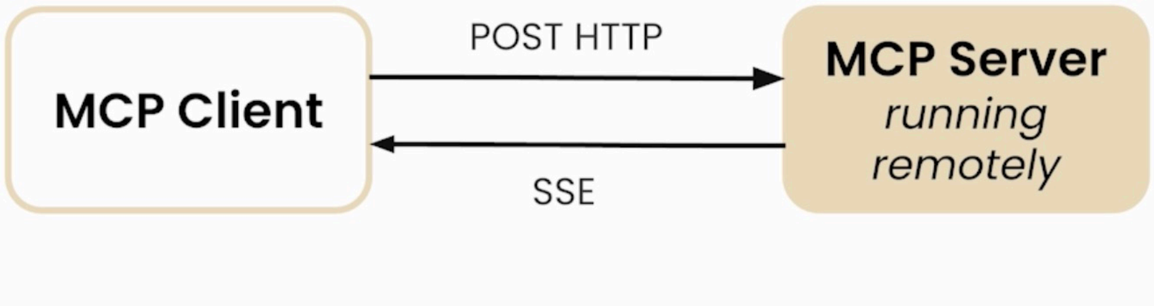
Standard IO (stdio) Transport

When running servers locally, stdio is most commonly used



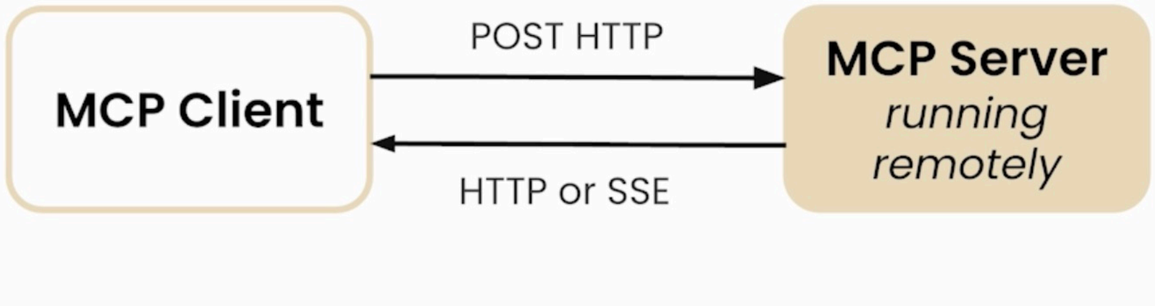
Transports for Remote Servers

HTTP + SSE
(from protocol version 2024-11-05)



Stateful Connection

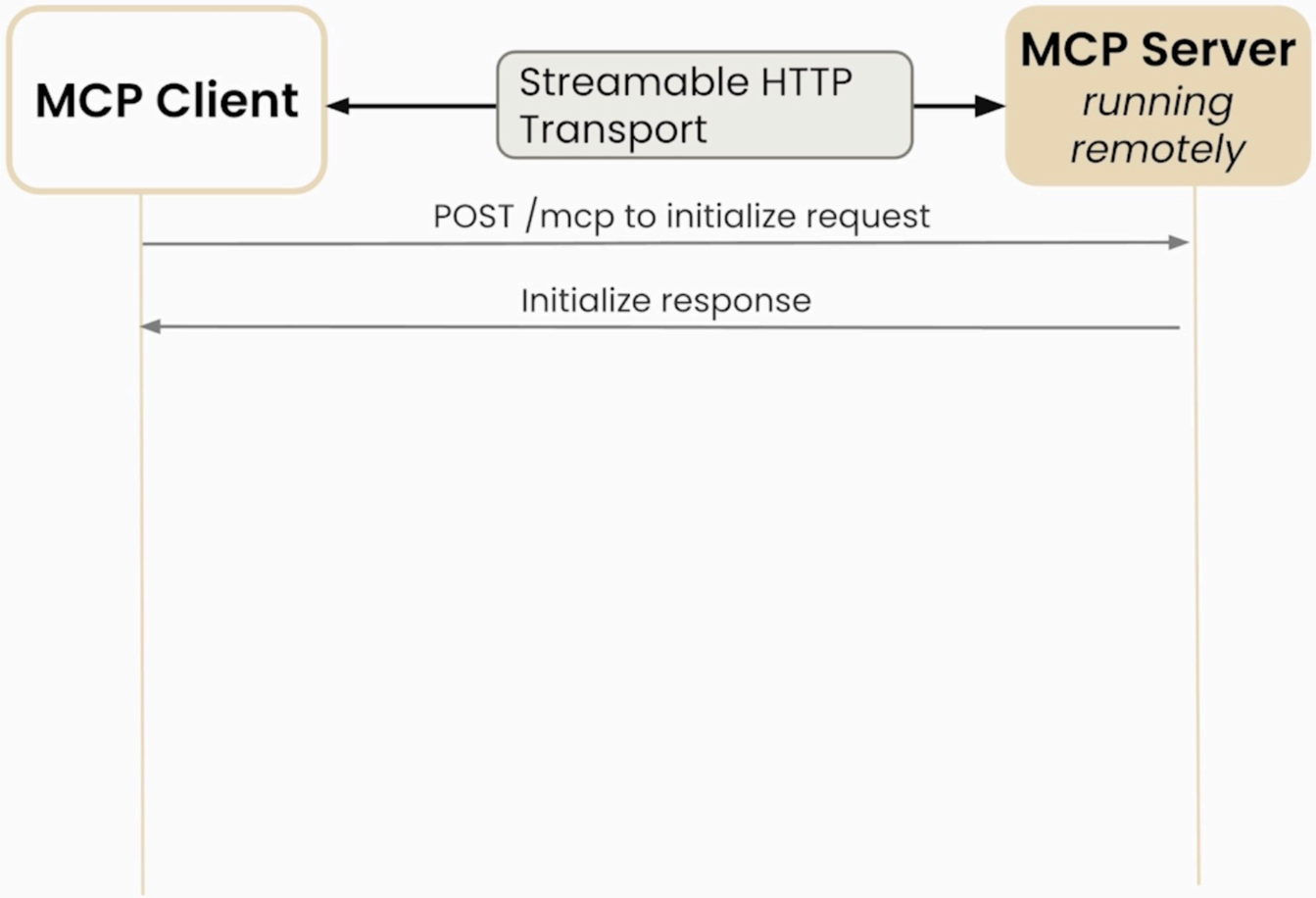
Streamable HTTP
(as of protocol version 2025-03-05)



