MCP: AI Apps as Easy as 1, 2, 3

INTRODUCTION TO MODEL CONTEXT PROTOCOL (MCP)



James Chapman
AI Curriculum Manager, DataCamp

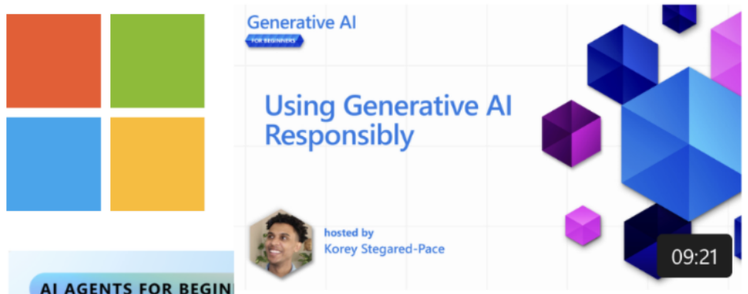
datacamp

Meet Your Instructor



Korey Stegared-Pace

Senior AI Cloud Advocate at Microsoft





EPISODE

Using Generative AI Responsibly (Part 3 of 18) | Generative AI for Beginners

Jun 25, 2024

Azure

EPISODE

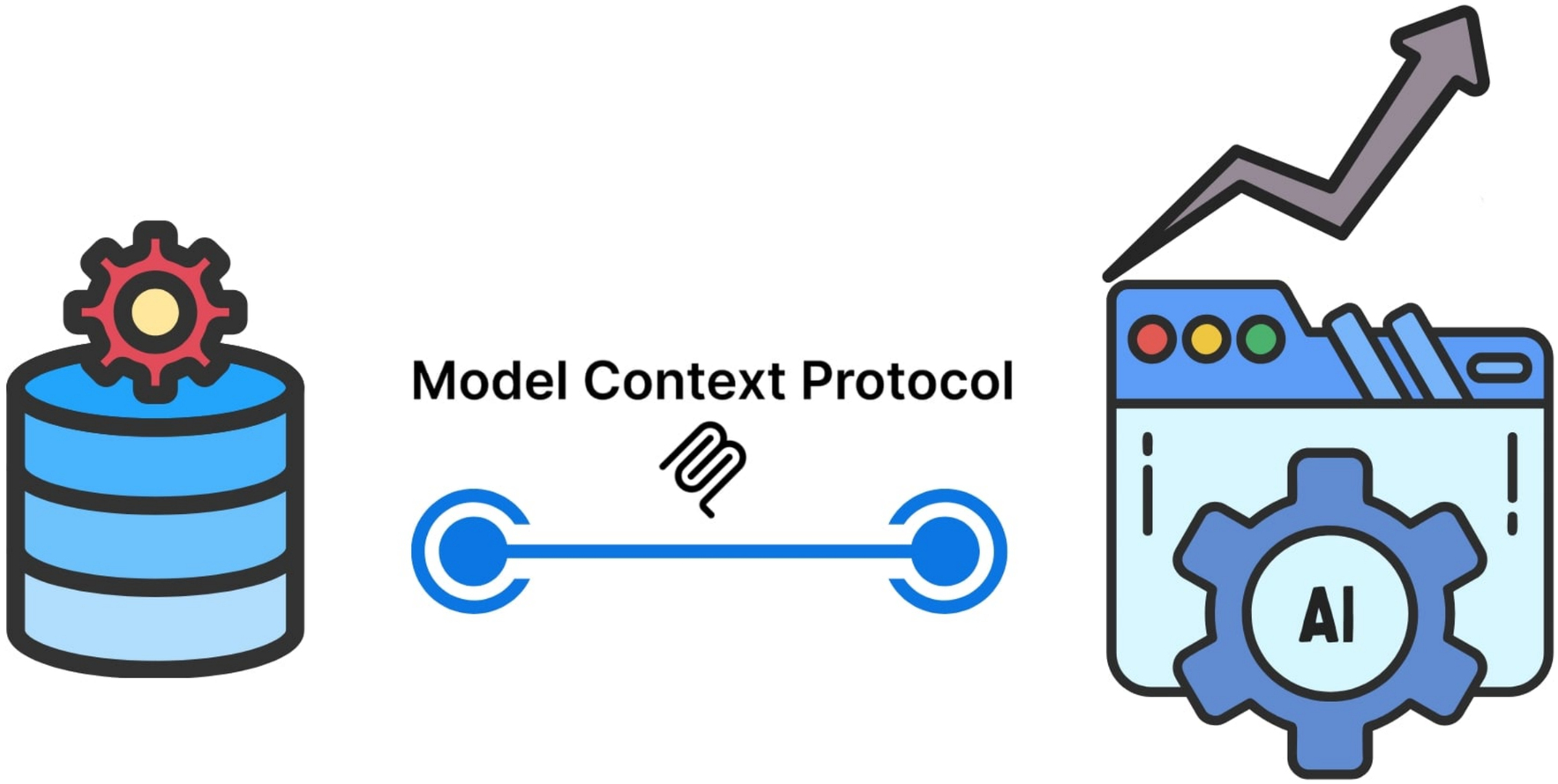
What are AI agents? (Part 1 of 10) | AI Agents for Beginners

