

# Docker home lab

Dokumentacja stworzenia własnego laboratorium opartego o kontenery Docker

Hubert Bojda

26 sierpnia 2024

# Spis treści

<b>1</b>	<b>Sprzęt</b>	<b>3</b>
<b>2</b>	<b>Oprogramowanie</b>	<b>4</b>
2.1	Docker . . . . .	4
2.2	Tailscale . . . . .	4
2.3	Kontenery . . . . .	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

# 1 Sprzęt

Celem projektu jest stworzenie własnego laboratorium, które można zrealizować na dowolnym urządzeniu wspierającym technologię konteneryzacji, taką jak Docker. Na początek sugeruję utworzenie wirtualnej maszyny z systemem Linux, na przykład Debianem lub Ubuntu. W przyszłości można rozważyć zakup dedykowanego sprzętu, takiego jak Raspberry Pi lub terminale, znane również jako "ciency klienci". W moim przypadku wybór padł na terminal HP T630, który w porównaniu z Raspberry Pi często można kupić za połowę ceny. Jest to duża zaleta, choć Raspberry Pi wygrywa pod względem niskiego zużycia energii dzięki procesorowi opartemu na architekturze ARM.

Ten terminal oferuje wiele możliwości, szczególnie biorąc pod uwagę jego przystępną cenę. Za około 160 zł otrzymujemy komputer z dużą ilością pamięci RAM i wielordzeniowym procesorem, co pozwala na realizację szerokiego zakresu zadań. Alternatywą może być Raspberry Pi, które jest co prawda niemal dwa razy droższe, ale charakteryzuje się niższym poborem energii. Wybór zależy od indywidualnych potrzeb i priorytetów.

W moim przypadku terminal zużywa około 11 W w trybie spoczynku i do 25 W przy maksymalnym obciążeniu. Przy założeniu średniego zużycia na poziomie 15 W, koszt rocznej, całodobowej pracy wynosi około 150 zł. To całkiem rozsądna kwota, zważywszy na możliwości i wszechstronność tego urządzenia, które świetnie sprawdzi się jako baza do budowy własnego laboratorium.



Rysunek 1: HP T630

Hardware	Opis
Procesor	AMD GX-420GI, 4 rdzenie
RAM	8GB
Dysk Systemowy	128GB NVMe

## 2 Oprogramowanie

Na ww. sprzęcie został zainstalowany system Ubuntu Server 24.04 LTS. Komunikacja odbywa się przez mój komputer przez usługę SSH.

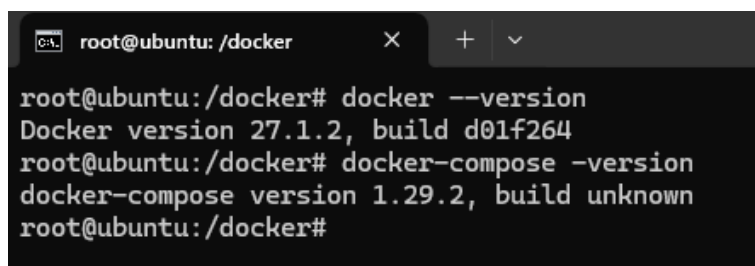
### 2.1 Docker

- Docker został zainstalowany wg. dokumentacji zawartej na stronie <https://docs.docker.com/engine/install/ubuntu> jest tam krok po kroku opisane co mamy wykonać aby zainstalować dockera.
- Kolejnym składnikiem uzupełniającym możliwości Dockera jest Docker Compose, aby pobrać odpowiednią paczkę wykorzystamy wiersz poleceń.

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

---

Sprawdzenie czy oprogramowanie poprawnie zostało zainstalowane:



```

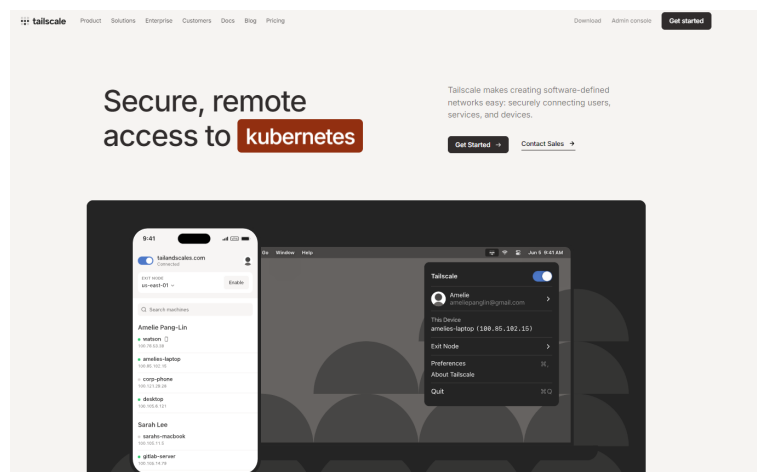
root@ubuntu: /docker
root@ubuntu:/docker# docker --version
Docker version 27.1.2, build d01f264
root@ubuntu:/docker# docker-compose -version
docker-compose version 1.29.2, build unknown
root@ubuntu:/docker#

```

Rysunek 2: Docker i Docker Compose

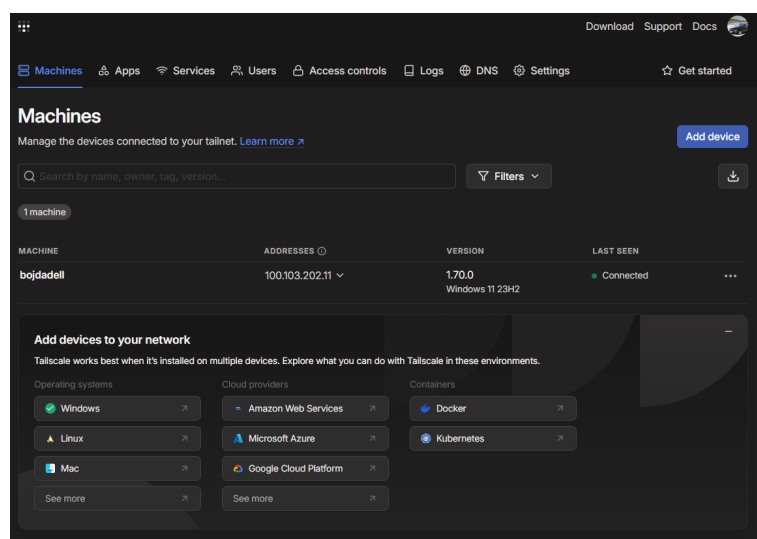
### 2.2 Tailscale

Tailscale to narzędzie do tworzenia prywatnych sieci VPN opartych na WireGuard. Jest łatwe w konfiguracji, umożliwia bezpieczne połączenie między urządzeniami bez potrzeby ustawiania serwerów VPN czy przekierowywania portów. Działa na wielu platformach (Windows, macOS, Linux, iOS, Android), tworząc sieć mesh, w której urządzenia komunikują się bezpośrednio. Tailscale jest idealne do zdalnego dostępu, zabezpieczania aplikacji i tworzenia prywatnych sieci dla zespołów, oferując wysoki poziom bezpieczeństwa i integrację z chmurą. [Tailscale docs](#)



