

# Docker home lab

dokumentacja stworzenia własnego laboratorium opartego o kontenery Docker

Hubert Bojda

18 sierpnia 2024

# Wstęp

## Sprzęt

Celem projektu jest stworzenie własnego laboratorium, które możemy stworzyć na dowolnym urządzeniu, które wspiera technologię konteneryzacji a mianowicie Docker. Na początek polecam stworzenie wirtualnej maszyny z dystrybucją linuxa np. debian lub ubuntu. Później można zakupić sobie już dedykowany sprzęt pod to rozwiązanie, w tej dziedzinie dobrze sprawdzi się RaspberryPi lub terminale inaczej nazywane "cieńkimi klientami". W moim przypadku będzie to terminal HP T630. W porównaniu z RaspberryPi, terminale często da się kupić połowę taniej co zdecydowanie będzie zaletą, natomiast RaspberryPi wygrywa w tej walce jeśli chodzi o pobór energii, zaletą jest procesor wykonany w architekturze ARM.



Rysunek 1: HP T630

## Oprogramowanie

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

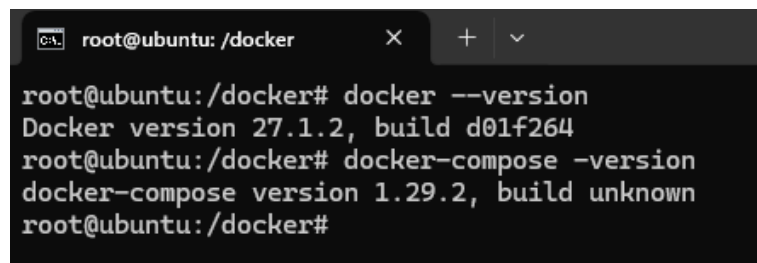
## Docker

- 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:

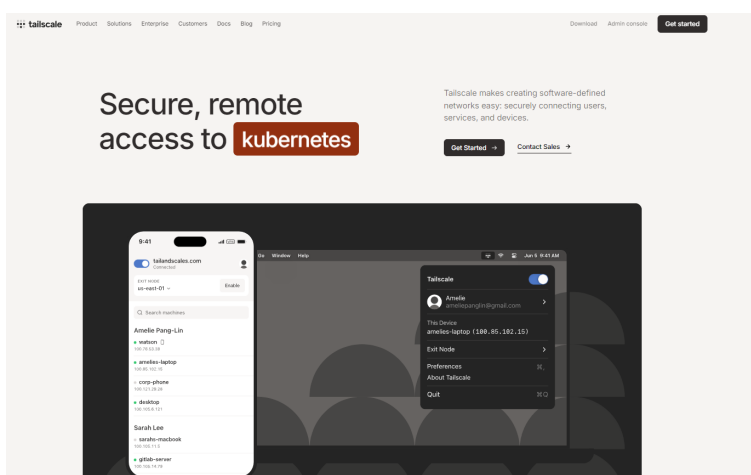


```
root@ubuntu:/docker# docker --version
Docker version 27.1.2, build d01f264
root@ubuntu:/docker# docker-compose -version
docker-compose version 1.29.2, build unknown
root@ubuntu:/docker#
```

Rysunek 2: Docker i Docker Compose

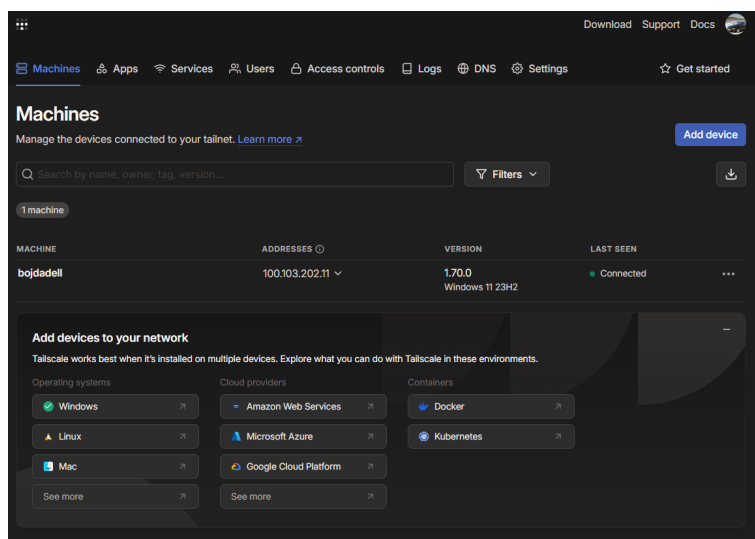
## Tailscale

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.