Allow for Stateless or Stateful Connection

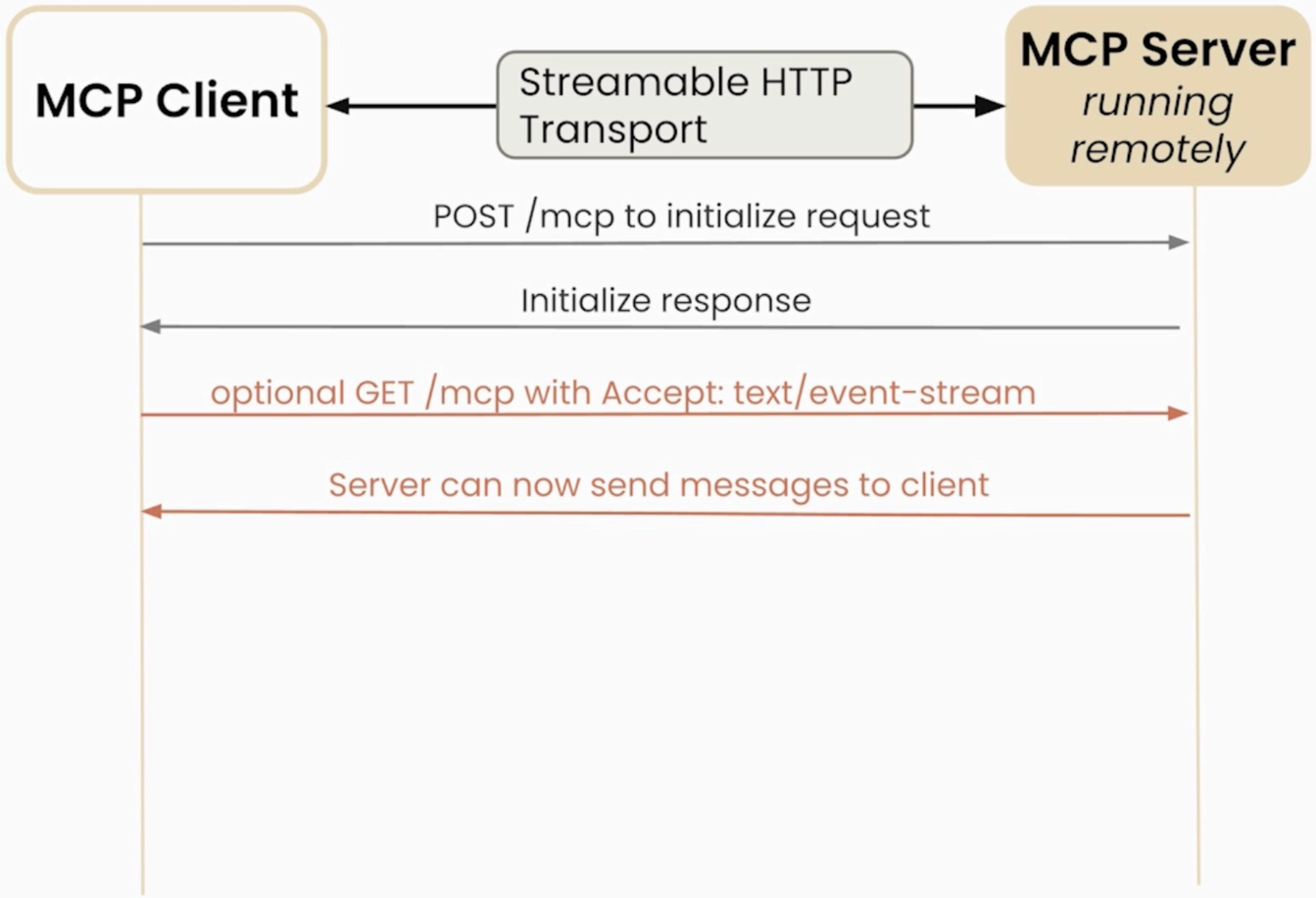
Streamable HTTP Transport

For remote MCP servers the Streamable HTTP transport is used. This supports stateless connections as well as stateful connections with the ability to opt into Server Sent Events



