

Docker home lab

Documentation for creating your own lab based on Docker containers

Hubert Bojda

September 18, 2024

Contents

1	Hardware	3
2	Software	4
2.1	Docker	4
2.2	Tailscale	4
2.3	Containers	8
2.3.1	Portainer	8
2.3.2	File Browser	9
2.3.3	NGINX Proxy Manager	11
2.3.4	Heimdall	17
2.3.5	Nextcloud	18
2.3.6	AdGuardHome	21
2.3.7	RustDesk	25

1 Hardware

The aim of the project is to create your own lab, which can be implemented on any device that supports containerization technology, such as Docker. To start, I suggest creating a virtual machine with a Linux system, for example, Debian or Ubuntu. In the future, you might consider purchasing dedicated hardware, such as a Raspberry Pi or terminals, also known as "thin clients." In my case, I chose the HP T630 terminal, which, compared to the Raspberry Pi, can often be purchased for half the price. This is a significant advantage, although the Raspberry Pi excels in terms of low power consumption due to its ARM-based processor.

This terminal offers many possibilities, especially considering its affordable price. For around 160 PLN, you get a computer with a large amount of RAM and a multi-core processor, allowing you to accomplish a wide range of tasks. An alternative could be the Raspberry Pi, which, although nearly twice as expensive, is characterized by lower energy consumption. The choice depends on individual needs and priorities.

In my case, the terminal consumes about 11 W in idle mode and up to 25 W under maximum load. Assuming an average consumption of 15 W, the cost of continuous operation for a year is approximately 150 PLN. This is a quite reasonable amount, considering the capabilities and versatility of this device, which will serve as an excellent foundation for building your own lab.



Figure 1: HP T630

Hardware	Description
Procesor	AMD GX-420GI, 4 rdzenie
RAM	8GB
System drive	128GB NVMe

2 Software

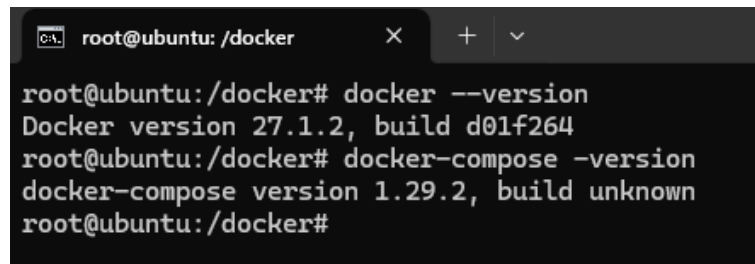
The aforementioned hardware has been installed with Ubuntu Server 24.04 LTS. Communication is carried out through my computer via the SSH service.

2.1 Docker

- Docker was installed according to the documentation provided on the website. <https://docs.docker.com/engine/install/ubuntu> there is a step-by-step description of what we need to do to install Docker.
- Another component that complements Docker's capabilities is Docker Compose. To download the appropriate package, we will use the command line.

```
1 apt update && apt upgrade -y && apt install docker-compose
```

Verifying whether the software has been installed correctly:



```

root@ubuntu: /docker
root@ubuntu:/docker# docker --version
Docker version 27.1.2, build d01f264
root@ubuntu:/docker# docker-compose -version
docker-compose version 1.29.2, build unknown
root@ubuntu:/docker#

```

Figure 2: Docker and Docker Compose

2.2 Tailscale

Tailscale is a tool for creating private VPN networks based on WireGuard. It is easy to configure and enables secure connections between devices without the need for setting up VPN servers or port forwarding. It works on multiple platforms (Windows, macOS, Linux, iOS, Android), creating a mesh network where devices communicate directly with each other. Tailscale is ideal for remote access, securing applications, and creating private networks for teams, offering a high level of security and cloud integration. [Tailscale docs](#)

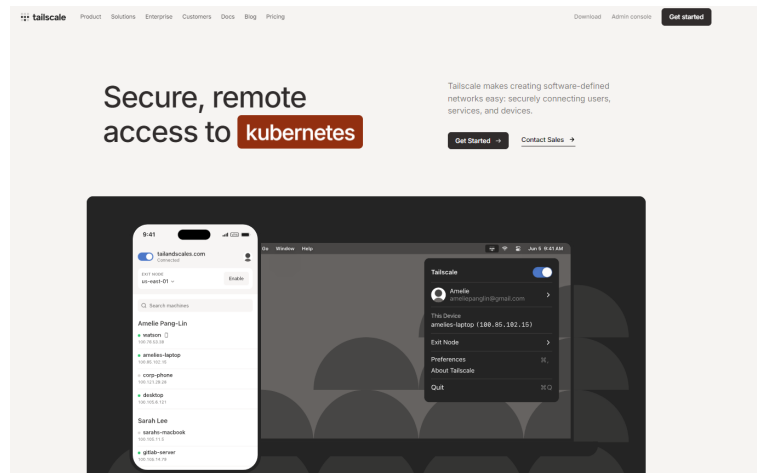


Figure 3: Tailscale

After registering, you will be prompted to add your device to the Tailscale network. Additionally, a client for the service will be installed on your device, from which you can manage other devices on the network or, for example, view the IP address of a particular device.

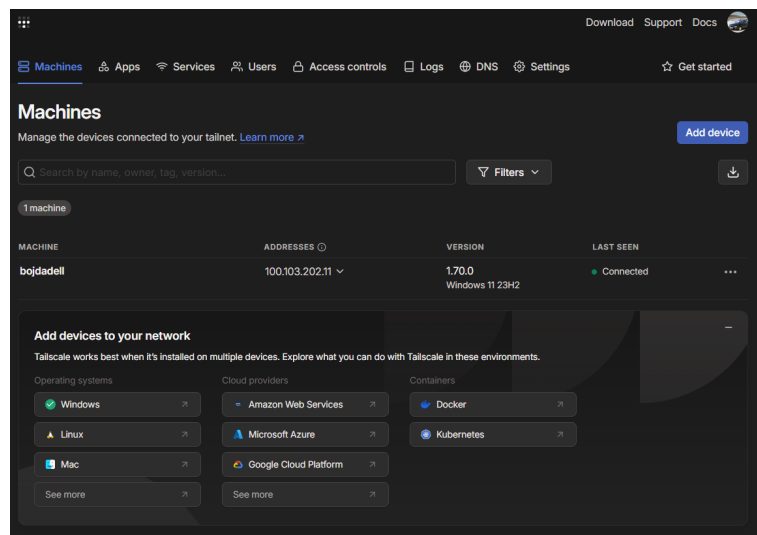


Figure 4: Tailscale Dashboard

To add additional devices to our network, we are provided with a simple guide on how to do so. There is a separate instruction for each system on how to connect to the Tailscale network.

