**5 things I do to simplify self-hosting - Containerization**

Story by Ayush Pande

From executing CLI commands to modifying long YAML files, hosting services on local hardware can seem like a difficult undertaking if you’ve never worked with containers. But the truth is that you don’t need to be a home server maestro to create an arsenal of self-hosted services.

Even discarding the highly detailed tutorials and guides created by the home lab community, there’s a plethora of tools and services designed to help newcomers navigate their way through the seemingly treacherous self-hosted landscape. As someone who broke free from premium, privacy-intrusive third-party apps, here’s a general outline of how I grew accustomed to self-hosting my own services.

**Start with containerization platforms**

**Plenty of these to choose from**



Accessing the CasaOS web UI from a PC

Confession time: When I first dove into the home server rabbit hole as a teenager, I was terrified of YAML files and Docker scripts. The only experience I had with Linux up until that point was updating packages with the good ol’ *apt update && apt upgrade* commands, and I’d have to open multiple forum posts trying to understand anything beyond *wget* and *curl*.

That’s when I stumbled upon containerization platforms, which are designed to make self-hosting as simple as choosing my favorite apps, modifying a couple of settings from a menu, and pressing the Install button. I started with a CasaOS instance running inside a Debian VM, and I still use an LXC variant of the app (via [Proxmox](https://www.xda-developers.com/proxmox-guide/?utm_source=syndication&pubDate=20250707" \t "_blank)) to this day. However, there are tons of other options worth checking out. If you’re on a Raspberry Pi, I recommend flashing a spare microSD card with [YunoHost](https://www.xda-developers.com/yunohost-guide/?utm_source=syndication&pubDate=20250707" \t "_blank) or [UmbrelOS](https://www.xda-developers.com/turn-your-raspberry-pi-into-a-self-hosting-hub-using-umbrelos/?utm_source=syndication&pubDate=20250707" \t "_blank). The former has a massive set of apps to choose from, though I prefer UmbrelOS’ aesthetically pleasing UI.

**Then move on to Docker/Podman Desktop**

**CLI commands can be challenging in the beginning**



Docker Desktop running on Windows 11

Before I begin this section, let me add that terminal commands are borderline essential if you want to familiarize yourself with the nitty-gritty of self-hosted environments. But for folks who simply wish to deploy services that are missing from their favorite containerization platforms, the GUI variants of Docker or Podman are more than adequate.

Personally, I’m more biased towards [Podman Desktop](https://www.xda-developers.com/podman-desktop-guide/?utm_source=syndication&pubDate=20250707" \t "_blank), as it’s not only compatible with the Docker runtime, but also supports the titular “pods,” which are groups of closely associated containers. Either way, both are fairly easy to set up, regardless of whether you’re on Windows and Linux, and include most of the features you’ll need at the beginning of your self-hosting journey. Plus, these GUI container runtimes can help you get familiar with the basic containerization jargon, which comes in handy when you make the switch to a CLI-only environment.

**Use container management UIs**

**Cockpit and Portainer are my favorites**



Accessing Podman containers via Cockpit

If you’re not satisfied with Docker or Podman Desktop and want a simple web UI that includes all the facilities you’ll ever need to tinker with your self-hosted arsenal, I recommend checking out container management UIs. In fact, I rely on a mixture of [Portainer](https://www.xda-developers.com/portainer-guide/?utm_source=syndication&pubDate=20250707" \t "_blank) and [Cockpit](https://www.xda-developers.com/4-reasons-i-use-cockpit-to-manage-my-podman-containers/?utm_source=syndication&pubDate=20250707) to manage my container suite.

Since most of my non-CasaOS services run via Podman (which, in turn, runs inside virtual machines and SBCs), I prefer using the Cockpit-Podman extension to manage my containers on Cockpit. In addition to simplifying pod management, Cockpit also lets me keep an eye on the host machines. For the rest of my Docker-based services, Portainer works incredibly well. If you’re starting your container journey with Docker, Portainer is a great way to manage your self-hosted empire.

**Update containers via Watchtower**

**The monitor-only label is my best friend**



Deploying a Watchtower server

As much as I love containers, updating them is a tedious process. After pulling the updated image from a repository, I’ll have to stop and destroy the container associated with it. Then, I’ll have to spin up a new container using the updated image and mount the storage volumes associated with the old container to it. While it’s not an extremely complex process, doing so for every container becomes a chore when your library of self-hosted services is as large as mine.

That’s where [Watchtower](https://www.xda-developers.com/watchtower-guide/?utm_source=syndication&pubDate=20250707) comes in handy, as it automates every step of the updation process. Unfortunately, broken images are far from rare in the self-hosting landscape. Since Watchtower has no way to determine whether an image is stable enough for a container, setting up automatic updates can result in the automation app pulling faulty images and breaking containers. Thankfully, Watchtower includes a *monitor-only* label, which informs me about new updates before installing them, and I’ve enabled it for every mission-critical service in my arsenal.

**Deploy a dashboard**

**Homarr combines a sleek UI with an easy setup process**



a laptop showing the homarr website

Once your army of self-hosted containers grows sufficiently large, it can be difficult to keep track of the IP addresses and port numbers of their web UIs. Combining reverse proxy services like Caddy and Nginx with custom domain names is a valid option, but doing so can be quite challenging if you’re not used to creating self-signed certificates or dealing with multiple configuration files.

My favorite fix involves mapping the IP addresses of my VMs, containers, and virtualization platforms. I’ve been using [Homepage](https://www.xda-developers.com/homepage-is-the-best-dashboard-app/?utm_source=syndication&pubDate=20250707) as my dashboard for the last couple of months, though I wouldn’t recommend newcomers to jump straight into this YAML-heavy platform (even though it’s worth the extra hassle). Instead, Homarr has the right balance between functionality, aesthetics, and ease of setup. Adding apps and services to Homarr is a walk in the park, and so is configuring extra integrations for them. Plus, it’s possible to create multiple dashboards inside the same Homarr instance.

**Despite its seemingly complex nature, self-hosting is fairly simple**



The UmbrelOS web UI

As someone who writes about containerized services on a weekly basis, I can confirm that it’s pretty easy to get them up and running. Container platforms trivialize most of the challenge with deploying apps on local hardware, and CasaOS, [Runtipi](https://www.xda-developers.com/runtipi-raspberry-pi-guide/?utm_source=syndication&pubDate=20250707" \t "_blank), Cosmos, and other popular environments include most of the essential services you could ask for. In fact, I’d go so far as to say that if you’re just looking to ditch premium cloud-based services and aren’t all that interested in obscure container images or home servers, containerization platforms can satiate your FOSS self-hosting needs.