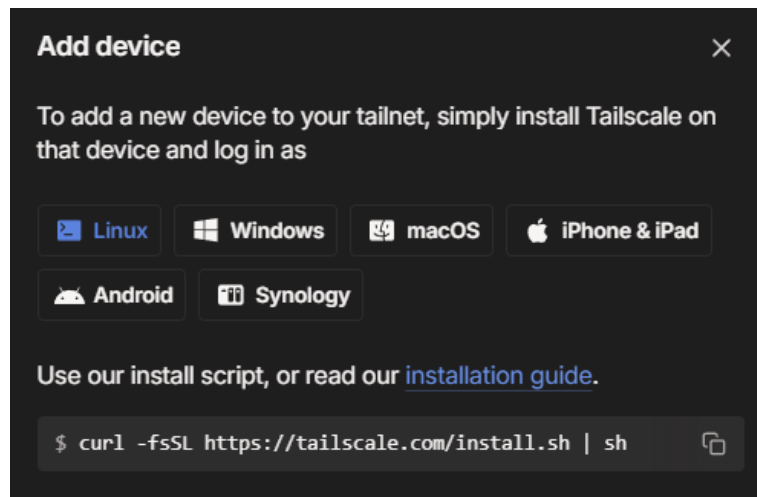
Rysunek 3: Tailscale

Po zarejestrowaniu zostajemy proszeni o dodaniu naszego urządzenia do sieci tailscale, dodatkowo na nasze urządzenie zostanie zainstalowany klient usługi z którego możemy zarządzać innymi urządzeniami w sieci lub np. możemy podejrzeć adres ip danego urządzenia.

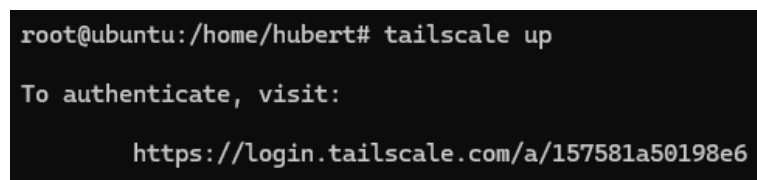


Rysunek 4: Tailscale Dashboard

Aby dodać kolejne urządzenia do naszej sieci, mamy pokazane w prosty sposób jak tego dokonać. Do każdego z systemów jest osobna instrukcja jak podłączyć się do sieci tailscale.

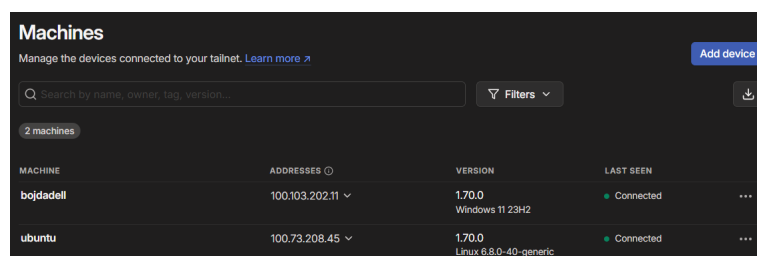


Rysunek 5: Podpinanie Linuxa



Rysunek 6: Tailscale Up Linux

Następnie musimy się zalogować poprzez wygenerowany link a po zatwierdzeniu nasze urządzenie jest już widoczne w sieci. Ciekawą opcją którą oferuje tailscale jest dostęp do urządzeń

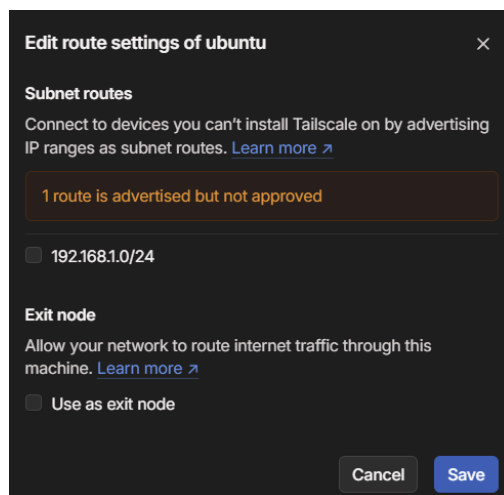


Rysunek 7: Nasze urządzenia

w naszej lokalnej sieci, możemy z poziomu panelu administracyjnego na stronie określić, które z naszych urządzeń będzie routerem dla naszej lokalnej sieci, która nie jest bezpośrednio podpięta do tailscale. [tailscale.com/kb/1406/quick-guide-subnets](https://tailscale.com/kb/1406/quick-guide-subnets)

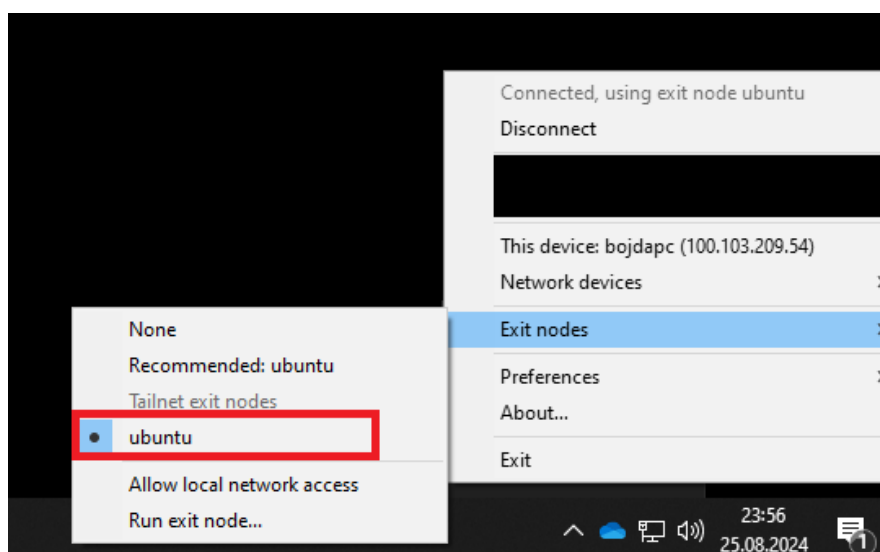
```
1 sudo tailscale up --advertise-routes=adres_sieci/maska_podsieci
```

