NetBox

Opowieść, instalacja w Dockerze, logowanie z AD, kopia i aktualizacja.

- NetBox: źródło prawdy o sieci i serwerowni
- Instalacja NetBoksa w Dockerze
- NetBox: logowanie z AD, kopia i aktualizacja

NetBox: źródło prawdy o sieci i serwerowni

Poziom: podstawy **Dotyczy: NetBox, projekt netbox-community/netbox-docker**

Czas: ~15 min

Ta strona otwiera w tej książce osobny wątek poświęcony NetBoksowi: skąd się wziął, kto go dziś rozwija i dlaczego Zielona w ogóle potrzebuje osobnej aplikacji, żeby wiedzieć, co stoi we własnej serwerowni.

Skróty użyte na tej stronie (9)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
DCIM	Data Center Infrastructure Management: dokumentacja tego, co fizycznie stoi w serwerowni, czyli szafy, urządzenia i okablowanie
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
IPAM	IP Address Management: dokumentacja tego, który adres IP jest zajęty przez co, żeby dwa urządzenia nie dostały tego samego adresu
IT	Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy
MAC	Media Access Control: unikalny adres fizyczny karty sieciowej, przypisany fabrycznie
REST	sposób budowania API przez zwykłe żądania HTTP (GET, POST) zamiast osobnego protokołu
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
VLAN	Virtual LAN: wydzielona logicznie sieć na tym samym sprzęcie

Skąd się wzięło

> **Pojęcie: NetBox.** Aplikacja klasy DCIM i IPAM, czyli jedno, autorytatywne miejsce zapisu tego, co fizycznie i logicznie istnieje w infrastrukturze: które urządzenie stoi w której szafie, jaki adres IP należy do którego interfejsu, który VLAN obowiązuje w którym segmencie sieci.

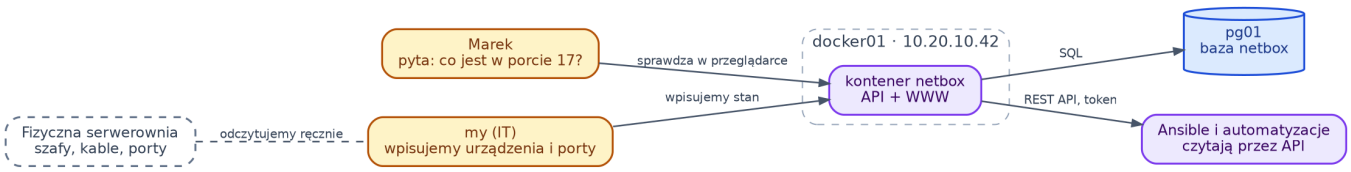
NetBox powstał w 2016 roku w firmie DigitalOcean, dostawcy chmury publicznej, który w tamtym czasie gwałtownie rozbudowywał własne centra danych na całym świecie. Autorem był Jeremy Stretch, inżynier sieciowy DigitalOcean, znany wcześniej w branży jako twórca popularnego serwisu edukacyjnego o sieciach komputerowych PacketLife.net. Zespół DigitalOcean potrzebował narzędzia, które łączyłoby dokumentację adresacji IP z dokumentacją fizycznego okablowania i rozmieszczenia sprzętu w szafach, bo istniejące wtedy narzędzia tej klasy traktowały te dwie rzeczy osobno, zmuszając administratorów do ręcznego utrzymywania dwóch niezależnych, rozjeżdżających się z czasem źródeł prawdy. DigitalOcean opublikował NetBox jako projekt otwartoźródłowy jeszcze w tym samym roku, zamiast trzymać go wyłącznie na własny użytek.

Przez kolejne lata NetBox rozwijał się jako projekt społecznościowy, z Jeremym Stretchem jako głównym opiekunem, zyskując status de facto standardu w branży dostawców usług sieciowych i operatorów centrów danych. W 2023 roku, żeby zapewnić projektowi stabilne, pełnoetatowe wsparcie rozwojowe, powstała spółka NetBox Labs, założona przez twórców NetBoksa przy wsparciu funduszy inwestycyjnych. NetBox Labs oferuje dziś płatne wersje NetBox Cloud i NetBox Enterprise z dodatkowym wsparciem i funkcjami dla dużych organizacji, ale sam NetBox w wersji community, ten, z którego korzysta Zielona, pozostaje wolnym oprogramowaniem na licencji Apache 2.0, rozwijanym w pełni jawnie na GitHubie.

