Poznajemy Zabbixa i instalacja

Opowieść, serwer w Dockerze i z pakietów, logowanie kontem z AD.

- Zabbix: od pracy dyplomowej na Łotwie do monitoringu całej serwerowni
- Instalacja serwera Zabbix w Dockerze
- Instalacja serwera Zabbix z pakietów na Rocky Linux 9
- Zabbix: logowanie kontem z Active Directory

Zabbix: od pracy dyplomowej na Łotwie do monitoringu całej serwerowni

Poziom: podstawy **Dotyczy: Zabbix, historia projektu** **Czas: ~10 min**

Ta strona otwiera książkę o Zabbiksie: skąd wziął się ten system monitoringu, dlaczego Zielona w ogóle go potrzebowała i co czeka na kolejnych stronach.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
B2B	business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych
GPL	General Public License: otwarta licencja oprogramowania, na której opierają się MySQL Community i MariaDB
IT	Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy
SIA	Sabiedrība ar Ierobežotu Atbildību: łotewski odpowiednik polskiej spółki z ograniczoną odpowiedzialnością, forma prawna spółki (np. Zabbix SIA)
SMART	Self-Monitoring, Analysis and Reporting Technology: wbudowany w dysk mechanizm zbierający liczniki o jego kondycji, na przykład temperaturę i liczbę realokowanych sektorów, odczytywany programami takimi jak smartctl
SNMP	Simple Network Management Protocol: protokół odczytu stanu urządzeń sieciowych
WMS	Warehouse Management System: system zarządzania magazynem

Skąd się wzięło

> **Pojęcie: Zabbix.** System monitoringu open source, który cyklicznie zbiera dane z serwerów i urządzeń sieciowych, ocenia je względem progów i alarmuje, gdy coś przekroczy normę.
>
> Pełny rozkład na czynniki pierwsze jest na stronie Architektura Zabbixa.

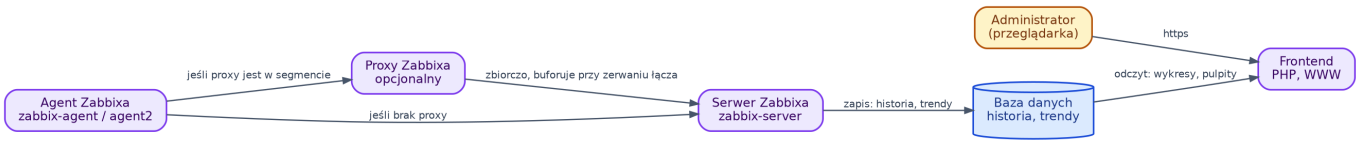
Autorem Zabbixa jest **Alexei Vladishev**, informatyk z Rygi, absolwent Uniwersytetu Łotewskiego. Pod koniec lat 90. pracował jako administrator systemów w instytucji finansowej i, jak wielu administratorów przed nim i po nim, miał dość ręcznego sprawdzania, czy serwery i usługi wciąż działają. W 1998 roku napisał na własny użytek prosty skrypt do automatycznego sprawdzania stanu maszyn, który z czasem rozrósł się w coś więcej niż jednorazowe narzędzie.

Trzy lata później, 7 kwietnia 2001 roku, Vladishev opublikował swój projekt publicznie jako wolne oprogramowanie na licencji GPL, pod nazwą Zabbix 1.0alpha1. Pierwsza w pełni stabilna wersja, oznaczona już jako 1.0, ukazała się dopiero w marcu 2004 roku: między publikacją a stabilnym wydaniem minęły trzy lata dopracowywania. W 2005 roku Vladishev założył w Rydze spółkę **Zabbix SIA** (SIA to łotewski odpowiednik polskiego „sp. z o.o."), żeby móc sprzedawać wsparcie techniczne i szkolenia, nie zamykając przy tym samego oprogramowania: silnik Zabbixa zostaje wolnym oprogramowaniem do dziś, tylko od wersji 7.0 z 2024 roku zmienił licencję z GPLv2 na AGPLv3.

Firma rozrosła się z jednoosobowego projektu do zespołu liczącego dziś ponad siedemdziesiąt osób, z siedzibą wciąż w Rydze i oddziałami w kilku innych krajach, obsługującego instalacje Zabbixa w firmach na całym świecie. Model biznesowy się nie zmienił od 2005 roku: oprogramowanie darmowe i otwarte, płatne wsparcie i certyfikacja dla tych, którzy chcą mieć na kogo zadzwonić w środku nocy.

Jak to działa w jednym obrazku

Zabbix rozkłada się na cztery elementy: agent zainstalowany na monitorowanej maszynie zbiera lokalne metryki, serwer centralny odpytuje agentów (albo odbiera od nich dane) i ocenia progi, baza danych trzyma całą zebraną historię, a frontend czyta z tej bazy wykresy i pulpity pokazywane w przeglądarce. Pełny opis tej architektury, razem z rozróżnieniem sprawdzeń pasywnych i aktywnych, jest na stronie Architektura Zabbixa.



