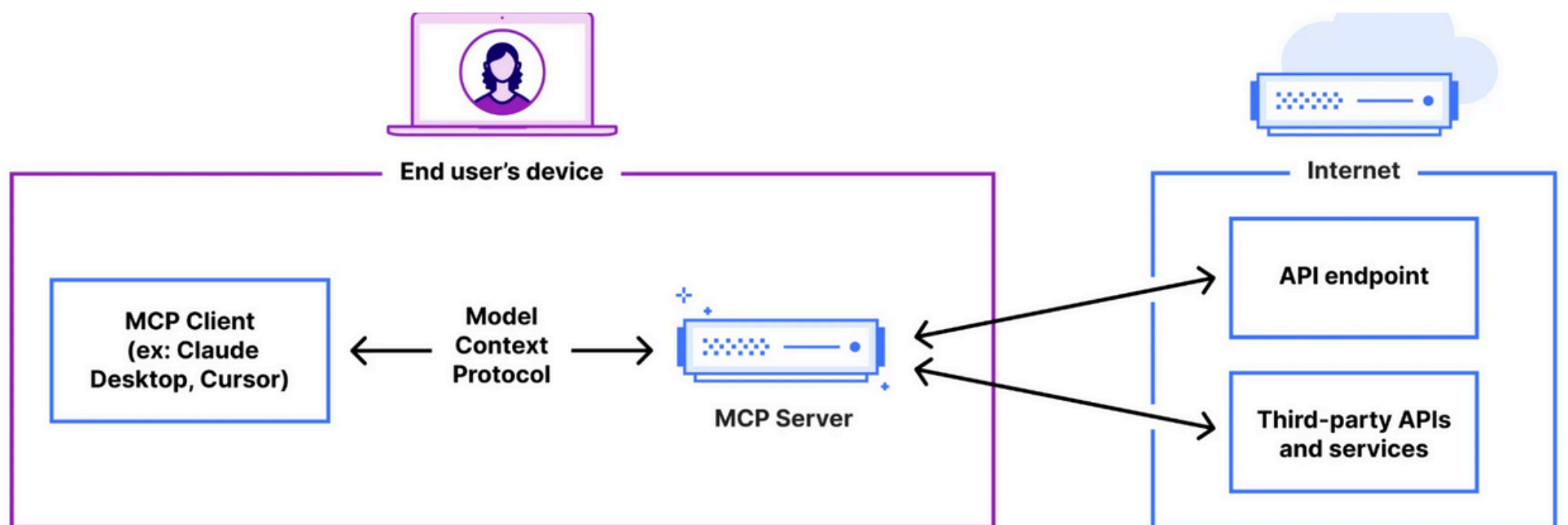
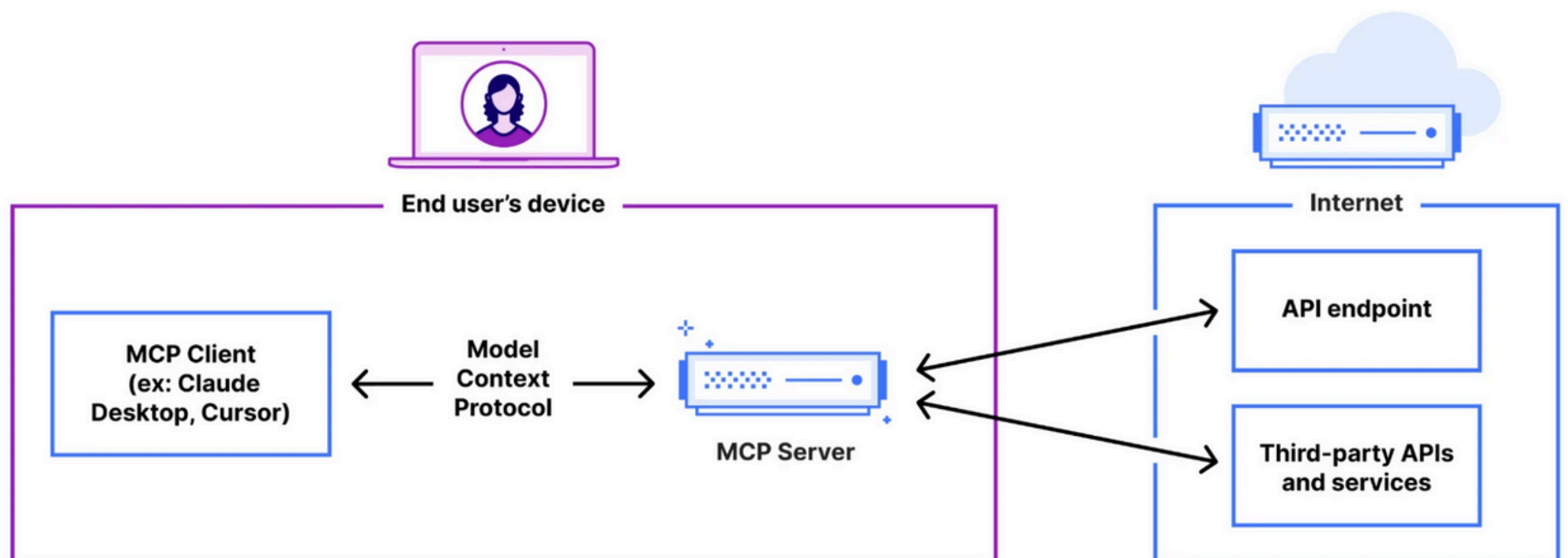