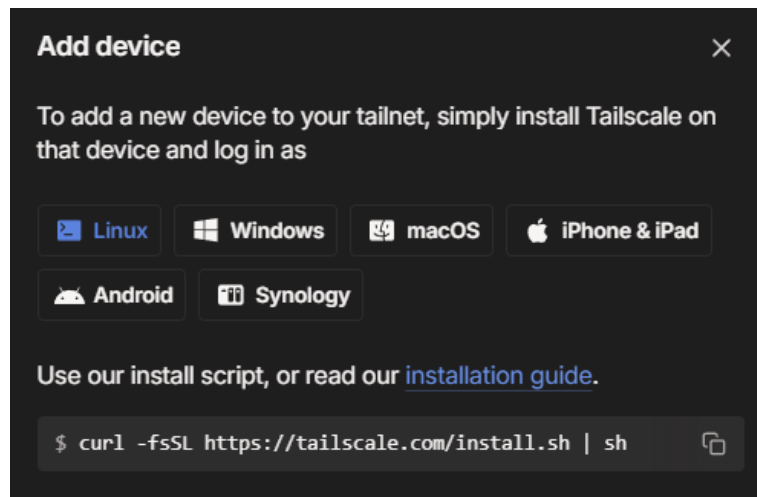


Figure 5: Connecting Linux

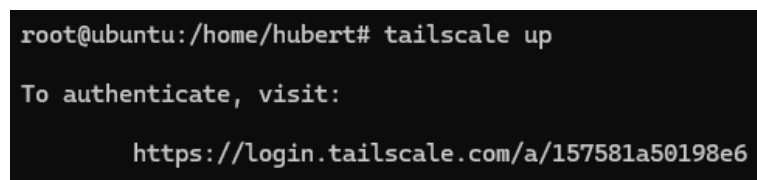


Figure 6: Tailscale Up Linux

Next, we need to log in via the generated link, and after confirmation, our device will be visible on the network. An interesting feature offered by Tailscale is access to devices on

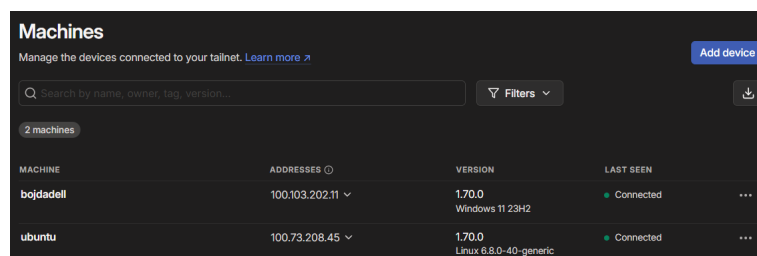


Figure 7: Our devices

our local network. From the administrative panel on the website, we can specify which of our devices will act as a router for our local network that is not directly connected to Tailscale. tailscale.com/kb/1406/quick-guide-subnets

```
1 sudo tailscale up --advertise-routes=adres_sieci/maska_podsieci
```

Next, we need to activate this option from the website:

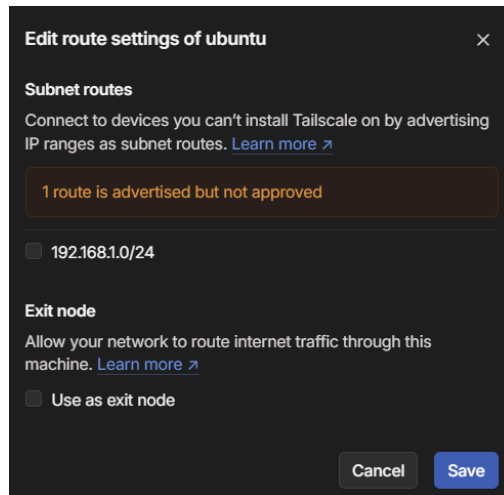


Figure 8: Accept

Tailscale offers a feature that allows one of your devices to act as an exit node to the internet. This means that network traffic can be routed through the selected device, increasing anonymity and security. For example, while traveling, you can use the internet through your home computer, which helps conceal your actual location and secure your data when using public networks. This feature is one of the many ways Tailscale helps protect user privacy and provides secure, remote access to the network.tailscale.com/kb/1103/exit-nodes

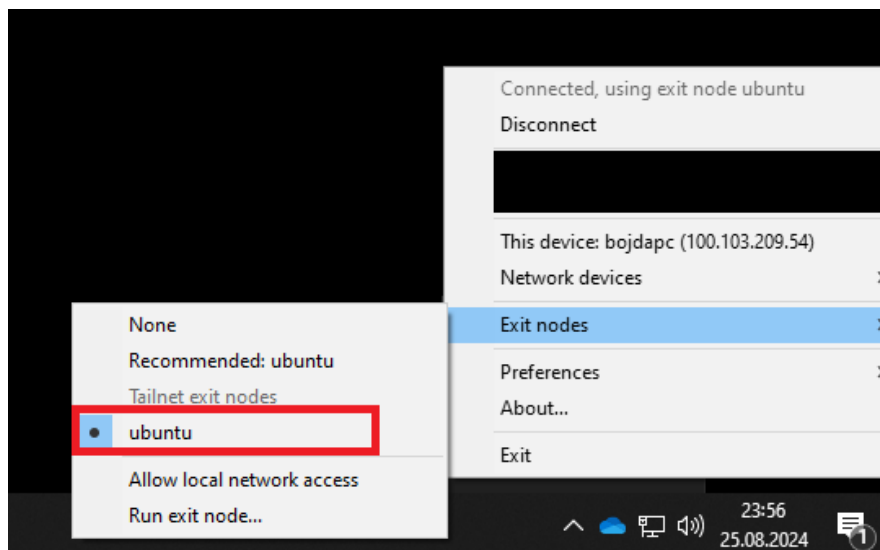


Figure 9: Exit Node

My public IP address when VPN is down.

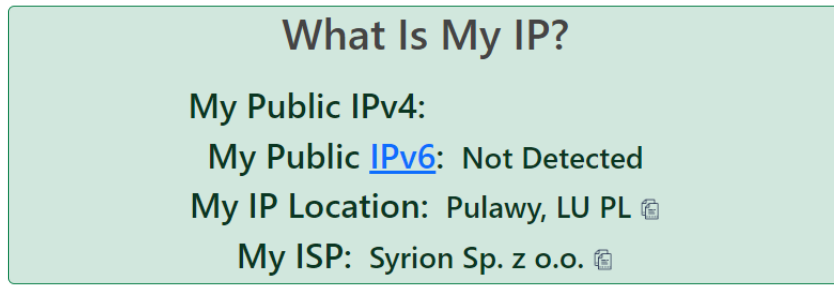


Figure 10: IP

Here this is my IP when VPN is up.

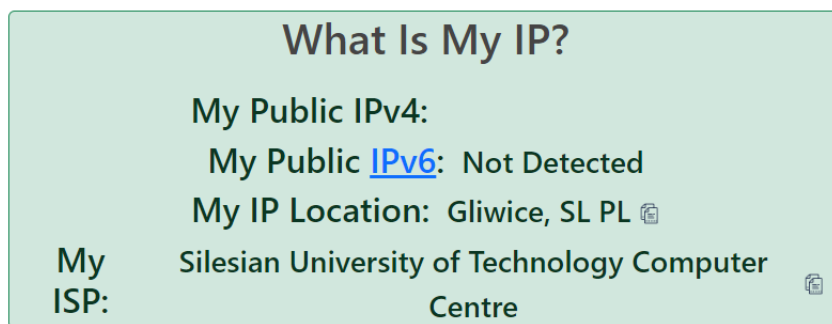


Figure 11: IP

2.3 Containers

In this section, I will discuss how to install various applications using Docker. I will show both docker-compose.yaml files and standard Docker commands to illustrate the different approaches to achieving the goal. Installation details are available in the documentation for each application on the website. hub.docker.com.