Instalację NetBoksa w kontenerach ułatwia osobny, również otwartoźródłowy projekt `netbox-community/netbox-docker`, utrzymywany przez społeczność równolegle do samego NetBoksa: to on dostarcza gotowy obraz kontenera i przykładowy komplet plików Compose, z którego korzysta Zielona, opisany na kolejnej stronie tej książki.

Jak to działa w jednym obrazku

NetBox sam w sobie nie łączy się z żadnym przełącznikiem ani serwerem: to zespół IT ręcznie wpisuje do niego, co fizycznie stoi w serwerowni, a NetBox zapisuje ten stan w swojej bazie i udostępnia go przez API innym narzędziom, np. Ansible do automatyzacji.



NetBox jako źródło prawdy: zespół IT wpisuje stan serwerowni, inne narzędzia go odczytują (kliknięcie otwiera pełny rozmiar)

Dlaczego Zielona

Z ŻYCIA WZIĘTE

Wtorek, 9:00. Marek dzwoni z magazynu w Swarzędzu: nowy przełącznik od dostawcy trzeba podłączyć do sieci serwerowej, ale nikt nie pamięta, który port starego przełącznika w porcie 17 obsługuje który skaner. Tomek loguje się przez SSH na przełącznik, przegląda tabelę adresów MAC port po porcie, dzwoni do Marka z pytaniem „czy teraz coś przestało działać", i powtarza to jeszcze dwa razy, zanim trafia na właściwy port.

Taka sytuacja w Zielonej powtarzała się przy każdej zmianie okablowania: jedynym źródłem prawdy o tym, co jest podłączone gdzie, była pamięć administratora albo, w najlepszym razie, rozsypane notatki w OneNote sprzed lat, nigdy w pełni nieaktualne. Przy 180 osobach, dwóch lokalizacjach i infrastrukturze rosnącej od 1998 roku, żaden pojedynczy człowiek nie jest już w stanie pamiętać tego wszystkiego na bieżąco.

NetBox rozwiązuje to jednym miejscem zapisu: każda szafa, każde urządzenie i każdy port mają swój wpis, wprowadzany raz przy instalacji sprzętu, a nie odtwarzany za każdym razem z pamięci. Dodatkowa korzyść, której Zielona jeszcze w pełni nie wykorzystuje, ale do której się przygotowuje: te same dane, dostępne przez REST API opisane na istniejącej stronie NetBox, mogą zasilać inwentarz Ansible zamiast ręcznie utrzymywanego pliku `inventory.ini`, więc dodanie serwera do NetBoksa czyni go automatycznie widocznym dla automatyzacji.

Mapa tej książki

Ta część książki „Tożsamość i repozytoria" prowadzi od źródła danej wiedzy o NetBoksie do samodzielnej, działającej instalacji podłączonej do domeny Zielonej.

Strona	Na jakie pytanie odpowiada
NetBox: źródło prawdy o sieci i serwerowni	Skąd wziął się NetBox i dlaczego Zielona go potrzebuje
NetBox	Jaki jest model danych NetBoksa i jak wygląda praca na co dzień
Instalacja NetBoksa w Dockerze	Jak postawić NetBoksa od zera projektem netbox-docker
NetBox: logowanie z AD, kopia i aktualizacja	Jak podłączyć logowanie domenowe i bezpiecznie utrzymywać instalację

Powiązane: FreeIPA · Gitea · Active Directory: konta, grupy, jednostki organizacyjne

Źródła: github.com/netbox-community/netbox, plik AUTHORS i historia projektu · netboxlabs.com, strona „About" · github.com/netbox-community/netbox-docker

Instalacja NetBoksa w Dockerze

Typ: procedura

Dotyczy: netbox-docker v5.1.0, obraz netboxcommunity/netbox:v4.7-5.1.0 (sprawdzono w dokumentacji projektu 28.09.2026)

Czas: ~45 min

Efekt: działający NetBox na `docker01`, dostępny pod `netbox.zielona.example`, z kontem administratora i pierwszymi wpisanymi danymi: dwiema lokalizacjami Zielonej i dwoma urządzeniami.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
DCIM	Data Center Infrastructure Management: dokumentacja tego, co fizycznie stoi w serwerowni, czyli szafy, urządzenia i okablowanie
TLS	Transport Layer Security: szyfrowanie połączeń, np. w HTTPS

Zanim zaczniemy

> **Z ŻYCIA WZIĘTE**
>
> Tomek zakłada w NetBoksie pierwszą szafę i pierwsze urządzenie, ale najpierw musi w ogóle postawić samą aplikację. Znajduje w internecie dziesiątki poradników z różnymi wersjami pliku Compose, każdy z inną nazwą usługi bazy danych. Sięga po oficjalny projekt `netbox-community/netbox-docker`, zamiast składać własny plik od zera z fragmentów znalezionych w sieci.

