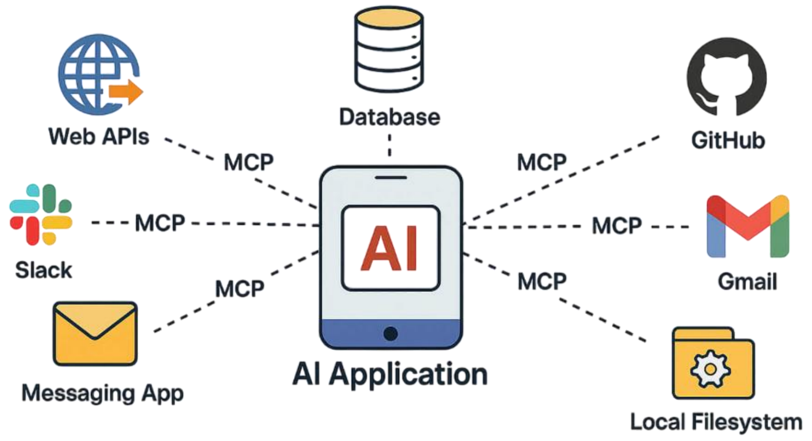
Model Context Protocol (MCP)

Ali Najafi

Sabancı Üniversitesi

VBYO 2025



Agents

	AI agent	AI assistant	Bot
Purpose	Autonomously and proactively perform tasks	Assisting users with tasks	Automating simple tasks or conversations
Capabilities	Can perform complex, multi-step actions; learns and adapts; can make decisions independently	Responds to requests or prompts; provides information and completes simple tasks; can recommend actions but the user makes decisions	Follows pre-defined rules; limited learning; basic interactions
Interaction	Proactive; goal-oriented	Reactive; responds to user requests	Reactive; responds to triggers or commands

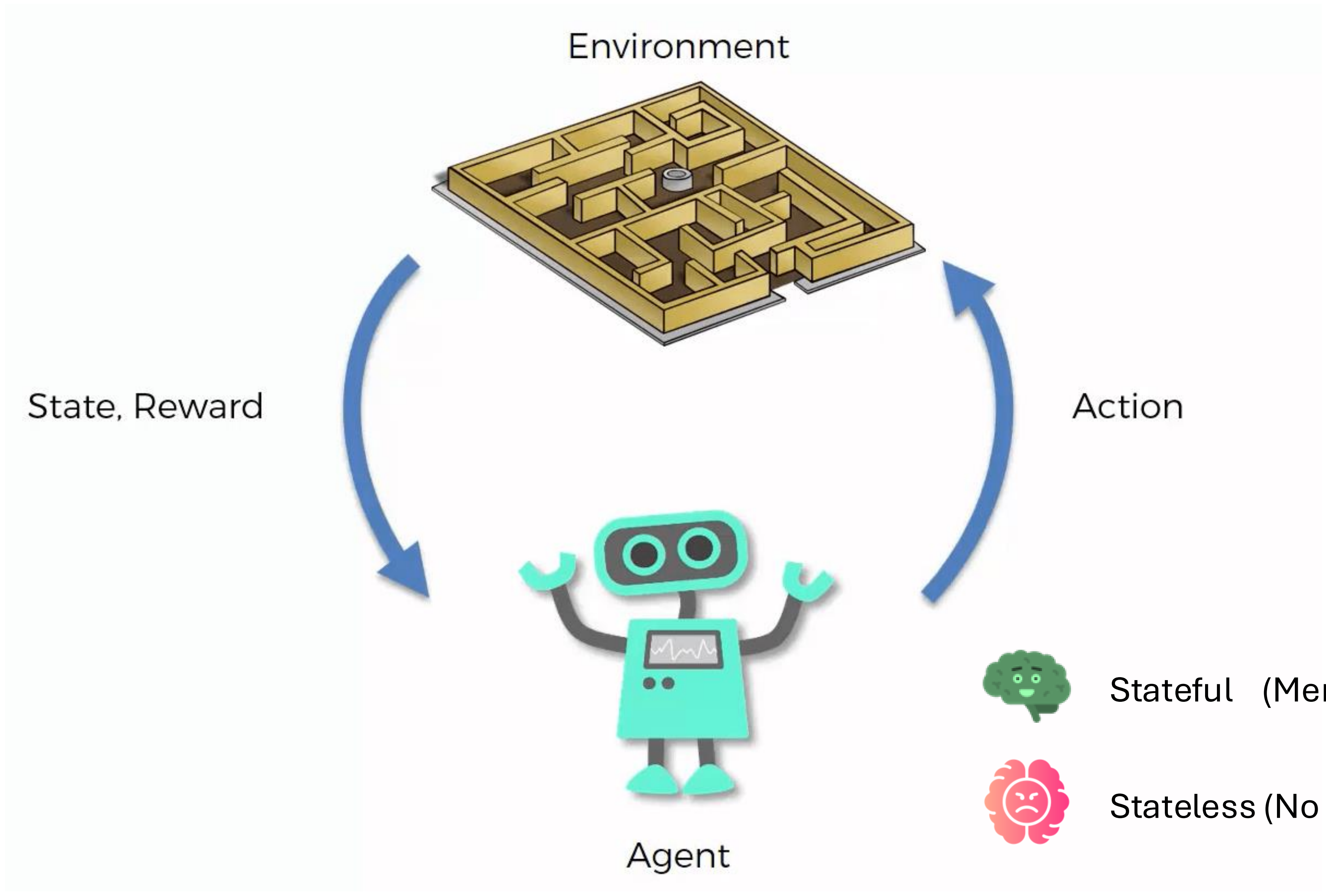
How do AI agents work?