2.3.1 Portainer

Portainer is a tool for managing Docker and Kubernetes environments through a user-friendly graphical interface. It allows for easy management of containers, images, networks, and volumes, as well as monitoring of resources. Portainer simplifies administration even in complex container environments, making it an ideal tool for both beginners and experienced Docker users.

Docker-compose.yaml

```
1 version: '2'
2 services:
3   portainer:
4     image: portainer/portainer
5     ports:
6       - 9000:9000
7     volumes:
8       - /var/run/docker.sock:/var/run/docker.sock
9       - /portainer:/data
10    restart: always
```

In the folder where the docker-compose file is located, we run the command:

```
1    docker compose up -d
```

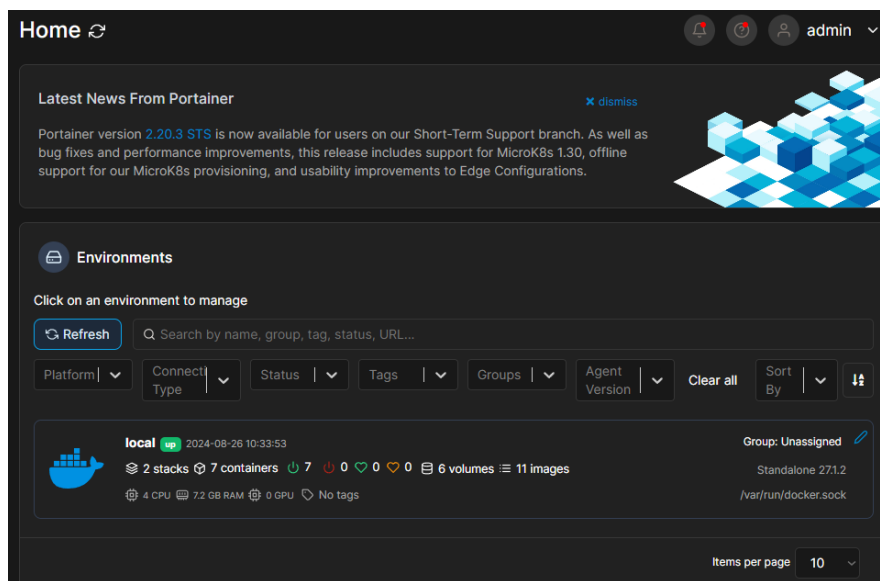


Figure 12: Portainer Dashboard

2.3.2 File Browser

This is a simple application used for visualizing the directory structure on our terminal's disk. It allows us to easily create new folders and files, and navigate through directories more efficiently. To install the application, we can use the templates offered by Portainer in the section **App Templates**.

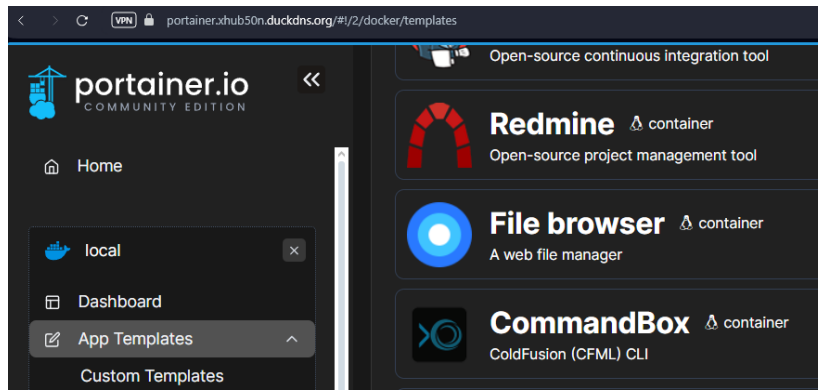


Figure 13: File Browser template

Before starting the container, it is advisable to adjust the advanced settings related to the port on which the application will run and which location on our disk will be mounted to the container using a Bind Mount. For our purposes, we can use the `/` location as the entire disk

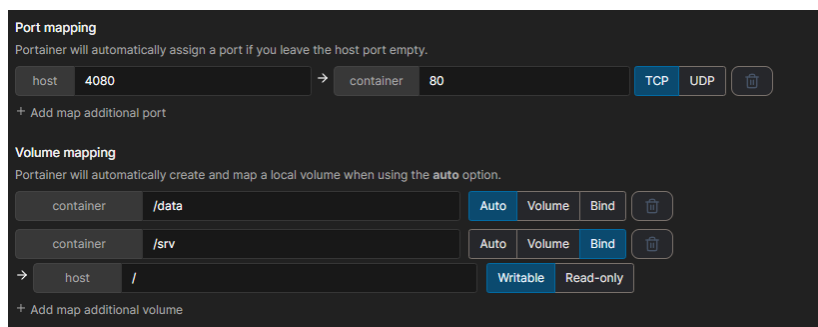


Figure 14: File Browser settings

on which the application is installed. And this is how the application looks. The default login is: **admin** and password **admin**

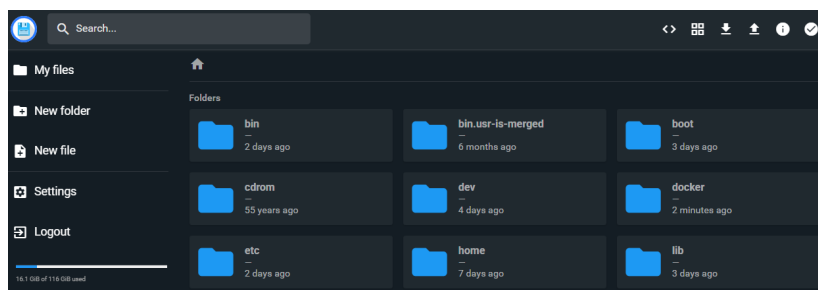


Figure 15: Application

2.3.3 NGINX Proxy Manager

NGINX Proxy Manager is a simple and user-friendly graphical interface for managing an NGINX proxy server. It allows you to create and manage reverse proxies, SSL certificates, redirects, and access rules without needing to manually edit configuration files. It is ideal for users who want to quickly set up and manage a proxy using an accessible control panel.