Instalację NetBoksa opieramy na oficjalnym projekcie `netbox-community/netbox-docker`, osobnym od samego NetBoksa, ale utrzymywanym przez tę samą społeczność. Projekt dostarcza gotowy `docker-compose.yml` z przypiętym obrazem, a własne odstępstwa wprowadzamy przez plik `docker-compose.override.yml`, zamiast edytować plik bazowy wprost: dzięki temu aktualizacja do nowszej

wersji repozytorium nie nadpisuje naszych zmian.

> 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.

W odróżnieniu od MailCow, opisanego na stronie Instalacja MailCow na docker01 od zera, repozytorium `netbox-docker` trzyma tylko sam plik Compose i konfigurację, bez katalogu z danymi wymieszanego w środku: dane NetBoksa (baza, media, raporty) domyślnie trafiają do wolumenów zarządzanych przez Dockera, a my przez `docker-compose.override.yml` przekierowujemy je pod `/docker/apps/netbox`, zgodnie ze zwykłą konwencją Zielonej.

Wymaganie	Uwaga
Docker Engine i wtyczka Compose	sprawdzamy `docker compose version`
Wolny port `8000` na hoście (tylko lokalnie)	NetBox stoi za Nginksem, patrz krok reverse proxy
Wariant b: osiągalny `pg01` na porcie `5432`	baza tworzona ręcznie przed pierwszym startem
`SECRET_KEY` wygenerowany osobno dla tej instalacji	nigdy nie kopiujemy klucza z dokumentacji ani z innego środowiska

Uważaj: obraz zawsze przypinamy do konkretnej wersji (`v4.7-5.1.0`), nigdy do `latest`. Aktualizacja to świadoma zmiana tagu w `docker-compose.override.yml`, opisana szerzej na stronie NetBox: logowanie z AD, kopia i aktualizacja.

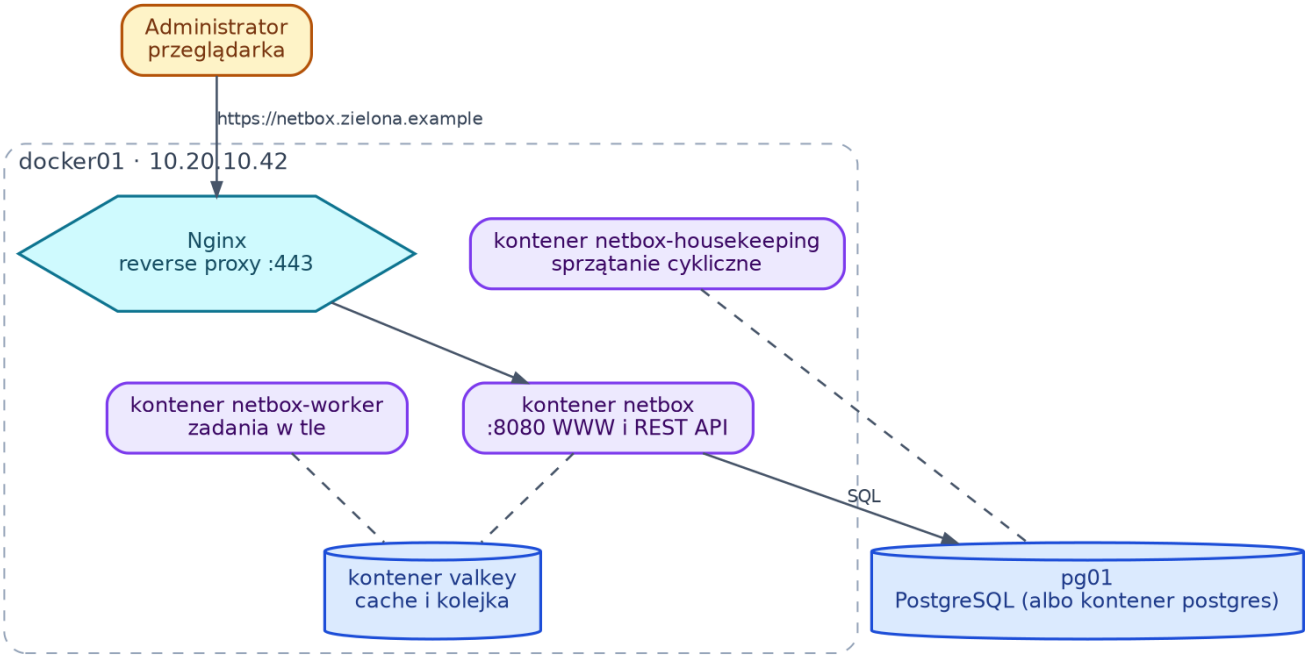
Przebieg

Klonowanie repozytorium: git clone

```
sudo mkdir -p /docker/compose
cd /docker/compose
sudo git clone -b release https://github.com/netbox-community/netbox-docker.git netbox
cd netbox
```

Gałąź `release` śledzi ostatnie stabilne wydanie projektu, zgodnie z zaleceniem samego `netbox-docker`.

