Remote MCP Servers

Remote MCP Servers (with Fast MCP)

Agenda of the Video

1. Build a simple remote MCP server (basic functions: add numbers, generate random numbers).
2. Replace the simple code with **Expense Tracker MCP server code**.
3. Deploy the server using **Fast MCP Cloud**.
4. Identify issues and fix them.

1. Local vs Remote MCP Servers

- **Local MCP Server**
  - Runs on the same machine as host and client.
  - Faster (communication stays on the same machine).
- **Remote MCP Server**
  - Runs on a different machine (usually over the internet).
  - Advantages:
    - Supports multiple clients simultaneously.
    - Runs on powerful machines → can handle compute-intensive tasks.
  - Disadvantages:
    - Slower than local servers (network latency).
  - In enterprises, most MCP servers are **remote**.

---

2. Steps to Create a Remote MCP Server

1. **Install dependencies**
    - `pip install uv`
    - `uv add mcp`
2. **Setup project**
    - Create a folder → open in VS Code.
    - Initialize with `uv init .`
3. **Write server code** (main.py)
    - Tools: add numbers, generate random numbers.
    - Key difference from local server:
        - Local → `mcp.run()` (default: stdio transport).
        - Remote → use `transport = HTTP` with host `0.0.0.0` and a defined port.
4. **Run server**
    - `fastmcp run main.py --transport http --host 0.0.0.0 --port <number>`
5. **Debug with MCP Inspector**
    - `uv run fastmcp dev main.py`
    - Select transport: `streamable http`.
    - Check tools & resources.

---

3. Deployment with Fast MCP Cloud

1. Push code to GitHub.
    - `git init`, `git add.`, `git commit -m "initial commit"`.
    - Add remote & push: `git remote add origin <url>` → `git push origin main`.
2. Go to fastmcp.cloud.
    - Connect GitHub account.
    - Select repository → Deploy.
    - Set entry point (`main.py`).
    - Optional: change server name, discoverability.
3. Once deployed → server URL generated → shareable.

---

4. Using Remote MCP Server with Claude Desktop

- Go to **Settings** → **Connectors** → **Add Custom Connector**.
- Provide server name + URL.
- Works only with **Pro plan** currently.
- Free plan users: workaround (explained later in video).

---

5. Converting Expense Tracker into Remote MCP

- Copy Expense Tracker code (from last video) into main.py.

- Add supporting `categories.json` file.
- Run locally → test with MCP Inspector.
- If working → push to GitHub → deploy with Fast MCP Cloud (same process as above).

Key Takeaways

- Remote MCP servers = necessary for enterprise setups.
- Minimal code difference from local MCP servers (main change = transport → HTTP).
- Fast MCP Cloud simplifies deployment.
- Once deployed, servers can be shared globally via a URL.