

MCP FOR THE HOMELAB



# MCP

FOR THE HOMELAB

Teach your AI assistant to actually run your lab

AXEL VERNE

FREE SAMPLE – INTRO + CHAPTERS 1 & 4

# Contents

*What's in this sample — and why these two chapters*

|   |                                                                |                  |
|---|----------------------------------------------------------------|------------------|
| — | Introduction: Your Lab Is Smart. Your AI Is Locked Outside.    | included         |
| 1 | Why Your Lab Needs MCP (And Why It Secretly Already Wanted It) | included         |
| 4 | Your First MCP Server, Tonight                                 | included         |
| 2 | The Bookmark Cemetery: A Theory of Homelab Pain                | in the full book |
| 3 | MCP Without the PowerPoint                                     | in the full book |
| 5 | The Homelab Shelf: Servers Worth Stealing                      | in the full book |
| 6 | Roll Your Own Front Desk                                       | in the full book |
| 7 | The Paranoid Chapter (Read Twice)                              | in the full book |
| 8 | A Day in the Lab, With Hands                                   | in the full book |
| 9 | When It Breaks: A Field Guide                                  | in the full book |
| + | Recipe Appendix · Five Weekend Projects                        | in the full book |

**Why chapter 4 is in the free sample:** because a sample should *work*. Chapter 4 is a complete, runnable MCP setup on hardware you already own — do it tonight, no purchase required. The full book adds the shelf of ready-made servers, the security chapter, the troubleshooting field guide, and ten recipe stacks. [Get the full book →](#)

# Your Lab Is Smart. Your AI Is Locked Outside.

Let's be honest about the relationship you have with your AI assistant right now.

You ask it to explain a Kubernetes error. It does, beautifully. You ask it to write a `systemd` unit. Done, better than yours. You ask it to actually *restart the container that's throwing the error* — and it hands you a nicely formatted command you get to walk over to a terminal and paste yourself, like some kind of digital errand boy.

You have a brilliant advisor locked in a closet. It can see everything through the door crack and touch nothing.

Meanwhile your lab — that lovingly assembled rack of Jadis-grade hobby computing — is a small city of services that all speak different dialects: a NAS with a web UI, a hypervisor with an API you swear you'll learn, eleven *arr* apps that each have their own config format, and a router whose interface requires a human sacrifice to change a port forwarding rule.

**MCP is the door.** It's an open standard (Anthropic, November 2024, open-sourced and subsequently adopted by every major AI platform) that lets your AI assistant call tools directly — your hypervisor, your Home Assistant, your download client, your DNS. Not "tell you what to type." *Do it*, with your approval where it matters.

Here's the deal I'm offering: **by the end of Chapter 4 — tonight, realistically — you will have an AI assistant that can list your Proxmox VMs, check your Docker containers, and toggle a Home Assistant light.**

And because you're exactly the kind of person who'll skip the safety chapter — it's Chapter 7, it's short, and it's the difference between "my assistant has hands" and "a PDF I downloaded has hands." Read it twice. Then yolo responsibly.

*Fair warning about my credentials: everything in this book worked at least once on my lab. Several things also failed spectacularly first, and those stories are included at no extra charge.*

# Why Your Lab Needs MCP (And Why It Secretly Already Wanted It)

## The confession that opens every homelab conversation

I have, at current count, **seven** interfaces I log into regularly to manage services that are, in one meaningful sense, *right there*. I have a bookmark folder called "lab" that scrolls for three screens. I have a note in my phone titled "docker commands I always forget" that is just `docker compose logs -f` with different flags.

And I have an AI assistant that can explain any of those seven interfaces in perfect detail and operate none of them.

If your lab has more than four services, you already know the shape of this problem. It's not that automation is impossible — you've automated plenty. It's that automation is *bespoke*. Every script you've written works exactly one service, breaks on the second update, and lives in a git repo named something like `lab-scripts-2-FINAL`.

## What you've already built (and why it isn't enough)

The homelab automation stack, as of this writing, usually looks like:

- **A cron job graveyard.** Each entry made sense once. Collectively they're a haunted forest.
- **Home Assistant automations** — genuinely great, and completely siloed inside Home Assistant.
- **N8N / Node-RED flows** — the "I'll wire it visually" era. Powerful, and the debugging experience of a Rube Goldberg device.
- **Bash scripts with curl and prayer.** The raw API calls are documented *somewhere*, probably.

Notice what all of these have in common: **each one is a point solution between you and one service.** The API glue exists, but the glue is the product. Nobody's script collection ever became a general-purpose way to *ask the lab for things*.

And none of it is *conversational*. You can't ask any of it a question. You can't say "why is Plex buffering?" — you can only run the three commands that would reveal the answer, one at a time, in three different UIs, while the family waits.

## What MCP changes (the honest version)

MCP doesn't replace your scripts. It replaces the *ad hoc* nature of them.

When your hypervisor, Docker host, Home Assistant, and download client each expose an MCP server — the "front desk" pattern from its sibling book, minus the hotel analogy, you get it — your assistant can discover their capabilities automatically and call them like functions. One standard, N services, zero bespoke glue per tool.

The three concrete changes:

1. **Discovery beats documentation.** The assistant reads each server's menu of tools. You stop being the integration layer between your own services.
2. **Conversation beats UI.** "Which containers restarted in the last 24 hours and which ones died more than twice?" becomes one question instead of a terminal session.
3. **Approval where it matters.** Read operations flow freely; anything destructive waits for your tap. Your assistant gets hands, not unrestricted sudo.