Następnie musimy tą opcję aktywować już z poziomu strony internetowej:



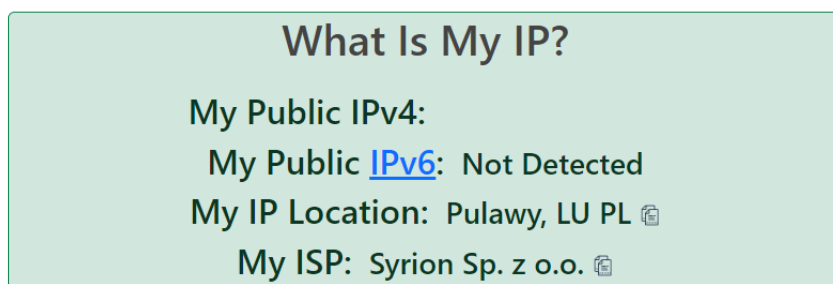
Rysunek 8: Akceptacja

Tailscale oferuje funkcję, która umożliwia wykorzystanie jednego z urządzeń jako węzeł wyjściowy do internetu. Oznacza to, że ruch sieciowy może być przekierowany przez wybrane urządzenie, co zwiększa anonimowość i bezpieczeństwo połączenia. Na przykład, będąc w podróży, możesz korzystać z internetu przez swój domowy komputer, co pozwala ukryć rzeczywistą lokalizację i zabezpieczyć dane podczas korzystania z publicznych sieci. Ta funkcja jest jednym z wielu sposobów, w jakie Tailscale pomaga chronić prywatność użytkowników i zapewniać bezpieczny, zdalny dostęp do sieci. [tailscale.com/kb/1103/exit-nodes](https://tailscale.com/kb/1103/exit-nodes)



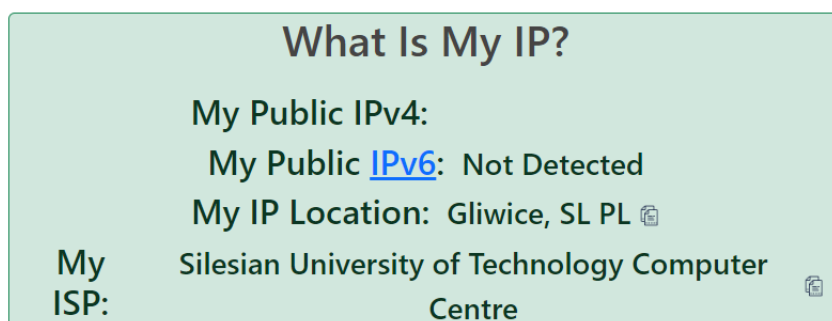
Rysunek 9: Exit Node

Mój adres ip przed uruchomieniem usługi vpn.



Rysunek 10: IP domowe

A tak prezentuje się ip po włączeniu vpn.



Rysunek 11: IP poprzez VPN

## 2.3 Kontenery

W tej sekcji omówię, jak instalować różne aplikacje za pomocą Dockera. Pokażę zarówno pliki docker-compose.yaml, jak i standardowe polecenia Dockera, aby zilustrować różnorodność podejść do osiągnięcia celu. Szczegóły instalacji są dostępne w dokumentacji każdej z aplikacji na stronie [hub.docker.com](https://hub.docker.com).

### 2.3.1 Portainer

Portainer to narzędzie do zarządzania środowiskami Docker i Kubernetes za pomocą przyjaznego interfejsu graficznego. Umożliwia łatwe zarządzanie kontenerami, obrazami, sieciami i wolumenami, a także monitorowanie zasobów. Portainer upraszcza administrację nawet w złożonych środowiskach kontenerowych, co czyni go idealnym narzędziem zarówno dla początkujących, jak i doświadczonych użytkowników Dockera.

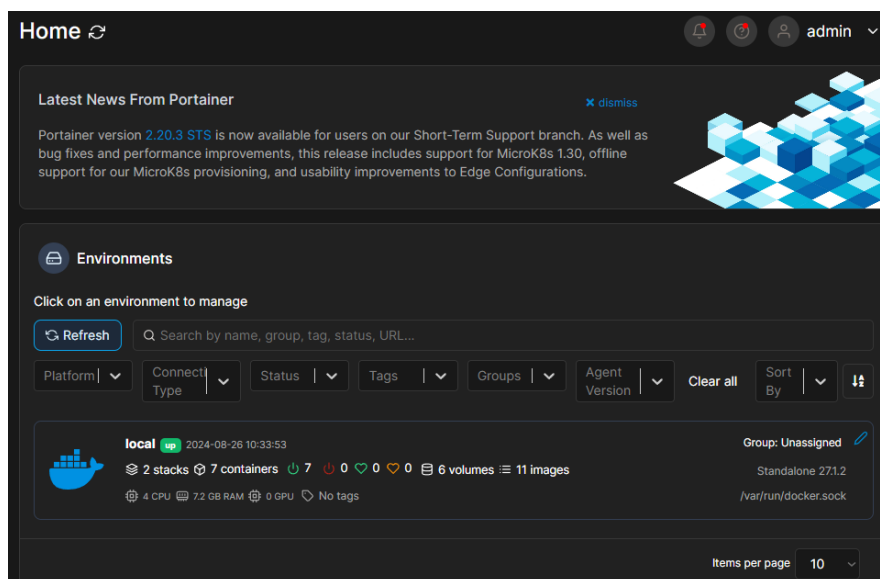
Plik docker-compose.yaml



```
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
```

W folderze, w którym znajduje się plik docker-compose uruchamiamy polecenie:

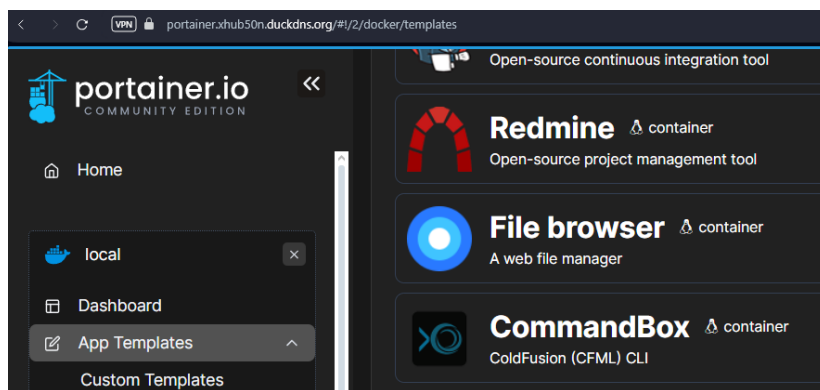
```
1 docker compose up -d
```



Rysunek 12: Portainer Dashboard

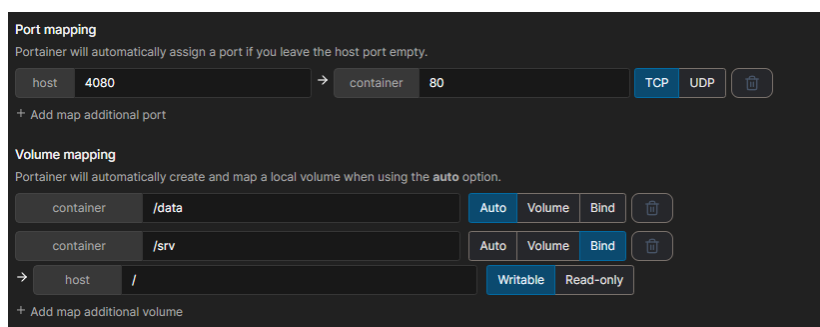
### 2.3.2 File Browser

Jest to prosta aplikacja służąca do wizualizacji struktury katalogów na dysku naszego terminala, możemy w niej w łatwy sposób tworzyć nowe foldery i pliki oraz będziemy mogli sprawniej poruszać się po katalogach. Aby zainstalować aplikację możemy użyć szablonów które oferuje nam portainer w sekcji **App Templates**.



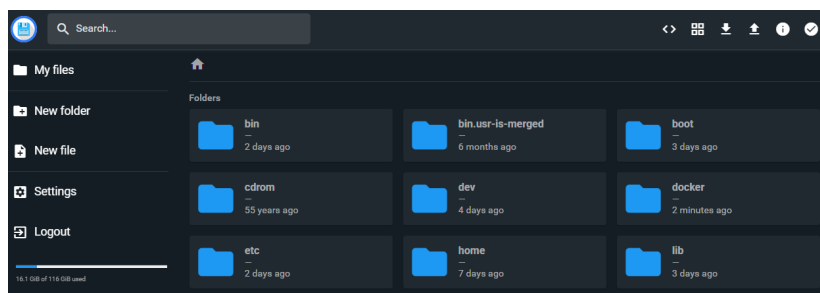
Rysunek 13: File Browser szablon

Przed uruchomieniem kontenera warto zmienić ustawienia zaawansowane dotyczące portu pod którym będziemy uruchamiać aplikację oraz która lokalizacja naszego dysku będzie podpinana za pomocą Bind Mount do kontenera. Na nasze potrzeby możemy wykorzystać lokalizację /



Rysunek 14: Ustawienia File Browser

jako cały dysk na którym jest zainstalowany dysk. A tak prezentuje się aplikacja. Domyślny login to **admin** i hasło **admin**



Rysunek 15: Aplikacja

### 2.3.3 NGINX Proxy Manager

NGINX Proxy Manager to prosty i łatwy w obsłudze interfejs graficzny do zarządzania serwerem proxy NGINX. Umożliwia tworzenie i zarządzanie reverse proxy, certyfikatami SSL, przekierowaniami oraz regułami dostępu, bez potrzeby ręcznego edytowania plików konfiguracyjnych. Jest idealny dla użytkowników, którzy chcą szybko skonfigurować proxy i zarządzać nim za pomocą przystępnego panelu.

Aby w poprawny sposób stworzyć aplikację u siebie trzeba wykonać odpowiednią strukturę folderów a mianowicie:

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

---

Zawartość pliku config.json

```
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 }
```

---

Zawartość pliku docker-compose.yaml

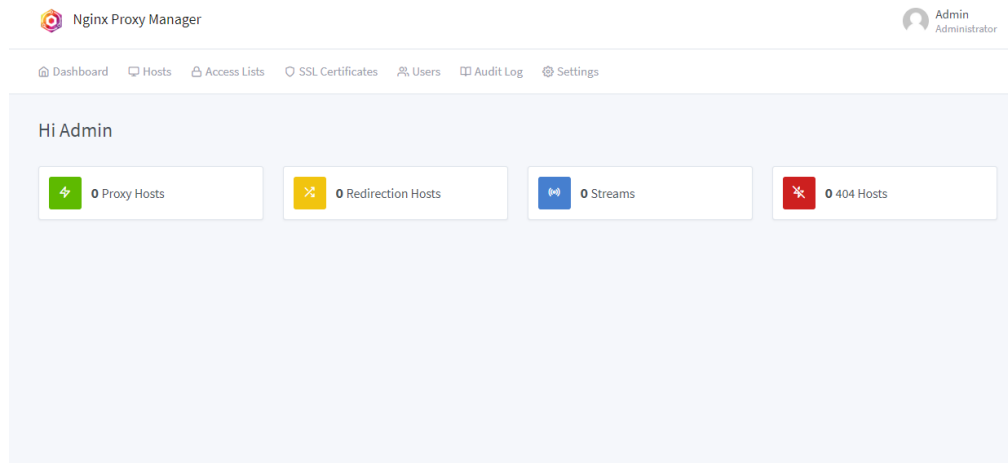
```
1 version: '2'
2 services:
3   app:
4     image: 'jc21/nginx-proxy-manager:latest'
5     restart: always
6     ports:
7       - '80:80'
8       - '443:443'
9       - '81:81'
10    volumes:
11      - './config.json:/app/config/production.json'
12      - './data:/data'
13      - './letsencrypt:/etc/letsencrypt'
14    depends_on:
15      - db
16
17   db:
18     image: 'jc21/mariadb-aria:latest'
19     restart: always
20     environment:
21       MYSQL_ROOT_PASSWORD: 'Zaq12wsx'
22       MYSQL_DATABASE: 'npm'
```

```

23     MYSQL_USER: 'npm'
24     MYSQL_PASSWORD: 'Zaq12wsx'
25     volumes:
26     - './data/mysql:/var/lib/mysql'