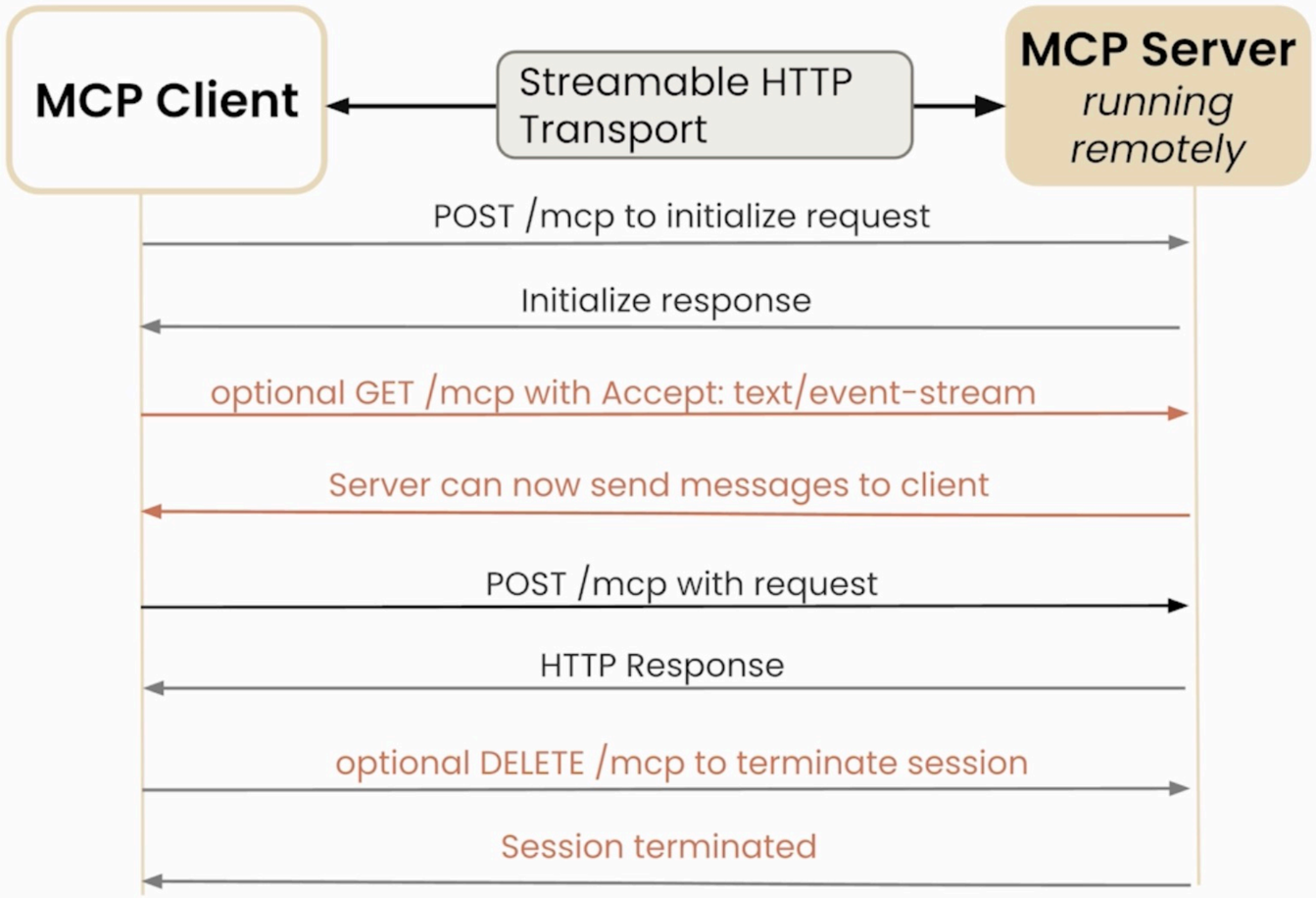
Streamable HTTP Transport

For remote MCP servers the Streamable HTTP transport is used. This supports stateless connections as well as stateful connections with the ability to opt into Server Sent Events



Streamable HTTP Transport

For remote MCP servers the Streamable HTTP transport is used. This supports stateless connections as well as stateful connections with the ability to opt into Server Sent Events



package-lock.json	Update `server-everything` to use the latest ver...	2 days ago
package.json	removed unused files, this has been moved to it...	5 months ago
tsconfig.json	Create package for each server	6 months ago

README | Code of conduct | MIT license | Security

Model Context Protocol servers

This repository is a collection of *reference implementations* for the Model Context Protocol (MCP), as well as references to community built servers and additional resources.

The servers in this repository showcase the versatility and extensibility of MCP, demonstrating how it can be used to give Large Language Models (LLMs) secure, controlled access to tools and data sources. Each MCP server is implemented with either the Typescript MCP SDK or Python MCP SDK.

> Note: Lists in this README are maintained in alphabetical order to minimize merge conflicts when adding new items.

🌟 Reference Servers

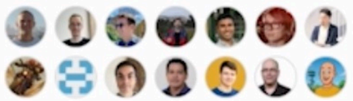
These servers aim to demonstrate MCP features and the TypeScript and Python SDKs.

- **AWS KB Retrieval** - Retrieval from AWS Knowledge Base using Bedrock Agent Runtime
- **Brave Search** - Web and local search using Brave's Search API
- **EverArt** - AI image generation using various models
- **Everything** - Reference / test server with prompts, resources, and tools
- **Fetch** - Web content fetching and conversion for efficient LLM usage
- **Filesystem** - Secure file operations with configurable access controls
- **Git** - Tools to read, search, and manipulate Git repositories
- **GitHub** - Repository management, file operations, and GitHub API integration
- **GitLab** - GitLab API, enabling project management
- **Google Drive** - File access and search capabilities for Google Drive
- **Google Maps** - Location services, directions, and place details
- **Memory** - Knowledge graph-based persistent memory system
- **PostgreSQL** - Read-only database access with schema inspection
- **Puppeteer** - Browser automation and web scraping
- **Redis** - Interact with Redis key-value stores
- **Sentry** - Retrieving and analyzing issues from Sentry.io
- **Sequential Thinking** - Dynamic and reflective problem-solving through thought sequences
- **Slack** - Channel management and messaging capabilities
- **Sqlite** - Database interaction and business intelligence capabilities
- **Time** - Time and timezone conversion capabilities

🤝 Third-Party Servers

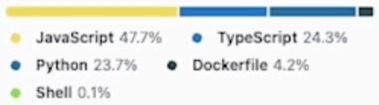
🎖️ Official Integrations

Contributors 461



+ 447 contributors

Languages



README.md

Fetch MCP Server

A Model Context Protocol server that provides web content fetching capabilities. This server enables LLMs to retrieve and process content from web pages, converting HTML to markdown for easier consumption.

