Agentic AI – Unit V

Agentic Frameworks: Model Context Protocol

AI Tooling

- Before the introduction of MCP, AI applications relied on various methods, such as manual API wiring, plugin-based interfaces, and agent frameworks, to interact with external tools.
- These approaches required integrating each external service with a specific API, leading to increased complexity and limited scalability.
- MCP addresses these challenges by providing a standardized protocol that enables seamless and flexible interaction with multiple tools.

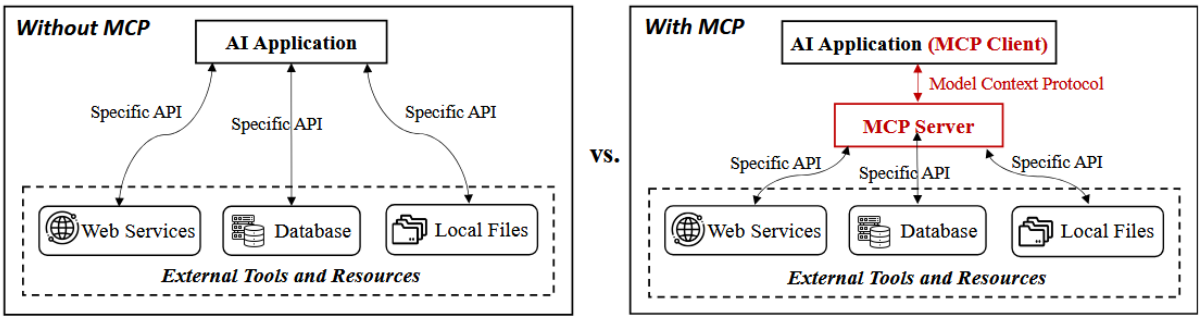


Figure. Tool invocation with and without MCP

Manual API Wiring

- In traditional implementations, developers had to establish manual API connections for each tool or service that an AI application interacted with.
- This process required custom authentication, data transformation, and error handling for every integration.
- As the number of APIs increased, the maintenance burden became significant, often leading to tightly coupled and fragile systems that were difficult to scale or modify.
- MCP eliminates this complexity by offering a unified interface, allowing AI models to connect with multiple tools dynamically without the need for custom API wiring.

AI Agent Tool Integration

- The emergence of AI agent frameworks like LangChain and similar tool orchestration frameworks provided a structured way for models to invoke external tools through predefined interfaces, improving automation and adaptability.
- However, integrating and maintaining these tools remained largely manual, requiring custom implementations and increasing complexity as the number of tools grew.

- MCP simplifies this process by offering a standardized protocol that enables AI agents to seamlessly invoke, interact with, and chain multiple tools through a unified interface.
- This reduces manual configuration and enhances task flexibility, allowing agents to perform complex operations without extensive custom integration.

Standardized Plugin Interfaces

- To reduce the complexity of manual wiring, plugin-based interfaces such as OpenAI ChatGPT Plugins, introduced in November 2023, allowed AI models to connect with external tools through standardized API schemas like OpenAPI.
- For example, in the OpenAI Plugin ecosystem, plugins like Zapier allowed models to perform predefined actions, such as sending emails or updating CRM records. However, these interactions were often one-directional and could not maintain state or coordinate multiple steps in a task. New LLM app stores such as ByteDance Coze and Tencent Yuanqi have also emerged, offering a plugin store for web services.
- While these platforms expanded available tool options, they created isolated ecosystems where plugins are platform-specific, limiting cross-platform compatibility and requiring duplicate maintenance efforts.
- MCP stands out by being open-source and platform-agnostic, enabling AI applications to engage in rich two-way interactions with external tools, facilitating complex workflows.