```

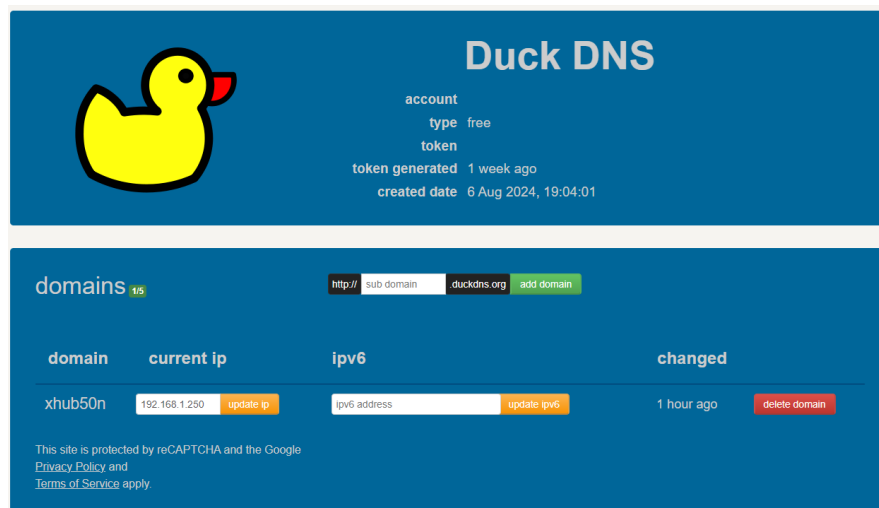
Po zainstalowaniu aplikacji musimy się do niej zalogować, domyślny login to: **admin@example.com** a hasło to **changeme**, po zalogowaniu jesteśmy proszeni o zmianę loginu i hasła na nasze własne, a poniżej prezentuje się główny widok aplikacji:



Rysunek 16: NGNIX

Teraz przedstawię konfigurację dostępu do aplikacji Portainer z poziomu NGNIX.

- W pierwszej kolejności co nam się przyda do dostęp poprzez domenę DNS, możemy w tym celu wykorzystać serwis [www.duckdns.org](https://www.duckdns.org), serwis pozwala na stworzenie darmowej dynamicznej domeny DNS, która zdecydowanie wystarczy nam na realizację tego projektu.



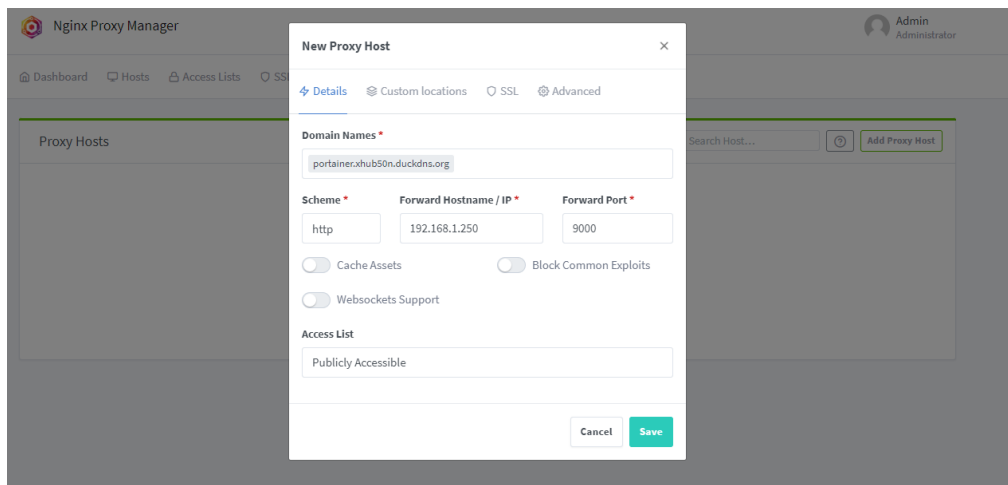
Rysunek 17: DuckDNS

Wykonując polecenie ping dowolna nazwa.nazwadomeny.org będzie zwracany nasz adres który podaliśmy podczas konfiguracji duckdns.

```
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
```

Rysunek 18: Ping

- Kolejnym krokiem będzie utworzenie nowego hosta proxy w NGINX Proxy Manager. Gdy mamy już naszą domenę, będziemy mogli również skorzystać z możliwości tworzenia certyfikatów SSL za pomocą usługi Let's Encrypt. W nazwie domeny wpisujemy naszą domenę stworzoną w DuckDNS oraz unikalną nazwę dla danej usługi, później przypisujemy adres ip pod którym będzie nasza aplikacja oraz port.



Rysunek 19: Dodawanie host-a

- Ostatnim krokiem będzie utworzenie certyfikatu SSL za pomocą usługi Let's Encrypt, niektóre aplikacje takie jak np.NextCloud wymagają tego aby aplikacja działała na szyfrowanym połączeniu. Let's Encrypt to darmowa usługa, która automatycznie wydaje certyfikaty SSL/TLS, pozwalające na szyfrowanie połączeń między serwerem a użytkownikiem. Dzięki prostemu narzędziu, jakim jest NGINX, można łatwo uzyskać i odnawiać certyfikaty, co sprawia, że każda strona internetowa może szybko przejść na HTTPS. Główne zalety Let's Encrypt to brak kosztów, automatyzacja procesu oraz zgodność z większością przeglądarek i urządzeń, co znacząco zwiększa bezpieczeństwo stron internetowych. W miejscu 'Credentials File Content' podajemy token, który wygenerował nam DuckDNS oraz wymuszamy aby połączenie było zawsze po 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.

Rysunek 20: Tworzenie certyfikatu

- Możemy też wykorzystać certyfikat tzw. wild card to oznacza że nie będziemy musieli tworzyć certyfikatu dla każdej ze stron tylko jeden który podepniemy pod wszystkie witryny. Decyzja zależy tylko i wyłącznie od nas w jaki sposób chcemy tworzyć certyfikaty.