The fetch tool will truncate the response, but by using the `start_index` argument, you can specify where to start the content extraction. This lets models read a webpage in chunks, until they find the information they need.

Available Tools

- `fetch` - Fetches a URL from the internet and extracts its contents as markdown.
  - `url` (string, required): URL to fetch
  - `max_length` (integer, optional): Maximum number of characters to return (default: 5000)
  - `start_index` (integer, optional): Start content from this character index (default: 0)
  - `raw` (boolean, optional): Get raw content without markdown conversion (default: false)

Prompts

- **fetch**
  - Fetch a URL and extract its contents as markdown
  - Arguments:
    - `url` (string, required): URL to fetch

Installation

Optionally: Install node.js, this will cause the fetch server to use a different HTML simplifier that is more robust.

Using uv (recommended)

When using `uv` no specific installation is needed. We will use `uvx` to directly run *mcp-server-fetch*.

Using PIP

Alternatively you can install `mcp-server-fetch` via pip:

```
pip install mcp-server-fetch
```

After installation, you can run it as a script using:

```
python -m mcp_server_fetch
```

Configuration

Configure for Claude.app

Add to your Claude settings:

▶ Using uvx

burkeholland Move VS Code below Claude 090b6b7 · last month History

Name	Last commit message	Last commit date
..		
Dockerfile	fix warnings: - FromAsCasing: 'as' and 'FROM' keywords' casing do not...	4 months ago
README.md	Move VS Code below Claude	last month
index.ts	allow ~ to be used in config	2 months ago
package.json	Add Dockerfiles for the 17 sample MCP servers	5 months ago
tsconfig.json	Updated Filesystem	6 months ago

README.md

Filesystem MCP Server

Node.js server implementing Model Context Protocol (MCP) for filesystem operations.

Features

- Read/write files
- Create/list/delete directories
- Move files/directories
- Search files
- Get file metadata

Note: The server will only allow operations within directories specified via `args`.

API

Resources

- `file://system`: File system operations interface

Tools

- **read_file**
  - Read complete contents of a file
  - Input: `path` (string)
  - Reads complete file contents with UTF-8 encoding

burkeholland Move VS Code below Claude 090b6b7 · last month History

Name	Last commit message	Last commit date
..		
Dockerfile	fix warnings: - FromAsCasing: 'as' and 'FROM' keywords' casing do not...	4 months ago
README.md	Move VS Code below Claude	last month
index.ts	allow ~ to be used in config	2 months ago
package.json	Add Dockerfiles for the 17 sample MCP servers	5 months ago
tsconfig.json	Updated Filesystem	6 months ago

README.md

Filesystem MCP Server

Node.js server implementing Model Context Protocol (MCP) for filesystem operations.

Features

- Read/write files
- Create/list/delete directories
- Move files/directories
- Search files
- Get file metadata

Note: The server will only allow operations within directories specified via `args`.

API

Resources

- `file://system`: File system operations interface

Tools

- **read_file**
  - Read complete contents of a file
  - Input: `path` (string)
  - Reads complete file contents with UTF-8 encoding

```
{
  "mcpServers": {
    "filesystem": {
      "command": "docker",
      "args": [
        "run",
        "-i",
        "--rm",
        "--mount", "type=bind,src=/Users/username/Desktop,dst=/projects/Desktop",
        "--mount", "type=bind,src=/path/to/other/allowed/dir,dst=/projects/other/allowed/dir,ro",
        "--mount", "type=bind,src=/path/to/file.txt,dst=/projects/path/to/file.txt",
        "mcp/filesystem",
        "/projects"
      ]
    }
  }
}
```

NPX

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "/Users/username/Desktop",
        "/path/to/other/allowed/dir"
      ]
    }
  }
}
```

Usage with VS Code

For quick installation, click the installation buttons below...

VS Code NPM VS Code Insiders NPM

VS Code Docker VS Code Insiders Docker

For manual installation, add the following JSON block to your User Settings (JSON) file in VS Code. You can do this by pressing `Ctrl + Shift + P` and typing `Preferences: Open Settings (JSON)`.

Optionally, you can add it to a file called `.vscode/mcp.json` in your workspace. This will allow you to share the configuration with others.

> Note that the `mcp` key is not needed in the `.vscode/mcp.json` file.

You can provide sandboxed directories to the server by mounting them to `/projects`. Adding the `ro` flag will make the directory readonly by the server.

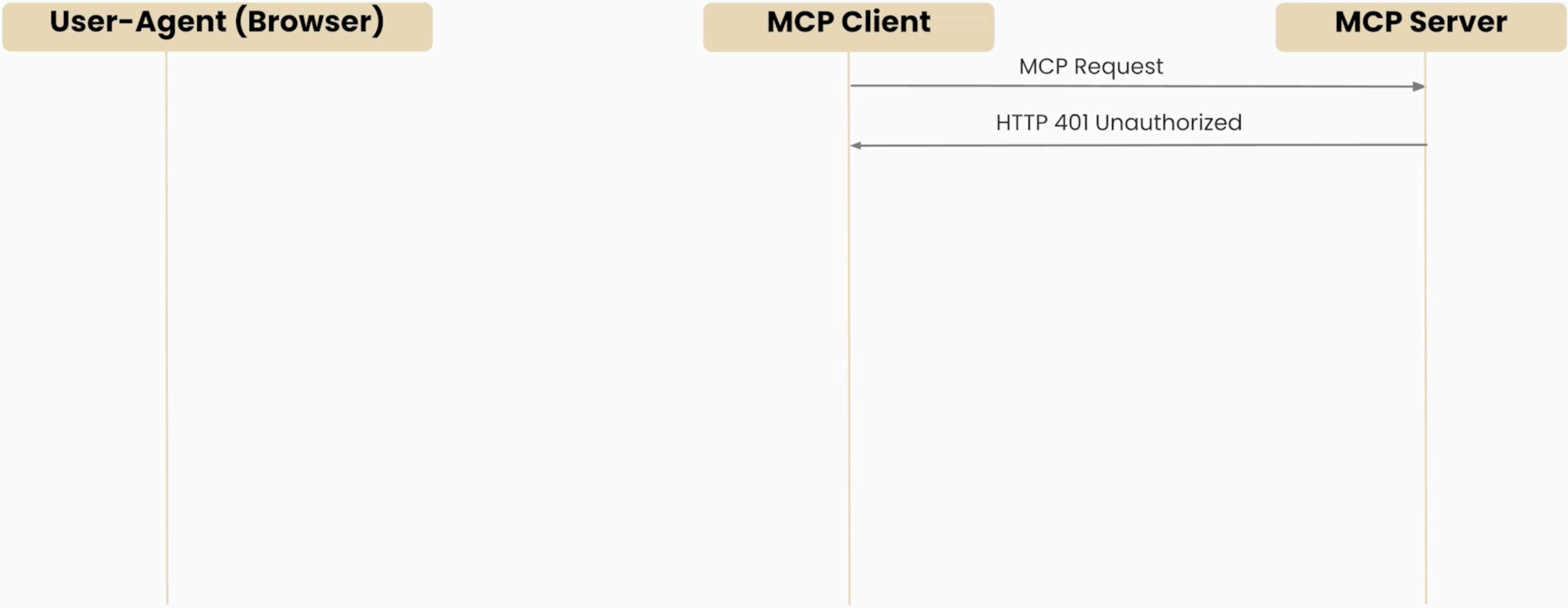
Docker

Note: all directories must be mounted to `/projects` by default.