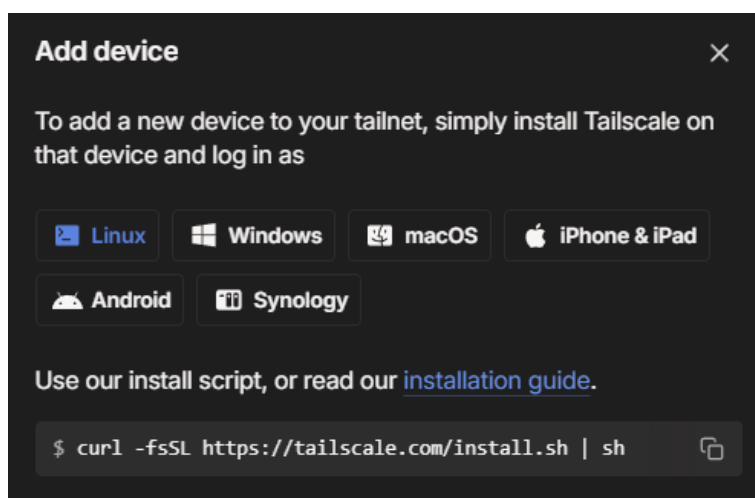
Rysunek 3: Tailscale

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

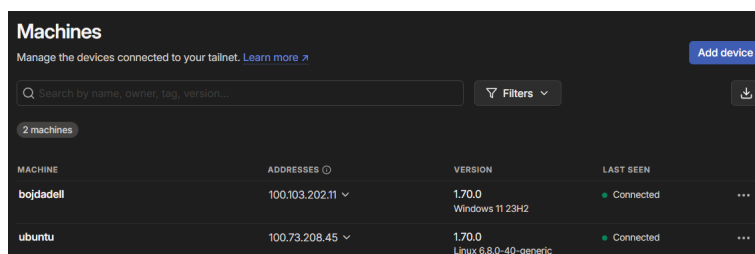
```
root@ubuntu:/home/hubert# tailscale up

To authenticate, visit:

https://login.tailscale.com/a/157581a50198e6
```

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



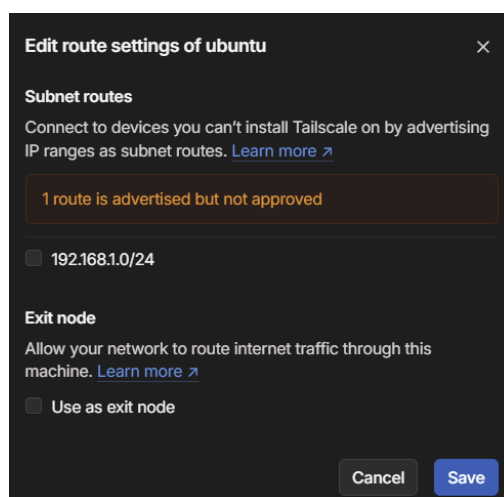
MACHINE	ADDRESSES	VERSION	LAST SEEN
bojdadell	100.103.202.11	1.70.0 Windows 11 23H2	Connected
ubuntu	100.73.208.45	1.70.0 Linux 6.8.0-40-generic	Connected

Rysunek 7: Nasze urządzenia

w naszej lokalnej sieci, np. kiedy chcemy zmienić ustawienia na routerze z dowolnego miejsca na świecie to dzięki tej opcji będziemy mogli to zrobić. Na naszym urządzeniu musimy wykonać następujące polecenie:

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

Następnie musimy tą opcję aktywować już z poziomu strony internetowej:



**Edit route settings of ubuntu**

**Subnet routes**  
Connect to devices you can't install Tailscale on by advertising IP ranges as subnet routes. [Learn more](#)

1 route is advertised but not approved

☐ 192.168.1.0/24

**Exit node**  
Allow your network to route internet traffic through this machine. [Learn more](#)

☐ Use as exit node

Cancel Save

Rysunek 8: Akceptacja

## 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 <https://hub.docker.com>.

### 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
```

---

### 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": "Zaq12wsx",
9       "port": 3306
10    }
11 }
```