Mar 24, 2025

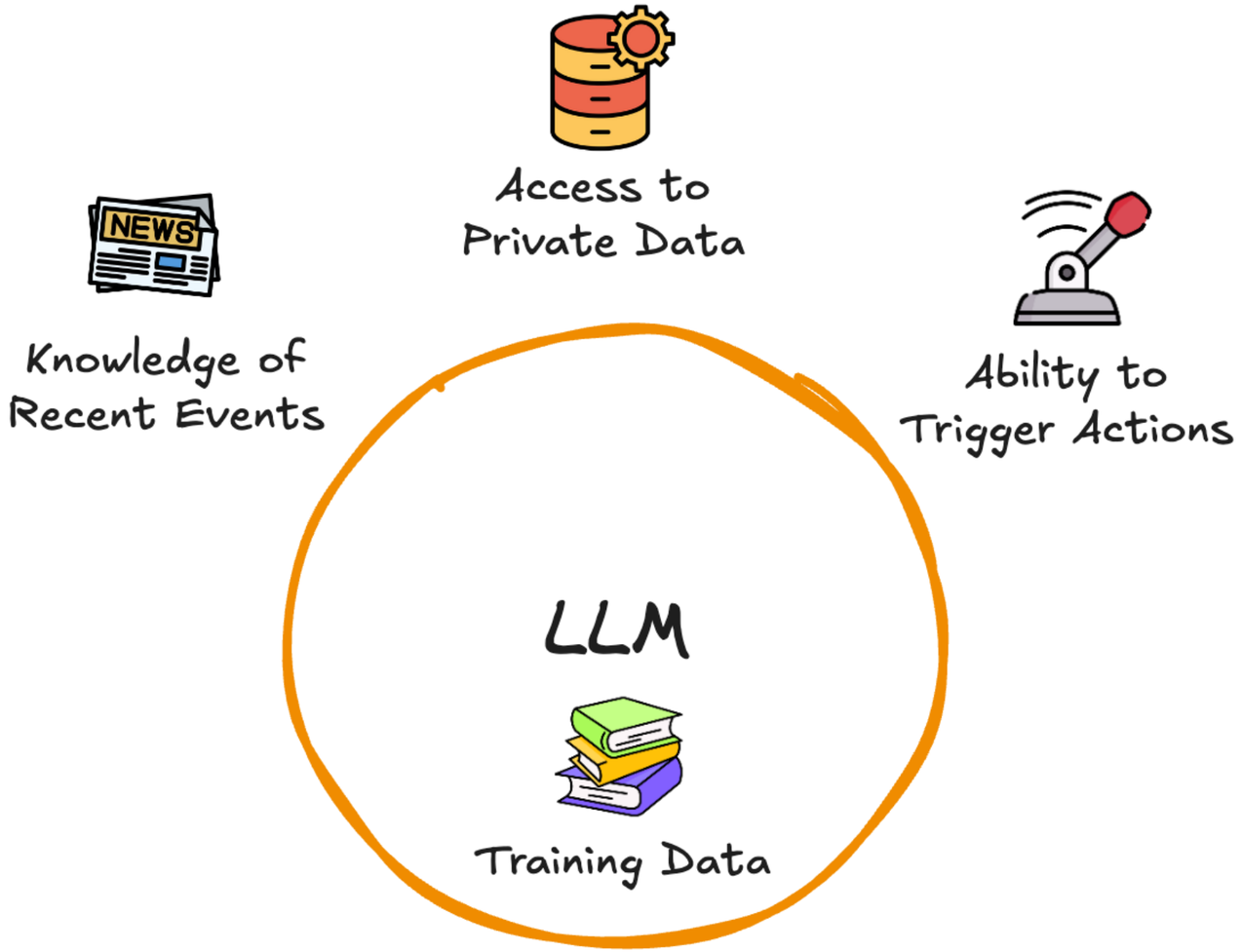
Beginner

[1] https://developer.microsoft.com/en-us/advocates/korey-stegared-pace

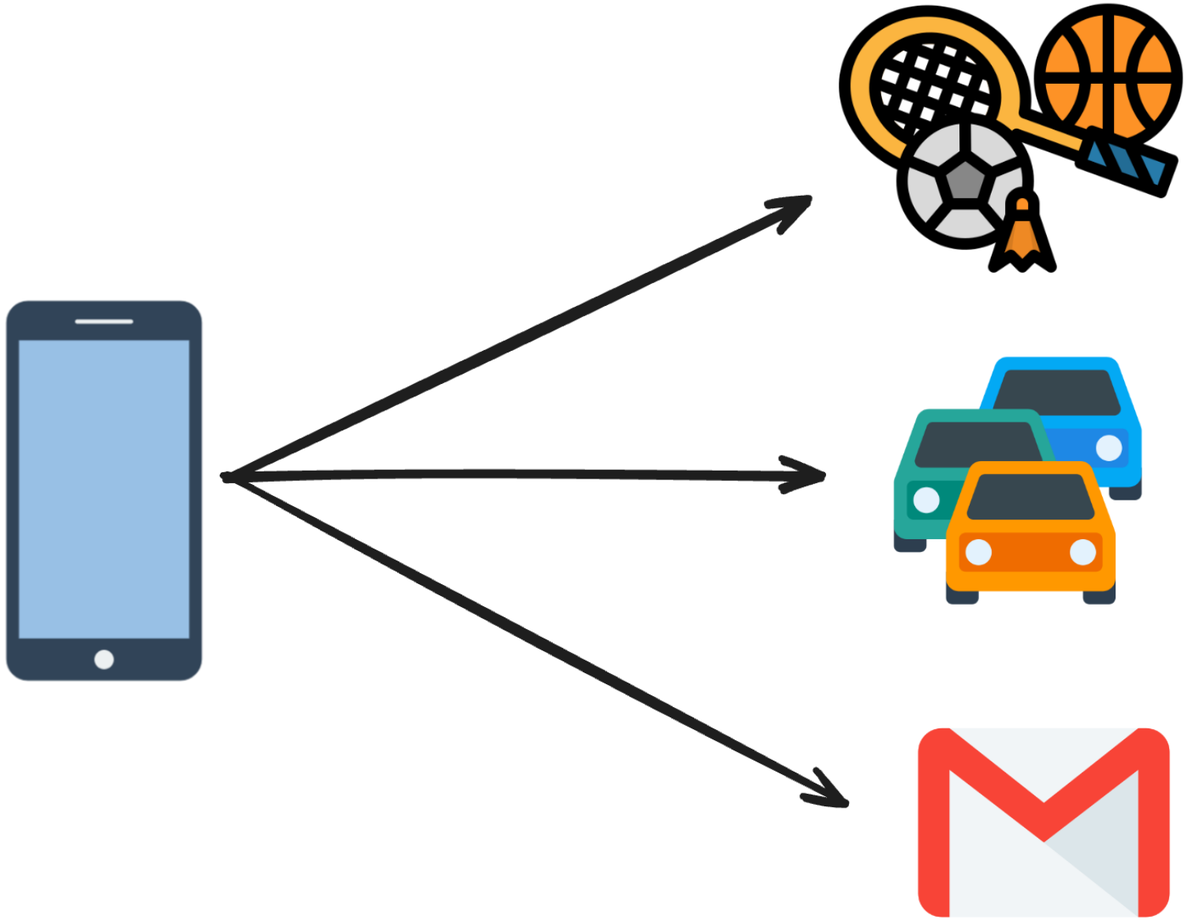
Looking Ahead



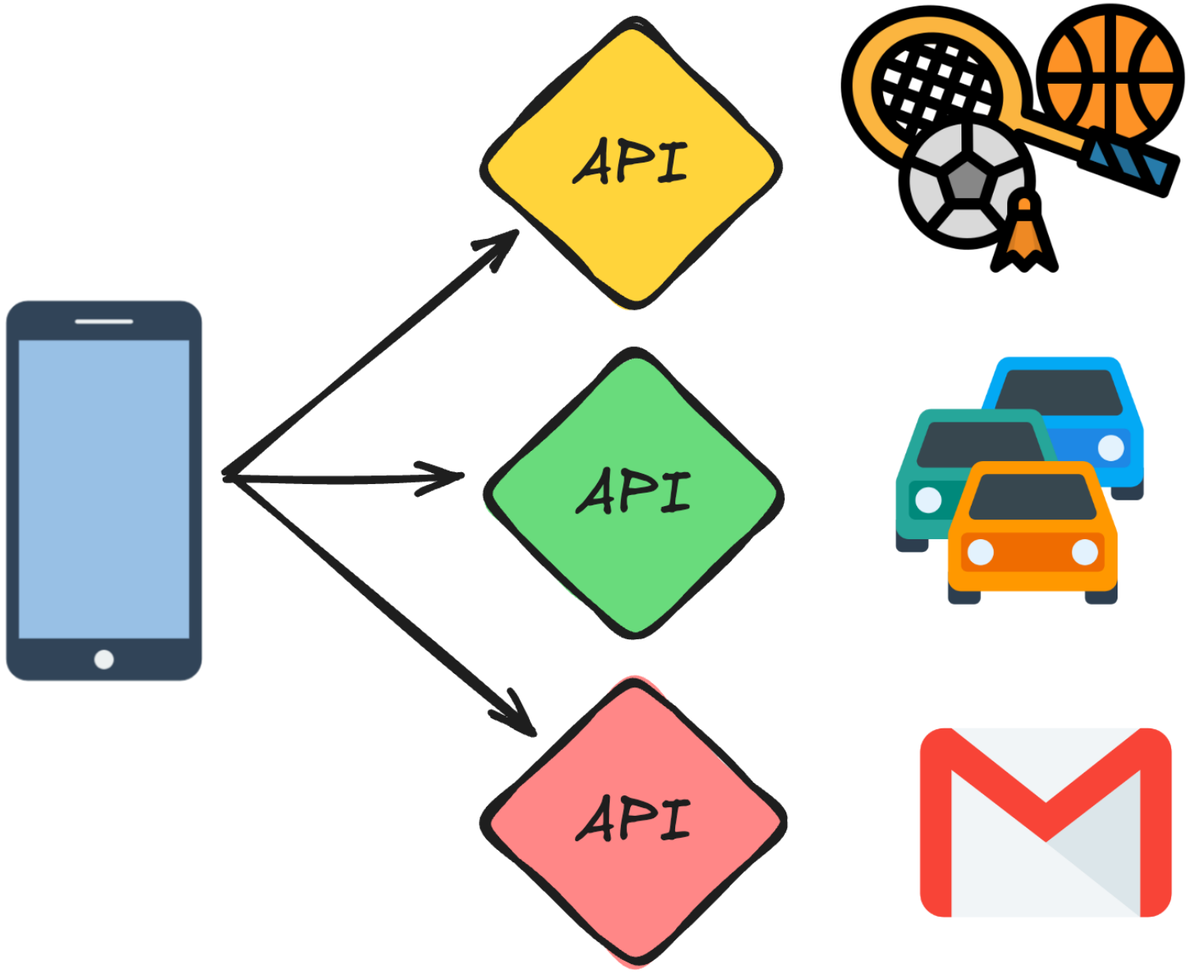
The Big LLM Problem



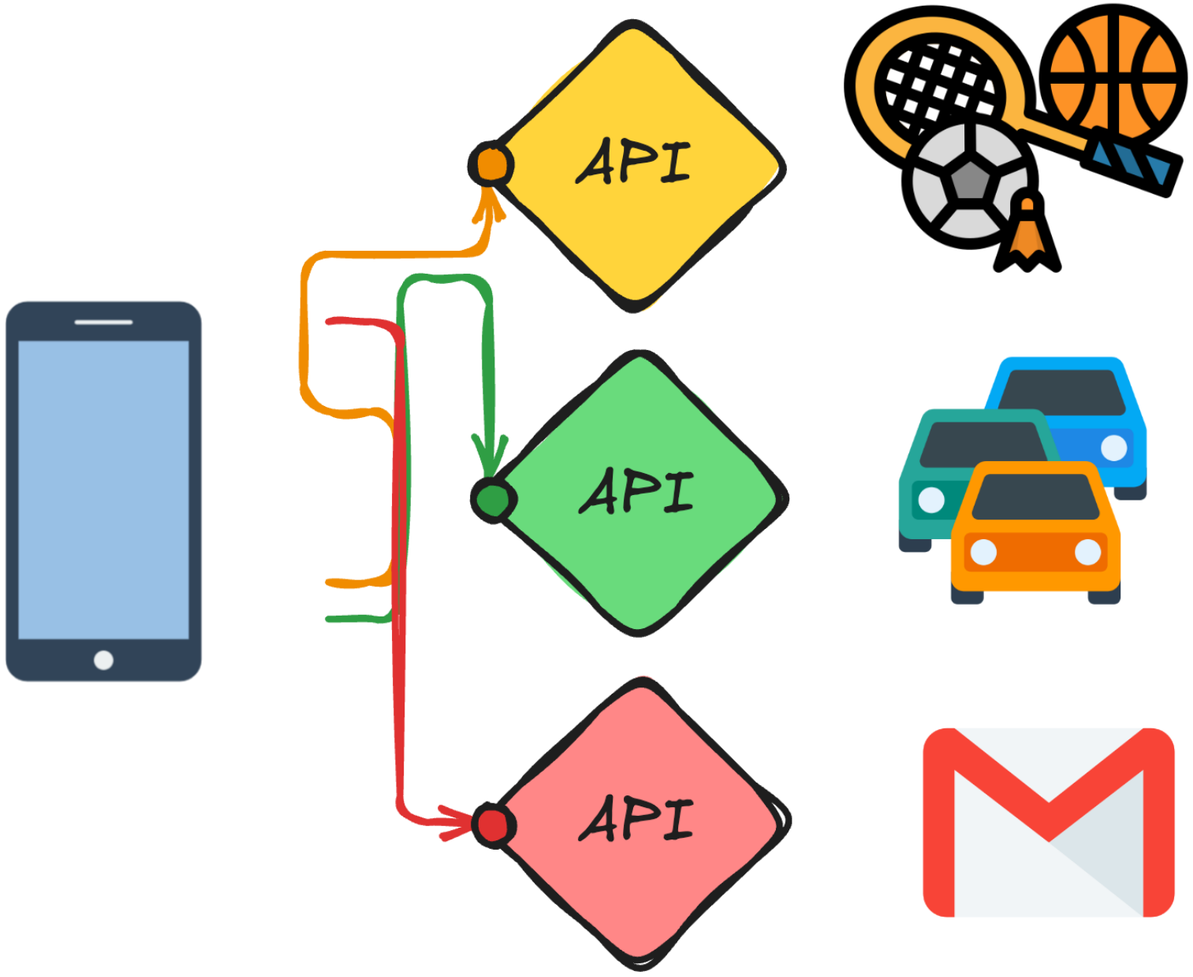
The Big LLM Problem



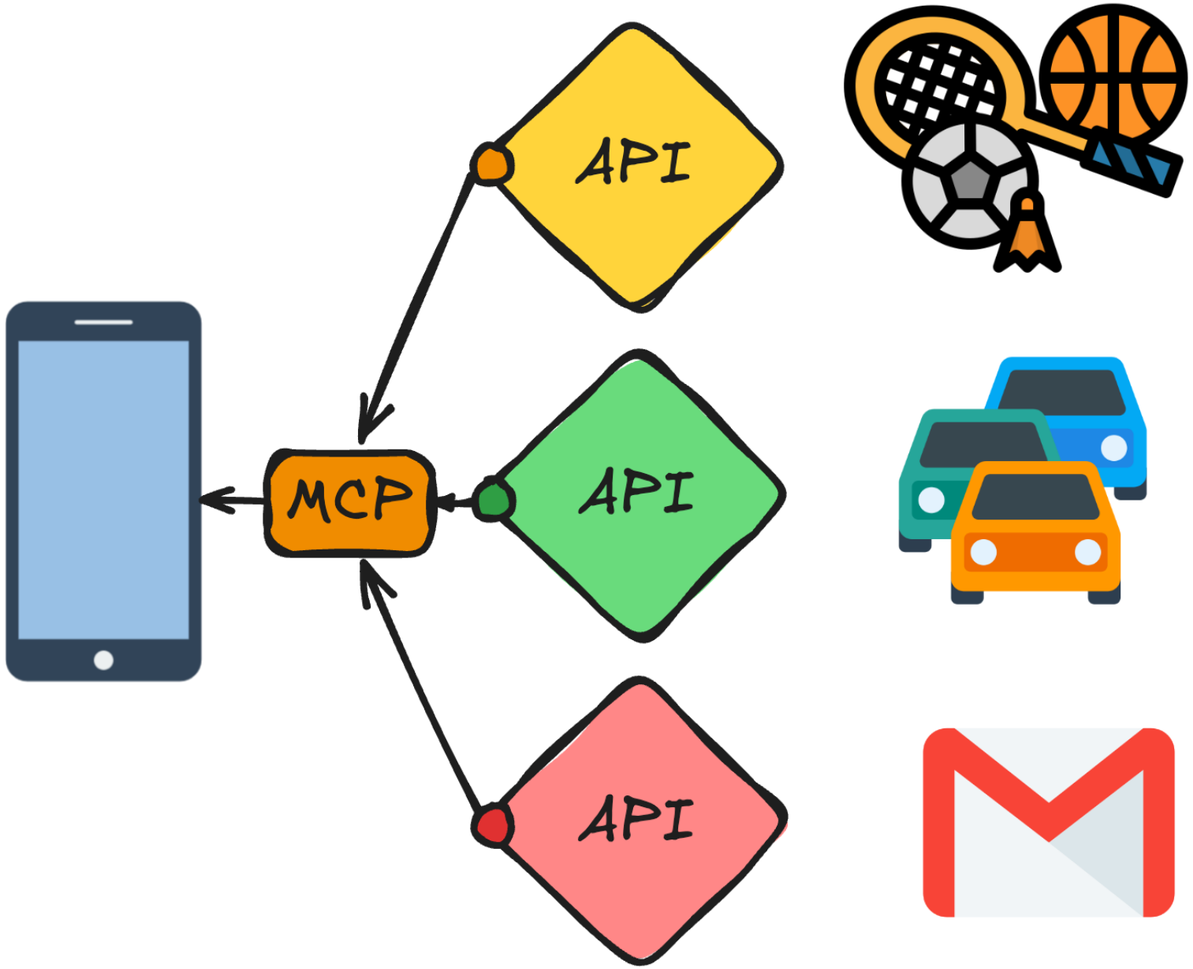
The Big LLM Problem



The Big LLM Problem



MCP to the Rescue!

