Instalacja Vaultwardena w Dockerze

Typ: procedura **Dotyczy: Vaultwarden 1.37.3, obraz vaultwarden/server:1.37.3**

Czas: ~30 min

Efekt: działający kontener Vaultwarden wystawiony przez Nginksa pod `https://hasla.zielona.example`, z wyłączoną otwartą rejestracją, pocztą przez MailCow i zahaszowanym tokenem administratora, gotowy do pierwszej konfiguracji przez przeglądarkę.

Skróty użyte na tej stronie (9)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
HTTP	Hypertext Transfer Protocol: protokół stron WWW
HTTPS	HTTP zabezpieczony szyfrowaniem TLS
SMTP	Simple Mail Transfer Protocol: protokół wysyłania poczty
SQLite	mała baza danych zapisana w jednym pliku; Home Assistant trzyma w niej historię w pliku home-assistant_v2.db
TLS	Transport Layer Security: szyfrowanie połączeń, np. w HTTPS
WebSocket	stałe, dwukierunkowe połączenie przeglądarki z serwerem, przez które Home Assistant na żywo odświeża panele; serwer pośredniczący musi przekazywać nagłówki Upgrade i Connection
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

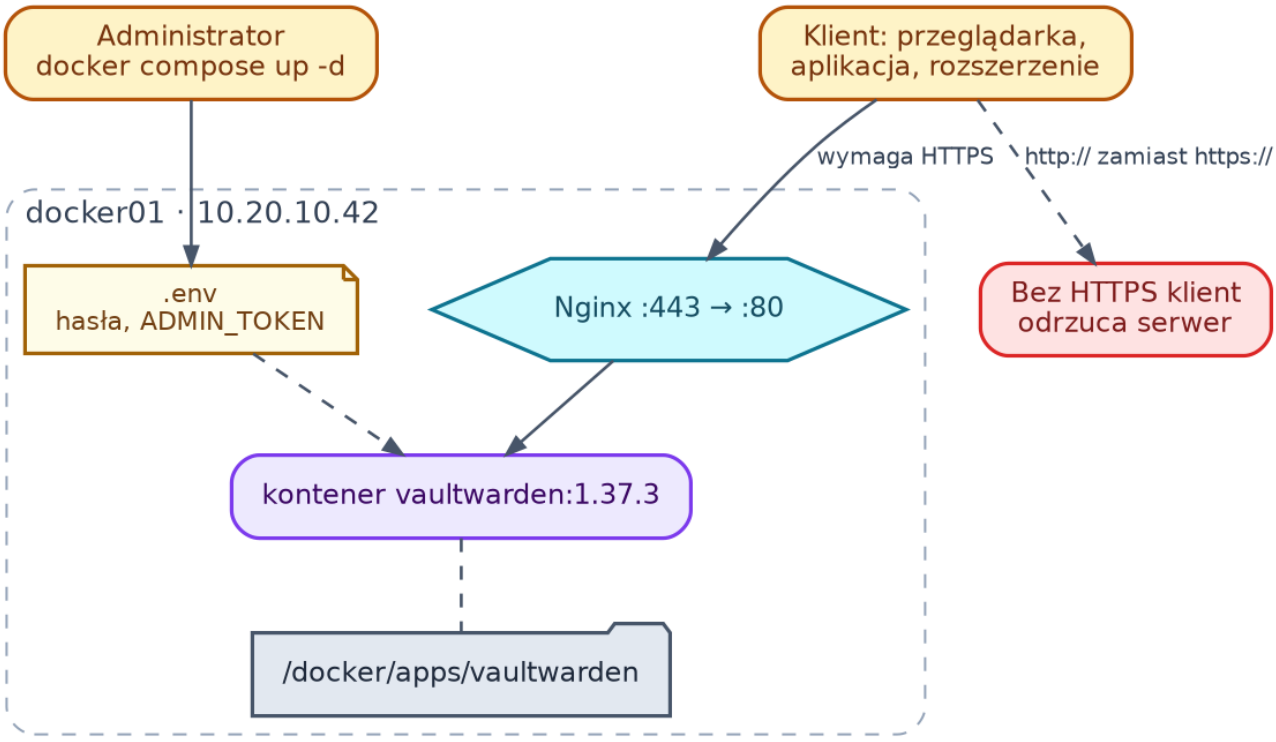
Zanim zaczniemy

Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo albo root, na maszynie z zainstalowanym Dockerem i wtyczką Compose. Vaultwarden, w odróżnieniu od Gitei, nie potrzebuje osobnej bazy danych: cały sejf trzyma w jednym pliku SQLite na własnym wolumenie, więc kontener w Zielonej jest tylko jeden.