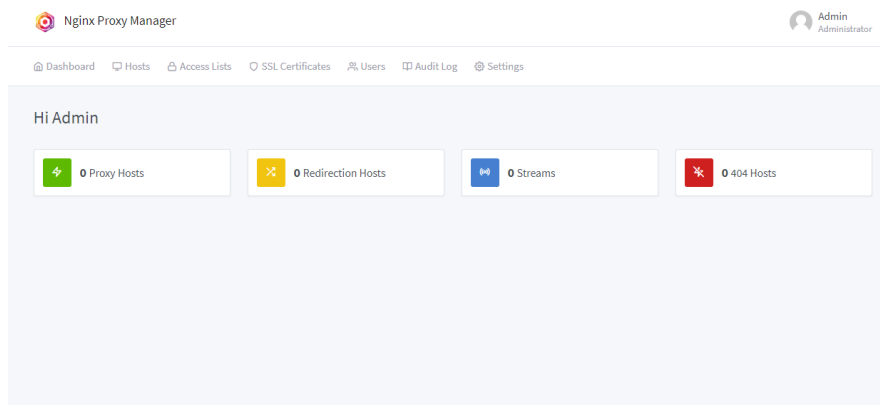
---

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'
```

---

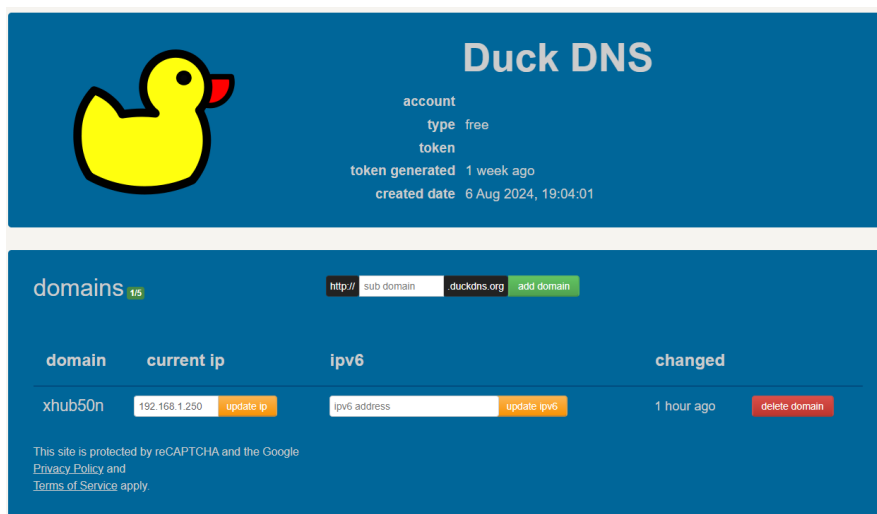
Po zainstalowaniu aplikacji musimy się do niej zalogować, domyślny login to: admin@example.com a hasło to changeme, po zalogowaniu jesteśmy proszeni o zmianę loginu i hasła na nasze własne, a poniżej prezentuje się główny widok aplikacji:



Rysunek 9: NGNIX

Teraz przedstawię konfigurację dostępu do aplikacji Portainer z poziomu NGNIX.

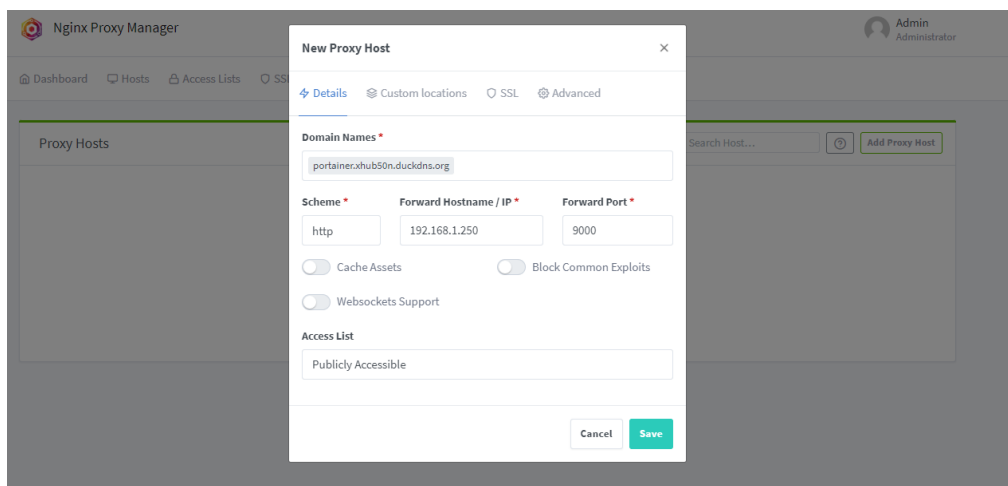
- W pierwszej kolejności co nam się przyda do dostęp poprzez domenę DNS, możemy w tym celu wykorzystać serwis DuckDNS, serwis pozwala na stworzenie darmowej dynamicznej domeny DNS, która zdecydowanie wystarczy nam na realizację tego projektu.



Rysunek 10: DuckDNS

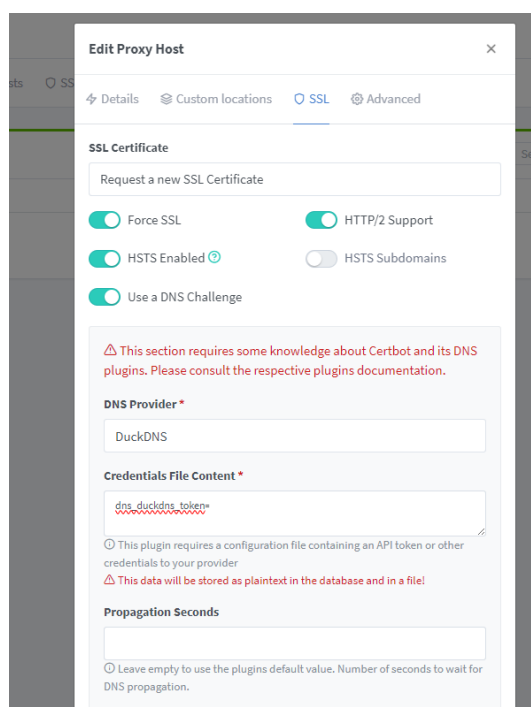