And the ecosystem math is the same as the enterprise side, just with more acrylic and RGB: MCP launched in late 2024, every major AI platform adopted it through 2025, and by mid-2026 the protocol's SDKs were pulling close to half a billion downloads a month. This is not an experimental hobby protocol. It's the standard the hobby is standardizing on.

## The objection you're already typing

*"I already have this. I SSH in and I have an agent that runs shell commands."*

Sure — and shell agents are how we all got our first taste of "AI with hands." I ran one for a month. It was fantastic right up until it wasn't: an innocuous cleanup request produced a `rm -rf` argument that was *technically* correct if you squinted, and I learned about backups with my heart instead of my schedule.

Here's the actual difference, and it's the thesis of this book:

***A shell agent has one tool: everything. An MCP setup has many tools: each one exactly as big as it should be.***

With MCP, restarting Plex is a tool called `restart_container` scoped to containers. Deleting a VM is *not on that menu*. The assistant can't fat-finger what isn't exposed. That's the whole security model, and it's why Chapter 7 is short rather than nonexistent.

# Your First MCP Server, Tonight

**The promise, restated with a stopwatch:** twenty minutes from now you will ask your AI assistant "what's in my media folder?" and it will *know*. Thirty minutes from now it will check your Docker containers on request. Tonight. On hardware you already own.

We're going to stand up two servers — filesystem (read-only) and Docker — and wire them into one client. This is the pattern for everything in the rest of the book: **pick a server, hand it scoped credentials, add a config stanza, ask a question**. Master this chapter and you've mastered the whole book's mechanics.

## Step 0 — Pick your client (2 minutes)

Claude Desktop / Claude Code is the most mature MCP client; several open clients (Cherry Studio et al.) work if you want local models in the mix. The config format below is identical across all of them. Pick whatever's installed; client religion is out of scope.

## Step 1 — The filesystem server (5 minutes)

```
{
  "mcpServers": {
    "media-files": {
      "command": "docker",
      "args": [
        "run", "--rm", "-i",
        "--network", "none",
        "-v", "/tank/media:/tank/media:ro",
        "mcp/filesystem:latest",
        "/tank/media"
      ]
    }
  }
}
```

Details that matter, in the order you'll be tempted to skip them:

- **:ro** — read-only mount. The server physically cannot modify your files. Start every filesystem server this way.
- **--network none** — this server has no earthly reason to phone home, so it can't.
- **The trailing /tank/media argument** — the directories the server is allowed to see. Everything else stays invisible.

## Step 2 — The Docker server (8 minutes)

```
{
  "mcpServers": {
    "docker": {
      "command": "docker",
      "args": [
        "run", "--rm", "-i",
        "-v", "/var/run/docker.sock:/var/run/docker.sock",
        "-e", "ALLOW_LIST=plex,radarr,sonarr,qbittorrent,homeassistant",
        "docker-mcp:latest"
      ]
    }
  }
}
```

The **ALLOW\_LIST** is the chapter's teaching moment: this server *could* control every container on the host, but we've scoped it to the media stack. If the assistant has a brainstorm about postgres, that conversation goes nowhere, by design.

Restart the client and run the graduation exam:

*"List my containers and tell me which ones have restarted recently."*

Then the fun one: *"What's eating the most CPU right now, and should I care?"* — it'll call `list_containers`, then `stats`, then `synthesize`. That joining is the product.

## Step 3 — The approval reflex

Your client will ask before most tool calls. **Do not turn this off yet.** Reads → auto-approve eventually. Writes → keep the prompt. That's the entire operational philosophy of the book: **scope the menu, keep the click.**

## When it doesn't work

Three failure modes cover 90% of first-night issues: (1) run the exact `docker run` line by hand — it's always the mount path; (2) restart the client — stale cached tool catalogs; (3) your user isn't in the `docker` group. The full field guide is Chapter 9.

### TONIGHT'S HOMEWORK

Add one more read-only server for anything except Docker — the Pi-hole or Home Assistant server from Chapter 5's shelf — and ask it your wishlist question #1. If it answers in one sentence,

THE SETUP WORKS. HERE'S THE REST OF THE MAP.

## Chapter 4 gave you hands. The next six give you judgment.

### In the full book:

- **Ch. 5** — the Homelab Shelf: ready-made servers for Proxmox, HA, \*arr, qBit, Pi-hole, TrueNAS, Grafana — with the 5-step scoping ritual
- **Ch. 6** — roll your own front desk: the complete custom-server pattern, with the write-tool promotion ritual
- **Ch. 7** — the Paranoid Chapter: prompt injection, the socket-is-root rule, and the ten-second fridge version
- **Ch. 8** — a day in the lab, with hands: the "is it worth it" chapter
- **Ch. 9** — when it breaks: the accumulated error catalog and a 5-step recovery drill
- **Appendices** — ten copy-paste recipe stacks and five weekend projects ranked by bragging rights

**\$8.99, DRM-free, no course upsell:**

**Get the full ebook → \$8.99**

**Companion repo — free, MIT, smoke-tested →**

Instant download · PDF + Kindle formats · if chapter 4 didn't work for you, chapter 9 has the fixes, and email gets you a refund.

ONE FAVOR

## Know someone whose bookmark folder also scrolls for three screens?

Hand them this sample. It's free, chapter 4 works, and the companion repo is MIT — there's nothing to upsell them, which is exactly why they'll thank you.

The highest compliment this book can receive is a forwarded PDF and a starred repo. — Axel Verne

[Get the full ebook →](#)

MCP for the Homelab © 2026 Axel Verne · axelverne.com