```
{
  "mcp": {
```

Authentication

- **The March 2025 specification update enables MCP clients and servers to make use of OAuth 2.1**



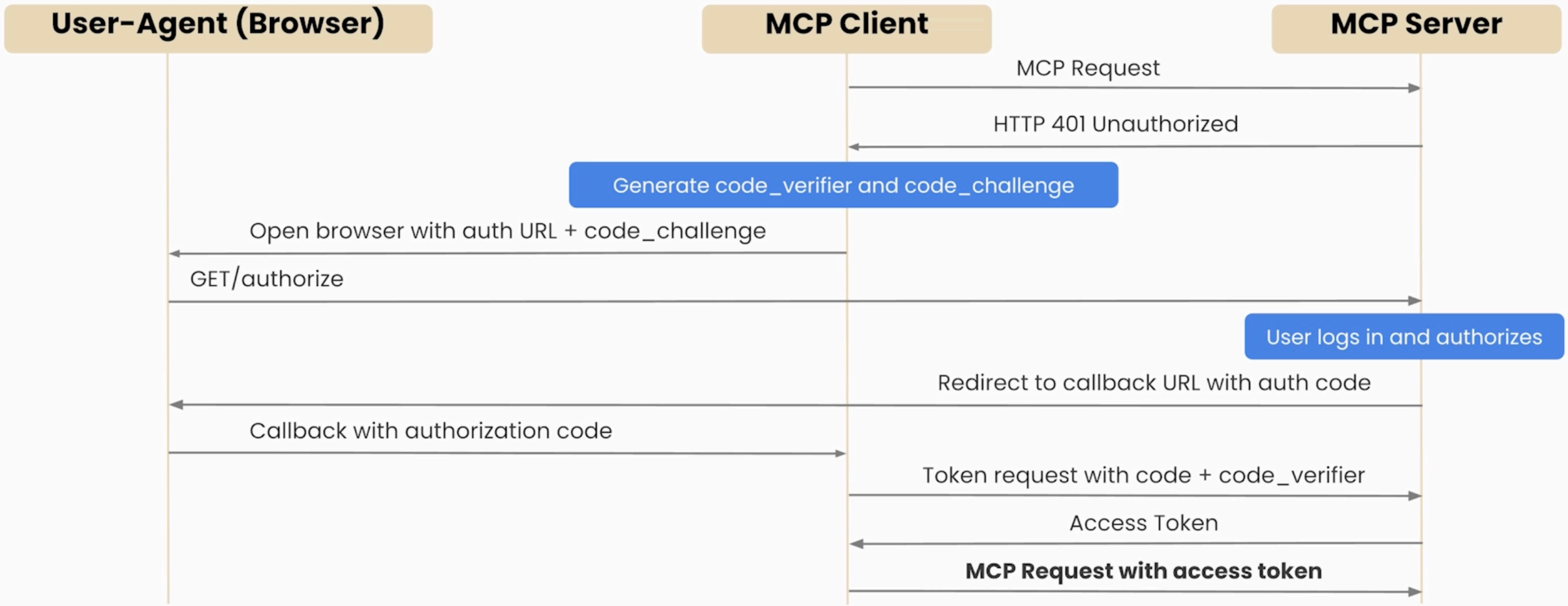
ANTHROP\C

DeepLearning.AI

Authentication

It's an active area of development as of March 2025. Please check git for latest updates

- **The March 2025 specification update enables MCP clients and servers to make use of OAuth 2.1**



Authentication

- **The March 2025 specification update enables MCP clients and servers to make use of OAuth 2.1**
- MCP servers implementing the new spec can authorize users to perform specific tasks and actions on existing services built on top of OAuth such as Auth0, Google APIs, GitHub, etc.
  - Optional feature in Model Context Protocol
  - STDIO transport can use environment variables
  - Remote servers need something different