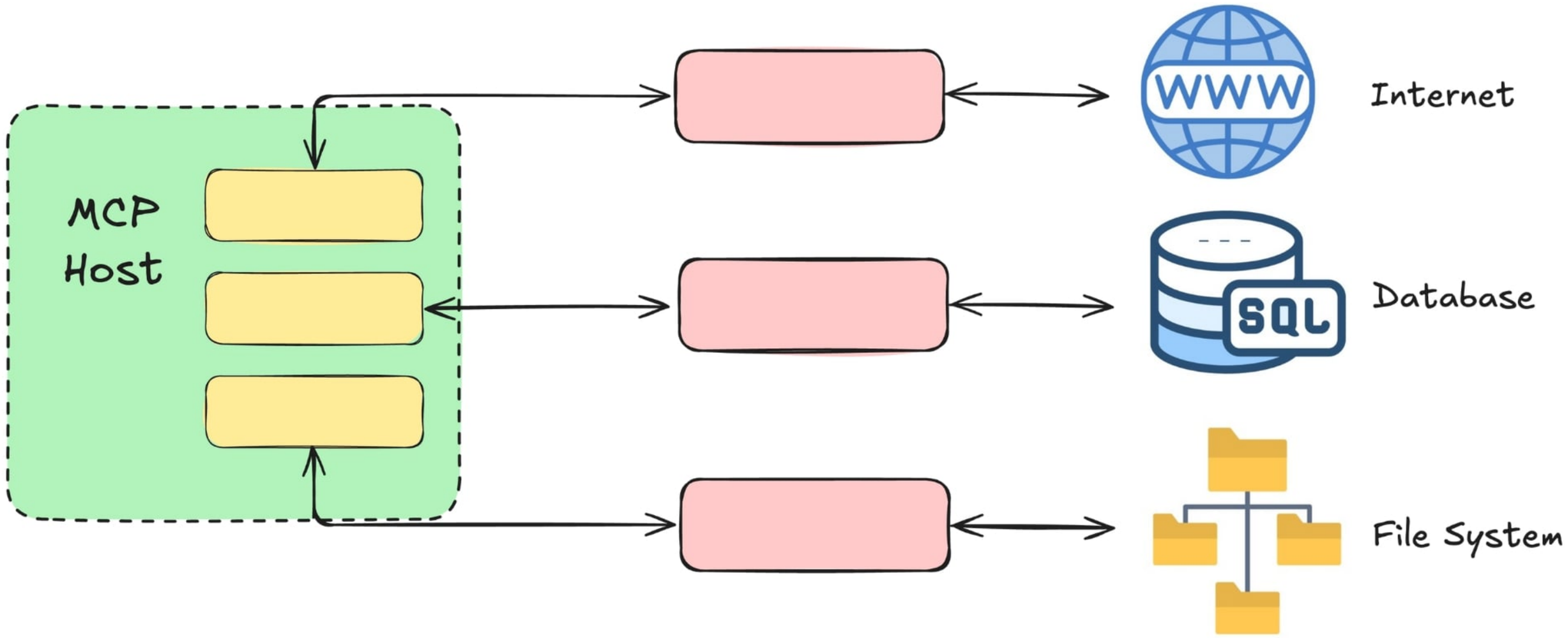


MCP to the Rescue!

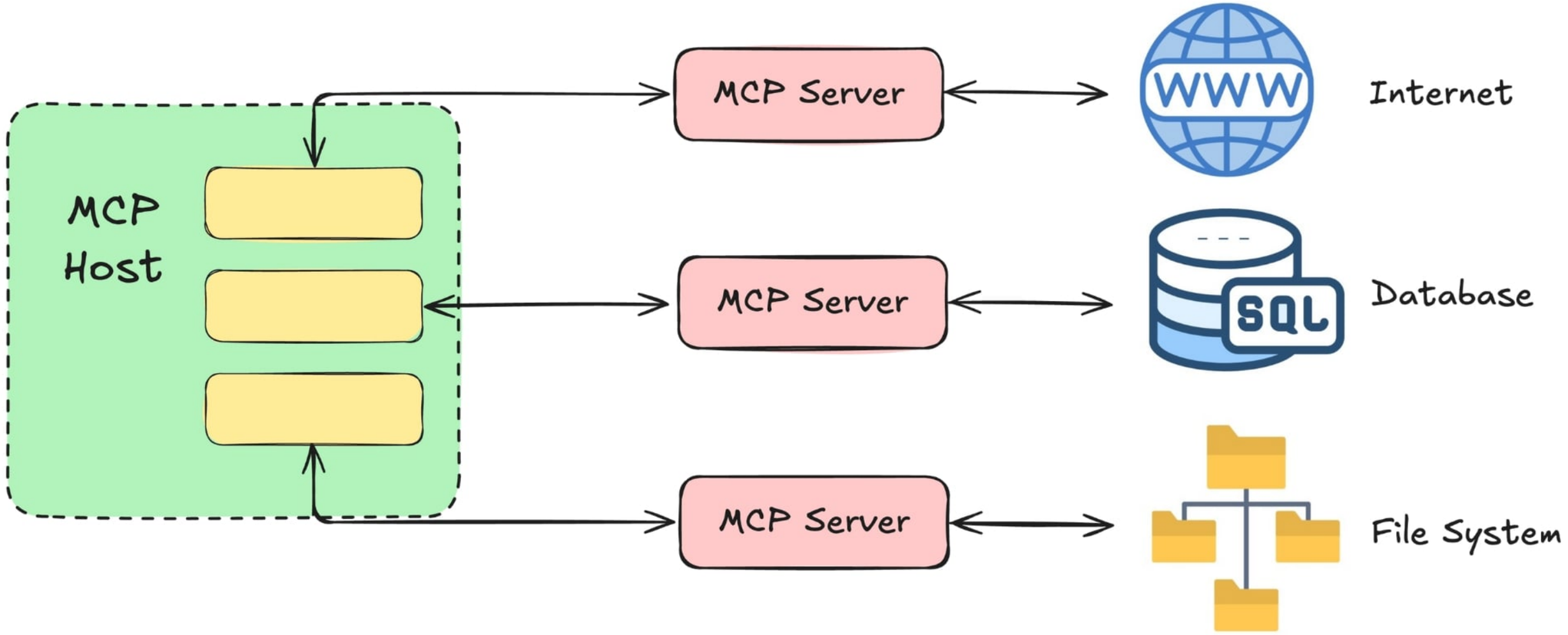
- **Simplifies connectivity** and adds features
  - *Dynamic tool discovery* → AI can "see" what functionality is available
- MCP components:
  - Host, client, and the server