To create an application correctly on your system, you need to set up the appropriate folder structure, namely:

```
1 /nginx
2 /data
3 /mysql
4 /letsencrypt
5 config.json
6 docker-compose.yaml
```

config.json file content

```
1 {
2   "database":
3     {
4       "engine": "mysql",
5       "host": "db",
6       "name": "npm",
7       "user": "npm",
8       "password": "YOUR PASSWORD",
9       "port": 3306
10    }
11 }
```

docker-compose.yaml file

```
1 version: '2'
2 services:
3   app:
4     image: 'jc21/nginx-proxy-manager:latest'
5     restart: always
6     ports:
7       - '80:80'
8       - '443:443'
9       - '81:81'
10    volumes:
11      - './config.json:/app/config/production.json'
12      - './data:/data'
13      - './letsencrypt:/etc/letsencrypt'
14    depends_on:
15      - db
16
17   db:
18     image: 'jc21/mariadb-aria:latest'
19     restart: always
20     environment:
21       MYSQL_ROOT_PASSWORD: 'your password'
22       MYSQL_DATABASE: 'npm'
23       MYSQL_USER: 'npm'
```

```

24     MYSQL_PASSWORD: 'your password'
25     volumes:
26     - './data/mysql:/var/lib/mysql'

```

After installing the application, we need to log in. The default login is: **admin@example.com** and the password is **changeme**. After logging in, you will be prompted to change the username and password to your own. Below is the main view of the application:

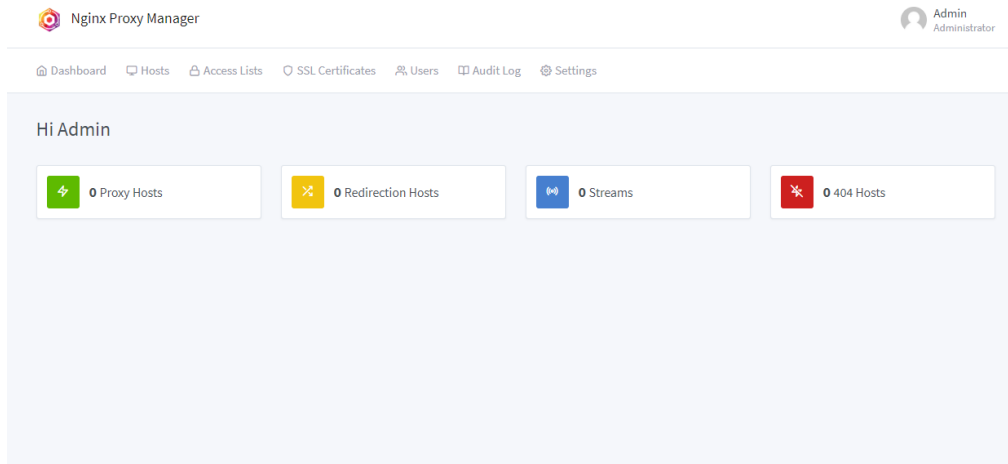


Figure 16: NGNIX

Now, I will present the configuration for accessing the Portainer application through NGINX.

- First, we need a domain for DNS access. For this purpose, we can use the service www.duckdns.org, which allows you to create a free dynamic DNS domain. This will be more than sufficient for this project.

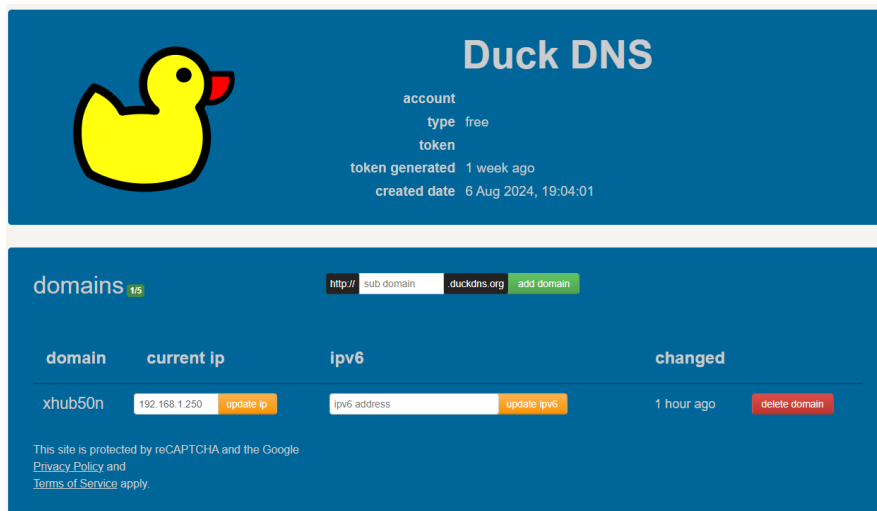


Figure 17: DuckDNS

When executing the command `ping any-name.your-domain.org`, it will return the address you provided during the DuckDNS configuration.

```
Pinging x.xhub50n.duckdns.org [192.168.1.250] with 32 bytes of data:  
Reply from 192.168.1.250: bytes=32 time=6ms TTL=64  
Reply from 192.168.1.250: bytes=32 time=6ms TTL=64  
Reply from 192.168.1.250: bytes=32 time=6ms TTL=64  
Reply from 192.168.1.250: bytes=32 time=6ms TTL=64
```

Figure 18: Ping

- The next step is to create a new proxy host in NGINX Proxy Manager. Once we have our domain, we can also take advantage of the SSL certificate creation feature provided by Let's Encrypt. In the domain name field, enter the domain created with DuckDNS along with a unique name for the specific service. Then, assign the IP address where your application is hosted and the port number.

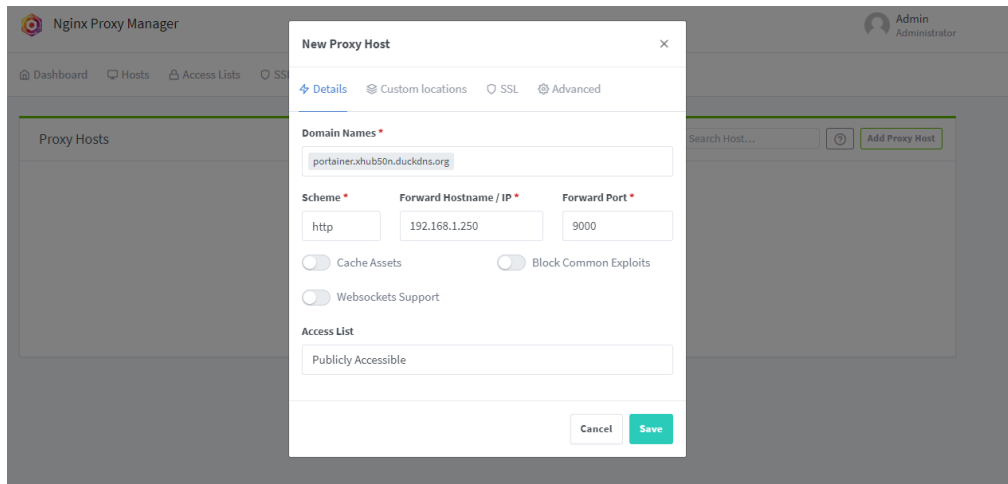


Figure 19: Adding host

- The final step is to create an SSL certificate using the Let's Encrypt service. Some applications, such as NextCloud, require the application to operate over an encrypted connection. Let's Encrypt is a free service that automatically issues SSL/TLS certificates, enabling encryption between the server and the user. With a straightforward tool like NGINX, you can easily obtain and renew certificates, allowing any website to quickly switch to HTTPS.

The main advantages of Let's Encrypt include no cost, automation of the process, and compatibility with most browsers and devices, which significantly enhances website security. In the 'Credentials File Content' field, enter the token provided by DuckDNS, and ensure that the connection is always enforced to use HTTPS.

Edit Proxy Host [X]

Details Custom locations **SSL** Advanced

SSL Certificate

Request a new SSL Certificate

☒ Force SSL ☒ HTTP/2 Support

☒ HSTS Enabled ⓘ ☐ HSTS Subdomains

☒ Use a DNS Challenge

⚠ This section requires some knowledge about Certbot and its DNS plugins. Please consult the respective plugins documentation.

