Search Engine & Public Access

Search Engine

Around this point in my homelab journey, I had the baseline established and was ready to focus on applications that would bring more meaning to my homelab and actually be of use. Throughout graduate school, a big focus of mine was cybersecurity and privacy, as well as data analytics. Once I gained a better understanding of what data we leave behind every time we search something on Google, it became clear to me that I needed something more private and within my control. While I mostly use Brave browser and DuckDuckGo, I still wanted something I could fully own. That led me to SearXNG.

SearXNG is a free and open-source meta search engine that I discovered through watching homelab videos on YouTube, specifically from NetworkChuck. After watching his video, the setup seemed pretty straightforward — install Docker, run a simple bash command, and you have a containerized application up and running. I already had everything in place: a domain connected through my reverse proxy, notifications set up to monitor the container in case it lost connection or went down, and a virtual machine that was already provisioned and ready to go.

Since this is something I wanted to access remotely and potentially share with friends, I deployed it on my External VM, which was already in the proper VLAN with the appropriate permissions and security controls in place.

While it all looked simple at first, there were a few setbacks. I had to dig through Docker logs to figure out what errors were occurring and how to fix them. One valuable lesson I learned is that every environment is different, just because something works for someone else doesn't mean it will work for you without adjustment. After fixing errors and using ChatGPT for help in understanding the issues, and reading through hundreds of lines of logs and configuration, I finally got it working. SearXNG was live.

It's a simple search engine with no data collection or trackers. The application works by fetching results from multiple pre-defined search engines, creating an anonymous one-time session profile, and returning those results without leaving any cookie crumbs behind — no record of who searched, from which IP address, or anything else that could be traced back to you.

Tunnels

While the traditional way of exposing applications to the internet is through port forwarding, it isn't the safest approach. With security and privacy at the forefront of my homelab philosophy, I wanted to do this safely and securely. To achieve that, I went with a more modern approach: application tunneling.

My domain was already registered with Cloudflare, and their free tier offers a lot of useful features — one of which is Cloudflare Tunnel. This allowed me to connect a forward proxy service running on Cloudflare's infrastructure back to my personal network, exposing my self-hosted application to the internet without any port forwarding required. Nothing on my router needed to be opened, which significantly reduces the attack surface.

To make this work, I had to install a lightweight agent cloudflared, running in its own container. This agent communicates outbound to Cloudflare to establish a secure tunnel and expose whichever application I've configured. Once everything was in place, I followed the steps in the Cloudflare dashboard and set up the appropriate connection. Cloudflare automatically created a CNAME DNS record, which allowed anyone to access my search engine through a public URL. Even though SearXNG was running on a private IP (e.g., 10.0.0.x), requests from the outside world resolve to Cloudflare's public infrastructure, making it much harder for anyone to discover or target my actual server.

On the Cloudflare side, I took every available step to lock things down as tightly as possible. I created access rules for my tunnel applications and restricted incoming traffic to only a handful of countries, specifically, the ones I'd realistically expect legitimate traffic to originate from. This makes the service more resilient to automated attacks and probing. Cloudflare also provides native DDoS protection, keeping the application safe from botnet floods without any additional configuration on my end.

Docker Management

This project was fairly straightforward to get running, but it highlighted several gaps in my initial foundational setup. While Docker containers can be managed entirely through the command line, deploying and maintaining them through a GUI is much more practical for day-to-day operations. This led me to install Portainer.

I set up the main Portainer instance on my Overseer VM, with lightweight agents deployed on both the Internal and External VMs. This gave me a single pane of glass to manage all three environments from one application, without needing to SSH into each machine separately to check on container status or make changes.

Portainer helped me build a deeper understanding of how Docker management works; things like network bridges defined in Compose YAML files, container resource settings, volume mounts, and log inspection. Portainer does have a commercial side and the interface can feel cluttered at first with features you may never use, but with time I was able to identify the parts that mattered and work efficiently within it to maintain my homelab applications.

Revision #1
Created 2026-05-12 00:31:53 UTC by lumxux
Updated 2026-05-12 00:33:43 UTC by lumxux