- Kolejnym krokiem będzie utworzenie nowego hosta proxy w NGINX Proxy Manager. Gdy mamy już naszą domenę, będziemy mogli również skorzystać z możliwości tworzenia certyfikatów SSL za pomocą usługi Let's Encrypt. W nazwie domeny wpisujemy naszą domenę stworzoną w DuckDNS oraz unikalną nazwę dla danej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





Rysunek 11: Dodawanie host-a

- Ostatnim krokiem będzie utworzenie certyfikatu SSL w celu szyfrowania połączenia, niektóre aplikacje takie jak np. Vaultwarden wymagają tego aby aplikacja działała na szyfrowanym połączeniu. W miejscu 'Credentials File Content' podajemy token, który wygenerował nam DuckDNS oraz wymuszamy aby połączenie było zawsze po HTTPS.



Rysunek 12: Tworzenie certyfikatu

Tworzenie wpisów dla każdej aplikacji, którą stworzymy wygląda w taki sam sposób.

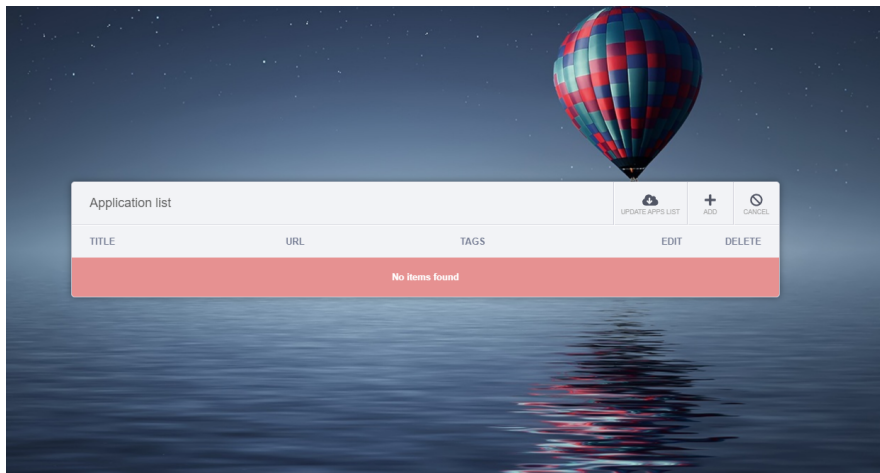
## Heimdall

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.

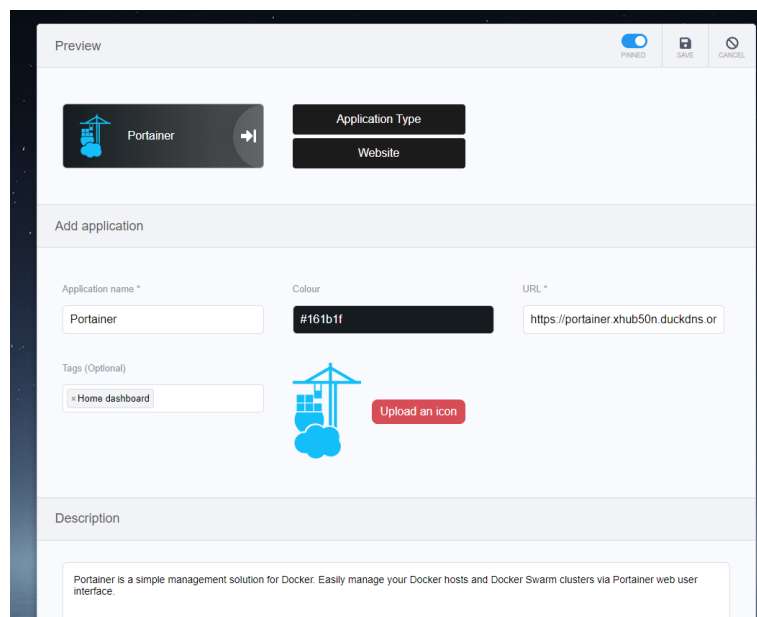
Zawartość pliku docker-compose.yaml