DNS Provider *

DuckDNS

Credentials File Content *

`dns_duckdns_token=`

ⓘ This plugin requires a configuration file containing an API token or other credentials to your provider

⚠ This data will be stored as plaintext in the database and in a file!

Propagation Seconds

ⓘ Leave empty to use the plugins default value. Number of seconds to wait for DNS propagation.

Figure 20: Create certificate

- We can also use a wildcard certificate, which means we won't need to create a certificate for each individual subdomain—one certificate can cover all subdomains. The choice is entirely up to us on how we want to create and manage certificates.

Add Let's Encrypt Certificate

Domain Names *

*.xhub50n.duckdns.org

These domains must be already configured to point to this installation

Email Address for Let's Encrypt *

☒ Use a DNS Challenge

This section requires some knowledge about Certbot and its DNS plugins. Please consult the respective plugins documentation.

DNS Provider *

DuckDNS

Credentials File Content *

dns_duckdns_token=your-duckdns-token

This plugin requires a configuration file containing an API token or other credentials to your provider

This data will be stored as plaintext in the database and in a file!

Figure 21: Wild Card

SSL Certificates			
		Search Certificate...	Add SSL Certificate
NAME	CERTIFICATE PROVIDER	EXPIRES	
<div> <div> </div> <div> *.xhub50n.duckdns.org </div> <div> Created: 26th August 2024 </div> </div>	Let's Encrypt - DuckDNS	24th November 2024, 9:01 am	

Figure 22: Certificate

IF THE CERTIFICATE CREATION DOES NOT WORK IMMEDIATELY, IT'S WORTH REPEATING THE PROCESS AND ADDING A WAIT TIME FOR THE CREATION TO COMPLETE!!!

Next, when assigning a new host to the proxy, you can select the certificate that you created earlier.

Edit Proxy Host

×

⚡ Details

📍 Custom locations

🔒 SSL

⚙️ Advanced

SSL Certificate

*.xhub50n.duckdns.org

🔴

Force SSL

🔴

HTTP/2 Support

🔴

HSTS Enabled ?

🔴

HSTS Subdomains

Cancel

Save

Figure 23: Create new host

Here is how everything should look after correctly configuring our services.

Proxy Hosts					
SOURCE		DESTINATION	SSL	ACCESS	STATUS
👤	filebrowser.xhub50n.duckdns.org Created: 26th August 2024	http://192.168.1.250:4080	Let's Encrypt	Public	● Online
👤	heimdall.xhub50n.duckdns.org Created: 26th August 2024	http://192.168.1.250:8080	Let's Encrypt	Public	● Online
👤	nextcloud.xhub50n.duckdns.org Created: 26th August 2024	http://192.168.1.250:5000	Let's Encrypt	Public	● Online
👤	nginx.xhub50n.duckdns.org Created: 26th August 2024	http://192.168.1.250:81	Let's Encrypt	Public	● Online
👤	portainer.xhub50n.duckdns.org Created: 26th August 2024	http://192.168.1.250:9000	Let's Encrypt	Public	● Online

Figure 24: NGINX

2.3.4 Heimdall

Heimdall is a dashboard application that serves as a central access point for all applications and services running on a server. It allows you to create visual shortcuts to various web applications, websites, or services, with the option to add icons, descriptions, and monitor statuses. Heimdall is user-friendly and doesn't require complex configuration, making it an ideal tool for organizing access to resources in a home network or business.

Docker-compose.yml

```
1 services:
2   heimdall:
3     image: lscr.io/linuxserver/heimdall:latest
4     container_name: heimdall
5     environment:
6       - PUID=1000
7       - PGID=1000
8       - TZ=Etc/UTC
9     volumes:
10      - /docker/heimdall/config:/config
11     ports:
12       - 8080:80
13       - 8443:443
14     restart: unless-stopped
```

After starting the application, you will see a blank screen. This means the application is running but still needs to be configured. We need to add the applications we want to connect to.

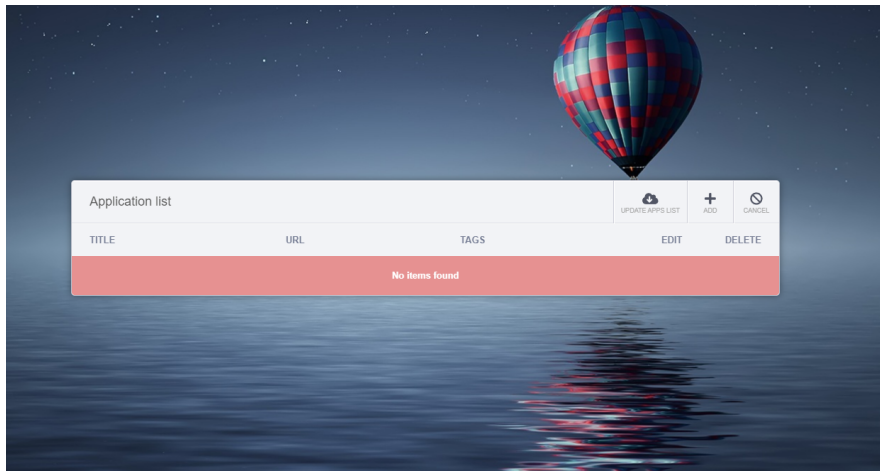


Figure 25: Heimdall

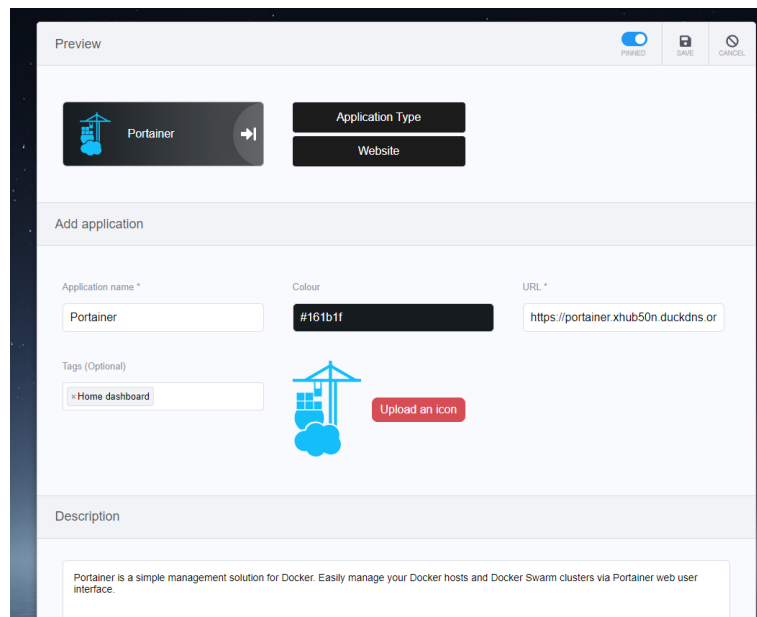


Figure 26: Adding page

As you can see while creating an entry, you can notice a link containing HTTPS.

