MCP

Model Context Protocol (MCP)

What is Model Context Protocol ?

It standardizes how applications provide context to LLMs. It provides a standardized way to connect AI models to different data sources and tools.

In simpler terms, we can say it a new way to expose tools and services to LLMs

Just like HTTPS or MQTT, it connects clients and servers but it especially designed for AI applications

The protocol uses JSON-RPC 2.0 messages to establish communication between the

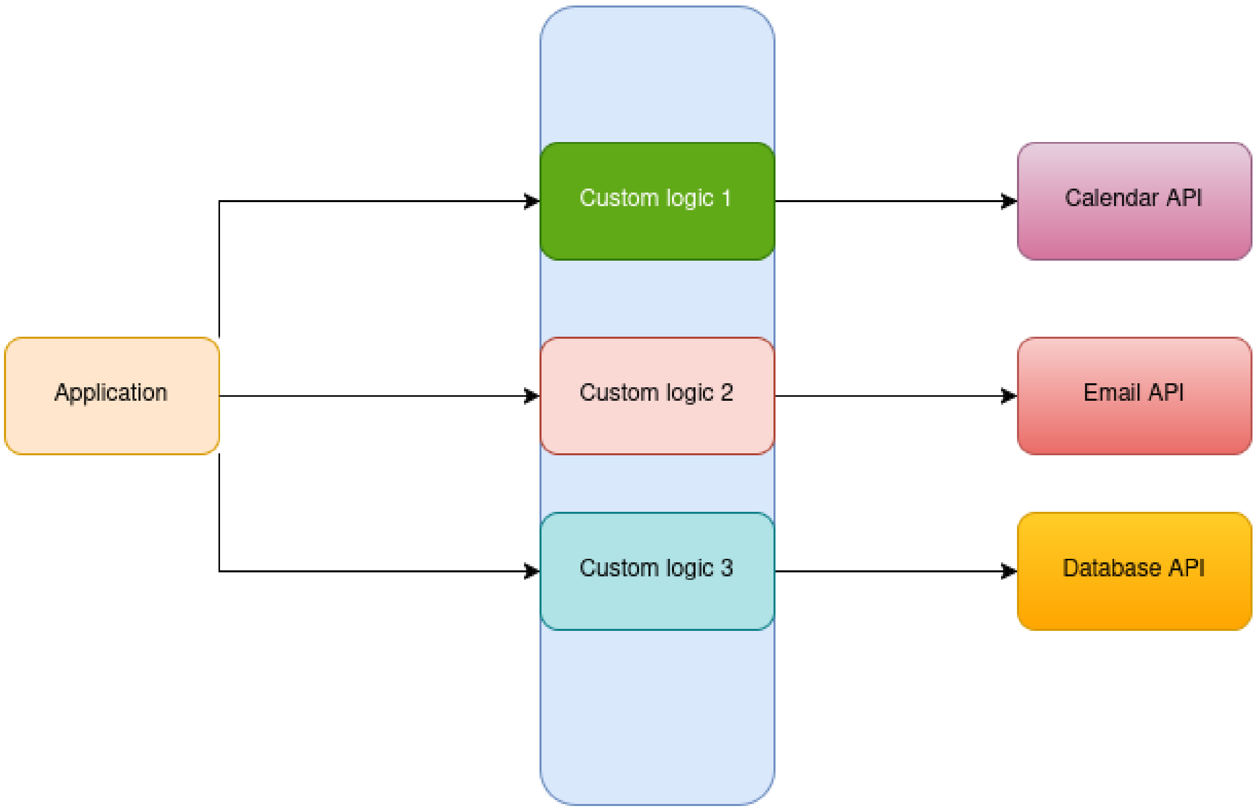
- **Host**: LLM applications that initiate connections
- **Client**: Connectors within the host application
- **Server**: Services that provide context and capabilities

Traditional way

Before MCP, we had to manually write the functions to integrate the external services like databases which are exposed as tool to the LLMs.

This has 2 main issues:

1. Custom implementations for each AI application to hook into its required context, leading to a lot of duplicated effort.
2. Change in external API leads to change in developer’s code base also.