Add Let's Encrypt Certificate

Domain Names \*

\*.xhub50n.duckns.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!

Rysunek 21: Tworzenie certyfikatu 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	

Rysunek 22: Certyfikat

**JEŚLI TWORZENIE CERTYFIKATU NIE DZIAŁA OD RAZU TO WARTO POWTÓRZYĆ CZYNNOŚĆ I DODAĆ CZAS OCZEKIWANIA NA TWO-  
WORZENIE!!!**

Następnie podczas przypisywania nowego hosta do proxy możemy wybrać certyfikat który stworzyliśmy przed chwilą.

Edit Proxy Host

×

⚡ Details

📍 Custom locations

🔒 SSL

⚙️ Advanced

SSL Certificate

\*.xhub50n.duckdns.org

Force SSL

HTTP/2 Support

HSTS Enabled ⓘ






HSTS Subdomains

Cancel

Save

Rysunek 23: Tworzenie nowego hosta

A tak powinna prezentować się całość po poprawnej konfiguracji naszych usług.

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	⋮

Rysunek 24: NGINX



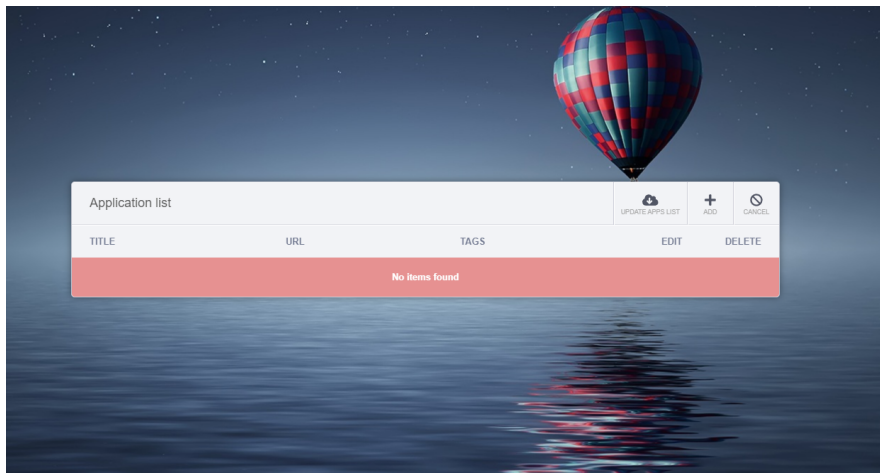
### 2.3.4 Heimdall

Heimdall to aplikacja typu dashboard, która służy jako centralny punkt dostępu do wszystkich aplikacji i usług uruchomionych na serwerze. Pozwala na tworzenie wizualnych skrótów do różnych aplikacji webowych, stron internetowych czy usług, z opcją dodawania ikon, opisów i monitorowania statusu. Heimdall jest prosty w obsłudze i nie wymaga skomplikowanej konfiguracji, co czyni go idealnym narzędziem do organizowania dostępu do zasobów w sieci domowej lub w firmie.

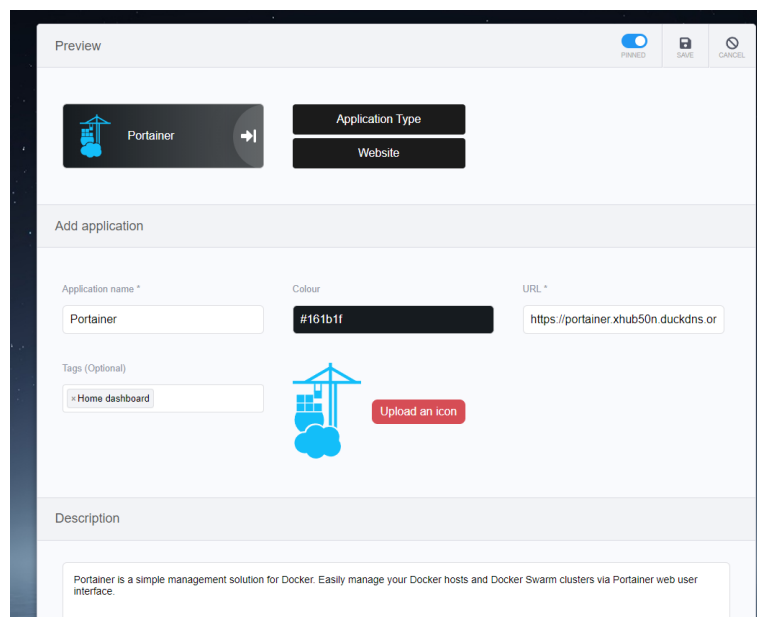
Zawartość pliku docker-compose.yaml

```
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
```

Po uruchomieniu aplikacji ukaże nam się pusty ekran, to oznacza że aplikacja działa ale jeszcze wymaga konfiguracji z naszej strony. Musimy dodać nasze aplikacje do których będziemy chcieli się podłączyć.

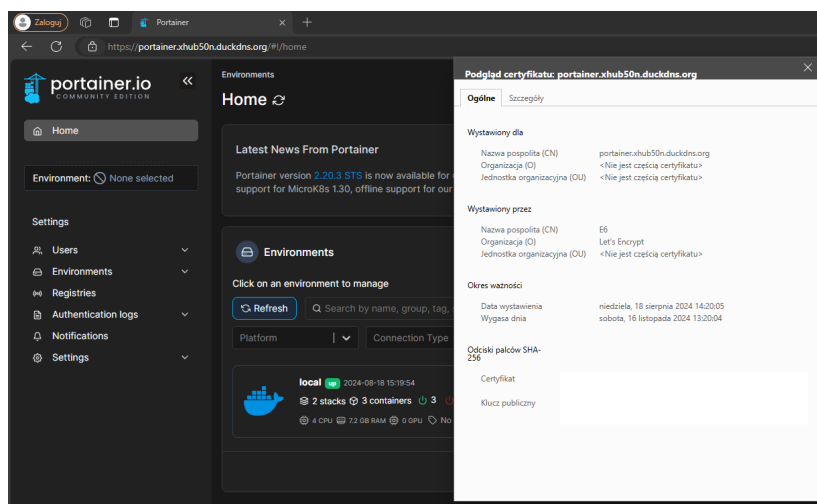


Rysunek 25: Heimdall



Rysunek 26: Dodawanie strony

Jak widać podczas tworzenia wpisu można dostrzec link zawierający HTTPS.