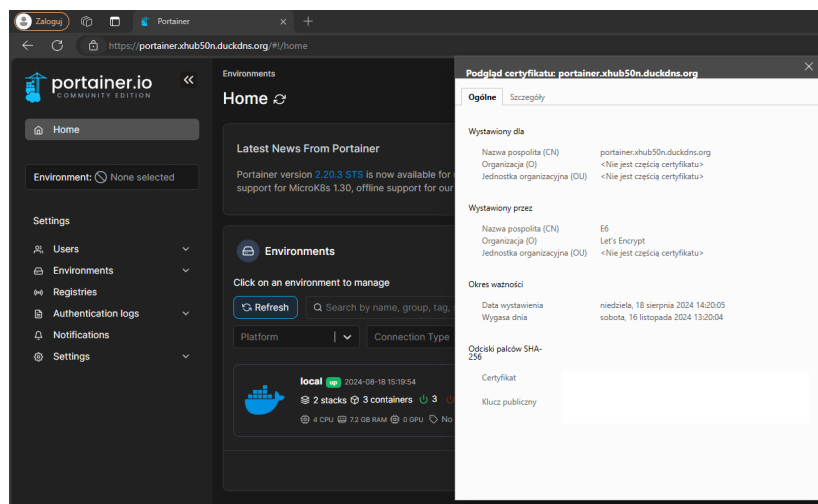


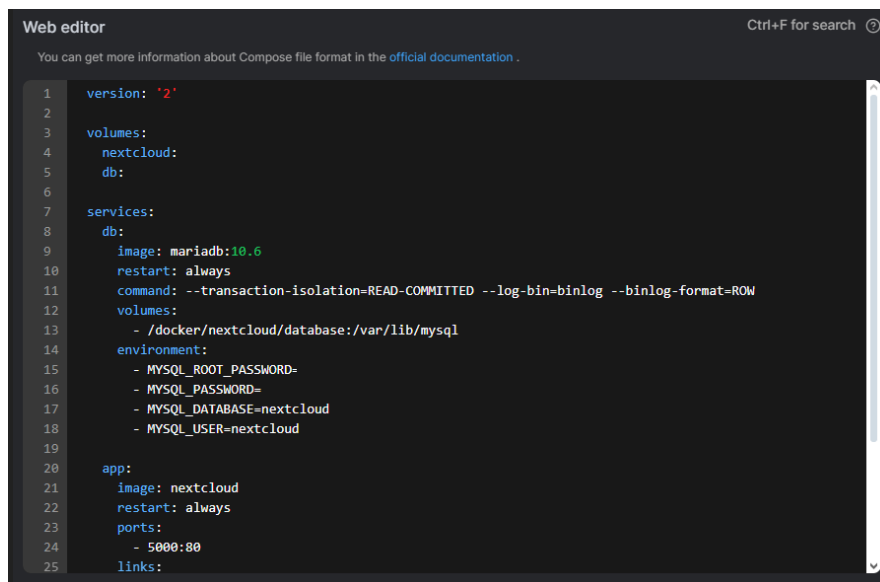
Figure 27: Portainer HTTPS

2.3.5 Nextcloud

Nextcloud is a powerful alternative to popular cloud services such as those offered by Google or Microsoft. Unlike commercial solutions, Nextcloud gives users full control over their data, allowing you to run a private cloud on your own server—whether it's local or remote. As open-source software, Nextcloud enables file synchronization across various devices, ensuring that your documents, photos, and other files are always accessible and up-to-date, regardless of whether you're using a computer, smartphone, or tablet.

However, Nextcloud is much more than just a file synchronization tool. It is a comprehensive platform offering a wide range of features that can replace many other online services. It includes tools for managing calendars, contacts, and notes, as well as applications supporting team collaboration, such as document editors, chat, video conferencing, and more. This makes Nextcloud not only a data storage solution but also a hub for management and collaboration. One of Nextcloud's greatest strengths is its remarkable flexibility. With an extensive plugin system, the software can be customized to meet very specific needs of users or businesses. The ability to integrate with other tools and full control over configuration make Nextcloud an ideal solution for both individuals and large enterprises. Furthermore, complete control over the infrastructure ensures that your data remains private and protected from unauthorized access. In an era when privacy awareness is increasing, Nextcloud offers a level of certainty that many commercial cloud solutions lack.

From within the Portainer application, you can install your applications using a file docker-compose.yaml

A screenshot of a web editor interface with a dark theme. The editor displays a Docker Compose file for Nextcloud. The file includes a version declaration, volume definitions for 'nextcloud' and 'db', and two service definitions: 'db' (MariaDB) and 'app' (Nextcloud). The 'db' service is configured with a specific image, restart policy, command, and environment variables. The 'app' service is configured with the 'nextcloud' image, restart policy, and port mapping. A search bar is visible in the top right corner.

```
1 version: '2'
2
3 volumes:
4   nextcloud:
5     db:
6
7 services:
8   db:
9     image: mariadb:10.6
10    restart: always
11    command: --transaction-isolation=READ-COMMITTED --log-bin=binlog --binlog-format=ROW
12    volumes:
13      - /docker/nextcloud/database:/var/lib/mysql
14    environment:
15      - MYSQL_ROOT_PASSWORD=
16      - MYSQL_PASSWORD=
17      - MYSQL_DATABASE=nextcloud
18      - MYSQL_USER=nextcloud
19
20   app:
21     image: nextcloud
22     restart: always
23     ports:
24       - 5000:80
25     links:
```

Figure 28: Nextcloud

Docker-compose.yaml

```
1 version: '2'
2
3 volumes:
4   nextcloud:
5     db:
6
7 services:
8   db:
9     image: mariadb:10.6
10    restart: always
11    command: --transaction-isolation=READ-COMMITTED --log-bin=binlog --
12             binlog-format=ROW
13    volumes:
14      - /your/path/database:/var/lib/mysql
```

```

14     environment:
15         - MYSQL_ROOT_PASSWORD={YOUR_PASSWORD}
16         - MYSQL_PASSWORD={YOUR_PASSWORD}
17         - MYSQL_DATABASE=nextcloud
18         - MYSQL_USER=nextcloud
19
20     app:
21         image: nextcloud
22         restart: always
23         ports:
24             - 5000:80
25         links:
26             - db
27         volumes:
28             - /your/path/nextcloud-data:/var/www/html
29         environment:
30             - MYSQL_PASSWORD={YOUR_PASSWORD}
31             - MYSQL_DATABASE=nextcloud
32             - MYSQL_USER=nextcloud
33             - MYSQL_HOST=db

```

Attempting to launch the application from a web browser on a phone:

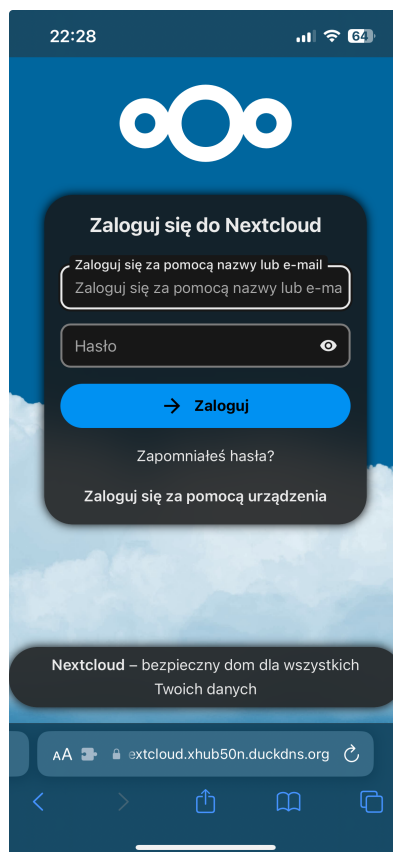


Figure 29: Nextcloud in safari

To use the application with HTTPS, we need to add a few lines of code to our application. In the local folder, we need to edit the file: `/nextcloud-data/config/config.php` after that

you need to restart container.

```
1 'trusted_domains' =>
2   array (
3     0 => '192.168.1.250:5000',
4     1 => 'nextcloud.xhub50n.duckdns.org',
5   ),
6 'overwrite.cli.url' => 'https://nextcloud.xhub50n.duckdns.org',
7 'overwriteprotocol' => 'https'
```

2.3.6 AdGuardHome

AdGuard Home is software for ad-blocking and privacy protection that operates as a local DNS server. It filters ads, tracking, and malicious software at the network level, providing enhanced privacy and faster browsing. AdGuard Home is easy to configure and manage, supporting multiple devices in a home or office network and allowing centralized content filtering.


We will deploy this application using a Docker command.

```
1 docker run --name adguardhome -v /docker/adguard/workdir:/opt/
   adguardhome/work -v /docker/adguard/confdir:/opt/adguardhome/
   conf --network=host --restart always -d adguard/adguardhome
```

The application will be connected to the host network, not the Docker network, and it will always run even if the terminal is restarted. Alternatively, we can use a docker-compose file for deployment.

```
1 ####
2 # It's required to release 53 port
3 # $ sudo systemctl stop systemd-resolved
4 # $ sudo systemctl disable systemd-resolved.service
5 # $ sudo reboot
6 ####
7
8 version: "2"
9 services:
10   adguardhome:
11     image: adguard/adguardhome:latest
12     container_name: adguardhome
13     restart: always
14     ports:
15       - 53:53/tcp
16       - 53:53/udp
17       - 853:853/tcp
18       - 3000:3000/tcp
19     volumes:
20       - /your/own/dir/work:/opt/adguardhome/work
21       - /your/own/dir/conf:/opt/adguardhome/conf
```

Here, the container will also start with the system, and Docker will ensure that the container remains running at all times.



Interfejs internetowy administratora

Interfejs sieciowy

Port

Wszystkie interfejsy

3000

Twój interfejs www AdGuard Home Admin będzie dostępny pod następującymi adresami:

- <http://127.0.0.1:3000>
- <http://172.18.0.2:3000>

Serwer DNS

Interfejs sieciowy

Port

Wszystkie interfejsy

53

Konieczne będzie skonfigurowanie urządzenia lub routera do korzystania z serwera DNS pod następującymi adresami:

- 127.0.0.1
- 172.18.0.2

Statyczny adres IP

AdGuard Home to serwer, więc do poprawnego działania potrzebuje statycznego adresu IP. W przeciwnym razie router może przypisać temu urządzeniu inny adres IP.

Wróć

Dalej

Figure 30: Adguard installation

To take advantage of this application, we need to add the IP address of our terminal in the router's DHCP settings so that all devices on our network can benefit from AdGuard.

22

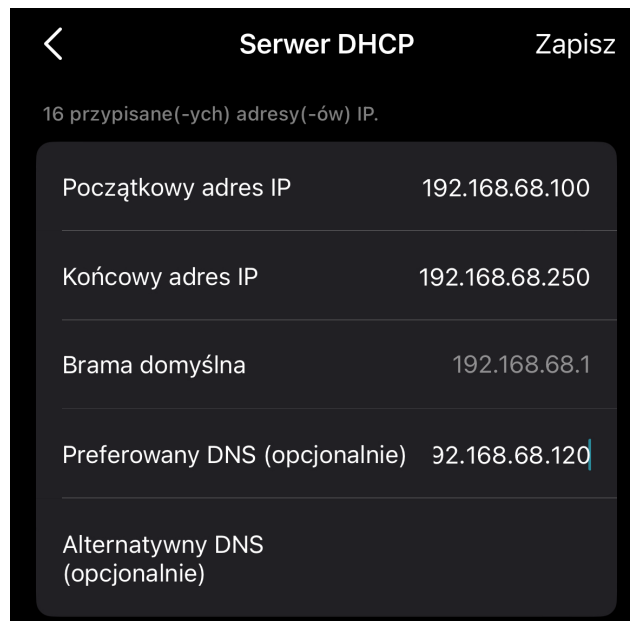


Figure 31: Change the DNS server in the router settings

After installing the application and logging in, you will see a dashboard displaying detailed information about network traffic and blocked sites.

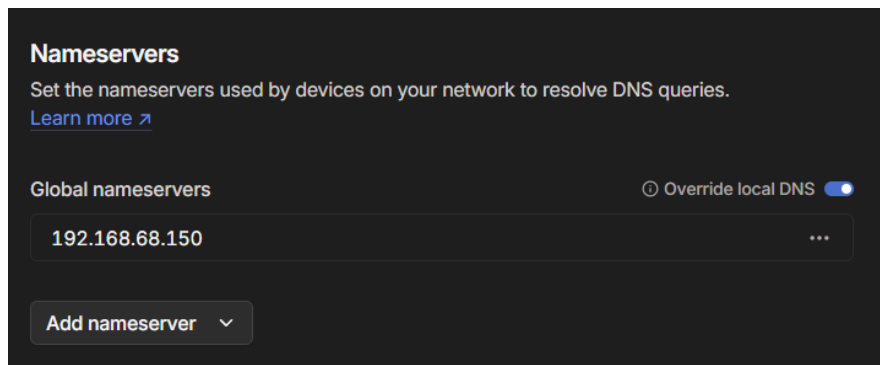


Figure 32: Dodanie serwera DNS do sieci Tailscale

To ensure all our devices on the Tailscale network use the Adguard DNS server, we need to go to the DNS options in the Tailscale admin panel and add the IP address of our device in the Nameservers section.

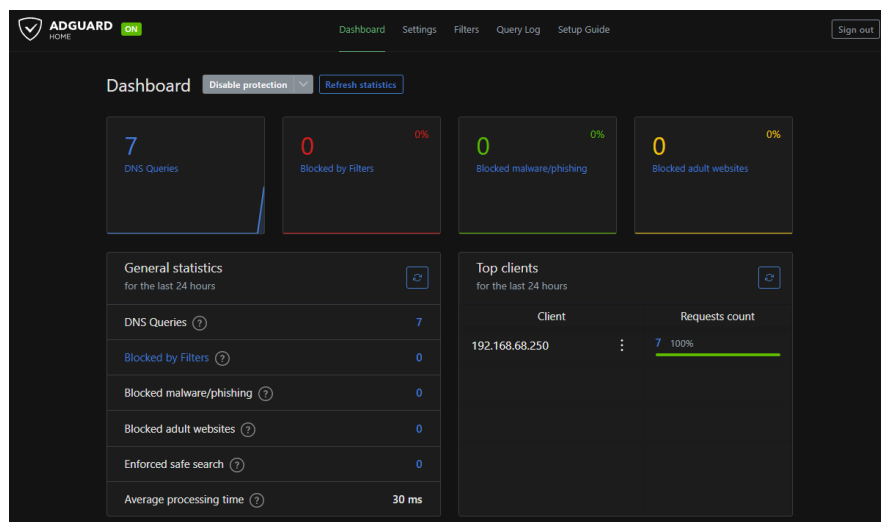


Figure 33: Dashboard

Another feature is the addition of filters, which are publicly available on the internet and allow for blocking ads, pop-ups, and cookies.