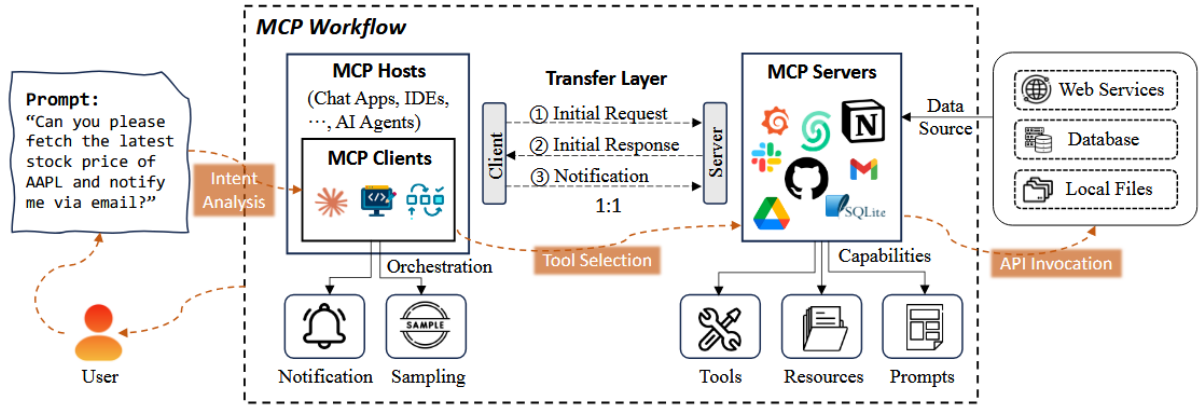
Retrieval-Augmented Generation (RAG) and Vector Database

- Contextual information retrieval methods, such as RAG, leverage vector-based search to retrieve relevant knowledge from databases or knowledge bases, enabling models to supplement responses with up-to-date information.
- While this approach addressed the problem of knowledge cutoff and improved model accuracy, it was limited to passive retrieval of information. It did not inherently allow models to perform active operations, such as modifying data or triggering workflows.
- For example, a RAG-based system could retrieve relevant sections from a product documentation database to assist a customer support AI.
- However, if the AI needed to update customer records or escalate an issue to human support, it could not act beyond providing textual responses.
- MCP extends beyond passive information retrieval by enabling AI models to interact with external data sources and tools actively, facilitating both retrieval and action in a unified workflow.

MCP Architecture

Core Components

- The MCP architecture is composed of three core components: MCP host, MCP client, and MCP server.
- These components collaborate to facilitate seamless communication between AI applications, external tools, and data sources, ensuring that operations are secure and properly managed.
- In a typical workflow, the user sends a prompt to the MCP client, which analyzes the intent, selects the appropriate tools via the MCP server, and invokes external APIs to retrieve and process the required information before notifying the user of the results.



The workflow of MCP

MCP Host

- The MCP host is an AI application that provides the environment for executing AI-based tasks while running the MCP client.
- It integrates interactive tools and data to enable smooth communication with external services.
- Examples include Claude Desktop for AI-assisted content creation, Cursor, an AI-powered IDE for code completion and software development, and AI agents that function as autonomous systems for executing complex tasks. The MCP host hosts the MCP client and ensures communication with external MCP servers.

MCP Client

- The MCP client acts as an intermediary within the host environment, managing communication between the MCP host and one or more MCP servers.
- It initiates requests to MCP servers, queries available functions, and retrieves responses that describe the server's capabilities.

- This ensures seamless interaction between the host and external tools.In addition to managing requests and responses, the MCP client processes notifications from MCP servers, providing real-time updates about task progress and system status.
- It also performs sampling to gather data on tool usage and performance, enabling optimization and informed decision-making.
- The MCP client communicates through the transport layer with MCP servers, facilitating secure, reliable data exchange and smooth interaction between the host and external resources.

MCP Server

- The MCP server enables the MCP host and client to access external systems and execute operations, offering three core capabilities: tools, resources, and prompts.

Tools: Enabling external operations

- Tools allow the MCP server to invoke external services and APIs to execute operations on behalf of AI models. When the client requests an operation, the MCP server identifies the appropriate tool, interacts with the service, and returns the result.
- For instance, if an AI model requires real-time weather data or sentiment analysis, the MCP server connects to the relevant API, retrieves the data, and delivers it to the host.
- Unlike traditional function calling, which requires multiple steps and separates invocation from execution, Tools of MCP servers streamline this process by allowing the model to autonomously select and invoke the appropriate tool based on context.
- Once configured, these tools follow a standardized supply-and-consume model, making them modular, reusable, and easily accessible to other applications, enhancing system efficiency and flexibility.

Resources: Exposing data to AI models

- Resources provide access to structured and unstructured datasets that the MCP server can expose to AI models.
- These datasets may come from local storage, databases, or cloud platforms. When an AI model requests specific data, the MCP server retrieves and processes the relevant information, enabling the model to make data-driven decisions.
- For example, a recommendation system may access customer interaction logs, or a document summarization task may query a text repository.

Prompts: Reusable templates for workflow optimization

- Prompts are predefined templates and workflows that the MCP server generates and maintains to optimize AI responses and streamline repetitive tasks. They ensure consistency in responses and improve task execution efficiency.