- **Persona**: A well-defined persona allows an agent to maintain a consistent character and behave in a manner appropriate to its assigned role, evolving as the agent gains experience and interacts with its environment.

- **Memory**: The agent is equipped in general with short term, long term, consensus, and episodic memory. Short term memory for immediate interactions, long-term memory for historical data and conversations, episodic memory for past interactions, and consensus memory for shared information among agents. The agent can maintain context, learn from experiences, and improve performance by recalling past interactions and adapting to new situations.

- **Tools**: Tools are functions or external resources that an agent can utilize to interact with its environment and enhance its capabilities. They allow agents to perform complex tasks by accessing information, manipulating data, or controlling external systems, and can be categorized based on their user interface, including physical, graphical, and program-based interfaces. Tool learning involves teaching agents how to effectively use these tools by understanding their functionalities and the context in which they should be applied.

- **Model**: Large language models (LLMs) serve as the foundation for building AI agents, providing them with the ability to understand, reason, and act. LLMs act as the "brain" of an agent, enabling them to process and generate language, while other components facilitate reason and action.



🟩 Stateful (Memory)

🟥 Stateless (No Memory)

What is MCP?

MCP is an open protocol that standardizes how applications provide context to large language models (LLMs). Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect your devices to various peripherals and accessories, MCP provides a standardized way to connect AI models to different data sources and tools. MCP enables you build agents and complex workflows on top of LLMs and connects your models with the world.

- **MCP Host**: The AI application that coordinates and manages one or multiple MCP clients.