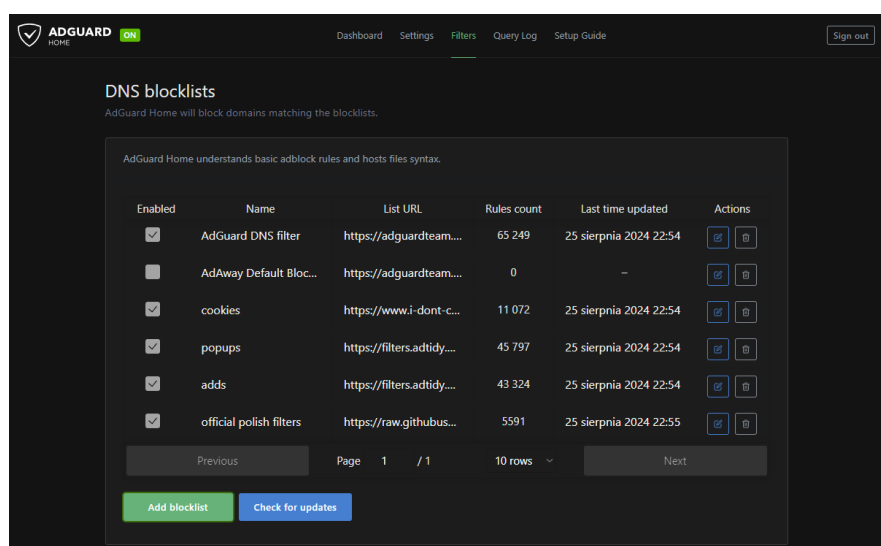


Figure 34: Filters

2.3.7 RustDesk

RustDesk is an open-source remote desktop application that allows remote control of another computer or device. It is an alternative to popular tools such as TeamViewer or AnyDesk. RustDesk is characterized by ease of use, security, and the ability to self-host a server, giving greater control over privacy and the security of connections.

- **Remote access and control:** Allows controlling remote devices as if the user were sitting in front of them.
- **Open-source:** The application's source code is open, allowing for review and customization.
- **Private servers:** The ability to run your own server to avoid using public servers, enhancing security.
- **Portability:** Works on multiple operating systems, such as Windows, macOS, Linux, and mobile devices (Android, iOS).
- **Encryption:** Supports encrypted connections for data security.

It is a very useful tool for system administrators, technical support teams, and individuals who need access to their computers from anywhere.

Here is how the YAML manifest for running the RustDesk service looks:

```
1 services:
2   hbbs:
3     container_name: hbbs
4     image: rustdesk/rustdesk-server:latest
5     command: hbbs
6     volumes:
7       - /docker/rustdesk:/root
8     network_mode: "host"
9     depends_on:
10      - hbbr
11     restart: always
12
13   hbbr:
14     container_name: hbbr
15     image: rustdesk/rustdesk-server:latest
16     command: hbbr
17     volumes:
18       - /docker/rustdesk:/root
19     network_mode: "host"
20     restart: always
```

The containers that have been created serve as a locally hosted server. To connect to the server, we need to install the RustDesk client on the local client machines. The client can be downloaded from [github.com/rustdesk](https://github.com/rustdesk/rustdesk). Choose the installation for your operating system and install it after downloading.

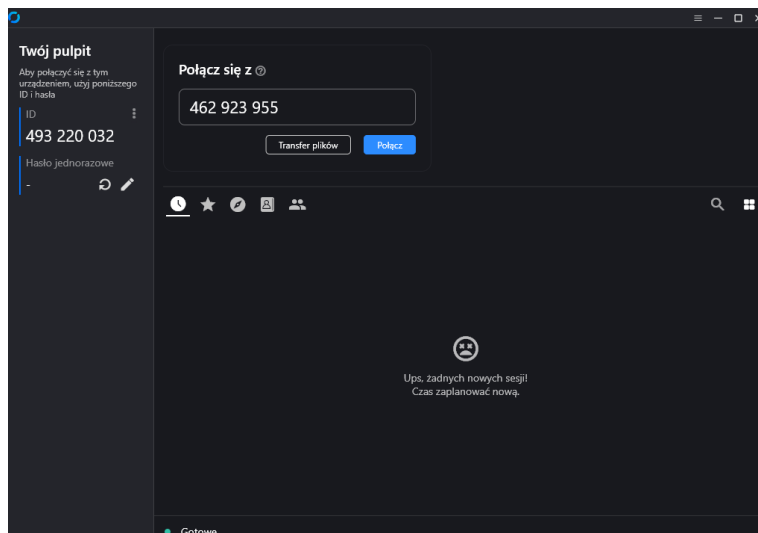


Figure 35: RustDesk Windows client

To work with our RustDesk server, we need to provide two pieces of information for the client to start functioning.

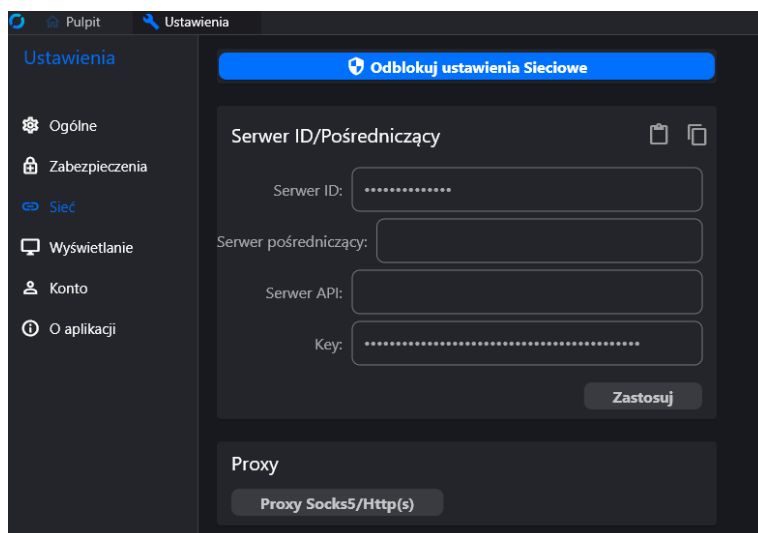


Figure 36: RustDesk credentials

We need to provide the IP address of our server and the public key used for encrypted transmission, ensuring that no one outside the network can view the data. An additional benefit of the Tailscale network is that it provides encrypted transmission by itself. The public key must be obtained from the folder on the server that is attached to the container. Additionally, we can set permissions for those who want to connect to our host. For example, we can disable mouse control, turn off sound, and much more. If we want to start a session with a new host, we need to know its ID displayed in the RustDesk application and the access password. By default, the option is enabled for a one-time password, which generates a new password with each connection. However, this can be disabled, and a custom password can be set, which simplifies managing devices on a home network.