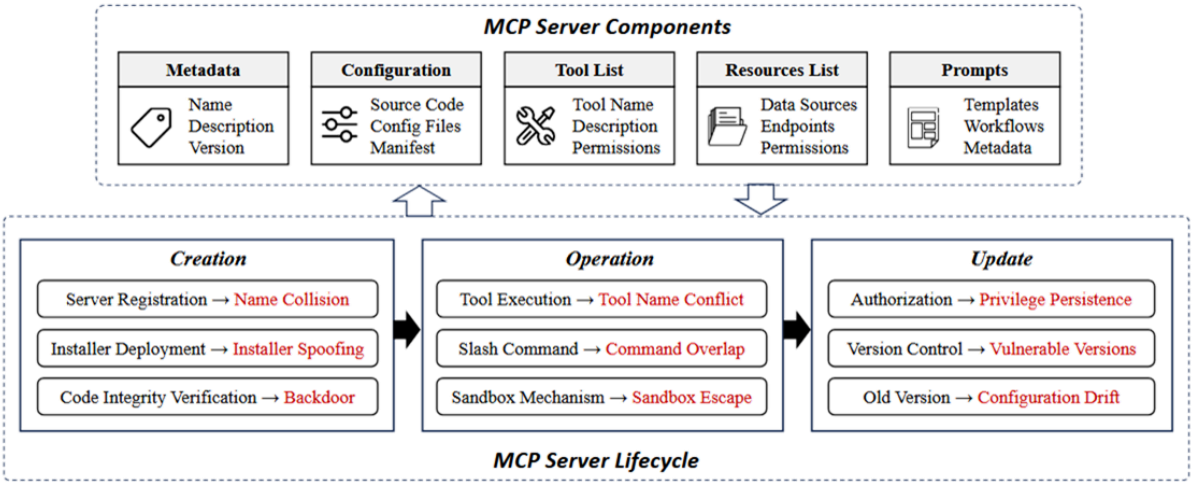
- For instance, a customer support chatbot may use prompt templates to provide uniform and accurate responses, while an annotation task may rely on predefined prompts to maintain consistency in data labeling.

Transport Layer and Communication

- The transport layer ensures secure, bidirectional communication, allowing for real-time interaction and efficient data exchange between the host environment and external systems.
- The transport layer manages the transmission of initial requests from the client, the delivery of server responses detailing available capabilities, and the exchange of notifications that keep the client informed of ongoing updates.
- Communication between the MCP client and the MCP server follows a structured process, beginning with an initial request from the client to query the server's functionalities.
- Upon receiving the request, the server responds with an initial response listing the available tools, resources, and prompts the client can leverage.
- Once the connection is established, the system maintains a continuous exchange of notifications to ensure that changes in server status or updates are communicated back to the client in real time.
- This structured communication ensures high-performance interactions and keeps AI models synchronized with external resources, enhancing the effectiveness of AI applications.

MCP Server Lifecycle

The MCP server lifecycle consists of three key phases: creation, operation, and update. Each phase defines critical activities that ensure the secure and efficient functioning of the MCP server, enabling seamless interaction between AI models and external tools, resources, and prompts.



MCP Server Components

- The MCP server is responsible for managing external tools, data sources, and workflows, providing AI models with the necessary resources to perform tasks efficiently and securely.
- It comprises several key components:
    - **Metadata:** Includes essential information about the server, such as its name, version, and description, allowing clients to identify and interact with the appropriate server.
    - **Configuration:** Involves the source code, configuration files, and manifest, which define the server's operational parameters, environment settings, and security policies.
    - **Tool list:** Stores a catalog of available tools, detailing their functionalities, input-output formats, and access permissions, ensuring proper tool management and security.
    - **Resources list:** Governs access to external data sources, including web APIs, databases, and local files, specifying allowed endpoints and their associated permissions.
    - **Prompts and Templates:** Include pre-configured task templates and workflows that enhance the efficiency of AI models in executing complex operations.Together, these components enable MCP servers to provide seamless tool integration, data retrieval, and task orchestration for AI-powered applications.

Creation Phase

- The creation phase is the initial stage of the MCP server lifecycle, where the server is registered, configured, and prepared for operation.
- This phase involves three key steps:
    1. **Server registration:** Assigns a unique name and identity to the MCP server, allowing clients to discover and connect to the appropriate server instance.
    2. **Installer deployment:** Involves installing the MCP server and its associated components, ensuring that the correct configuration files, source code, and manifests are in place.
    3. **Code integrity verification:** Validates the integrity of the server's codebase to prevent unauthorized modifications or tampering before the server becomes operational. Successful completion of the creation phase ensures that the MCP server is ready to handle requests and interact securely with external tools and data sources.