Rysunek 27: Portainer HTTPS

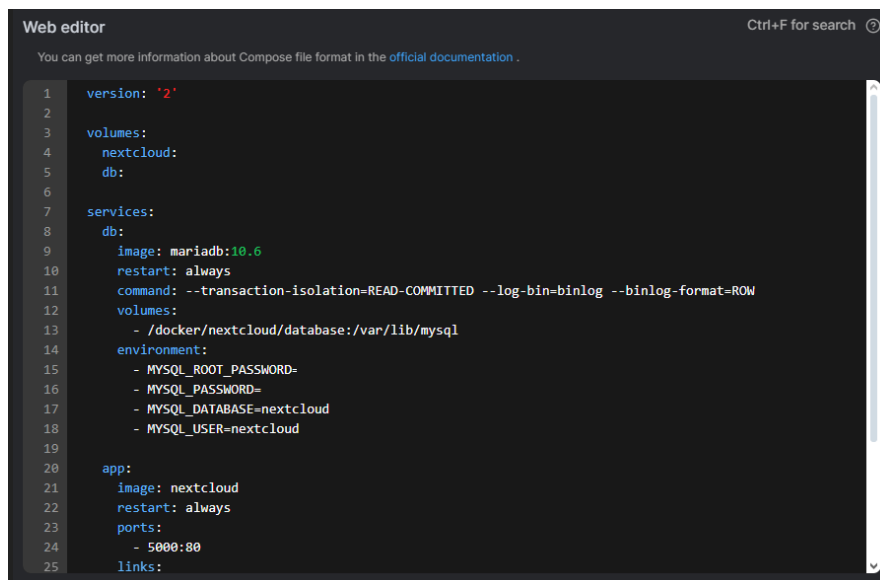
### 2.3.5 Nextcloud

Nextcloud to potężna alternatywa dla popularnych usług chmurowych, takich jak te oferowane przez Google czy Microsoft. W przeciwieństwie do komercyjnych rozwiązań, Nextcloud oddaje pełną kontrolę nad danymi w ręce użytkownika, umożliwiając uruchomienie prywatnej chmury na własnym serwerze – czy to lokalnym, czy zdalnym. Jako oprogramowanie open-source, Nextcloud umożliwia synchronizację plików między różnymi urządzeniami, co zapewnia, że Twoje dokumenty, zdjęcia i inne pliki są zawsze dostępne i aktualne, niezależnie od tego, czy korzystasz z komputera, smartfona, czy tabletu.

Jednak Nextcloud to znacznie więcej niż tylko narzędzie do synchronizacji plików. To kompleksowa platforma oferująca szeroki wachlarz funkcji, które mogą zastąpić wiele innych usług online. Znajdziesz tu narzędzia do zarządzania kalendarzem, kontaktami, notatkami, a także aplikacje wspierające współpracę zespołową, takie jak edytory dokumentów, czaty, wideokonferencje i wiele innych. Dzięki temu Nextcloud staje się nie tylko miejscem przechowywania danych, ale także centrum zarządzania i współpracy.

Jednym z największych atutów Nextcloud jest jego niesamowita elastyczność. Dzięki rozbudowanemu systemowi wtyczek, oprogramowanie można dostosować do bardzo specyficznych potrzeb użytkownika lub firmy. Możliwość integracji z innymi narzędziami oraz pełna kontrola nad konfiguracją sprawiają, że Nextcloud może być idealnym rozwiązaniem zarówno dla osób prywatnych, jak i dużych przedsiębiorstw. Co więcej, pełna kontrola nad infrastrukturą gwarantuje, że Twoje dane pozostają prywatne i zabezpieczone przed dostępem niepowołanych osób. W czasach, gdy świadomość dotycząca ochrony prywatności rośnie, Nextcloud oferuje pewność, której brakuje w wielu komercyjnych rozwiązaniach chmurowych.

Z poziomu aplikacji portainer możemy instalować nasze aplikacje wykorzystując plik docker-compose.yaml



Rysunek 28: Nextcloud

Docker-compose.yaml

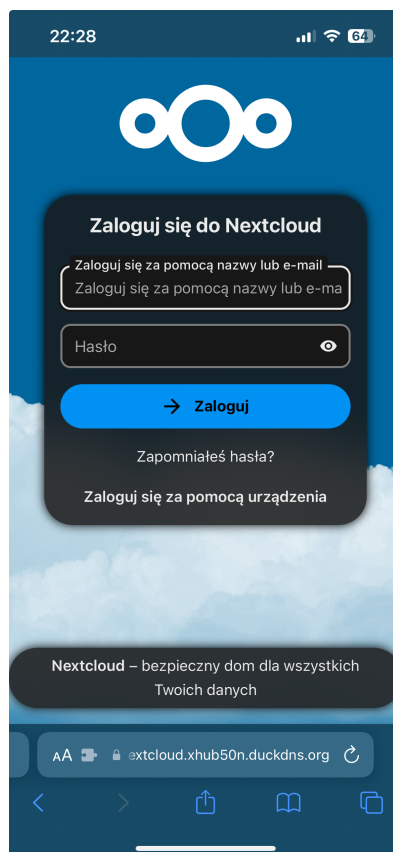
```
1 version: '2'
2
3 volumes:
4   nextcloud:
5   db:
6
7 services:
8   db:
9     image: mariadb:10.6
10    restart: always
11    command: --transaction-isolation=READ-COMMITTED --log-bin=binlog --
    binlog-format=ROW
```

```

12     volumes:
13         - /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

```

Próba uruchomienia aplikacji z poziomu przeglądarki w telefonie:



Rysunek 29: Nextcloud przeglądarka w telefonie

Aby móc korzystać z aplikacji w protokole HTTPS, musimy dodać kilka linii kodu w naszej aplikacji. W folderze lokalnym musimy edytować plik `/nextcloud-data/config/config.php` a następnie trzeba uruchomić ponownie terminal.

```
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 to oprogramowanie do blokowania reklam i ochrony prywatności w sieci, które działa jako lokalny serwer DNS. Filtruje reklamy, śledzenie i złośliwe oprogramowanie na poziomie sieciowym, zapewniając lepszą prywatność i przyspieszenie przeglądania internetu. AdGuard Home jest łatwy w konfiguracji i obsługuje wiele urządzeń w sieci domowej lub biurowej, umożliwiając centralne zarządzanie filtracją treści.

Tą aplikację wdrożymy poprzez polecenie docker

