Create and host a Streamable HTTP MCP Server in a virtual machine

Introduction

In this lesson, we tackled how to create and host a **Streamable HTTP MCP Server** using a virtual machine. Here's a step-by-step summary of what we covered.

Setting Up

- **Create a Virtual Environment**: We started by setting up a new virtual environment, ensuring everything is isolated and ready for our MCP server.
- **Activate the Environment**: We made sure to activate our newly created environment, which is a routine step but important for consistency.
- **Install MCP Library**: We added the `MCP CLI`, crucial for running our server.

Coding the MCP Server

- **Create `server.py`**: In this step, we created a simple Python file named `server.py` and pasted basic code to initialize our MCP server.
- **FastMCP Initialization**: The server initializes a **FastMCP** instance. This will interact using functions—like one that returns a personalized greeting.

Transport Mechanism

- **Understanding Transports**: We explored how to specify different transport mechanisms, such as `STDIO`, `SSE`, or `streamable HTTP`. The default is `STDIO`.
- **Switch to Streamable HTTP**: We changed the transport argument to `streamable HTTP` to enable network communication over HTTP streams.

Testing the Server

- **Initial Testing**: Initially, we ran the server using `MCP dev server.py`, noting that it defaults to `STDIO`.
- **Server Execution**: To test our `streamable HTTP` setup, we had to deploy the server using a command like `UV run server.py`.
- **Inspect and Connect**: Using a new terminal, we connected to our server via a specific IP and port, making sure to access it via `/MCP`.

Deployment Possibilities

- **Local to Web Transition**: While the server runs locally, we discussed moving it to the cloud. This makes it accessible globally via platforms like Azure, AWS, or Cloudflare.

Customization

- **Modify Defaults**: We can customize the default host settings (localhost and port 8000). Adjusting these settings aligns with deployment needs.

Client Compatibility

- **Client Access Issues**: We noted that some MCP clients, including **CLAWD**, may not support the `streamable HTTP` protocol yet. We hinted at workarounds to be discussed in the next video.

This lesson outlined configuring and deploying a streamable HTTP MCP server. Now, we're set to explore client connections and hosting solutions to ensure broad access and functionality.