Podstawowa architektura instalacji, wspólna dla obu wariantów bazy danych opisanych niżej, wygląda tak:



Architektura NetBoksa w Dockerze: Nginx, kontenery netbox, netbox-worker, netbox-housekeeping, valkey i baza PostgreSQL (kliknięcie otwiera pełny rozmiar)

Wariant a: laboratorium, baza w kontenerze PostgreSQL

Zakładamy katalogi na dane, zgodnie z konwencją ścieżek:

```
sudo mkdir -p /docker/apps/netbox/postgres /docker/apps/netbox/media
/docker/apps/netbox/reports /docker/apps/netbox/scripts
cp docker-compose.override.yml.example docker-compose.override.yml
```

Wygenerowanie klucza `SECRET_KEY`:

```
openssl rand -base64 45
```

Plik `docker-compose.override.yml`, wersja skrócona do zmian względem pliku bazowego:

```
services:
  netbox:
    image: docker.io/netboxcommunity/netbox:v4.7-5.1.0
```

```
    ports:
      - "127.0.0.1:8000:8080"
    environment:
      SECRET_KEY: "WygenerowanyKluczSecretKey"
      SUPERUSER_NAME: admin
      SUPERUSER_EMAIL: it@zielona.example
      SUPERUSER_PASSWORD: "HasloAdministratora"
    volumes:
      - /docker/apps/netbox/media:/opt/netbox/netbox/media:z
      - /docker/apps/netbox/reports:/opt/netbox/netbox/reports:z
      - /docker/apps/netbox/scripts:/opt/netbox/netbox/scripts:z
  netbox-worker:
    image: docker.io/netboxcommunity/netbox:v4.7-5.1.0
  netbox-housekeeping:
    image: docker.io/netboxcommunity/netbox:v4.7-5.1.0
  postgres:
    image: docker.io/postgres:18-alpine
    volumes:
      - /docker/apps/netbox/postgres:/var/lib/postgresql/data
```

(gdzie usługi `netbox-worker` i `netbox-housekeeping` muszą wskazywać ten sam obraz co `netbox`, inaczej mieszają się wersje schematu bazy danych między kontenerami tej samej instalacji)

Wariant b: produkcja Zielonej, baza na zewnętrznym pg01

Zamiast usługi `postgres` w pliku override wskazujemy istniejący serwer `pg01`. Zanim uruchomimy kontenery, zakładamy bazę i użytkownika:

```
psql -h pg01 -U postgres
```

```
CREATE DATABASE netbox;
CREATE USER netbox WITH ENCRYPTED PASSWORD 'HasloDoBazy';
GRANT ALL PRIVILEGES ON DATABASE netbox TO netbox;
ALTER DATABASE netbox OWNER TO netbox;
```

Fragment `docker-compose.override.yml` różniący się od wariantu a:

```
services:
  netbox:
    image: docker.io/netboxcommunity/netbox:v4.7-5.1.0
    environment:
      DB_HOST: pg01
      DB_NAME: netbox
      DB_USER: netbox
      DB_PASSWORD: "HasloDoBazy"
    ports:
      - "127.0.0.1:8000:8080"
    volumes:
      - /docker/apps/netbox/media:/opt/netbox/netbox/media:z
      - /docker/apps/netbox/reports:/opt/netbox/netbox/reports:z
      - /docker/apps/netbox/scripts:/opt/netbox/netbox/scripts:z
```

Usługę `postgres` z pliku bazowego wyłączamy, dopisując w override `postgres: profiles: ["disabled"]`, żeby `docker compose up` jej nie uruchamiał.

Pierwsze uruchomienie: docker compose

```
cd /docker/compose/netbox
sudo docker compose pull
sudo docker compose up -d
```

Reverse proxy: netbox.zielona.example

NetBox nasłuchuje wyłącznie na `127.0.0.1:8000`, więc wystawiamy go na zewnątrz dopiero przez Nginksa, zgodnie ze stroną Reverse proxy i równoważenie obciążenia, z certyfikatem TLS dla `netbox.zielona.example`.