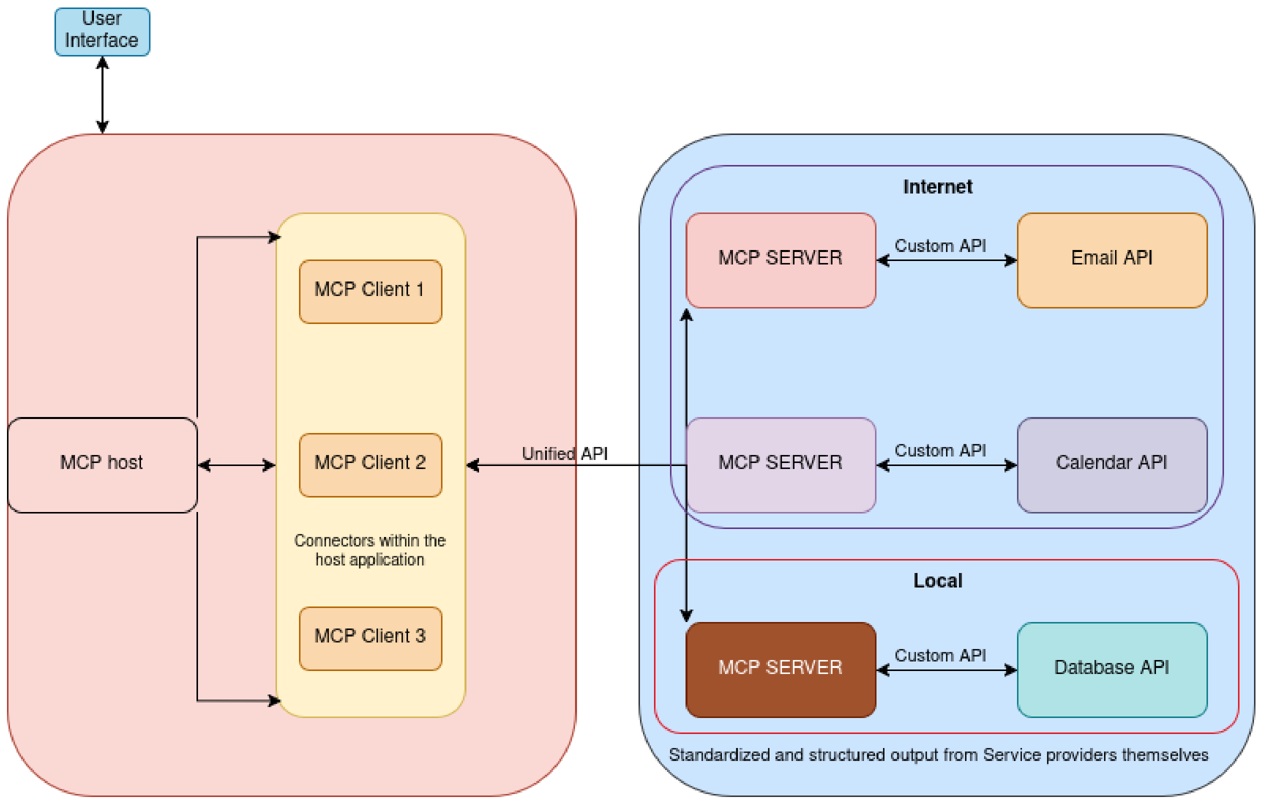


As represented in the diagram, each time we have to add any tool, we have to write a custom logic based on the external service provider APIs

There is no structured way for requesting and returning information between the applications and external services.

Model Context Protocol way

Instead of maintaining separate connectors for each data source, developers can now build against a standard protocol.



- MCP **Host** — LLM application (such as Cursor) that manages connections
- MCP **Client** — Maintains 1:1 connections with MCP servers
- MCP **Server** — Provides context, tools, and capabilities to the LLMs

A host application creates and manages multiple clients, with each client having a 1:1 relationship with a particular server.

MCP provides a stateful session protocol focused on context exchange and sampling coordination between clients and servers.

MCP is useful, as it allows:

- **Seamless Integration:** Clients can connect to a wide range of servers without needing to know the specifics of each underlying system.
- **Reusability:** Server developers can build integrations once and have them accessible to many different client applications.
- **Separation of Concerns:** Different teams can focus on building client applications or server integrations independently. For example, an infrastructure team can manage an MCP server for a vector database, which can then be easily used by various AI application development teams.

Getting started

- GitHub repository link: https://github.com/modelcontextprotocol

- Python SDK link: https://github.com/modelcontextprotocol/python-sdk
- Official documentation: https://modelcontextprotocol.io/introduction

List of MCP servers

Server Name	Server Type	Server Link
AWS KB Retrieval	Retrieval from AWS Knowledge Base using Bedrock Agent Runtime	https://github.com/modelcontextprotocol/servers/blob/main/src/aws-kb-retrieval-server
Git	Tools to read, search, and manipulate Git repositories	https://github.com/modelcontextprotocol/servers/blob/main/src/git
GitHub	Repository management, file operations, and GitHub API integration	https://github.com/modelcontextprotocol/servers/blob/main/src/github
Google Drive	File access and search capabilities for Google Drive	https://github.com/modelcontextprotocol/servers/blob/main/src/gdrive
Sqlite	Database interaction and business intelligence capabilities	https://github.com/modelcontextprotocol/servers/blob/main/src/sqlite
PostgreSQL	Read-only database access with schema inspection	https://github.com/modelcontextprotocol/servers/blob/main/src/postgres

AWS	Specialized MCP servers that bring AWS best practices directly to your development workflow.	https://github.com/awslabs/mcp
Chroma	Embeddings, vector search, document storage, and full-text search with the open-source AI application database	https://github.com/chroma-core/chroma-mcp

Complete list of servers can be found here: https://github.com/modelcontextprotocol/servers