- Built on established standards
  - OAuth 2.1 IETF DRAFT
  - OAuth 2.0 Authorization Server Metadata (RFC8414)
  - OAuth 2.0 Dynamic Client Registration Protocol (RFC7591)

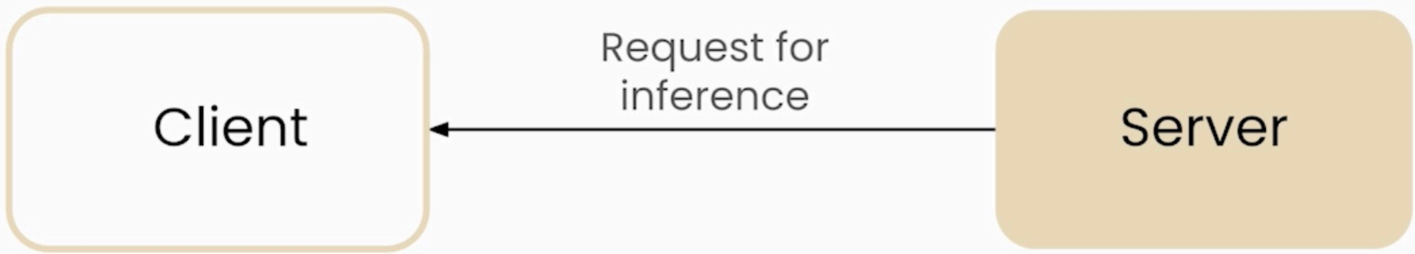
Client Primitives

Roots

- A root is a URI that a client suggests a server should operate.
- **"Only look in these specific folders for the files I need."**
- When a client connects to a server, it declares which roots the server should work with.
- While primarily used for filesystem paths, roots can be any valid URI including HTTP URLs
- **Benefits**:
  - Security: Limits server access to just the specified directories or endpoints
  - Clarity: Keeps the server focused on the relevant resources/locations
  - Versatility: While often used for file paths, roots can be any valid URI

Sampling

