Building Multi-Cloud Platform (MCP) Servers from Scratch

Presented by: Mohammad Arshad

Workshop Theme: Hands-On with LangChain, LangGraph & MCP Adapters

Key Libraries: langchain-mcp-adapters, langgraph, fast-mcp

What Are MCP Servers?

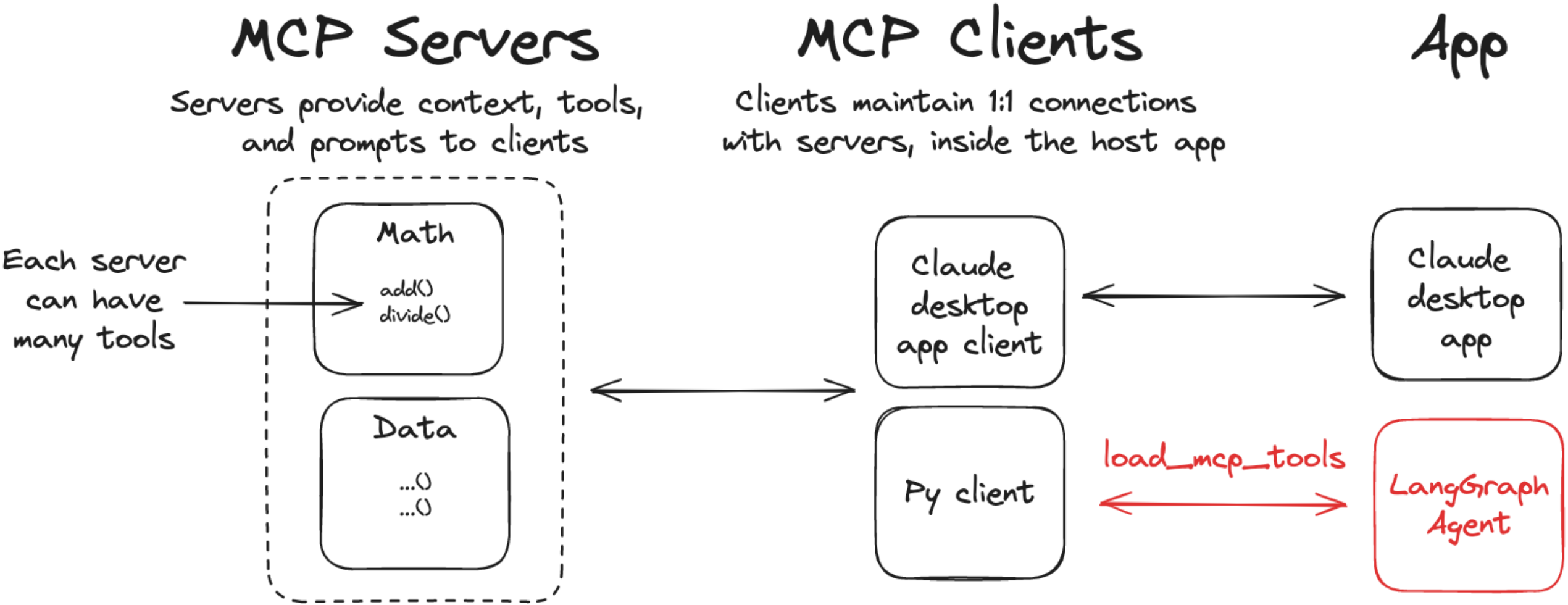
Definition:

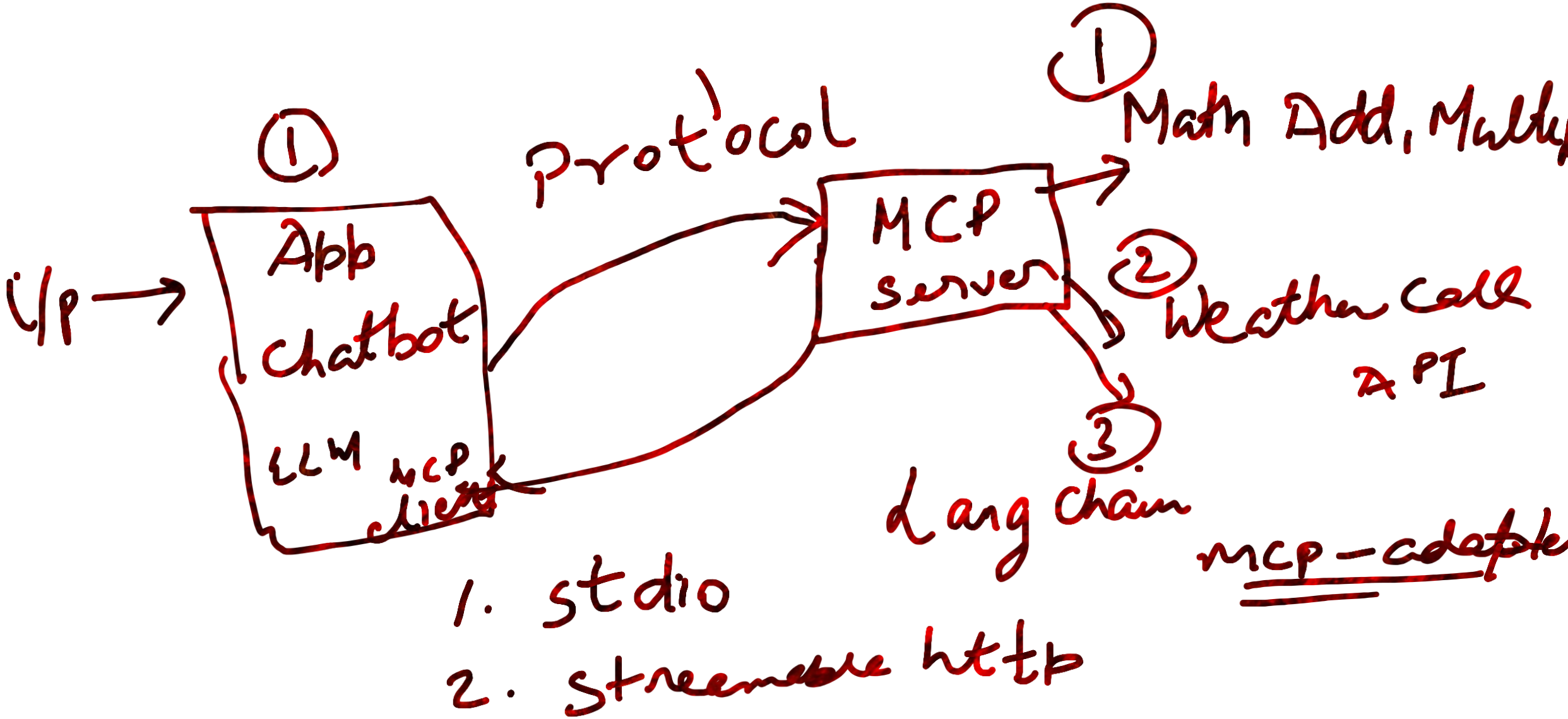
Multi-Cloud Platform (MCP) servers are modular components designed to extend the reasoning abilities of LLMs by offering specialized tools, prompts, and contextual information.

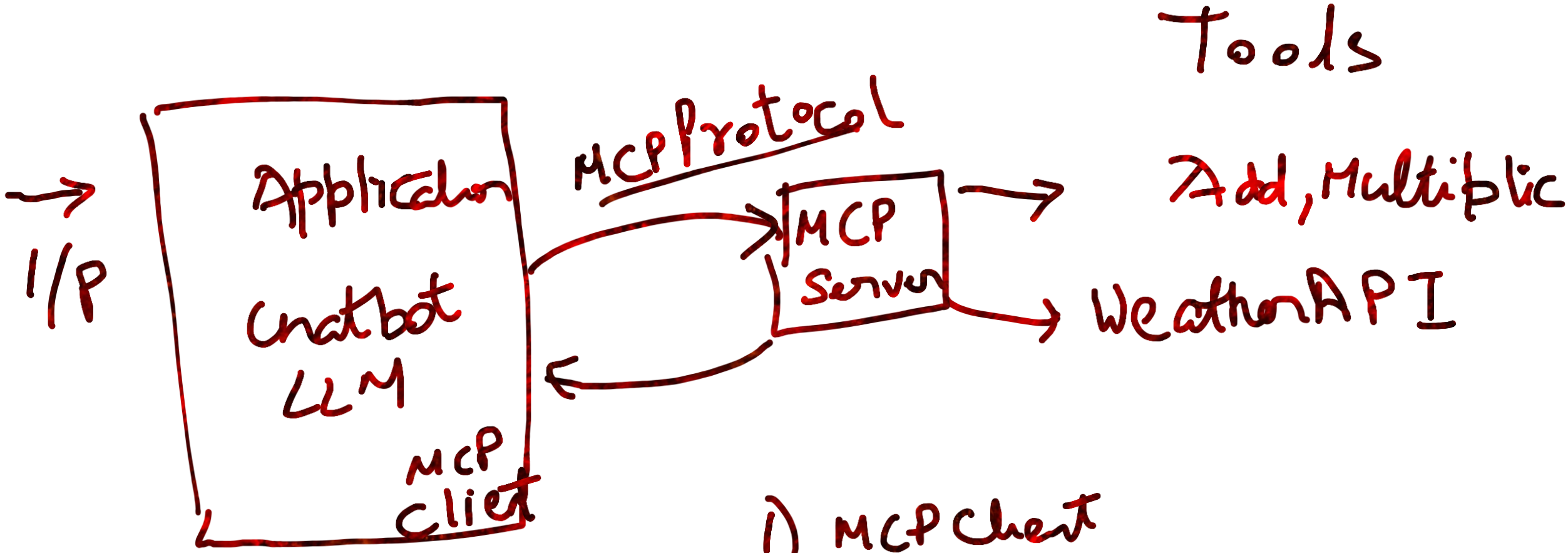
Key Capabilities:

- Perform domain-specific tasks (e.g., math, API calls, database queries).
- Provide a standardized interface for tools accessible to LLMs.
- Enable contextual and tool-augmented reasoning.

Analogy: Think of MCP Servers as specialized agents that LLMs can "outsource" tasks to.


MCP Servers
Servers provide context, tools, and prompts to clients
MCP Clients
Clients maintain 1:1 connections with servers, inside the host app
App
Each server can have many tools
Math
add()
divide()
Data
...()
...()
Claude desktop app client
Py client
Claude desktop app
load_mcp_tools
LangGraph Agent


1
Protocol
1
Math Add, Multy
I/p
App
Chatbot
LLM
MCP client
MCP server
2
Weather call API
3
Lang Chain
mcp-adapte
1. stdio
2. Streamable http



Transport protocol

1) stdio
2) http

1) MCP Client
2) MCP server

Lang chain ecosyst

The MCP Ecosystem

Components & Flow:

- User Interface / Application: Chatbot, Web UI, IDE plugin
- LLM (e.g., ChatGroq, GPT-4): Core reasoning engine
- MCP Client: MultiServerMcpClient coordinates server communication
- MCP Servers: Execute specific tasks via tools

Getting Started: Building Your First MCP Server

Required Libraries:
- langchain-mcp-adapters
- mcp
- langgraph

Setup Steps:
1. Initialize workspace: uv init
2. Create virtual environment: uv venv
3. Install dependencies: uv pip install -r requirements.txt
4. Create your server logic (e.g., math_server.py)

Choosing Your Communication Protocol

1. STDIO (Standard I/O)

- Reads input from stdin, writes output to stdout
- Best for local development and prototyping
- Simple to debug and lightweight

2. Streamable HTTP

- MCP server acts as an HTTP API (e.g., /mcp endpoint)
- Supports structured, real-time responses
- Best for scalable and production-ready deployments

The MCP Client: Your Central Hub

Tool: MultiServerMcpClient (from langchain_mcp_adapters.client)

What it does:
- Connects to multiple MCP servers
- Assigns aliases and routes tasks accordingly
- Enables seamless tool usage inside LangGraph agent chains

Example Config:

```
MultiServerMcpClient(
  servers={
    "math": {"cmd": "python math_server.py", "transport": "stdio"},
    "weather": {"cmd": "python weather_server.py", "transport": "http"}
  }
)
```

Live Demonstration: MCP in Action!

- Demo 1: Math Server (STDIO)
- - Input: "What is 25 * 8?"
- - Output from MCP server returned via stdout

- Demo 2: Weather Server (HTTP)
- - Input: "What's the weather in Dubai?"
- - Weather server responds via HTTP route

- Each response is passed back to the LLM and shown to the user.

Key Takeaways & Future Possibilities

- MCP enables modular, flexible agent design
- Use STDIO for dev, HTTP for prod
- LangChain + LangGraph allow scalable agentic workflows

What's Next:
- Build multi-agent chains
- Integrate with enterprise APIs or DBs
- Deploy on cloud using Docker + FastAPI

Q&A / Resources

Resources:
- LangChain Docs: https://docs.langchain.com/
- LangGraph Docs: https://docs.langgraph.dev/
- MCP Adapters: https://github.com/langchain-ai/langchain-mcp-adapters
- FastMCP Server: https://github.com/langchain-ai/fast-mcp
- Workshop Code Repo: [Insert GitHub link]

Thank You!
Open for questions.