```
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.

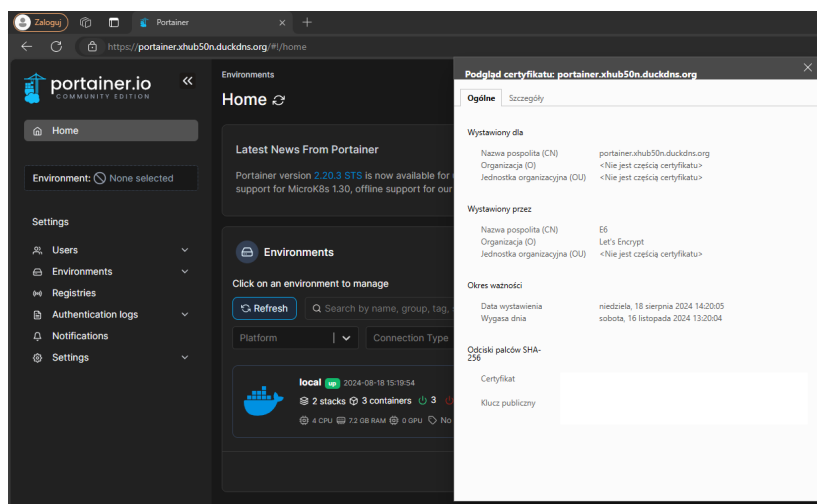


Rysunek 13: Heimdall



Rysunek 14: Dodawanie strony

Jak widać podczas tworzenia wpisu można dostrzec link zawierający HTTPS.

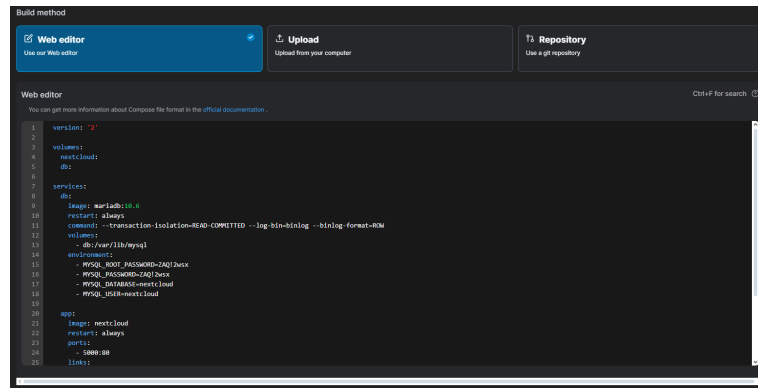


Rysunek 15: Portainer HTTPS

## Nextcloud

Nextcloud to platforma do zarządzania plikami i współpracy online, która umożliwia tworzenie własnej chmury prywatnej. Pozwala na bezpieczne przechowywanie, synchronizowanie i udostępnianie plików, dokumentów, kalendarzy, kontaktów i innych danych. Nextcloud oferuje również funkcje komunikacyjne, takie jak czat, videokonferencje i zarządzanie zadaniami, co czyni go kompleksowym rozwiązaniem dla pracy zespołowej i prywatnego zarządzania danymi.