> W naszym środowisku dane aplikacji trzymamy w `/docker/apps/<nazwa>`, a plik Compose w `/docker/compose/<nazwa>/compose.yaml`. To konwencja, nie wymóg Dockera: można ją zmienić, byle w całej firmie była jedna. Katalog `/docker` to osobny wolumen `vgDocker/lvDocker`, żeby pełny dysk aplikacji nie zatrzymał systemu.

Wymaganie	Uwaga
Docker Engine i wtyczka Compose	sprawdzamy `docker compose version`
Nginx na `docker01` jako reverse proxy	ten sam Nginx, który stoi już przed Giteą i innymi aplikacjami
Certyfikat TLS dla `hasla.zielona.example`	z CA Zielonej albo Let's Encrypt, patrz Reverse proxy i równoważenie obciążenia
Osiągalny MailCow po SMTP z `docker01`	do wysyłki zaproszeń, resetu hasła i powiadomień

Uważaj: obraz zawsze przypinamy do konkretnej wersji (`1.37.3`, sprawdzonej w rejestrze Docker Hub 28 września 2026 roku jako najnowsza stabilna), nigdy do `latest`. Aktualizacja to świadoma zmiana numeru w `.env`, opisana na stronie Kopia zapasowa i aktualizacja Vaultwardena, nie przypadkowe pobranie nowszej wersji przy kolejnym `docker compose pull`.



Instalacja Vaultwardena w Dockerze: kontener, wolumen danych, Nginx i wymóg HTTPS dla klientów (kliknięcie otwiera pełny rozmiar)

Dlaczego HTTPS jest obowiązkowy

Aplikacje kliencie Bitwardena i Vaultwardena, czyli rozszerzenie przeglądarki, aplikacja na telefon i aplikacja na komputer, szyfrują i odszyfrowują sejf lokalnie, w przeglądarce albo w silniku aplikacji, mechanizmem Web Crypto API. Przeglądarki udostępniają ten mechanizm tylko w tak zwanym bezpiecznym kontekście: na stronie otwartej przez `https://` albo na `localhost`. Otwarcie Vaultwardena po zwykłym `http://` z innego adresu niż `localhost` kończy się tym, że klient w ogóle nie odblokuje sejfu, niezależnie od tego, jak poprawnie skonfigurowany jest sam serwer. To nie jest ustawienie do włączenia w Vaultwardenie: to ograniczenie przeglądarki, które omija tylko prawdziwy certyfikat TLS przed aplikacją.

Przebieg

1. Katalogi na dane, zgodnie z konwencją ścieżek

```
sudo mkdir -p /docker/apps/vaultwarden
sudo mkdir -p /docker/compose/vaultwarden
```

2. Wygenerowanie zahaszowanego tokenu administratora: vaultwarden hash

```
docker run --rm -it vaultwarden/server:1.37.3 /vaultwarden hash
```

Komenda pyta o hasło do panelu `/admin` dwukrotnie i wypisuje gotowy do wklejenia hash w formacie Argon2 (`$argon2id$...`).

Przykład:

```
Password:
Confirm password:


ADMIN_TOKEN='$argon2id$v=19$m=65540,t=3,p=4$eKoW...'
```

Uważaj: wklejamy do `.env` cały ciąg zaczynający się od `$argon2id$`, nie hasło, które podaliśmy w kroku wyżej. Vaultwarden porówna przy logowaniu hasz wpisanego hasła z tym ciągiem, więc trzymanie w `ADMIN_TOKEN` samego hasła jawnym tekstem, choć wciąż dopuszczalne, nie chroni tokenu w przypadku wycieku pliku `.env`.

3. Plik zmiennych: /docker/compose/vaultwarden/.env

```
VAULTWARDEN_ADMIN_TOKEN='$argon2id$v=19$m=65540,t=3,p=4$eKoW...'
VAULTWARDEN_SMTP_PASSWORD=HasloSkrzynkiVaultwarden
```

4. Plik Compose: /docker/compose/vaultwarden/compose.yaml

```
services:
  vaultwarden:
    image: vaultwarden/server:1.37.3
    container_name: vaultwarden
    restart: always
    environment:
      - DOMAIN=https://hasla.zielona.example
      - SIGNUPS_ALLOWED=false
      - ADMIN_TOKEN=${VAULTWARDEN_ADMIN_TOKEN}
      - SMTP_HOST=poczta.zielona.example
      - SMTP_PORT=587
      - SMTP_SECURITY=starttls
      - SMTP_FROM=vaultwarden@zielona.example
      - SMTP_FROM_NAME=Vaultwarden Zielona
      - SMTP_USERNAME=vaultwarden@zielona.example
      - SMTP_PASSWORD=${VAULTWARDEN_SMTP_PASSWORD}
    volumes:
      - /docker/apps/vaultwarden:/data
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
```

```
    ports:
      - "8080:80"
```

`DOMAIN` to nie tylko kosmetyka: Vaultwarden wpisuje ten adres do linków w mailach z zaproszeniami i w konfiguracji dwuetapowego logowania metodą WebAuthn, więc musi zgadzać się dokładnie z adresem, pod jakim ludzie faktycznie otwierają sejf. `SIGNUPS_ALLOWED=false` wyłącza samodzielne zakładanie kont z ekranu logowania: w Zielonej konta zakłada administrator zaproszeniem z organizacji, opisanym na stronie Pierwsza konfiguracja: panel admina, organizacja i zaproszenia.