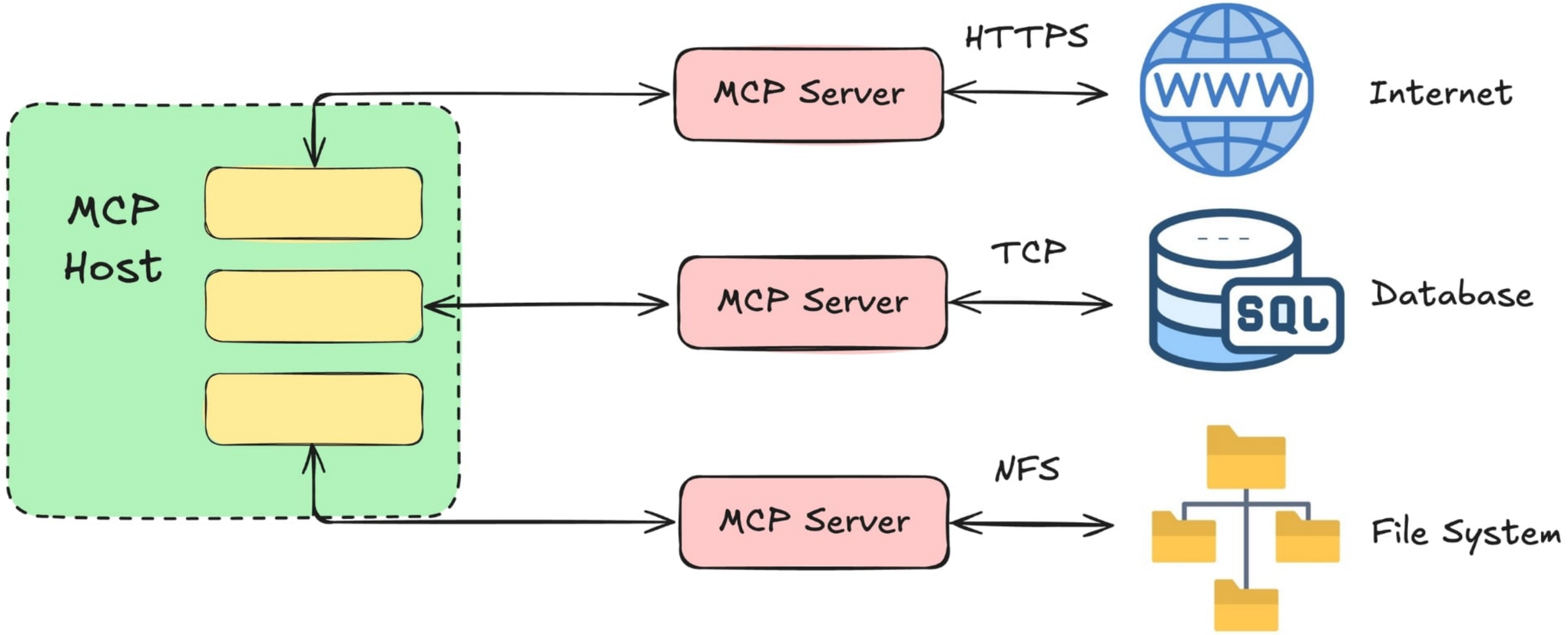
The MCP Architecture



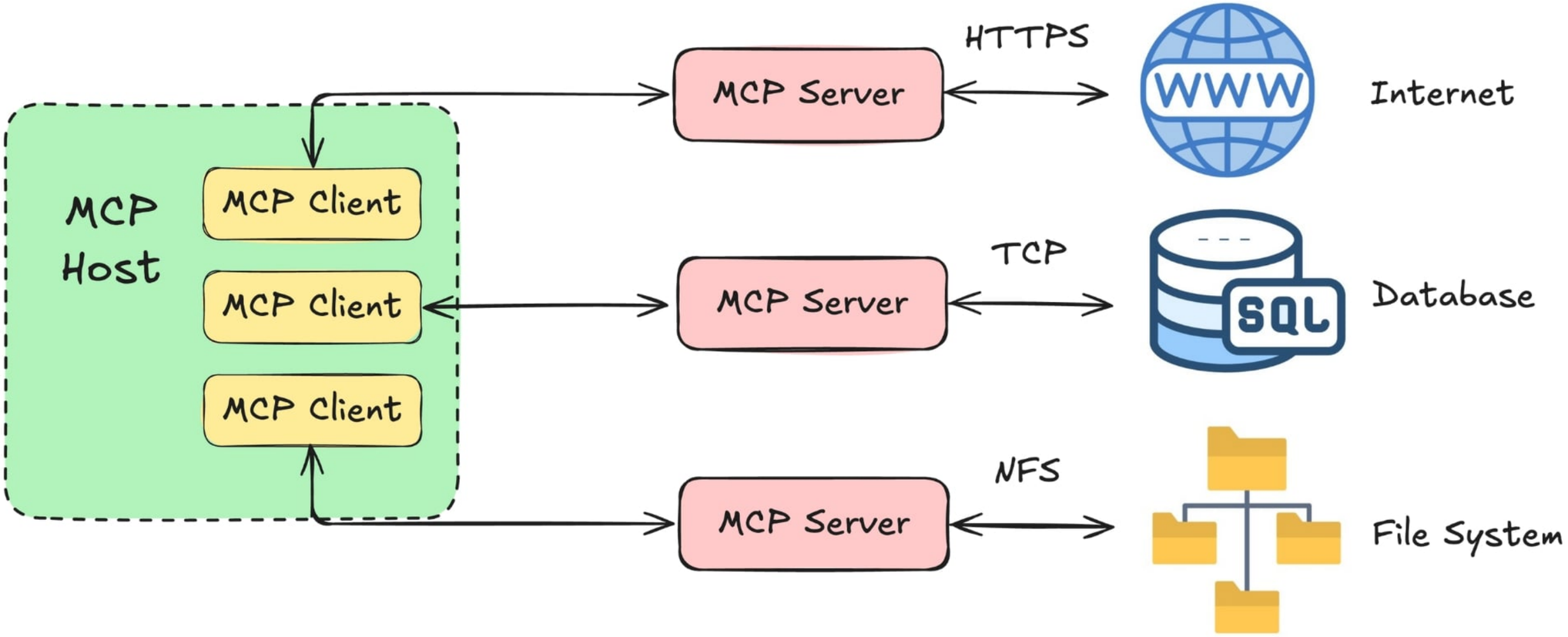
The MCP Architecture



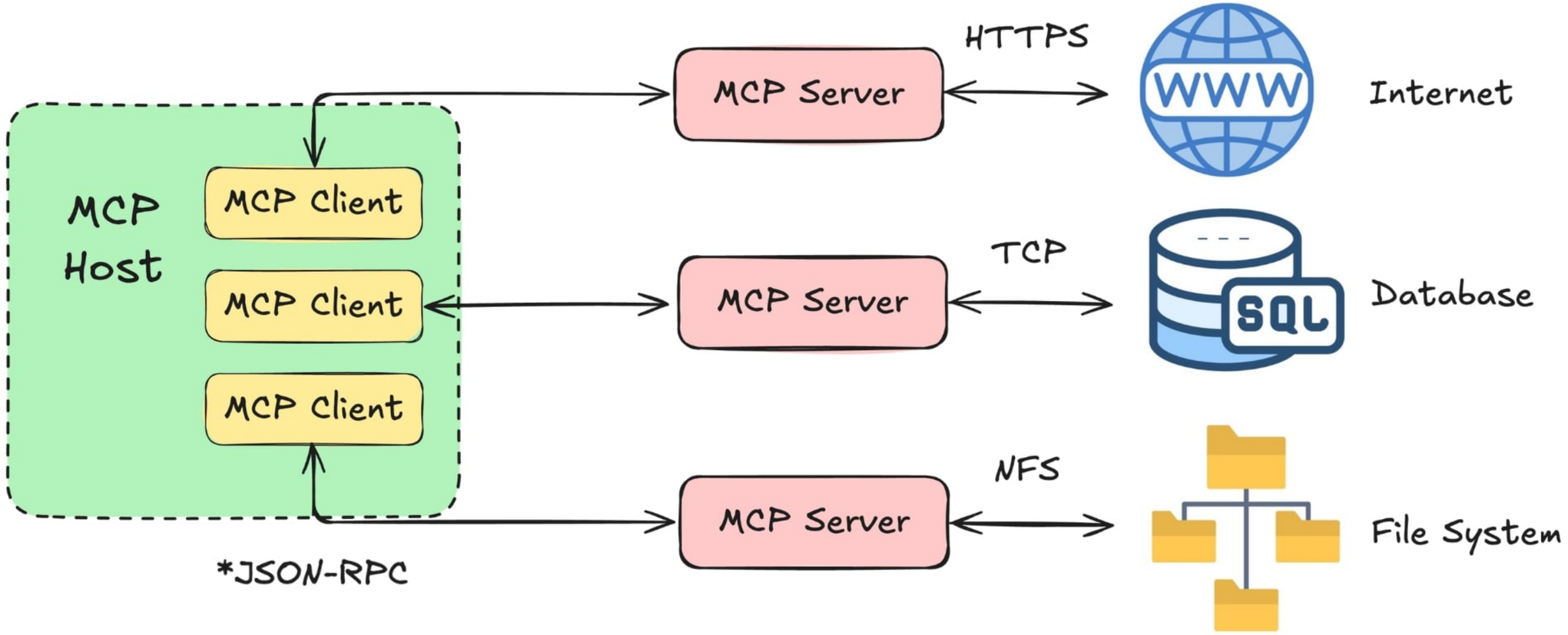
The MCP Architecture



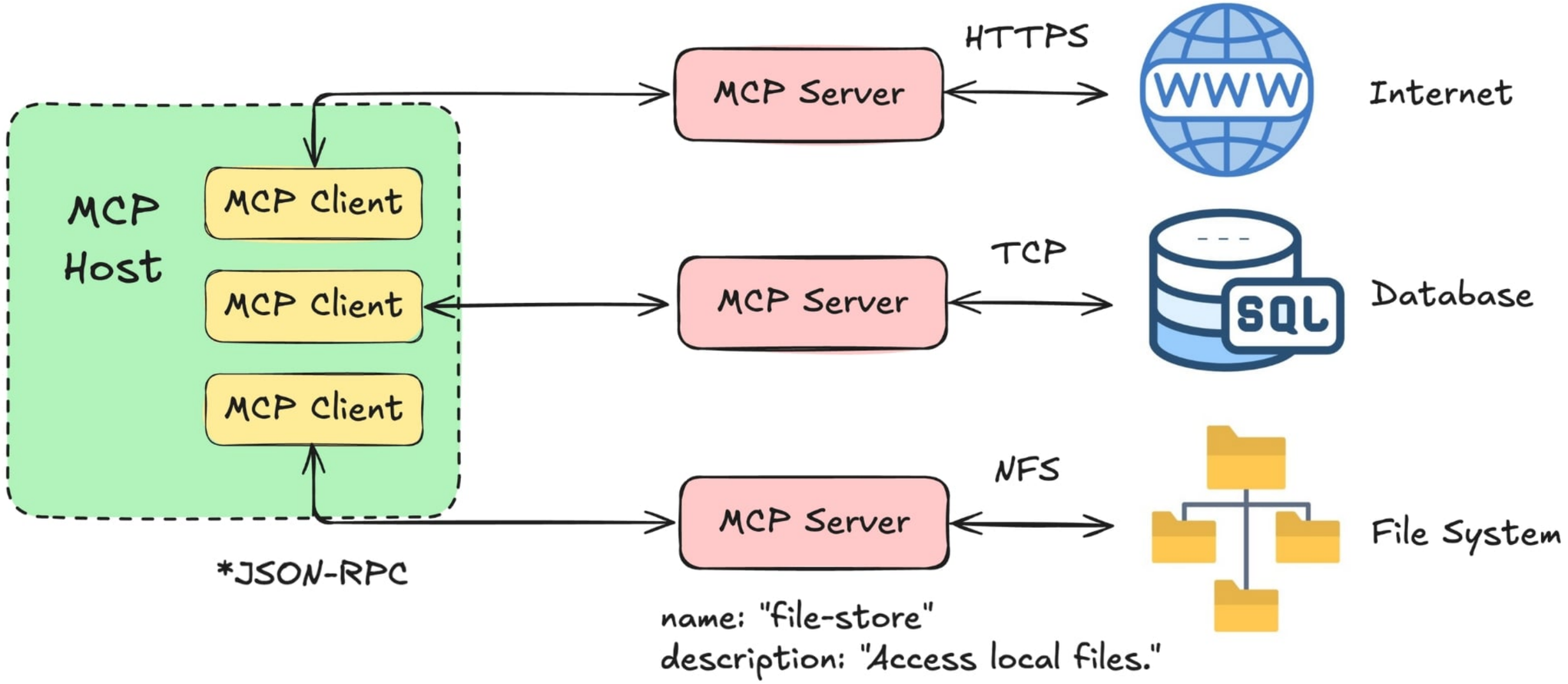
The MCP Architecture



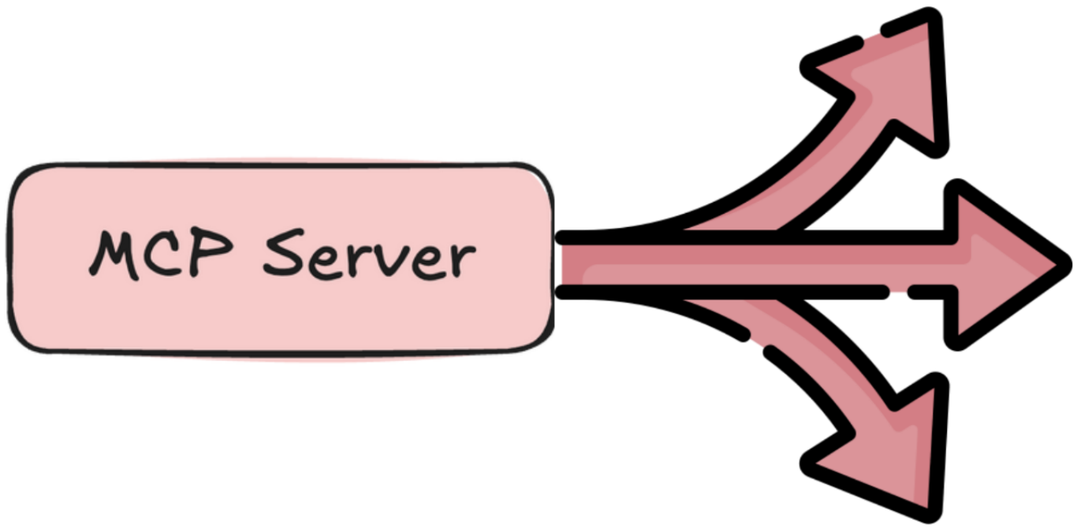
The MCP Architecture



The MCP Architecture



Primitives



Tools

-> Actions the LLM can perform
-> Querying a database, booking a flight, running code