(Claude Code, Claude Desktop, VS code Chat, …)

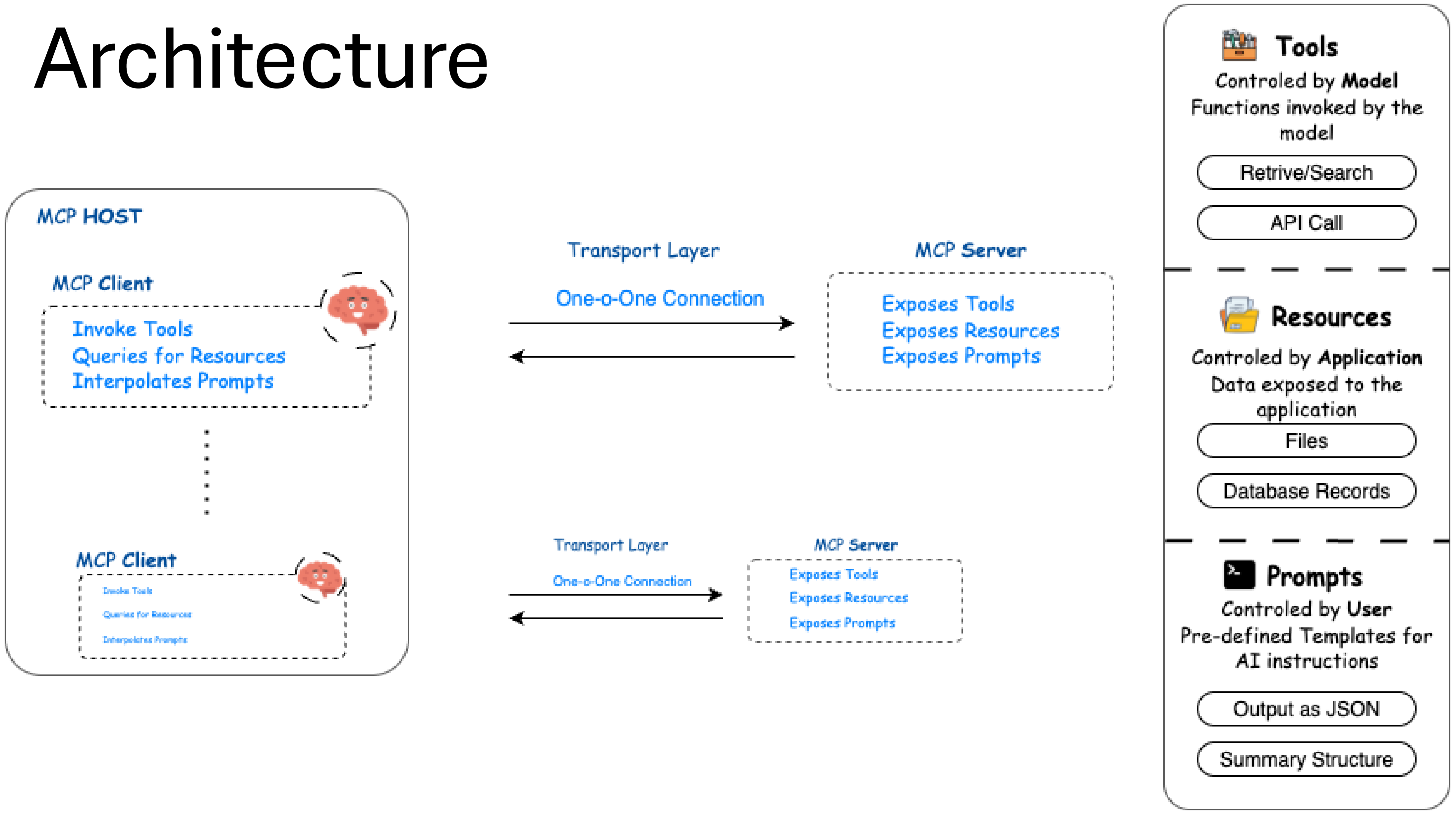
- **MCP Client**: A component that maintains a connection to an MCP server and obtains context from an MCP server for the MCP host to use

- **MCP Server**: A program that provides context to MCP clients

Why do we need MCP?

