Connect to MCP Server Streamable HTTP with MCP Client

Introduction

In this lesson, we explored how to connect an **MCP client** to an **MCP server** using the **streamable HTTP protocol**. We applied two different methods using distinct clients: **Anthropic Cloud** and **VS Code GitHub Copilot**. Here's what we did:

Using Anthropic Cloud with a Workaround

- **Setup**: Configured `cloud desktop config.json` to ensure the correct **IP address** and proper URL formatting with a **forward slash**.
- **Connection Test**: Successfully connected the client to the server, proving remote operation by monitoring server responses in a terminal session via SSH.
- **Execution**: Sent a greeting to "Henry," allowing the tool to run and observing the server's successful response.
- **Benefits**:
  - Avoided local execution, leveraging cloud-hosted capabilities.
  - Ensured automatic updates for clients without local redeployment.

Key Point

- The workaround was necessary for Anthropic due to the lack of native streamable HTTP support, while STDIO remains an alternative option.

Using VS Code GitHub Copilot

- **Integration**: Installed and configured VS Code GitHub Copilot to recognize MCP servers through **streamable HTTP**.
- **Tool Management**:
  - Added new tools, selecting HTTP as the communication method.
  - Configured `MCP.JSON` for seamless server connections.
- **Testing**: Verified connectivity by executing a greeting command to "Henry" and checked for successful server interaction.

Observations

- This method is more streamlined with native support, allowing full utilization of streamable HTTP without workarounds.

Discussion and Insights

- **Advantages of Streamable HTTP**:
  - Facilitates cloud-based applications without the need for local servers.
  - Enables seamless and scalable server connections.
- **Current Limitations**:
  - Limited client support for streamable HTTP, though this is expected to grow.
  - Essential for web-based applications that can't rely on local servers.

- **Technical Note**: Inclusion of a **forward slash** in URLs is crucial to ensure correct processing of requests. Missing it can cause sporadic issues.

Conclusion

This lesson highlighted the burgeoning potential of **streamable HTTP** in MCP servers while addressing present challenges such as client support and technical quirks. As more platforms adopt this protocol, it promises to simplify cloud-based client-server interactions vastly. Let's stay ahead by mastering these implementations!