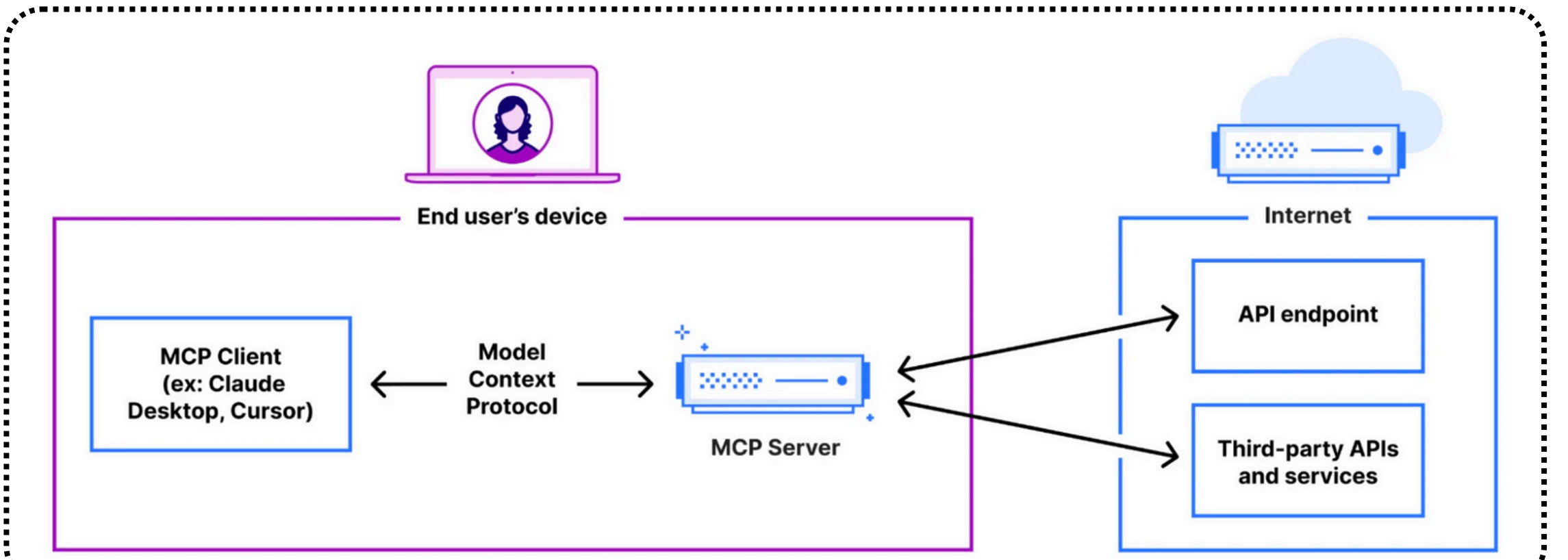
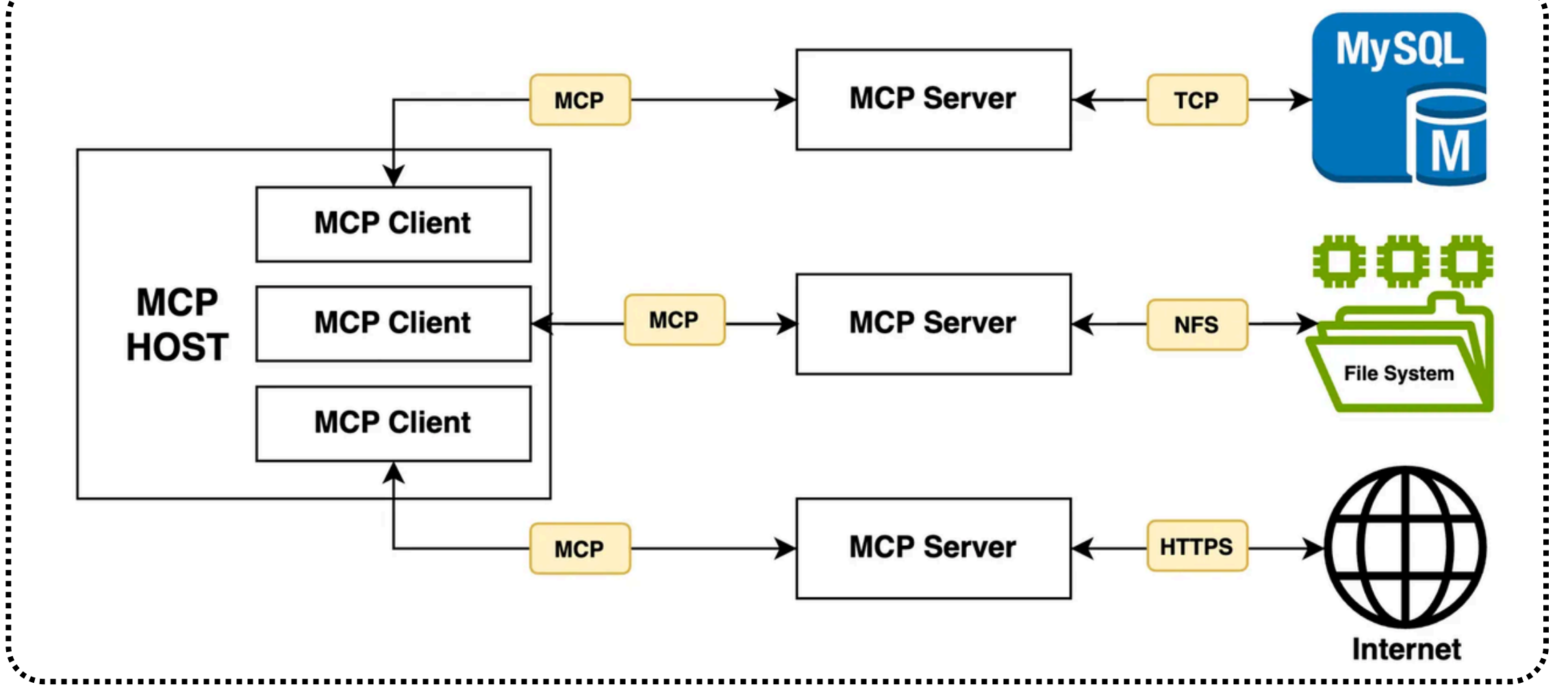
Analytics Vidhya