- **A growing list of pre-built integrations** that your LLM can directly plug into
- **A standardized way** to build custom integrations for AI applications
- **An open protocol** that everyone is free to implement and use
- **The flexibility to change** between different apps and take your context with you

Architecture



Tools

- **Each tool does a certain task.**
  - For instance, calculate the sum of two numbers (a+b)
- **Model-Controlled**: LLMs use these tools

Property	Description
name	A unique identifier (string) for the tool. The model uses this name to call it.
description	A short, natural language explanation of what the tool does. This helps the model understand when and how to use it.
parameters	A JSON Schema object that defines the expected inputs (arguments) for the tool. This includes types, required fields, descriptions, etc.
function	(Optional) The actual callable function (used in implementation, not model-exposed).
type	Usually set to "function" to match OpenAI's function_calling mechanism.
examples	(Optional, not always used) Example inputs/outputs to guide the model on usage.

Resources

- **Resources** are structured pieces of information—like documents, datasets, or memory—that provide context or knowledge for the model to reference during reasoning or task execution.
- **Application-Controlled**: Application manages resources

Property	Type	Description
name	string	A unique identifier for the resource. Used to reference the resource in the context.
type	string	Describes what kind of resource it is (e.g., "file", "document", "dataset", "memory", "calendar_event", etc.).
content_type	string	The MIME type or format of the content (e.g., text/markdown, application/json, application/pdf).
content	string or object	The actual resource content (raw text, structured data, etc.).
description	string	A human-readable description of the resource to help the model understand its relevance.
metadata	object	Optional additional metadata (e.g., creation date, source, author, etc.).
visibility	string	Indicates whether the resource is public, private, or restricted. Useful for access control in multi-agent settings.
embedding	array	(Optional) A vector embedding of the resource, if used for semantic search or retrieval.
scope	string	Specifies the scope of the resource (e.g., session, global, user, or project). Determines availability duration or user mapping.

Prompts

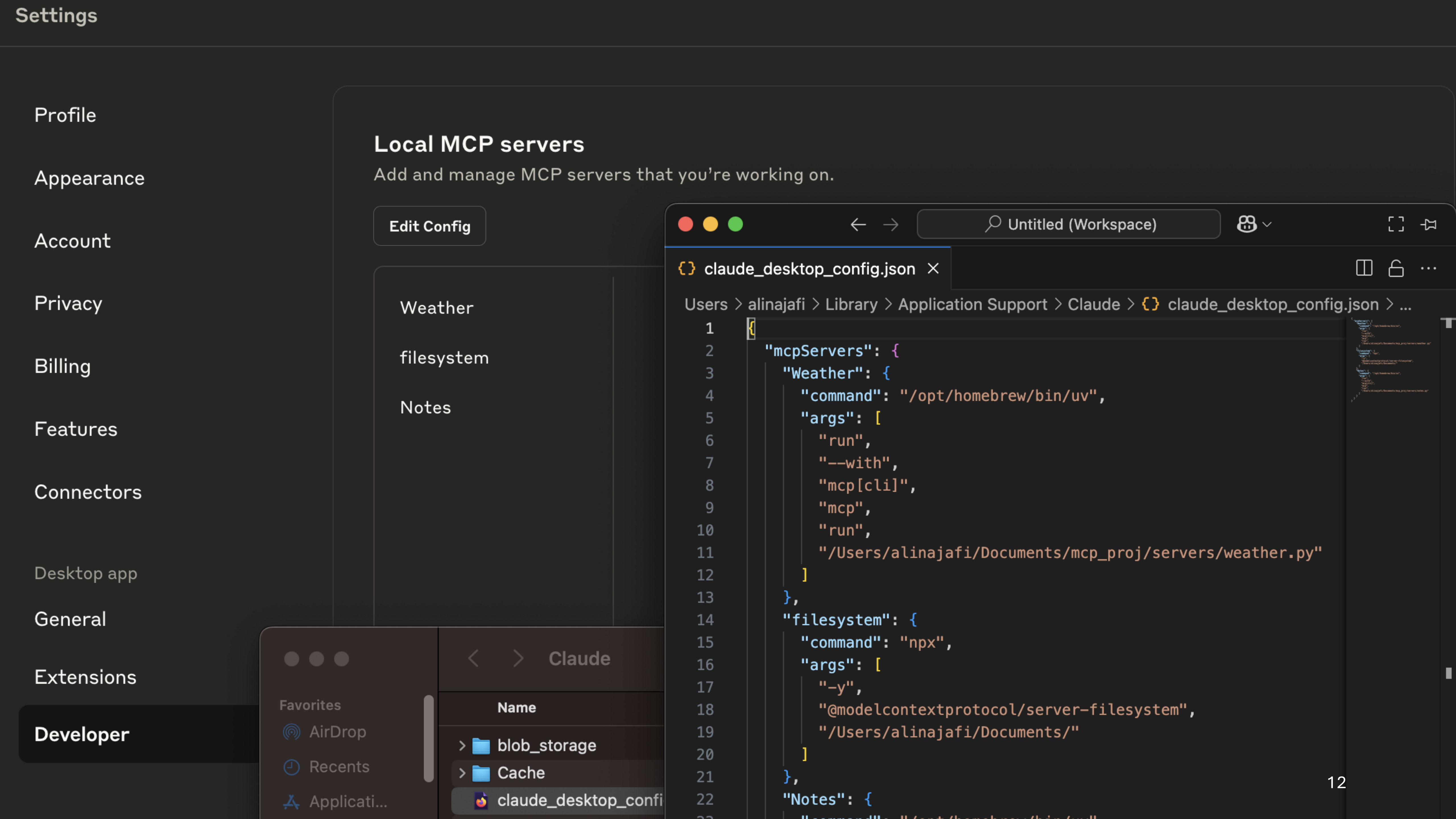
- **Prompts** are structured inputs that include the user's message, role, and optional metadata or context links to guide the model's response in a state-aware and controllable manner.
- **User-Controlled**: User manages resources