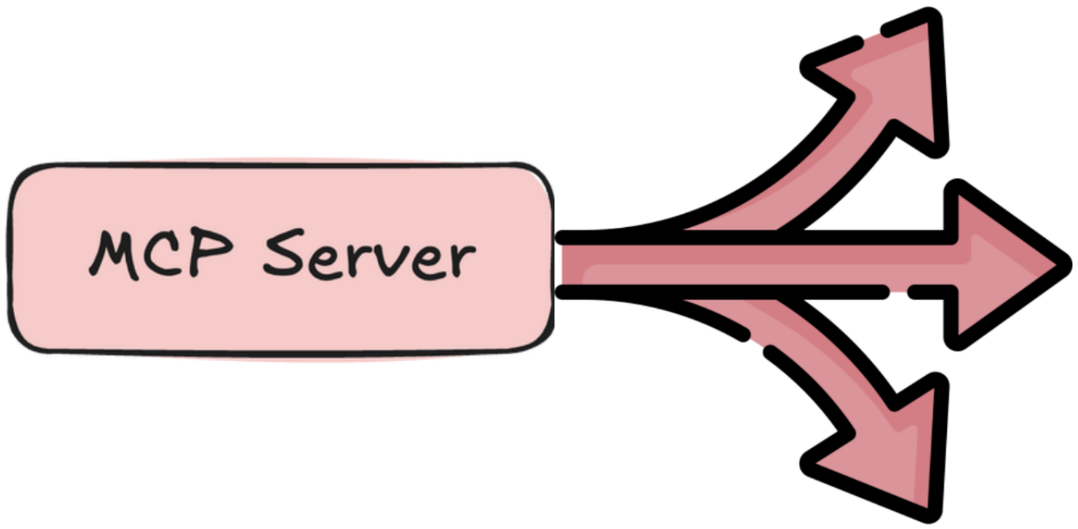
Resources

-> Read-only data or context
-> Documents, files, database records

Prompts

-> Pre-defined workflows and instructions
-> Saves users from needing to write long complex prompts

Primitives



Tools

-> Actions the LLM can perform
-> Querying a database, booking a flight, running code

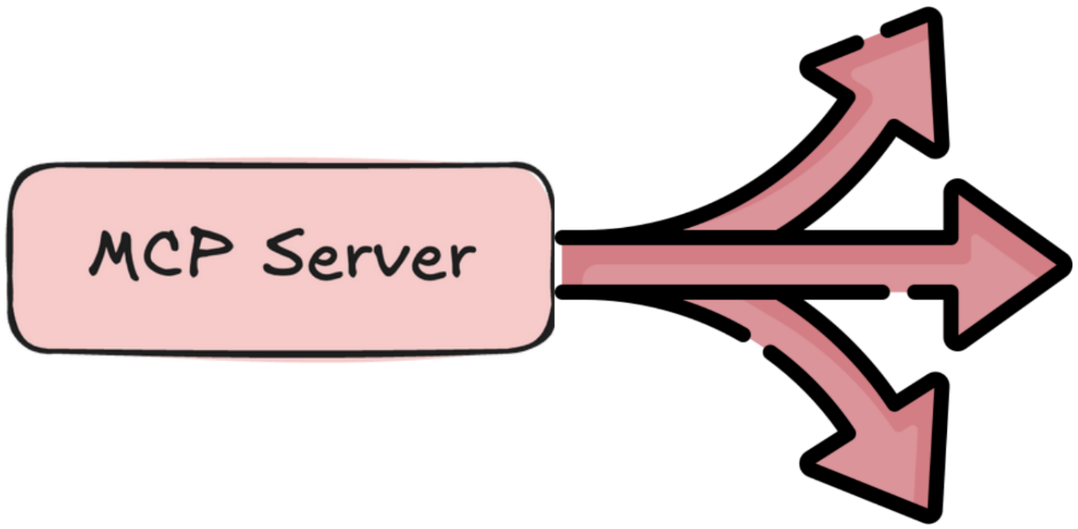
Resources

-> Read-only data or context
-> Documents, files, database records

Prompts

-> Pre-defined workflows and instructions
-> Saves users from needing to write long complex prompts

Primitives



Tools

-> Actions the LLM can perform
-> Querying a database, booking a flight, running code

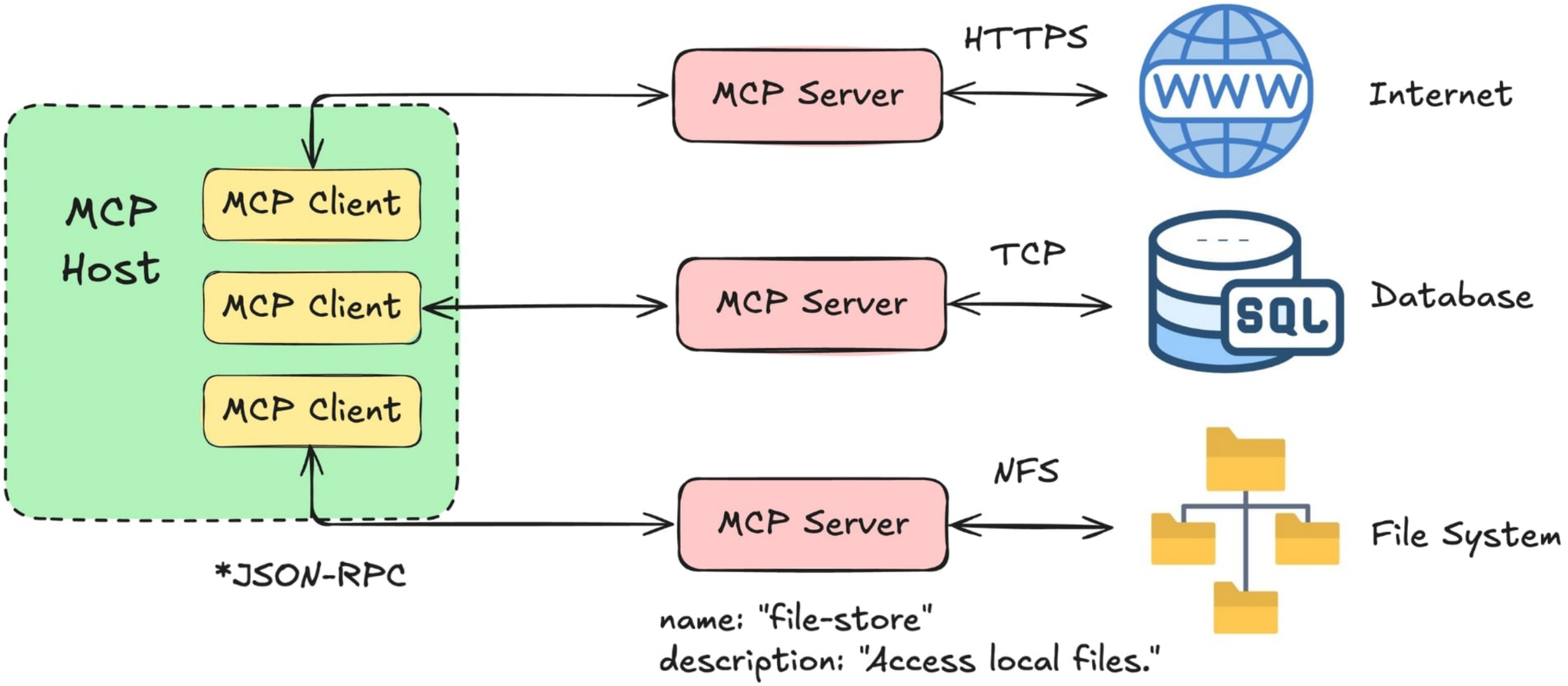
Resources

-> Read-only data or context
-> Documents, files, database records

Prompts

-> Pre-defined workflows and instructions
-> Saves users from needing to write long complex prompts

The MCP Architecture



Let's practice!

INTRODUCTION TO MODEL CONTEXT PROTOCOL (MCP)

Building Your First MCP Server

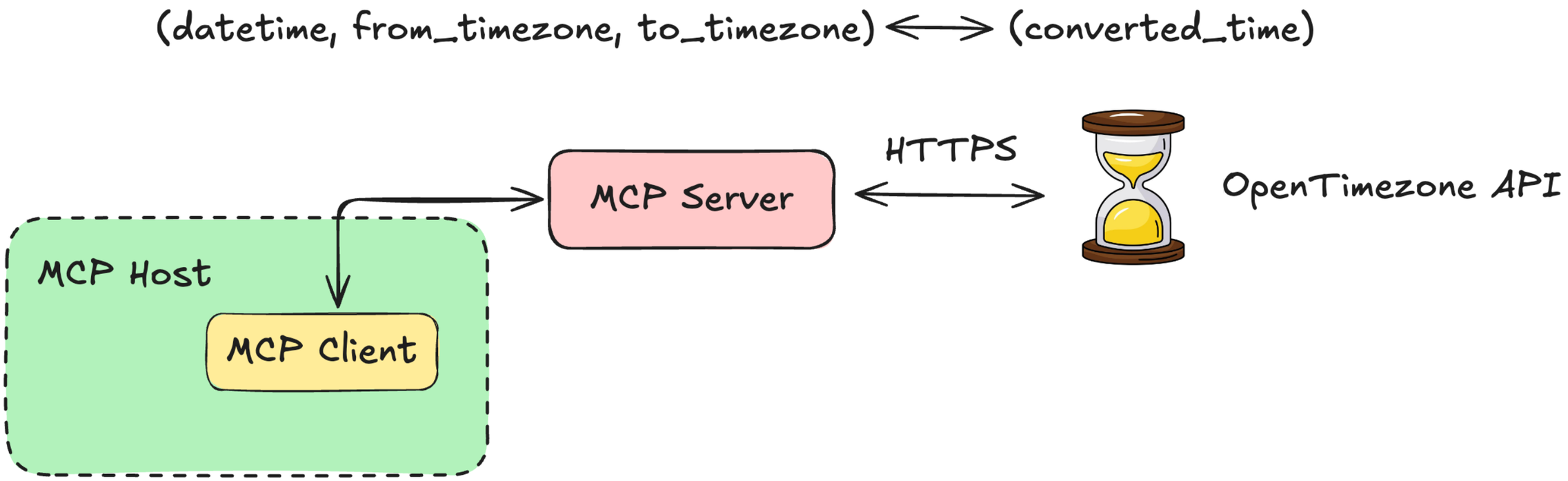
INTRODUCTION TO MODEL CONTEXT PROTOCOL (MCP)