How to Create an MCP Client Server Using LangChain





Setting Up the Environment

Before building our MCP client server using LangChain, let's prepare the environment. You need these items:

- Python version 3.11 or newer.
- Set up a new virtual environment (optional)
- An API key (e.g., OpenAI or Groq, depending on the model you choose).
- Specific Python libraries: langchain-mcp-adapters, langgraph, and an LLM library (like langchain-openai or langchain-groq) of your choice.

Install the needed libraries using pip. Open your terminal or command prompt and run:

```
pip install langchain-mcp-adapters langgraph langchain-groq # Or langchain-openai
```

Make sure you have the correct Python version and necessary keys ready.

Building the MCP Server

The MCP server's job is to offer tools the client can use. In our MCP client server using langchain example, we will build a simple server. This server will handle basic math operations as well as complex weather api to get weather details of a city. Understanding what is MCP server functionality starts here.

Create a Python file named mcp_server.py:

Let's import the required libraries

```
import math

import requests

from mcp.server.fastmcp import FastMCP
```

Initialize the FastMCP object

```
mcp= FastMCP("Math")
```

Let’s define the math tools

```python
@mcp.tool()
def add(a: int, b: int) -> int:
    print(f"Server received add request: {a}, {b}")
    return a + b

@mcp.tool()
def multiply(a: int, b: int) -> int:
    print(f"Server received multiply request: {a}, {b}")
    return a * b

@mcp.tool()
def sine(a: int) -> int:
    print(f"Server received sine request: {a}")
    return math.sin(a)
```

Now, Let’s define a weather tool, make sure you have API from here.

```python
WEATHER_API_KEY = "YOUR_API_KEY"

@mcp.tool()

def get_weather(city: str) -> dict:

    """

    Fetch current weather for a given city using WeatherAPI.com.

    Returns a dictionary with city, temperature (C), and condition.

    """

    print(f"Server received weather request: {city}")

    url = f"http://api.weatherapi.com/v1/current.json?key={WEATHER_API_KEY}&q={city}"

    response = requests.get(url)

    if response.status_code != 200:

        return {"error": f"Failed to fetch weather for {city}."}

    data = response.json()

    return {

        "city": data["location"]["name"],

        "region": data["location"]["region"],

        "country": data["location"]["country"],

        "temperature_C": data["current"]["temp_c"],

        "condition": data["current"]["condition"]["text"]

    }

    5. Now, instantiate the mcp server

if __name__ =="__main__":

    print("Starting MCP Server....")

    mcp.run(transport="stdio")
```

Explanation

This script sets up a simple MCP server named “Math”. It uses FastMCP to define four tools, add, multiply, sine and get_weather marked by the @mcp.tool() decorator. Type hints tell MCP about the expected inputs and outputs. The server runs using standard input/output (stdio) for communication when executed directly. This demonstrates what is MCP server in a basic setup.

Run the server: Open your terminal and navigate to the directory containing mcp_server.py. Then run:

```
python mcp_server.py
```

The server should start without any warnings. This server will keep on running for the client to access the tools

For more information, please visit this article

Advanced Generative AI Langchain

How to Create an MCP Client Server Using LangChain

Learn how to create an MCP client server using LangChain for improved AI interactions with external tools and data.

Harsh Mishra 19 Apr, 2025