# Convert Any FastAPI App into MCP Server using FastAPI-MCP



# FastAPI with MCP

FastAPI-MCP is a tool which lets you convert any FastAPI application into some tool that LLMs like **ChatGPT** or **Claude** can understand and use easily. By using FastAPI-MCP you can wrap your FastAPI endpoints in such a way that they will become a plug and play tool in an **AI ecosystem** utilizing LLMs.



With FastAPI-MCP, any FastAPI endpoint can be converted into a MCP tool for LLMs. These endpoints should include:

- **GET endpoints:** Converted into MCP resources.
- **POST, PUT, DELETE endpoints:** Converted into MCP tools.
- **Custom utility functions:** Can be added as additional MCP tools.

# Hands-on Application

---

This walkthrough shows how to create a FastAPI app, turn it into an MCP server using fastapi-mcp, and connect it to the Cursor IDE for testing. Here's a simplified breakdown of what's happening:

## 1. Setting Up FastAPI + MCP

- **Install Dependencies:** You install the necessary packages: `pip install fastapi fastapi_mcp uvicorn mcp-proxy`
- **Create main.py:** Import FastAPI, httpx, and fastapi\_mcp.

## 2. Build a Weather API

You define a FastAPI app (Weather Updates API) that:

- Accepts a city and **state code**.
- Looks up **latitude** and **longitude** from a predefined dictionary.
- Calls the **National Weather Service API** to get today's forecast.
- Returns **temperature**, **date**, and **forecast**.

### 3. Convert FastAPI to MCP

```
mcp = FastApiMCP(app, name="Weather Updates API", description="API for retrieving today's weather from weather.gov")
mcp.mount()
```

This wraps your FastAPI app into an MCP-compliant server, making your endpoints accessible to AI tools via MCP.

### 4. Run the Server

- **Run the app with:** `python main.py`
- It starts your FastAPI/MCP server at `http://127.0.0.1:8000`.

### 5. Connect MCP to Cursor IDE

- You configure Cursor IDE to use your local MCP server by editing the `mcp.json`:

```
{
  "mcpServers": {
    "National Park Service": {
      "command": "mcp-proxy",
      "args": ["http://127.0.0.1:8000/mcp"]
    }
  }
}
```

- Cursor now recognizes your MCP server and can route LLM prompts to it.

# Test the Integration

This MCP server can retrieve the weather updates. We just have to ask the Cursor IDE for the weather update on any location, and it will fetch that for us using the MCP server.

**Example 1:** *“Please tell me what is today’s weather in San Diego”*

Please tell me what is todays weather in San Diego

✓ Called MCP tool `get_weather_weather_get` ✓

Parameters:

```
{
  "stateCode": "CA",
  "city": "San Diego"
}
```

Result:

```
{
  "city": "San Diego",
```

Today's weather in San Diego, CA:

- Temperature: 55°F
- Forecast: Patchy drizzle, mostly cloudy, with a low around 55°F. South wind around 5 mph.

Let me know if you need more details or a forecast for another day!



## Example 2: “*New York weather?*”

New York weather?

✓ Called MCP tool `get_weather_weather_get` ✓

Parameters:

```
{  
  "stateCode": "NY",  
  "city": "New York"  
}
```

Result:

```
{  
  "city": "New York",  
  "state": "NY",
```

Today's weather in New York, NY:

- Temperature: 57°F (with temperatures rising to around 64°F overnight)
- Forecast: Patchy fog, showers, and thunderstorms. Cloudy. Southeast wind around 10 mph. Chance of precipitation is 80%. New rainfall amounts between a tenth and quarter of an inch possible.

Let me know if you need more details or a forecast for another city!

The outputs confirm that our MCP server is functioning correctly. Simply asking for weather details prompts the system to decide whether to use the MCP server. Even with a vague query like “New York weather?”, it understood the context from earlier prompts and automatically used the right MCP tool to respond accurately.