James Chapman
AI Curriculum Manager, DataCamp

datacamp

Converting Between Timezones



[1] https://opentimezone.com/

```python
# Create an MCP server instance
mcp = FastMCP("Timezone and Currency Converter")

@mcp.tool()
def convert_timezone(date_time: str, from_timezone: str, to_timezone: str) -> str:
    """
    Convert a datetime from one timezone to another.

    Args:
        date_time: The datetime string in ISO format (e.g., '2025-01-20T14:30:00')
        from_timezone: Source timezone (e.g., 'America/New_York')
        to_timezone: Target timezone (e.g., 'Europe/London')

    Returns:
        A string with the converted datetime and timezone information
    """
    # Define the API endpoint and format input data
    url = "https://api.opentimezone.com/convert"

    payload = {
        "dateTime": date_time,
        "fromTimezone": from_timezone,
        "toTimezone": to_timezone
    }

    # Make the API request
    response = requests.post(url, json=payload)

    data = response.json()
    converted_time = data.get('dateTime', 'N/A')

    return f"Time in {to_timezone}: {converted_time}"
```

Instantiate MCP Server

Define Tools

Doctrings and Typing

Step 1: Instantiating the Server

```python
from mcp.server.fastmcp import FastMCP
import requests

# Create an MCP server instance
mcp = FastMCP("Timezone Converter")
```

[1] https://github.com/modelcontextprotocol/python-sdk

Step 2: Adding the Tools

```python
from mcp.server.fastmcp import FastMCP
import requests

# Create an MCP server instance
mcp = FastMCP("Timezone Converter")

@mcp.tool()
def convert_timezone(date_time, from_timezone, to_timezone):
```

Step 2: Adding the Tools

```python
@mcp.tool()
def convert_timezone(date_time, from_timezone, to_timezone):

    # Define the API endpoint and format input data
    url = "https://api.opentimezone.com/convert"
    payload = {"dateTime": date_time, "fromTimezone": from_timezone, "toTimezone": to_timezone}

    response = requests.post(url, json=payload)
    data = response.json()
    converted_time = data.get('dateTime', 'N/A')
    return f"Time in {to_timezone}: {converted_time}"
```

Step 3: Docstrings and Type Hints

```python
@mcp.tool()
def convert_timezone(date_time: str, from_timezone: str, to_timezone: str) -> str:
    """
    Convert a datetime from one timezone to another.

    Args:
        date_time: The datetime string in ISO format (e.g., '2025-01-20T14:30:00')
        from_timezone: Source timezone (e.g., 'America/New_York')
        to_timezone: Target timezone (e.g., 'Europe/London')

    Returns:
        A string with the converted datetime and timezone information
    """
```

```python
# Create an MCP server instance
mcp = FastMCP("Timezone and Currency Converter")

@mcp.tool()
def convert_timezone(date_time: str, from_timezone: str, to_timezone: str) -> str:
    """
    Convert a datetime from one timezone to another.

    Args:
        date_time: The datetime string in ISO format (e.g., '2025-01-20T14:30:00')
        from_timezone: Source timezone (e.g., 'America/New_York')
        to_timezone: Target timezone (e.g., 'Europe/London')

    Returns:
        A string with the converted datetime and timezone information
    """
    # Define the API endpoint and format input data
    url = "https://api.opentimezone.com/convert"

    payload = {
        "dateTime": date_time,
        "fromTimezone": from_timezone,
        "toTimezone": to_timezone
    }

    # Make the API request
    response = requests.post(url, json=payload)

    data = response.json()
    converted_time = data.get('dateTime', 'N/A')

    return f"Time in {to_timezone}: {converted_time}"
```

Instantiate MCP Server

Define Tools

Doctrings and Typing

Let's practice!

INTRODUCTION TO MODEL CONTEXT PROTOCOL (MCP)

MCP Client-Server Communications

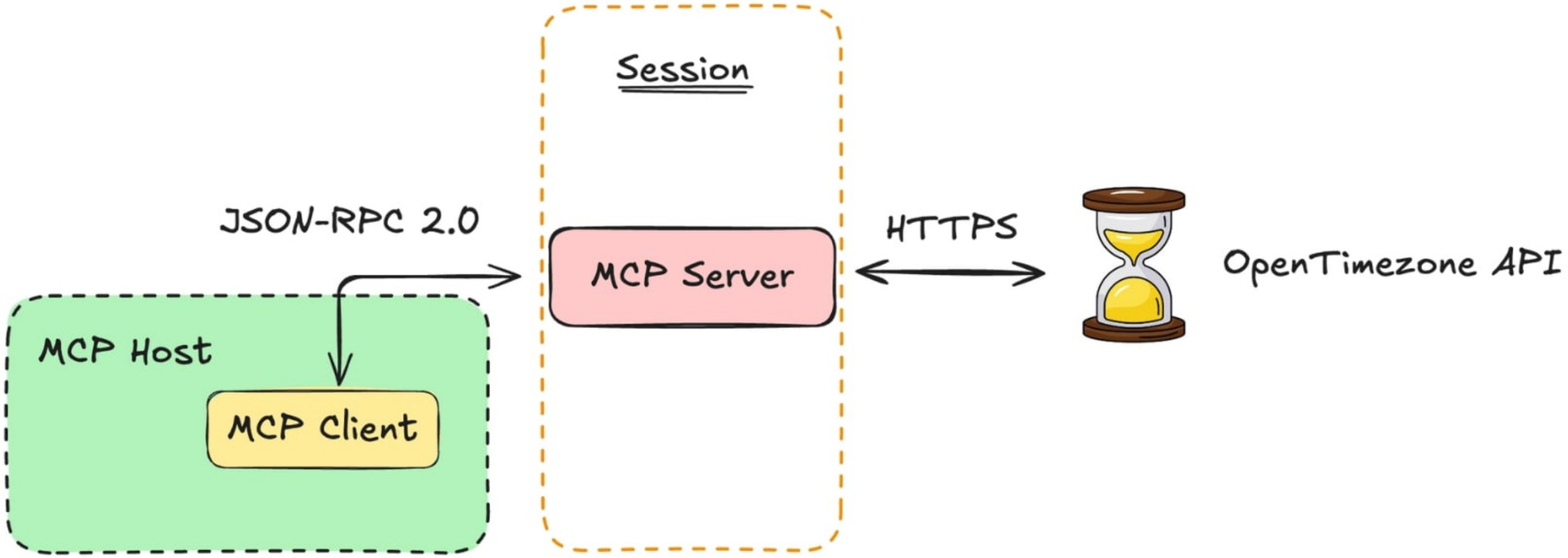
INTRODUCTION TO MODEL CONTEXT PROTOCOL (MCP)



