**What is MCP?**

MCP (Model Context Protocol) is a system that allows AI to interact with external tools and fetch information.

MCP servers can:

✅ Store data (like files or API responses)

✅ Run tools (functions that AI can execute)

✅ Use prompts (predefined templates for tasks)

**Installing Claude for Desktop**

First, install Claude for Desktop from Claude’s website. It allows us to test MCP integrations easily.

**Setting Up Python and the Right Tools**

We need Python 3.10 or higher and a tool called uv, which is a fast package manager for Python.

**Step 1: Install uv**

powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"  
After installation, restart your terminal so uv is recognized.

uv –version

**Step 2: Create the MCP Directory Structure**

We’ll organize our project with the following structure:

# Stores all MCP servers

mkdir -p ~/mcp/servers/terminal\_server

Here mcp is the root directory and inside this their servers and might make multiple servers and the current server is terminal\_server.

# Create a Workspace Directory

mkdir -p ~/mcp/workspace

In workspace directory all the work that claude desktop app does for us using our MCP server is done inside this workspace directory

# change to our terminal server directory

cd ~/mcp/servers/terminal\_server

Change to our terminal server directory So just to reiterate mcp will save everything related to mcp including our servers which will be inside the servers directory and there will be a dedicated workspace inside the workspace directory over here each server will have its own folder.

**Step 3: Set Up a Python Project inside terminal server**

uv init

**Step 4: Set Up and Activate the Virtual Environment**

uv venv

.venv\Scripts\activate # Windows

**Step 5: Install Required Packages**

uv add "mcp[cli]"

This installs the MCP package, which allows our server to communicate with Claude.

**Building the MCP Server That Executes Terminal Commands**

**Step 1: Create the Server File**

touch terminal\_server.py

**Step 2: Import the Necessary Code**

import os

import subprocess

from mcp.server.fastmcp import FastMCP

mcp = FastMCP("terminal")

DEFAULT\_WORKSPACE = os.path.expanduser("~/mcp/workspace")

**Step 3: Define the Function to Run Commands**

@mcp.tool()

def run\_command(command: str):

"""

Run a terminal command inside the workspace directory.

"""

try:

result = subprocess.run(command, shell=True, cwd=DEFAULT\_WORKSPACE, capture\_output=True, text=True)

return result.stdout or result.stderr

except Exception as e:

return str(e)

This function takes a command, runs it inside ~/mcp/workspace/, and returns the output.

**Step 4: Start the Server**

if *\_name\_* == "\_\_main\_\_":

mcp.run(transport='stdio')

Run this command to start your server:

uv run terminal\_server.py

📌 Connecting to Claude for Desktop

Now, let’s tell Claude how to use our server. Open this file:

code ~/Library/Application\ Support/Claude/claude\_desktop\_config.json

Add this configuration inside the file:

{

"mcpServers": {

"terminal": {

"command": "/Users/theailanguage/.local/bin/uv",

"args": [

"--directory", "/Users/theailanguage/mcp/servers/terminal\_server",

"run", "terminal\_server.py"

]

}

}

}

Save the file and restart Claude for Desktop. You should see a hammer icon, which means your tool is ready.

📌 Testing the MCP Server

Let’s test it out! In Claude, ask:

✅ Run the command ls in my workspace.

✅ Execute echo Hello World.

Claude will send the command to our server, which will execute it and return the response.

And that’s it! We built an MCP server that can execute terminal commands and connect it to Claude. Now, it can run real commands on your machine!