Pierwsze uruchomienie:

```
cd /docker/compose/vaultwarden
docker compose up -d
```

5. Reverse proxy Nginksa przed Vaultwardenem

Vaultwarden sam w sobie serwuje ruch HTTP na porcie 80 wewnątrz kontenera, bez obsługi TLS. Nowsze wersje Vaultwardena, w tym `1.37.3`, prowadzą powiadomienia na żywo tym samym portem co resztę ruchu, pod ścieżką `/notifications/hub`, więc blok proxy musi osobno przepuszczać nagłówki `Upgrade` i `Connection` właśnie dla tej ścieżki.

```
server {
    listen 443 ssl;
    server_name hasla.zielona.example;

    ssl_certificate     /etc/nginx/ssl/hasla.zielona.example.crt;
    ssl_certificate_key /etc/nginx/ssl/hasla.zielona.example.key;

    location /notifications/hub {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }

    location / {
        proxy_pass http://127.0.0.1:8080;
```

```
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

server {
    listen 80;
    server_name hasla.zielona.example;
    return 301 https://$host$request_uri;
}
```

Logi pierwszego startu

```
docker compose logs -f vaultwarden
```

Przykład:

```
vaultwarden  | [2026-09-28 10:02:14.221][start][INFO] Rocket has launched from
http://0.0.0.0:80
vaultwarden  | [2026-09-28 10:02:14.225][vaultwarden::db][INFO] Waiting for database to be
available
vaultwarden  | [2026-09-28 10:02:14.301][vaultwarden::db][INFO] Database is up to date
```

Weryfikacja

Sprawdzamy, że kontener wstał i widzi swój wolumen:

```
docker compose ps
```

Przykład:

```
NAME          IMAGE                          STATUS
vaultwarden   vaultwarden/server:1.37.3    Up 2 minutes
```

Sprawdzamy, że interfejs WWW odpowiada, zanim otworzymy go w przeglądarce:

```
curl -I http://localhost:8080
```

Przykład:

```
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
```

Sprawdzamy, że Nginx wystawia tę samą stronę po HTTPS na docelowym adresie:

```
curl -I https://hasla.zielona.example
```

Przykład:

```
HTTP/2 200
server: nginx
content-type: text/html; charset=utf-8
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: strona logowania otwarta pod `https://hasla.zielona.example`, z widoczną kłódką certyfikatu w pasku adresu przeglądarki i logo Vaultwardena.

Wycofanie zmian

Zatrzymanie i usunięcie kontenera, bez kasowania danych z wolumenu:

```
docker compose down
```

Pełne usunięcie razem z danymi, tylko gdy naprawdę chcemy zacząć od zera:

```
docker compose down
sudo rm -rf /docker/apps/vaultwarden
```

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa

Klient (rozszerzenie, aplikacja) nie odblokowuje sejfu mimo poprawnego hasła	dostęp po `http://`, nie po `https://`	otwieramy adres wyłącznie przez Nginksa po HTTPS, patrz sekcja „Dlaczego HTTPS jest obowiązkowy"
`curl -I http://localhost:8080` zwraca `Connection refused`	kontener jeszcze się uruchamia albo port zajęty przez inną usługę	`docker compose logs vaultwarden`, `ss -tlnp \`
Powiadomienia w interfejsie nie odświeżają się na żywo	Nginx nie przekazuje nagłówków `Upgrade` / `Connection` dla `/notifications/hub`	uzupełniamy blok `location /notifications/hub` z kroku 5

Zaproszenia mailowe nie wychodzą	złe dane SMTP w `compose.yaml` albo MailCow nieosiągalny z `docker01`	`docker compose logs vaultwarden \

Powiązane: Pierwsza konfiguracja: panel admina, organizacja i zaproszenia · Reverse proxy i równoważenie obciążenia · Konwencja Dockera w Zielonej

Źródła: github.com/dani-garcia/vaultwarden, wiki „Using Docker Compose" · github.com/dani-garcia/vaultwarden, wiki „Enabling admin page" · github.com/dani-garcia/vaultwarden, wiki „SMTP Configuration" · github.com/dani-garcia/vaultwarden, wiki „Enabling WebSocket notifications" · hub.docker.com/r/vaultwarden/server, sprawdzone 28 września 2026

Revision #23
Created 2026-09-28 16:58:00 UTC by Tomasz Konieczny
Updated 2026-10-26 14:18:27 UTC by Tomasz Konieczny