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

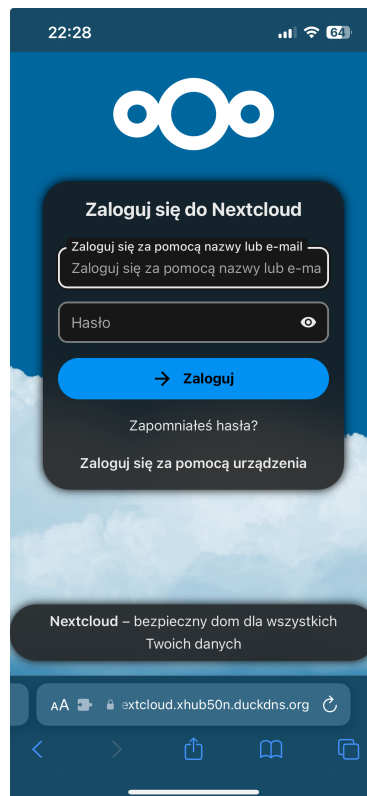


Rysunek 16: Nextcloud

### 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      - db:/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:
26       - db
27     volumes:
28       - nextcloud:/var/www/html
29     environment:
30       - MYSQL_PASSWORD=
31       - MYSQL_DATABASE=nextcloud
32       - MYSQL_USER=nextcloud
33       - MYSQL_HOST=db
```

Próba uruchomienia aplikacji z poziomu telefonu:



Rysunek 17: Nextcloud przeglądarka w telefonie

## AdGuardHome

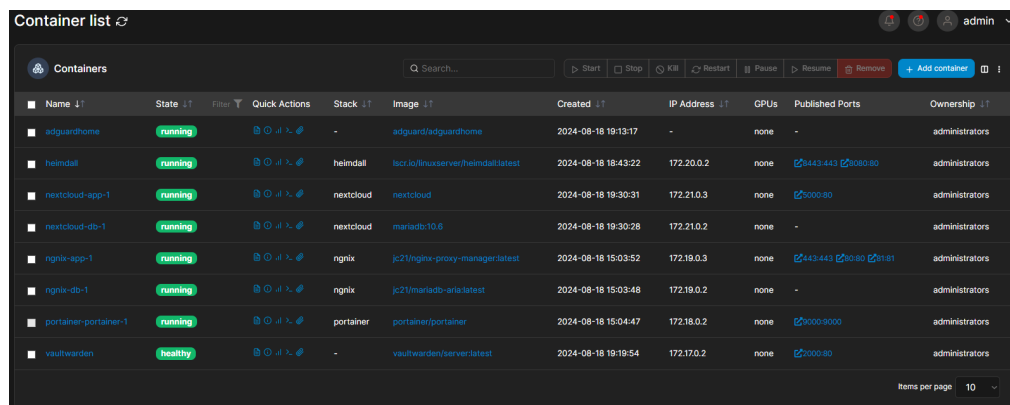
AdGuard Home to oprogramowanie do blokowania reklam i ochrony prywatności w sieci, które działa jako lokalny serwer DNS. Filtruje reklamy, śledzenie i złośliwe oprogramowanie na poziomie sieciowym, zapewniając lepszą prywatność i przyspieszenie przeglądania internetu. AdGuard Home jest łatwy w konfiguracji 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.

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



Name	State	Stack	Image	Created	IP Address	GPUs	Published Ports	Ownership
adguardhome	running	-	adguard/adguardhome	2024-08-18 19:13:17	-	none	-	administrators
heimdall	running	heimdall	laci.io/hauser/heimdall:latest	2024-08-18 18:43:22	172.20.0.2	none	443:443 80:80	administrators
nextcloud-app-1	running	nextcloud	nextcloud	2024-08-18 19:30:31	172.21.0.3	none	9000:80	administrators
nextcloud-db-1	running	nextcloud	mysql:10.6	2024-08-18 19:30:28	172.21.0.2	none	-	administrators
nginx-app-1	running	nginx	j21/nginx-proxy-manager:latest	2024-08-18 15:03:52	172.19.0.3	none	443:443 80:80 1181:1181	administrators
nginx-db-1	running	nginx	j21/mariadb-aria:latest	2024-08-18 15:03:48	172.19.0.2	none	-	administrators
portainer-portainer-1	running	portainer	portainer/portainer	2024-08-18 15:04:47	172.18.0.2	none	9000:9000	administrators
vaultwarden	healthy	-	vaultwarden/server:latest	2024-08-18 19:19:54	172.17.0.2	none	2000:80	administrators

Rysunek 18: Nextcloud przeglądarka w telefonie