- Allows a server to **request inference from the LLM** they're connected to via the MCP client, giving the user application full control over security, privacy, and cost



- Sampling is a way for servers to leverage the LLM's intelligence as part of their processing pipeline.

ANTHROP\C

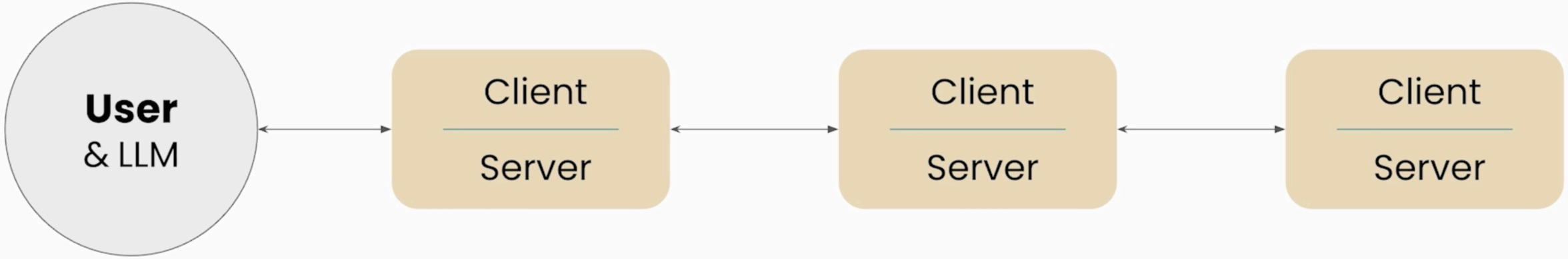
DeepLearning.AI

MCP will be the foundational protocol for agents

Building Effective Agents with MCP

Composability →

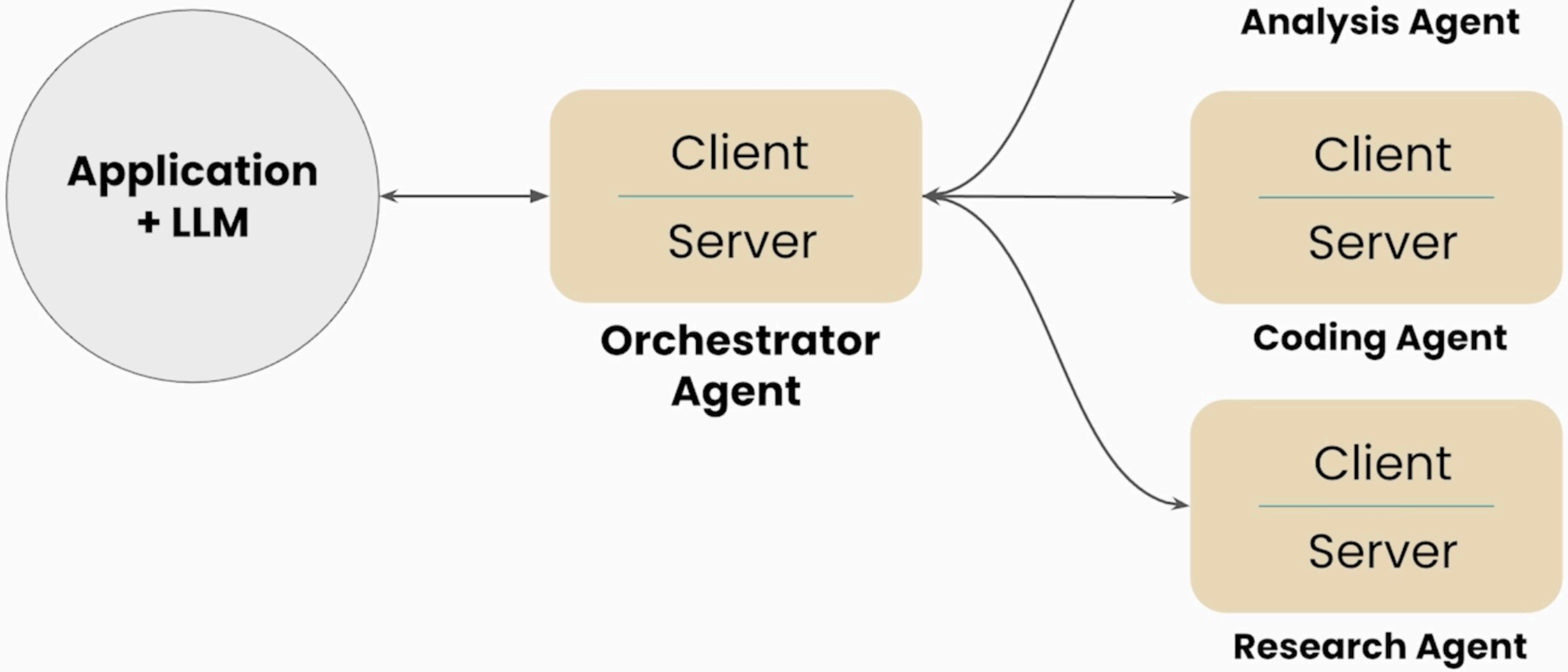
An MCP client can be a server and vice-versa



Building Effective Agents with MCP

Sampling + **Composability**

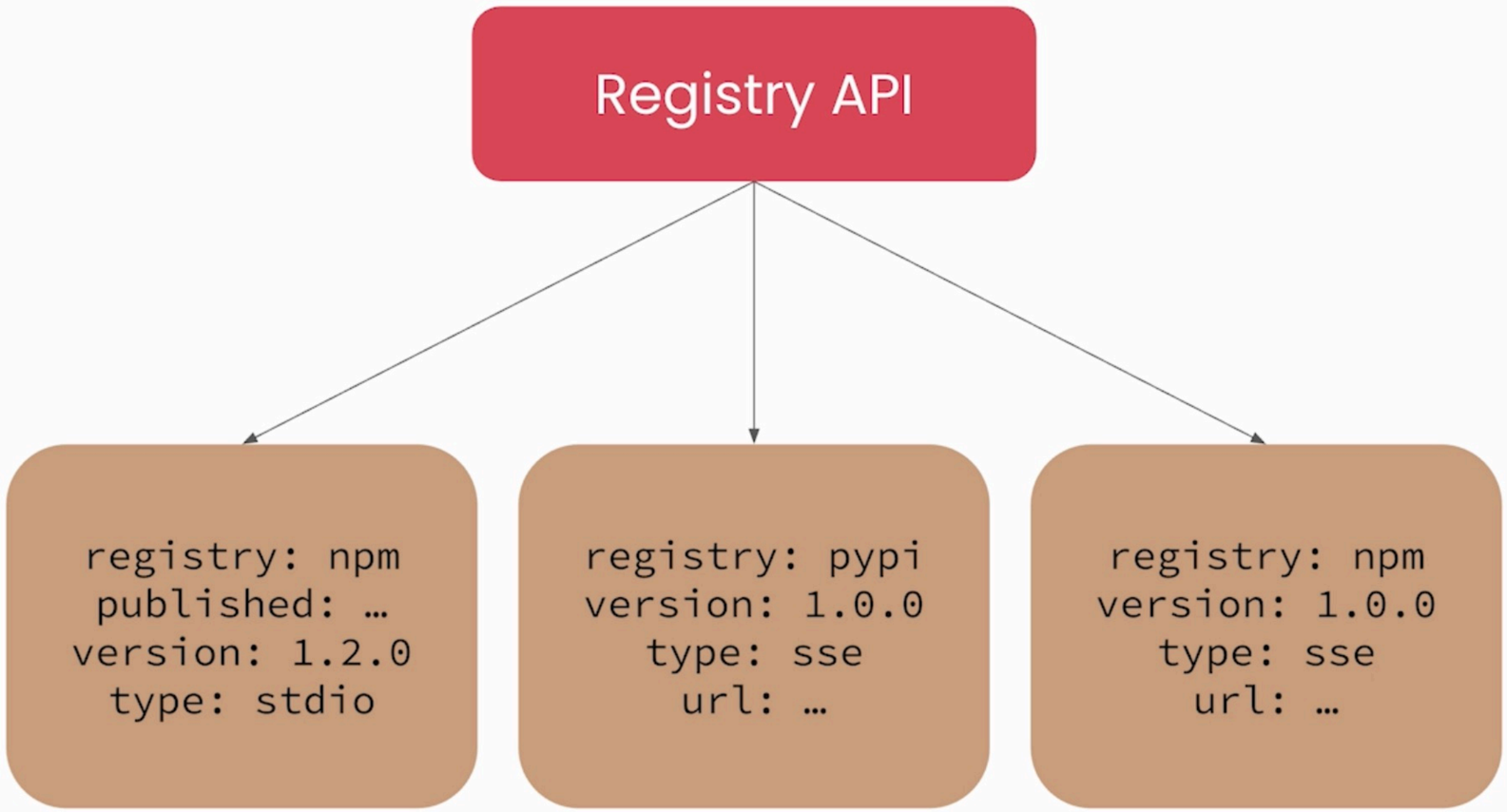
Chain agents together while making sure that the client application controls inference



An Official MCP Registry API

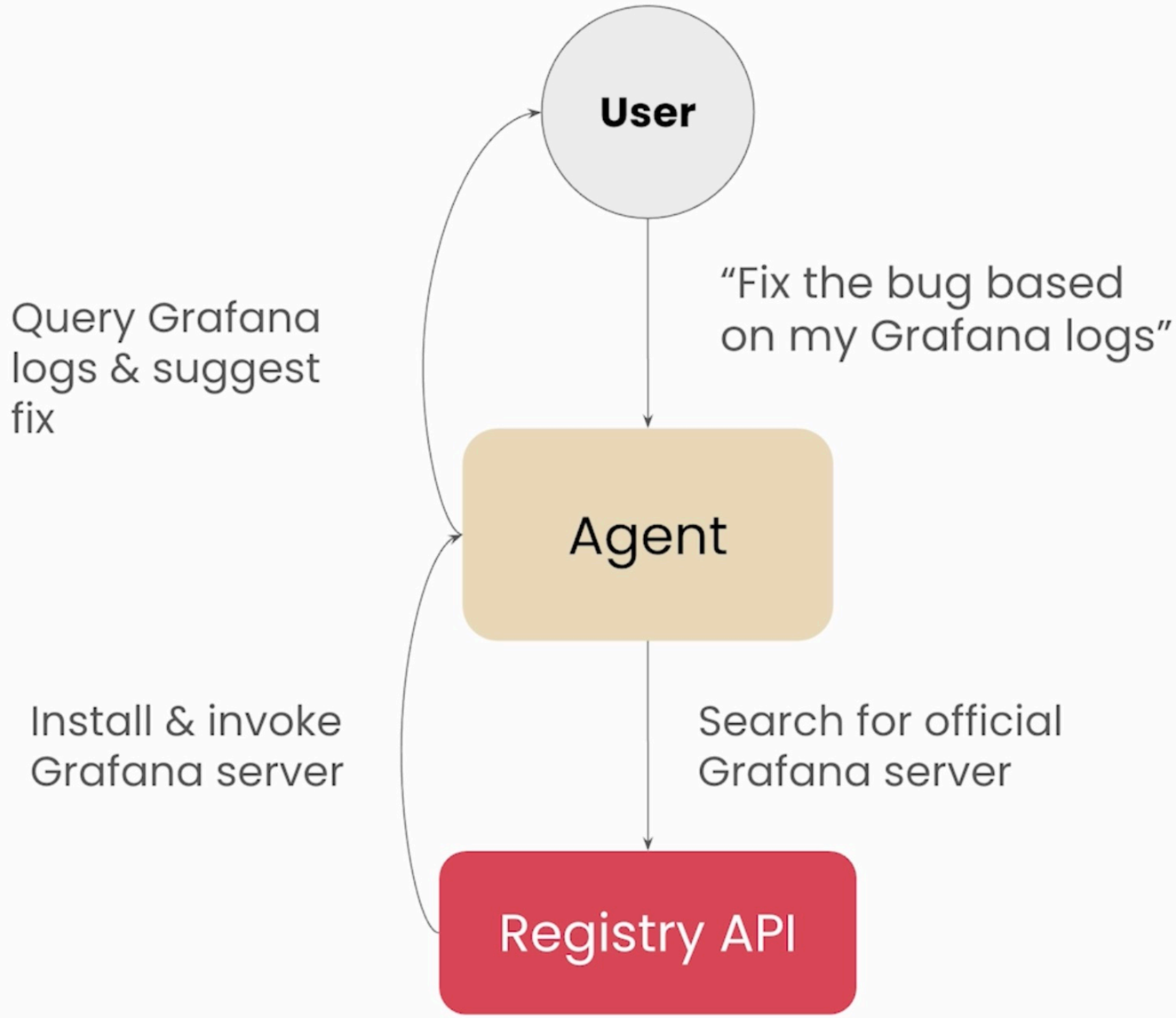
A unified, hosted metadata service which enables:

1. Discovery
2. Centralization
3. Verification