Operation Phase

- The operation phase is where the MCP server actively processes requests, executes tool invocations, and facilitates seamless interaction between AI applications and external resources.
- Tool execution allows the MCP server to invoke the appropriate tools based on the AI application's requests, ensuring that the selected tools perform their intended operations.
- Slash command handling: Enables the server to interpret and execute multiple commands, including those issued through user interfaces or AI agents, while managing potential command overlaps to prevent conflicts.
- Sandbox mechanism enforcement: Ensures that the execution environment is isolated and secure, preventing unauthorized access and mitigating potential risks.
- Throughout the operation phase, the MCP server maintains a stable and controlled environment, enabling reliable and secure task execution.

Update Phase

- The update phase ensures that the MCP server remains secure, up-to-date, and capable of adapting to evolving requirements.
- This phase includes three key tasks:

1. **Authorization management:** Verifies that post-update access permissions remain valid, preventing unauthorized use of server resources after updates.
2. **Version control:** Maintains consistency between different server versions, ensuring that new updates do not introduce vulnerabilities or conflicts.
3. **Old version management:** Deactivates or removes outdated versions to prevent attackers from exploiting known vulnerabilities in previous versions.

Use Cases

- MCP has become a vital tool for AI applications to effectively communicate with external tools, APIs, and systems.
- By standardizing interactions, MCP simplifies complex workflows, boosting the efficiency of AI-driven applications.

OpenAI: MCP Integration in AI Agents and SDKs

- OpenAI has adopted MCP to standardize AI-to-tool communication, recognizing its potential to enhance integration with external tools.
- Recently, OpenAI introduced MCP support in its Agent SDK, enabling developers to create AI agents that seamlessly interact with external tools.

- In a typical workflow, developers use the Agent SDK to define tasks that require external tool invocation. When an AI agent encounters a task like retrieving data from an API or querying a database, the SDK routes the request through an MCP server.
- The request is transmitted via the MCP protocol, ensuring proper formatting and real-time response delivery to the agent.
- OpenAI's plan to integrate MCP into the Responses API will streamline AI-to-tool communication, allowing AI models like ChatGPT to interact with tools dynamically without extra configuration.
- Additionally, OpenAI aims to extend MCP support to ChatGPT desktop applications, enabling AI assistants to handle various user tasks by connecting to remote MCP servers, further bridging the gap between AI models and external systems.

Cursor: Enhancing Software Development with MCP-Powered Code Assistants

- MCP is used to enhance software development by enabling AI-powered code assistants that automate complex tasks.
- With MCP, Cursor allows AI agents to interact with external APIs, access code repositories, and automate workflows directly within the integrated development environment.
- When a developer issues a command within the IDE, the AI agent evaluates whether external tools are needed.
- If so, the agent sends a request to an MCP server, which identifies the appropriate tool and processes the task, such as running API tests, modifying files, or analyzing code.
- The results are then returned to the agent for further action. This integration helps automate repetitive tasks, minimizing errors and enhancing overall development efficiency.
- By simplifying complex processes, Cursor boosts both productivity and accuracy, allowing developers to execute multi-step operations effortlessly.

Cloudflare: Remote MCP Server Hosting and Scalability

- Cloudflare has played a pivotal role in transforming MCP from a local deployment model to a cloud-hosted architecture by introducing remote MCP server hosting.
- This approach eliminates the complexities associated with configuring MCP servers locally, allowing clients to connect to secure, cloud-hosted MCP servers seamlessly.
- The workflow begins with Cloudflare hosting MCP servers in secure cloud environments that are accessible via authenticated API calls.
- AI agents initiate requests to the Cloudflare MCP server using OAuth-based authentication, ensuring that only authorized entities can access the server.

- Once authenticated, the agent dynamically invokes external tools and APIs through the MCP server, executing tasks such as data retrieval, document processing, or API integration.
- This approach not only reduces the risk of misconfiguration but also ensures seamless execution of AI-powered workflows across distributed environments.
- Furthermore, Cloudflare's multi-tenant architecture allows multiple users to securely access and manage their own MCP instances, ensuring isolation and preventing data leakage.
- Cloudflare's solution thus extends MCP's capabilities by enabling enterprise-grade scalability and secure multi-device interoperability.