James Chapman
AI Curriculum Manager, DataCamp

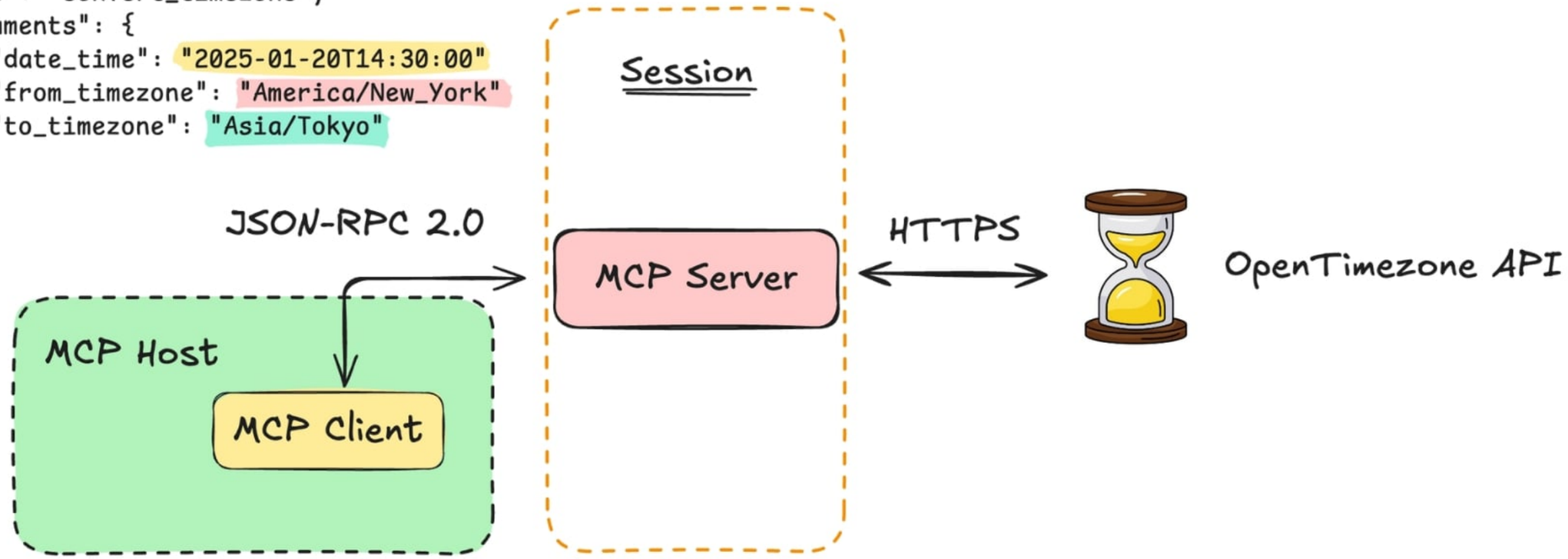
datacamp

Client-Server Communications

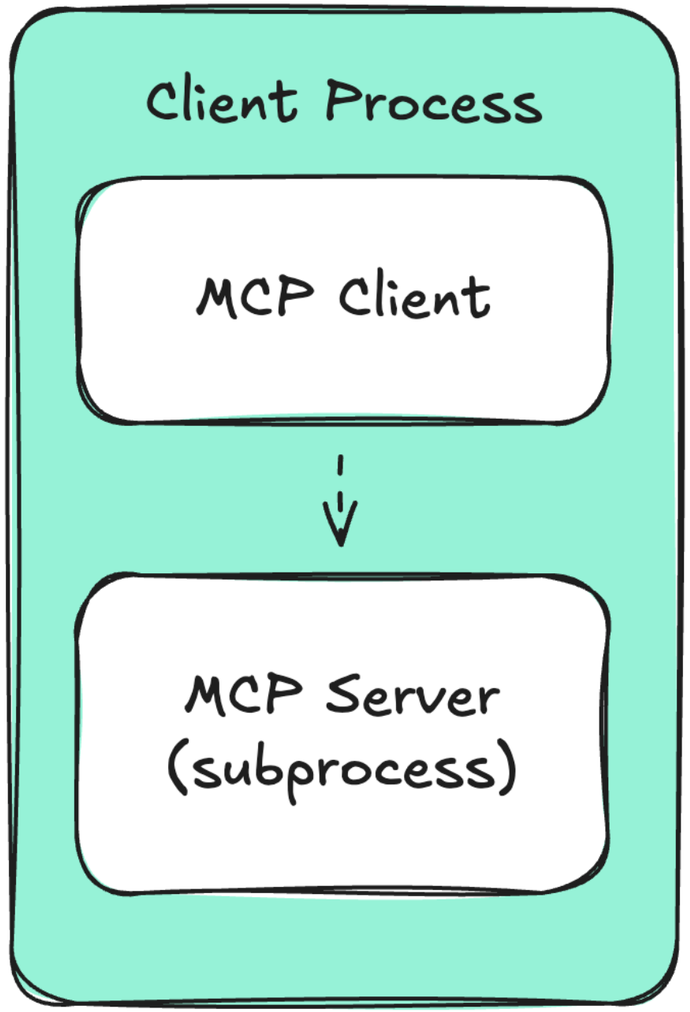


Client-Server Communications

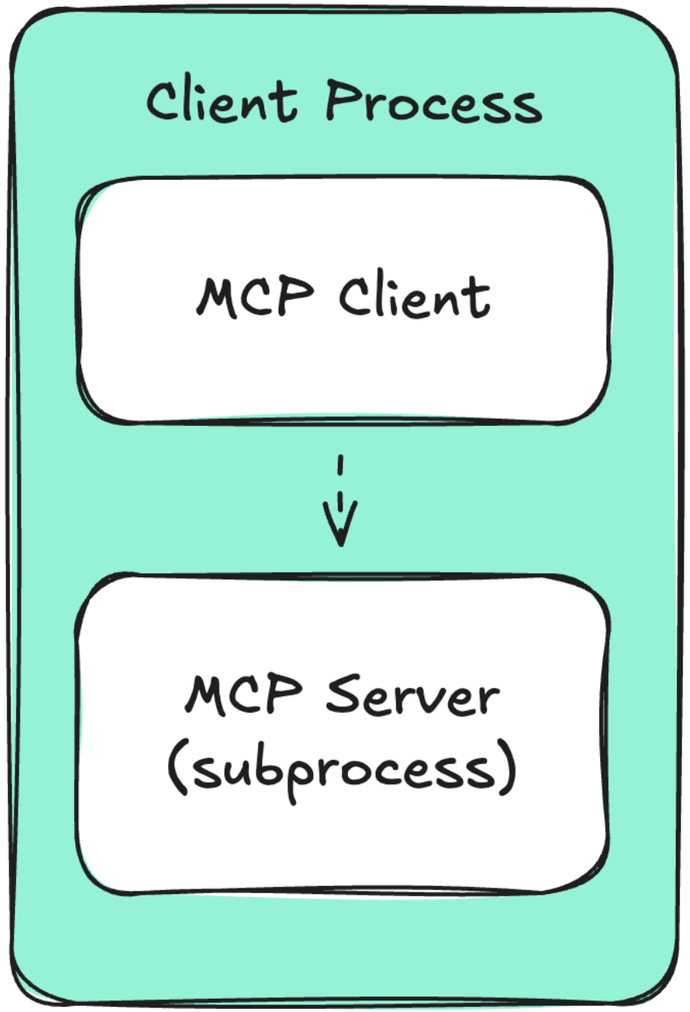
```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "tools/call",
    "params": {
        "name": "convert_timezone",
        "arguments": {
            "date_time": "2025-01-20T14:30:00"
            "from_timezone": "America/New_York"
            "to_timezone": "Asia/Tokyo"
        }
    }
}
```



Transports: Standard I/O (stdio)

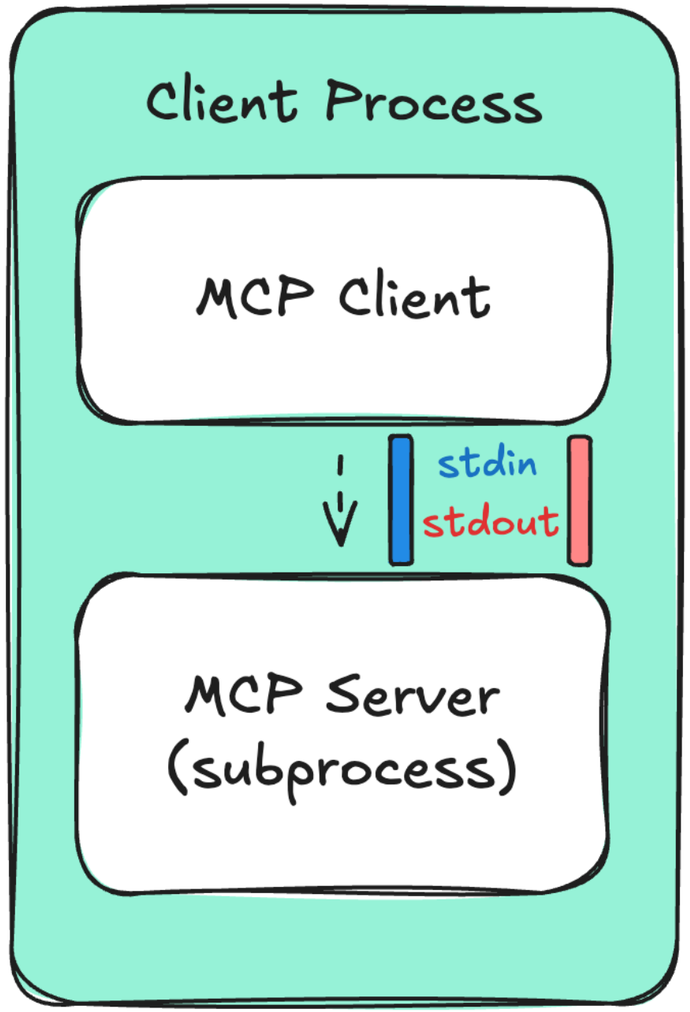


Transports: Standard I/O (stdio)



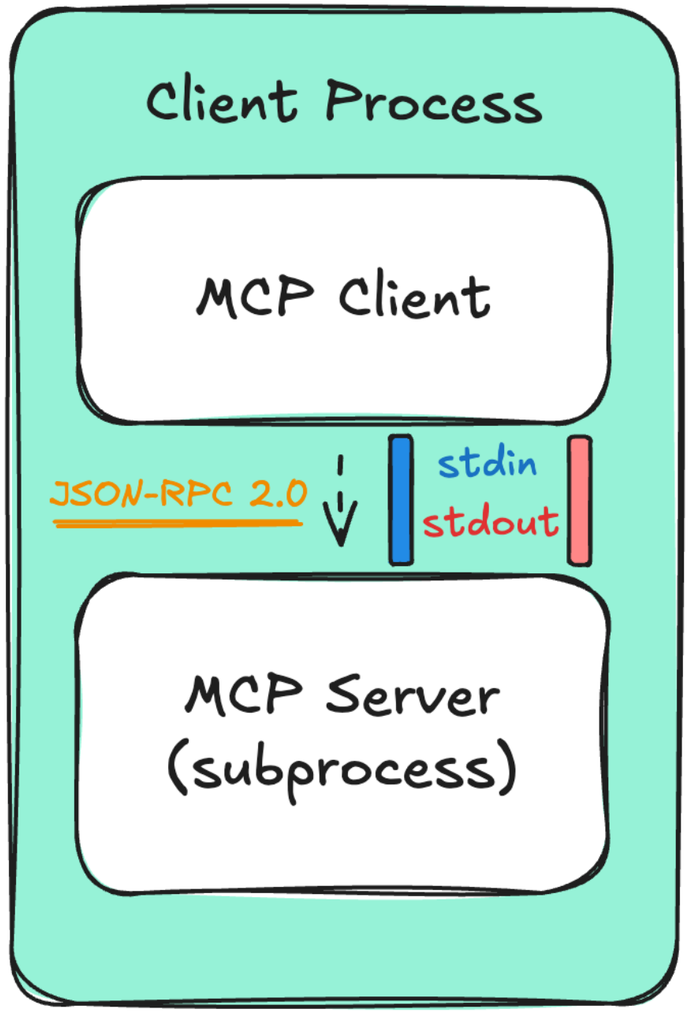
- Server runs as **child process**

Transports: Standard I/O (stdio)



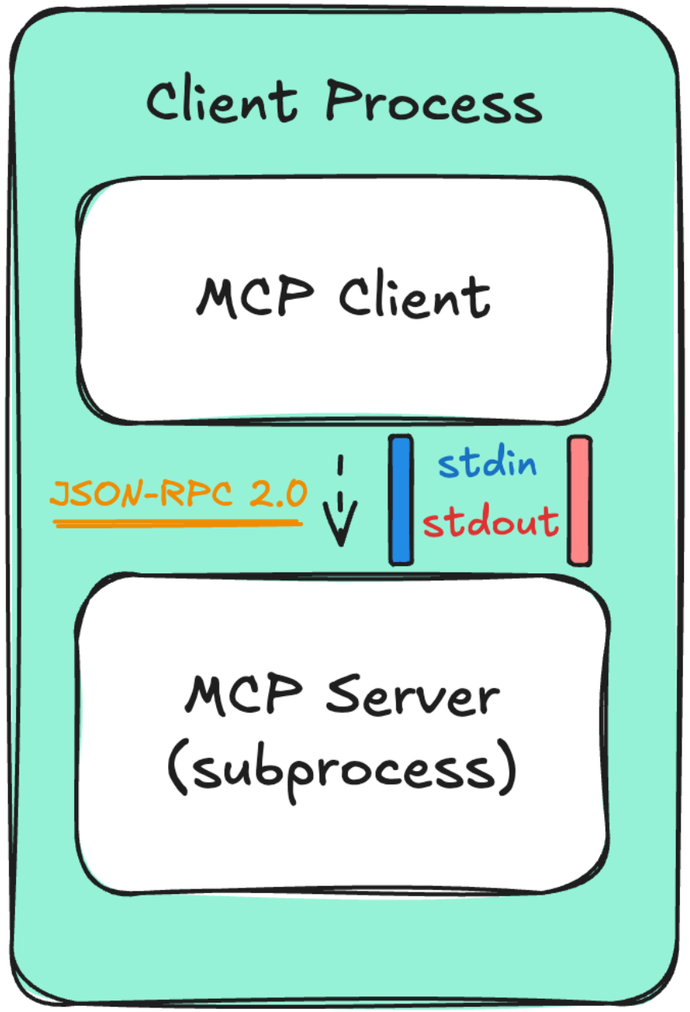
- Server runs as **child process**
- Communication via pipes (**stdin/stdout**)

Transports: Standard I/O (stdio)