Pierwsze dane: lokalizacje, szafa, urządzenia

Po zalogowaniu kontem administratora wprowadzamy dane w kolejności zgodnej z hierarchią modelu danych NetBoksa, opisaną na stronie NetBox: najpierw lokalizacje, potem szafę, na końcu urządzenia.

```
Organization → Sites → Add: Pamiątkowo (centrala)
Organization → Sites → Add: Swarzędz (magazyn)
Organization → Locations → Add: Serwerownia (site: Pamiątkowo)
DCIM → Racks → Add: R1 (site: Pamiątkowo, location: Serwerownia)
DCIM → Devices → Add: docker01 (site: Pamiątkowo, rack: R1)
DCIM → Devices → Add: fw01 (site: Pamiątkowo, rack: R1)
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran DCIM → Devices w NetBoksie po dodaniu urządzeń `docker01` i `fw01`, z widoczną kolumną „Site" pokazującą „Pamiątkowo" dla obu wierszy.

Weryfikacja

```
cd /docker/compose/netbox
sudo docker compose ps
```

Przykład:

```
NAME                        IMAGE                                  STATUS
netbox-netbox-1             netboxcommunity/netbox:v4.7-5.1.0      Up 2 minutes (healthy)
netbox-netbox-worker-1      netboxcommunity/netbox:v4.7-5.1.0      Up 2 minutes
netbox-netbox-housekeeping-1  netboxcommunity/netbox:v4.7-5.1.0    Up 2 minutes
netbox-postgres-1           postgres:18-alpine                     Up 2 minutes (healthy)
netbox-valkey-1             valkey/valkey:9.1-alpine               Up 2 minutes (healthy)
```

```
curl -I http://127.0.0.1:8000
```

Przykład:

```
HTTP/1.1 200 OK
```

Sprawdzamy dostęp do API tokenem administratora. Token v2, który zakładamy w profilu konta (zakładka API Tokens), ma postać `nbt_KluczTokenu.TokenAPI` i idzie w nagłówku `Authorization: Bearer`; pełnej wartości nie da się odczytać po jego utworzeniu, więc zapisujemy ją od razu.

```
curl -H "Authorization: Bearer nbt_KluczTokenu.TokenAPI"
https://netbox.zielona.example/api/dcim/devices/
```

Starszy token v1 w nagłówku `Authorization: Token TokenAPI` w NetBoksie 4.x nadal działa, ale od wersji 4.6 jest przestarzały i ma zniknąć w wersji 5.0, więc nowych tokenów w tym formacie nie zakładamy. Szczegóły tokenów opisuje strona Porządek i dokumentacja: patchpanele, etykiety, kolory i NetBox.

Przykład:

```
{"count": 2, "next": null, "previous": null, "results": [...]}
```

Wycofanie zmian

```
cd /docker/compose/netbox
sudo docker compose down
```

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