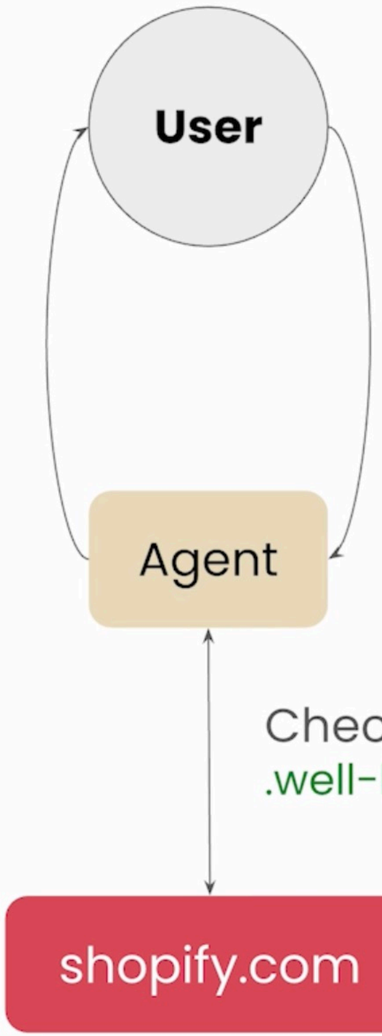
An Official MCP Registry API

An MCP server registry helps make agents **self-evolving** by letting them discover and choose their own tools



Server Discovery

Update store on Shopify via Remote MCP server



"Help me manage my store on Shopify"

shopify.com/.well-known/mcp.json

```
{
  "version": "1.0",
  "servers": [
    {
      "id": "main-server",
      "name": "Shopify Store Access",
      "endpoint": "https://shopify.com/api/mcp",
      "capabilities": ["resources", "tools"],
      "authType": "oauth2"
    },
    {
      "id": "admin-server",
      "name": "Shopify Admin API",
      "endpoint": "https://admin.shopify.com/api/mcp",
      "capabilities": ["resources", "tools"],
      "authType": "oauth2"
    }
  ],
  "organization": {
    "name": "Shopify"
  },
  "requirements": {
    "minProtocolVersion": "0.5.0"
  }
}
```