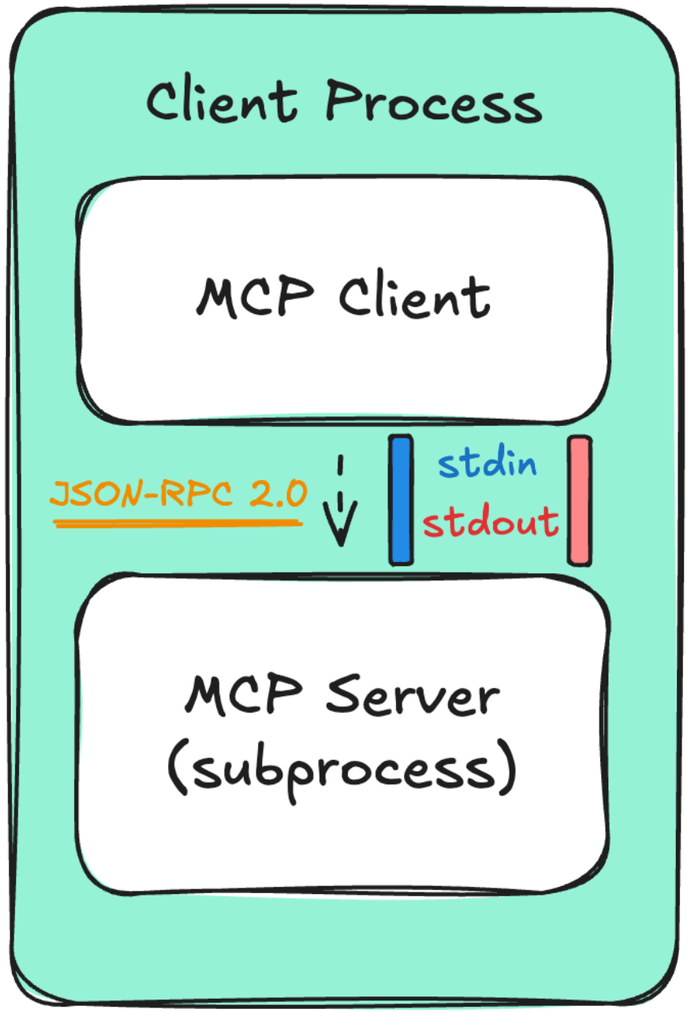
- Server runs as **child process**
- Communication via pipes (**stdin/stdout**)

Transports: Standard I/O (stdio)



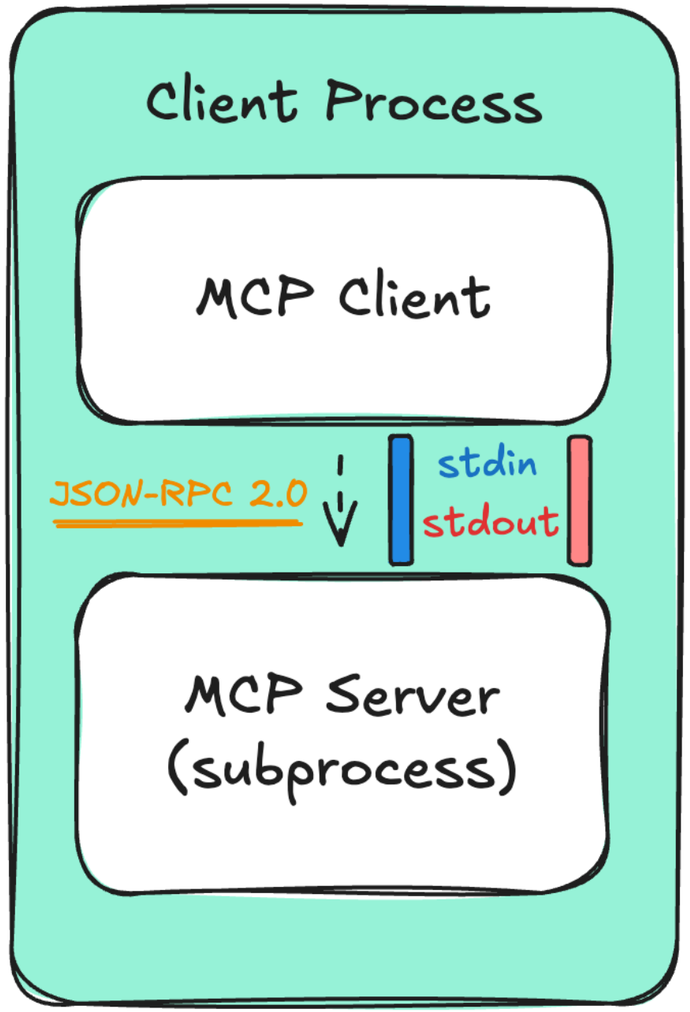
- Server runs as **child process**
- Communication via pipes (**stdin/stdout**)
- *Local only* - same machine

Transports: Standard I/O (stdio)



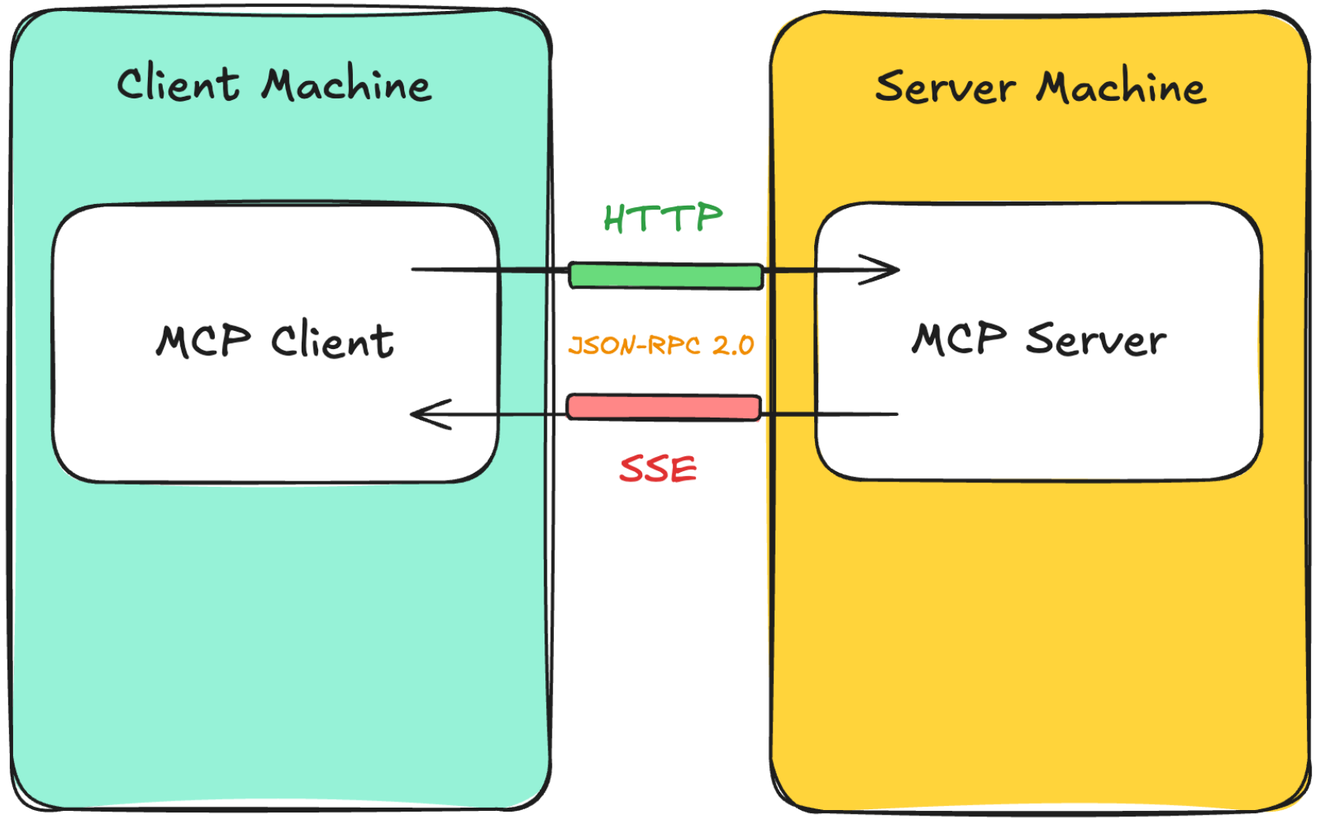
- Server runs as **child process**
- Communication via pipes (**stdin/stdout**)
- *Local only* - same machine
- Simple, no network configuration

Transports: Standard I/O (stdio)



- Server runs as **child process**
- Communication via pipes (**stdin/stdout**)
- *Local only* - same machine
- Simple, no network configuration
- **Server dies when client exits**

Transports: Streamable HTTP



MCP Server: timezone_server.py

```python
import mcp
from mcp.server.fastmcp import FastMCP
import requests

mcp = FastMCP("Timezone Converter")

@mcp.tool()
def convert_timezone(date_time: str, from_timezone: str, to_timezone: str) -> str:
    ...

if __name__ == "__main__":
    mcp.run(transport="stdio")
```

MCP Client: Listing Server Tools

```python
from mcp import ClientSession, StdioServerParameters
from mcp.client.stdio import stdio_client

async def get_tools_from_mcp():
    # Define the server parameters
    params = StdioServerParameters(command=sys.executable, args=["timezone_server.py"])

    # Connect to the MCP server and open a session
    async with stdio_client(params) as (reader, writer):
        async with ClientSession(reader, writer) as session:
            # Initialize the session
            await session.initialize()
            # Ask the server what tools it provides
            response = await session.list_tools()
```

MCP Client: Listing Server Tools

```
async def get_tools_from_mcp():
            # ...

            # Ask the server what tools it provides
            response = await session.list_tools()
            print("Connected to MCP server!")
            print("Available tools:")
            for tool in response.tools:
                print(f" - {tool.name}: {tool.description}")

            return response.tools

asyncio.run(get_tools_from_mcp())
```

MCP Client: Listing Server Tools

```
Connected to MCP server!
Available tools:
 - convert_timezone:
    Convert a datetime from one timezone to another.

    Args:
        date_time: The datetime string in ISO format (e.g., '2025-01-20T14:30:00')
        from_timezone: Source timezone (e.g., 'America/New_York')
        to_timezone: Target timezone (e.g., 'Europe/London')

    Returns:
        A string with the converted datetime and timezone information
```

MCP Client: Calling Server Tools

```python
async def call_mcp_tool(tool_name: str, arguments: dict) -> str:
    params = StdioServerParameters(command=sys.executable, args=["timezone_server.py"])

    async with stdio_client(params) as (reader, writer):
        async with ClientSession(reader, writer) as session:

            await session.initialize()

            result = await session.call_tool(tool_name, arguments)

            text_content = result.content[0].text

            print(f"Conversion Result: {text_content}")
            return str(text_content)
```

MCP Client: Calling Server Tools

```
asyncio.run(
    call_mcp_tool(
        "convert_timezone",
        {"date_time": "2025-01-20T14:30:00",
         "from_timezone": "America/New_York",
         "to_timezone": "Asia/Tokyo"}
    )
)
```

```
'2025-01-21T04:30:00+09:00'
```

Let's practice!

INTRODUCTION TO MODEL CONTEXT PROTOCOL (MCP)