```
sudo docker compose down -v
sudo rm -rf /docker/apps/netbox
```

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
Kontener `netbox` restartuje się w pętli zaraz po starcie	brak albo pusty `SECRET_KEY` w override	uzupełniamy `SECRET_KEY` wygenerowany osobno dla tej instalacji
`docker compose ps` pokazuje `netbox-worker` w innej wersji obrazu niż `netbox`	override nadpisał obraz tylko dla usługi `netbox`, nie dla `netbox-worker` i `netbox-housekeeping`	dopisujemy ten sam tag obrazu dla wszystkich trzech usług
Strona logowania wczytuje się bez stylów CSS	reverse proxy nie przekazuje poprawnie żądań do plików statycznych	sprawdzamy konfigurację Nginksa dla `netbox.zielona.example`, w szczególności nagłówki proxy
`password authentication failed for user netbox` (wariant b)	hasło w override inne niż w bazie na `pg01`	poprawiamy hasło w override albo w bazie przez `ALTER USER netbox WITH PASSWORD`

□□ Ćwiczenia

Ćwiczenie 1: kolejność wprowadzania pierwszych danych

Mamy już założoną lokalizację Pamiątkowo, ale nie mamy jeszcze żadnej szafy ani urządzenia. Chcemy dodać `fw01` do szafy, której jeszcze nie ma w systemie. Ustalamy kolejność kroków.

Odpowiedź

Najpierw zakładamy szafę (rack) przypisaną do lokalizacji Pamiątkowo, dopiero potem dodajemy urządzenie `fw01` wskazując tę już istniejącą szafę i konkretną pozycję w niej. Próba dodania urządzenia przed założeniem szafy nie ma się do czego odnieść, bo formularz urządzenia wymaga wskazania istniejącej szafy.

Powiązane: NetBox · NetBox: źródło prawdy o sieci i serwerowni · Reverse proxy i równoważenie obciążenia · NetBox: logowanie z AD, kopia i aktualizacja

Źródła: github.com/netbox-community/netbox-docker, plik `README.md` i `docker-compose.yml` (gałąź `release`) · github.com/netbox-community/netbox-docker/wiki, strona „Version"

NetBox: logowanie z AD, kopia i aktualizacja

Typ: procedura

Dotyczy: netbox-docker v5.1.0, moduł django-auth-ldap wbudowany w obraz od wersji 2.0 (sprawdzono w dokumentacji projektu 28.09.2026)

Czas: ~40 min

Efekt: logowanie do NetBoksa kontem z Active Directory z uprawnieniami zależnymi od grupy domenowej, oraz działająca procedura kopii zapasowej i aktualizacji instalacji.

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)

Zanim zaczniemy

> **Z ŻYCIA WZIĘTE**
>
> Ola dostaje w NetBoksie dostęp tylko do odczytu, żeby sprawdzać adresację IP swoich aplikacji, ale za każdym razem prosi Tomka o nowe hasło, bo zapomina poprzedniego. Tomek zakłada jej konto domenowe w NetBoksie, ale po miesiącu zauważa, że robi to już dla piątej osoby ręcznie, zamiast raz podłączyć logowanie do Active Directory.

Uczciwie o obrazie do LDAP. Starsze poradniki do `netbox-docker` każą budować osobny obraz z tagiem `-ldap`. Taki obraz nie jest już potrzebny: wsparcie dla LDAP, oparte na module `django-auth-ldap`, jest wbudowane w standardowy obraz `netboxcommunity/netbox` od wersji NetBoksa 2.0, a osobne obrazy `-ldap` z tamtego okresu nie są już utrzymywane. Cała konfiguracja LDAP sprowadza się do zestawu zmiennych środowiskowych w `docker-compose.override.yml`, bez budowania własnego obrazu.

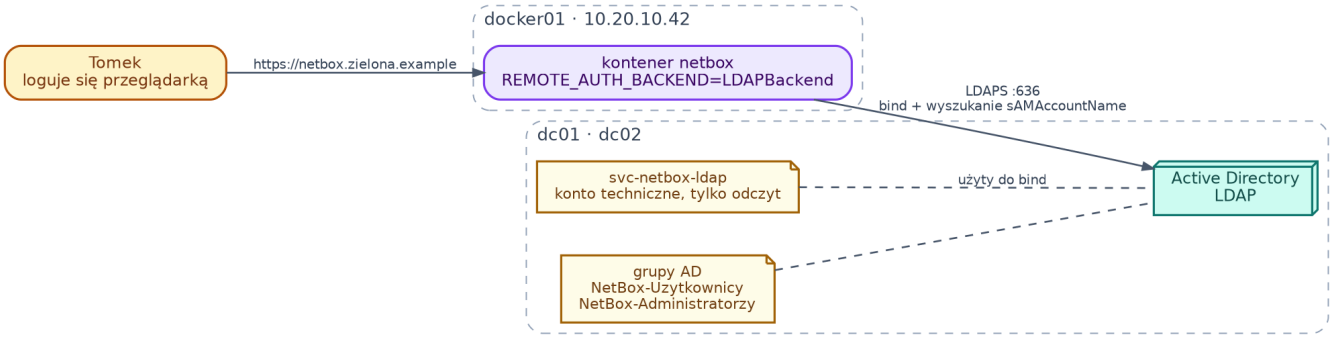
Wymaganie	Uwaga
Działający NetBox po instalacji z poprzedniej strony	patrz Instalacja NetBoksa w Dockerze
Konto techniczne w Active Directory, tylko do odczytu	`svc-netbox-ldap`
Grupy domenowe dla uprawnień NetBoksa	`NetBox-Uzytkownicy`, `NetBox-Administratorzy`, patrz niżej
Dostęp do `/docker/compose/netbox` z uprawnieniami do edycji override	zmiany wchodzą w życie dopiero po ponownym złożeniu kontenerów

Przebieg

Konfiguracja LDAP w docker-compose.override.yml

Dopisujemy zmienne środowiskowe do usługi `netbox` w pliku `docker-compose.override.yml`, założonym na poprzedniej stronie. Te same zmienne muszą trafić też do `netbox-worker` i `netbox-housekeeping`, inaczej zadania w tle uruchamiane przez te kontenery nie zobaczą tej samej konfiguracji uwierzytelniania.