```
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
```


---

Aplikacja ta zostanie podłączona do sieci hosta, nie do sieci dockera oraz aplikacja będzie zawsze uruchamiana nawet kiedy terminal zostanie uruchomiony ponownie. Lub możemy skorzystać z pliku docker-compose

```
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
```

---

Tutaj też kontener będzie uruchamiany wraz ze startem systemu, docker zadba o to aby kontener działał zawsze.



### 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

Rysunek 30: Instalacja Adguard

Aby wykorzystać możliwości tej aplikacji, musimy w ustawieniach DHCP routera dodać adres IP naszego terminala aby wszystkie urządzenia w naszej sieci mogły korzystać z zalet Adguard.

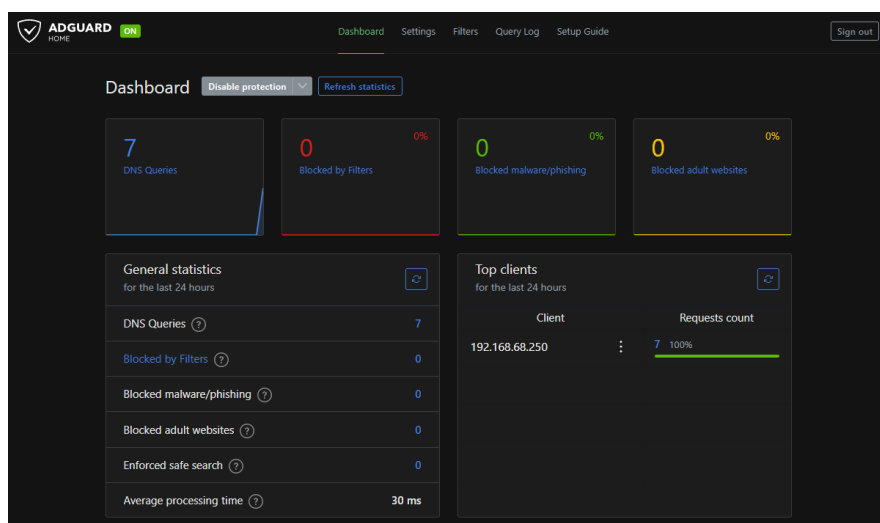
< Serwer DHCP Zapisz

16 przypisane(-ych) adresy(-ów) IP.

Początkowy adres IP	192.168.68.100
Końcowy adres IP	192.168.68.250
Brama domyślna	192.168.68.1
Preferowany DNS (opcjonalnie)	192.168.68.120
Alternatywny DNS (opcjonalnie)	

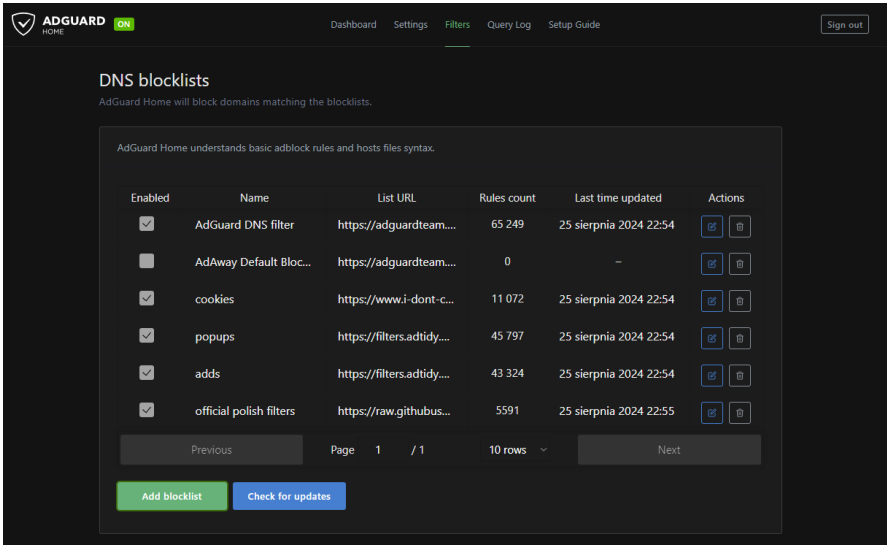
Rysunek 31: Zmiana serwera DNS w ustawieniach routera

Po instalacji aplikacji i zalogowaniu się do niej ukaze nam się dashboard w którym przedstawione są szczegółowe informacje o ruchu w naszej sieci i blokowanych stronach.



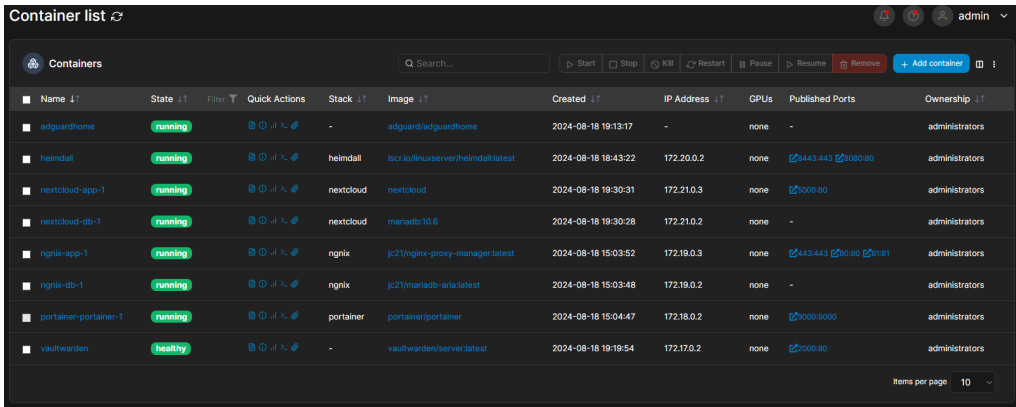
Rysunek 32: Dashboard

Kolejną funkcjonalnością jest dodawanie filtrów, które są publicznie dostępne w sieci i pozwalają na blokowanie reklam, wyskakujących okienek a także ciasteczka.



Rysunek 33: Filtry

A tak prezentuje się cały stack aplikacji, które zostały utworzone:



Rysunek 34: Portainer