20250913b FastMCP Quickstart

Welcome! This guide will help you quickly set up FastMCP, run your first MCP server, and deploy a server to FastMCP Cloud.If you haven' t already installed FastMCP, follow the [installation instructions]().

Create a FastMCP Server

A FastMCP server is a collection of tools, resources, and other MCP components. To create a server, start by instantiating the `FastMCP` class.Create a new file called `my_server.py` and add the following code:

my_server.py

```
from fastmcp import FastMCP

mcp = FastMCP("My MCP Server")
```

That' s it! You' ve created a FastMCP server, albeit a very boring one. Let' s add a tool to make it more interesting.

Add a Tool

To add a tool that returns a simple greeting, write a function and decorate it with `@mcp.tool` to register it with the server:

my_server.py

```
from fastmcp import FastMCP

mcp = FastMCP("My MCP Server")

@mcp.tool

def greet(name: str) -> str:

    return f"Hello, {name}!"
```

Run the Server

The simplest way to run your FastMCP server is to call its `run()` method. You can choose between different transports, like `stdio` for local servers, or `http` for remote access:

```
from fastmcp import FastMCP

mcp = FastMCP("My MCP Server")

@mcp.tool

def greet(name: str) -> str:

    return f"Hello, {name}!"

if __name__ == "__main__":

    mcp.run()
```

This lets us run the server with `python my_server.py`. The stdio transport is the traditional way to connect MCP servers to clients, while the HTTP transport enables remote connections.

Using the FastMCP CLI

You can also use the `fastmcp run` command to start your server. Note that the FastMCP CLI **does not** execute the `__main__` block of your server file. Instead, it imports your server object and runs it with whatever transport and options you provide.For example, to run this server with the default stdio transport (no matter how you called `mcp.run()`), you can use the following command:

```
fastmcp run my_server.py:mcp
```

To run this server with the HTTP transport, you can use the following command:

```
fastmcp run my_server.py:mcp --transport http --port 8000
```

Call Your Server

Once your server is running with HTTP transport, you can connect to it with a FastMCP client or any LLM client that supports the MCP protocol:

my_client.py

```
import asyncio

from fastmcp import Client

client = Client("http://localhost:8000")

async def call_tool(name: str):

    async with client:

        result = await client.call_tool("greet", {"name": name})

        print(result)

asyncio.run(call_tool("Ford"))
```

Note that:

- FastMCP clients are asynchronous, so we need to use `asyncio.run` to run the client
- We must enter a client context (`async with client:`) before using the client
- You can make multiple client calls within the same context

Deploy to FastMCP Cloud

FastMCP Cloud is a hosting service run by the FastMCP team at Prefect. It is optimized to deploy authenticated FastMCP servers as quickly as possible, giving you a secure URL that you can plug into any LLM client.

Please note that FastMCP Cloud is a commercial service, though it is completely free for most personal servers.

To deploy your server, you’ ll need a GitHub account. Once you have one, you can deploy your server in three steps:

1. Push your `my_server.py` file to a GitHub repository
2. Sign in to FastMCP Cloud with your GitHub account
3. Create a new project from your repository and enter `my_server.py:mcp` as the server entrypoint That’ s it! FastMCP Cloud will build and deploy your server, making it available at a URL like `https://your-project.fastmcp.app/mcp`. You can chat with it to test its functionality, or connect to it from any LLM client that supports the MCP protocol.For more details, see the FastMCP Cloud guide.

Installation Overview