```
services:
  netbox: &netbox-ldap
    environment:
      REMOTE_AUTH_ENABLED: "True"
      REMOTE_AUTH_BACKEND: "netbox.authentication.LDAPBackend"
      AUTH_LDAP_SERVER_URI: "ldaps://dc01.ad.zielona.example ldaps://dc02.ad.zielona.example"
      AUTH_LDAP_BIND_DN: "CN=svc-netbox-ldap,OU=ServiceAccounts,DC=ad,DC=zielona,DC=example"
      AUTH_LDAP_BIND_PASSWORD: "HasloKontaUslugowego"
      AUTH_LDAP_USER_SEARCH_BASEDN: "OU=Uzytkownicy,DC=ad,DC=zielona,DC=example"
      AUTH_LDAP_GROUP_SEARCH_BASEDN: "OU=Grupy,DC=ad,DC=zielona,DC=example"
      AUTH_LDAP_REQUIRE_GROUP_DN: "CN=NetBox-Uzytkownicy,OU=Grupy,DC=ad,DC=zielona,DC=example"
      AUTH_LDAP_IS_ADMIN_DN: "CN=NetBox-Administratorzy,OU=Grupy,DC=ad,DC=zielona,DC=example"
      AUTH_LDAP_IS_SUPERUSER_DN: "CN=NetBox-
Administratorzy,OU=Grupy,DC=ad,DC=zielona,DC=example"
  netbox-worker:
    <<: *netbox-ldap
  netbox-housekeeping:
    <<: *netbox-ldap
```



Logowanie do NetBoksa kontem z Active Directory przez LDAP, z uprawnieniami wynikającymi z grupy domenowej (kliknięcie otwiera pełny rozmiar)

Pojęcie: AUTH_LDAP_USER_SEARCH_ATTR. Atrybut, po którym NetBox dopasowuje wpisany login do konta w katalogu. Domyślna wartość tej zmiennej w `netbox-docker` to `sAMAccountName`, czyli login w formacie znanym z Active Directory (`tomek`, nie pełny adres e-mail), więc dla zwykłej domeny Active Directory, takiej jak w Zielonej, nie trzeba jej w ogóle nadpisywać.

Pojęcie: AUTH_LDAP_REQUIRE_GROUP_DN. Grupa, do której przynależność jest warunkiem koniecznym zalogowania się do NetBoksa w ogóle. Użytkownik istniejący w Active Directory, ale spoza tej grupy, nie zaloguje się mimo poprawnego hasła. **Pojęcie: AUTH_LDAP_IS_ADMIN_DN i AUTH_LDAP_IS_SUPERUSER_DN.** Osobne grupy nadające odpowiednio uprawnienia administracyjne panelu i pełne uprawnienia superużytkownika, na tej samej zasadzie najmniejszych uprawnień, co przy koncie serwisowym używanym do samego bind.

Założenie grup w Active Directory

```
New-ADGroup -Name "NetBox-Uzytkownicy" -GroupScope Global -Path
"OU=Grupy,DC=ad,DC=zielona,DC=example"
New-ADGroup -Name "NetBox-Administratorzy" -GroupScope Global -Path
"OU=Grupy,DC=ad,DC=zielona,DC=example"
Add-ADGroupMember -Identity "NetBox-Uzytkownicy" -Members "ola.wisniewska"
```

Ponowne złożenie kontenerów

```
cd /docker/compose/netbox
sudo docker compose up -d
```

Zmienne środowiskowe czytane są przy starcie kontenera, więc sama zmiana pliku override bez ponownego `docker compose up -d` nie wystarczy.

Weryfikacja

Logujemy się do NetBoksa kontem Oli, należącym do grupy `NetBox-Uzytkownicy`, ale nie do grupy administratorów:

```
https://netbox.zielona.example → login: ola.wisniewska
```

Sprawdzamy w logu kontenera, że próba logowania poszła przez LDAP:

```
sudo docker compose logs --tail 50 netbox | grep -i ldap
```

Przykład:

```
netbox  | INFO    netbox.auth_backends :: User ola.wisniewska successfully authenticated via
LDAP
```

Kopia zapasowa

Kopię NetBoksa wykonujemy dwuwarstwowo, tak jak w wersji opisanej na istniejącej stronie NetBox : zrzut bazy danych PostgreSQL oraz kopia katalogów z plikami niebędącymi częścią samej bazy.

Zrzut bazy danych: pg_dump

