

Docker Management

When I first started running applications in my homelab, I was managing everything through the command line. Spinning up a container meant crafting a `docker run` command with all the right flags, ports, volumes, environment variables, network bridges, and hoping you remembered every piece of it the next time you needed to touch it. For a container or two, it works fine. But as the number of applications grew, managing everything through CLI became increasingly difficult to keep organized. There was no easy way to see what was running at a glance, restart something quickly, or remember what flags a container had been started with six weeks ago.

That's what led me to **Portainer**.

Portainer

Portainer was a significant quality-of-life improvement. It gave me a visual interface for managing containers across multiple VMs from one place, with the main instance running on Overseer and lightweight agents deployed on the other VMs. I could see running containers, check logs, restart services, and deploy new stacks without touching the command line. For someone still getting comfortable with Docker, having that visual layer made a real difference. The learning curve was manageable, and the interface was intuitive enough to navigate without much friction.

That said, Portainer had its share of limitations. Updates were occasionally painful and required care to avoid breaking the setup. More notably, while it is open source, a significant portion of its more useful features are locked behind a paid Business license. For a homelab running on free and open-source software wherever possible, that always felt like a ceiling I kept bumping into. It worked, but it wasn't the long-term answer.

About a year into the homelab, I came across **Komodo**.

Komodo

Komodo was a different experience from the moment I opened it. The design was clean and modern, the interface was more intuitive, and it handled Docker Compose stacks in a way that finally felt right. Rather than treating stacks as an afterthought, Komodo put them front and center, easy to create, easy to edit, and easy to manage across multiple environments. The feature that really stood out, though, was built-in version control for stacks. Being able to track changes to Compose files, see what changed and when, and roll back if something broke was exactly the kind of workflow I hadn't realized I was missing.

Transitioning from Portainer to Komodo wasn't entirely painless. Docker managers aren't just passive viewers, they actively manage network bridges, container labels, and other configuration details that don't always transfer cleanly. Getting everything migrated across all my VMs and

configured correctly in Komodo took some time and careful attention, but once it was done, the day-to-day experience was noticeably better. Having VM snapshots, was a critical part of this process.

Then, at the beginning of 2026, I found **Dockhand** — and made the switch again.

Dockhand

On the surface, the idea is similar: a clean interface for managing Docker containers and Compose stacks across multiple environments. But there were a few specific things that made Dockhand stand out in practice. Container information, logs, and the built-in terminal are all surfaced much more directly — less clicking to get to what you actually need. The integrated file manager was the detail that pushed me over the edge. Being able to browse and edit files within a container's environment directly from the interface, without SSHing in separately, is a small thing that saves a surprising amount of time.

Around this same period, I also started taking Git more seriously. I'd had Forgejo deployed for a while, but it was mostly sitting idle. Komodo's version control feature had introduced me to the concept, but it was a self-contained system — I wasn't really thinking in terms of a proper Git workflow. When it came time to make the switch to Dockhand, I decided to use the transition as an opportunity to do it properly. I migrated all my Docker Compose stacks over to Forgejo, organized them into repositories, and set up Dockhand to pull from those repos for deployments.

The difference was immediate. Now every stack has a real commit history. Changes are tracked, documented, and reversible. If a deployment goes wrong, rolling back is a matter of seconds rather than trying to remember what the file looked like before you edited it. Dockhand is currently connected across six environments, a mix of VMs, LXC containers, and a cloud server — and deploying or updating a stack anywhere in that environment is fast and straightforward.

The one consistent limitation with both Komodo and Dockhand is resource monitoring. Neither tool reports CPU and RAM usage in a way that lines up consistently with what Proxmox shows at the hypervisor level. It's not a dealbreaker, but it does mean I keep an eye on resource usage from the Proxmox side rather than relying on the Docker manager for that. For everything else though, the current setup, Forgejo for version control, Dockhand for management, is the most capable and organized the homelab has ever been.

Revision #1

Created 2026-05-12 00:50:19 UTC by lumxux

Updated 2026-05-12 00:54:56 UTC by lumxux