Property	**Type**	**Description**
text	string	The core user input or natural language instruction provided to the model.
role	string	The role of the speaker, typically "user", "assistant", or "system"—used to format conversational turns.
name	string	(Optional) A custom name for the prompt turn, useful in multi-turn dialogues.
timestamp	string	(Optional) ISO 8601 timestamp indicating when the prompt was created.
metadata	object	(Optional) Additional information about the prompt (e.g., source, tone, priority).
context_refs	array	(Optional) References to specific contexts or resources to be included with the prompt.
id	string	

Transport Layer

- **STDIO**
  - 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:**

  In the **Streamable HTTP** transport, the server operates as an independent process that can handle multiple client connections. This transport uses HTTP POST and GET requests. Server can optionally make use of **Server-Sent Events** (SSE) to stream multiple server messages. This permits basic MCP servers, as well as more feature-rich servers supporting streaming and server-to-client notifications and requests.


Settings

Profile
Appearance
Account
Privacy
Billing
Features
Connectors

Desktop app
General
Extensions
Developer

Local MCP servers

Add and manage MCP servers that you're working on.

Edit Config

Weather
filesystem
Notes

Untitled (Workspace)

claude_desktop_config.json

Users > alinajafi > Library > Application Support > Claude > claude_desktop_config.json > ...

```json
{
  "mcpServers": {
    "Weather": {
      "command": "/opt/homebrew/bin/uv",
      "args": [
        "run",
        "--with",
        "mcp[cli]",
        "mcp",
        "run",
        "/Users/alinajafi/Documents/mcp_proj/servers/weather.py"
      ]
    },
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "/Users/alinajafi/Documents/"
      ]
    },
    "Notes": {
```

Claude

Favorites
AirDrop
Recents
Applicati...

Name
blob_storage
Cache
claude_desktop_confi

Thank you

- ali.najafi@sabanciuniv.edu
- **www.najafi-ali.com**

References

- https://cloud.google.com/discover/what-are-ai-agents?hl=en
- https://modelcontextprotocol.io