```
cd /docker/compose/netbox
sudo docker compose exec postgres pg_dump -U netbox netbox > /mnt/backup/netbox/netbox-$(date
+%F).sql
```

Dla wariantu produkcyjnego z bazą na `pg01` wykonujemy `pg_dump` bezpośrednio na `pg01`, bez wchodzenia do żadnego kontenera:

```
pg_dump -h pg01 -U netbox netbox > /mnt/backup/netbox/netbox-$(date +%F).sql
```

Kopia katalogu z mediami

```
sudo tar czf /mnt/backup/netbox/netbox-media-$(date +%F).tar.gz -C /docker/apps/netbox media reports scripts
```

Katalog `media` trzyma m.in. załączone pliki i obrazy urządzeń dodane ręcznie przez użytkowników, `reports` i `scripts` własne skrypty automatyzacji NetBoksa: żadne z nich nie odtworzą się z samej bazy danych.

Aktualizacja

Aktualizację wykonujemy zmianą przypiętego tagu obrazu w `docker-compose.override.yml`, zgodnie z listą zmian opublikowaną w repozytorium `netbox-docker`, a nie przez podmianę na `latest`.

```
cd /docker/compose/netbox
sudo git pull
```

Sprawdzamy w pliku `docker-compose.yml` repozytorium, jaka jest nowa domyślna wartość `VERSION`, i przepisujemy ten sam tag do własnego override dla usług `netbox`, `netbox-worker` i `netbox-housekeeping`. Dopiero potem:

```
sudo docker compose pull
sudo docker compose up -d
```

> Przed każdą aktualizacją wykonujemy świeżą kopię zapasową opisaną wyżej. NetBox, tak jak większość aplikacji opartych na frameworku Django, migruje schemat bazy danych automatycznie przy starcie nowej wersji, a operacja ta nie zawsze da się bezpiecznie cofnąć bez kopii sprzed zmiany.

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
Logowanie LDAP działa dla zwykłych użytkowników, ale nie dla administratorów	konto nie należy do grupy wskazanej w `AUTH_LDAP_IS_ADMIN_DN` albo `AUTH_LDAP_IS_SUPERUSER_DN`	sprawdzamy członkostwo grupy `getent group` na `dc01` albo `Get-ADGroupMember`

Objaw	Przyczyna	Co robimy
Zmiana w override nie ma żadnego efektu po zapisaniu pliku	brak ponownego `docker compose up -d`, kontener wciąż działa ze starymi zmiennymi	wykonujemy `docker compose up -d`, samo zapisanie pliku nie restartuje kontenera
`netbox-worker` loguje inaczej niż `netbox`, niespójne uprawnienia w tle	zmienne LDAP nadpisane tylko dla usługi `netbox`, nie dla `netbox-worker` i `netbox-housekeeping`	kopiujemy ten sam blok `environment` do wszystkich trzech usług, najlepiej kotwicą YAML jak w przykładzie wyżej
Aktualizacja kończy się kontenerem restartującym się w pętli	migracja schematu bazy przerwana w trakcie, np. brakiem miejsca na dysku	przywracamy kopię `pg_dump` sprzed aktualizacji, sprawdzamy wolne miejsce na `/docker`

□□ Ćwiczenia

Ćwiczenie 1: dobór zmiennych LDAP

Zakładamy w NetBoksie logowanie domenowe, ale chcemy, żeby tylko członkowie zespołu Oli mogli się w ogóle zalogować, a reszta firmy nie widziała NetBoksa wcale, nawet z poprawnym hasłem domenowym. Wskazujemy właściwą zmienną.

Odpowiedź

`AUTH_LDAP_REQUIRE_GROUP_DN` wskazujący grupę zespołu Oli. Ta zmienna działa jako warunek konieczny samego zalogowania, niezależnie od `AUTH_LDAP_IS_ADMIN_DN` i `AUTH_LDAP_IS_SUPERUSER_DN`, które rozstrzygają wyłącznie poziom uprawnień wewnątrz NetBoksa dla osób, które i tak już przeszły ten pierwszy warunek.

Powiązane: Instalacja NetBoksa w Dockerze · NetBox · Active Directory: konta, grupy, jednostki organizacyjne · FreeIPA

Źródła: github.com/netbox-community/netbox-docker/wiki/LDAP · github.com/netbox-community/netbox-docker, plik `configuration/ldap/ldap_config.py` (gałąź `release`) · github.com/netbox-community/netbox-docker/wiki/Updating