Wartość płynie od agenta do serwera, dalej do bazy danych, a frontend czyta ją dopiero z bazy (kliknięcie otwiera pełny rozmiar)

Dlaczego Zielona

> **Z ŻYCIA WZIĘTE**
>
> Noc z poniedziałku na wtorek, tuż po trzeciej. Na jednym z dysków w klastrze `hv02` pojawia się pierwszy sygnał awarii, jeszcze zanim dysk faktycznie odmówi posłuszeństwa. Zabbix, monitorujący stan SMART i pulę magazynu Storage Spaces, wysyła alarm od razu: administrator dyżurny widzi wiadomość na telefonie, zanim ktokolwiek z biura w ogóle usiądzie do komputera. Pełną historię tej nocy, łącznie z liczeniem zapasu w puli, opisujemy na stronie S2D: wymiana dysku bez przestoju, gdy pula jest prawie pełna.

Zanim Zielona miała cokolwiek monitorujące infrastrukturę, o awarii dowiadywano się na dwa sposoby: albo ktoś zauważył objaw (Optima wisi, WMS muli), albo nikt nie zauważył niczego, dopóki drugi dysk w tej samej puli też nie zaczął szwankować, i wtedy nagle liczyła się każda godzina. Żadna z tych dwóch dróg nie jest sposobem, w jaki dwuosobowy zespół IT chce prowadzić serwerownię obsługującą księgowość, magazyn i sklep B2B naraz.

Zielona wybrała akurat Zabbixa, a nie płatną usługę monitoringu w chmurze, z trzech praktycznych powodów. Po pierwsze, dane telemetryczne serwerów, haseł do baz i adresacji sieci wewnętrznej nie muszą wtedy opuszczać sieci firmy. Po drugie, jeden system obsługuje naraz Windows, Linuksa, urządzenia sieciowe po SNMP (przełączniki, UPS-y) i własne aplikacje zespołu Oli przez proste skrypty, bez kupowania osobnej licencji na każdy rodzaj hosta. Po trzecie, koszt nie rośnie liniowo z liczbą monitorowanych maszyn, co przy planach dołożenia Kubernetesa i kolejnych baz danych ma znaczenie.

Do tego dochodzi jedna praktyczna cecha, którą docenia się dopiero po pierwszym większym audycie: Zabbix trzyma pełną historię, więc na pytanie pana Wojtka „a jak wyglądało obciążenie serwera Optimy w listopadzie zeszłego roku" zespół odpowiada wykresem, a nie wzruszeniem ramion.

Mapa tej książki

Książka o Zabbiksie prowadzi od instalacji serwera do codziennej pracy z monitoringiem: hostów, elementów danych, wyzwalaczy i utrzymania samej instalacji.

Strona	Na jakie pytanie odpowiada
Zabbix: od pracy dyplomowej na Łotwie do monitoringu całej serwerowni	Skąd wziął się Zabbix i po co go mamy
Instalacja serwera Zabbix w Dockerze	Jak postawić serwer, frontend i bazę w kontenerach

Strona	Na jakie pytanie odpowiada
Instalacja serwera Zabbix z pakietów na Rocky Linux 9	Jak postawić to samo jako usługi systemd, z pakietów dystrybucji
Zabbix: logowanie kontem z Active Directory	Jak podłączyć logowanie przez LDAP zamiast kont lokalnych
Architektura Zabbixa	Z jakich elementów składa się Zabbix i którędy płyną dane
Agent, instalacja i konfiguracja	Jak podłączyć nowy host do monitoringu
Host, element danych, wyzwalacz, akcja	Jak to się składa w jeden spójny system pojęć
Monitorowanie typowych usług	Jak monitorować bazy, porty, certyfikaty i logi
Utrzymanie serwera Zabbix	Jak dbać o rosnącą bazę, retencję i wydajność, jak robić kopię i aktualizować
SMART dysków na Windows przez agenta 2	Jak dołożyć monitoring zdrowia dysków tam, gdzie ma to sens

Powiązane: Architektura Zabbixa · S2D: wymiana dysku bez przestoju, gdy pula jest prawie pełna

Źródła: The story of how Zabbix software became one of the world's most demanded IT monitoring solutions · Alexei Vladishev, bio Zabbix · Zabbix, endoflife.date · About Zabbix LLC

Instalacja serwera Zabbix w Dockerze

Typ: procedura

Dotyczy: Zabbix 7.0.31 LTS, obrazy zabbix/zabbix-server-pgsql, zabbix/zabbix-web-nginx-pgsql, zabbix/zabbix-agent2 (sprawdzone na Docker Hub 28.09.2026)

Czas: ~35 min

Efekt: działający serwer, frontend i agent Zabbixa w kontenerach, z bazą PostgreSQL w wariancie laboratoryjnym albo na `pg01` w wariancie produkcyjnym, gotowe do pierwszego logowania.

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
LTS	Long Term Support: wersja z długim wsparciem poprawkami

Zanim zaczniemy

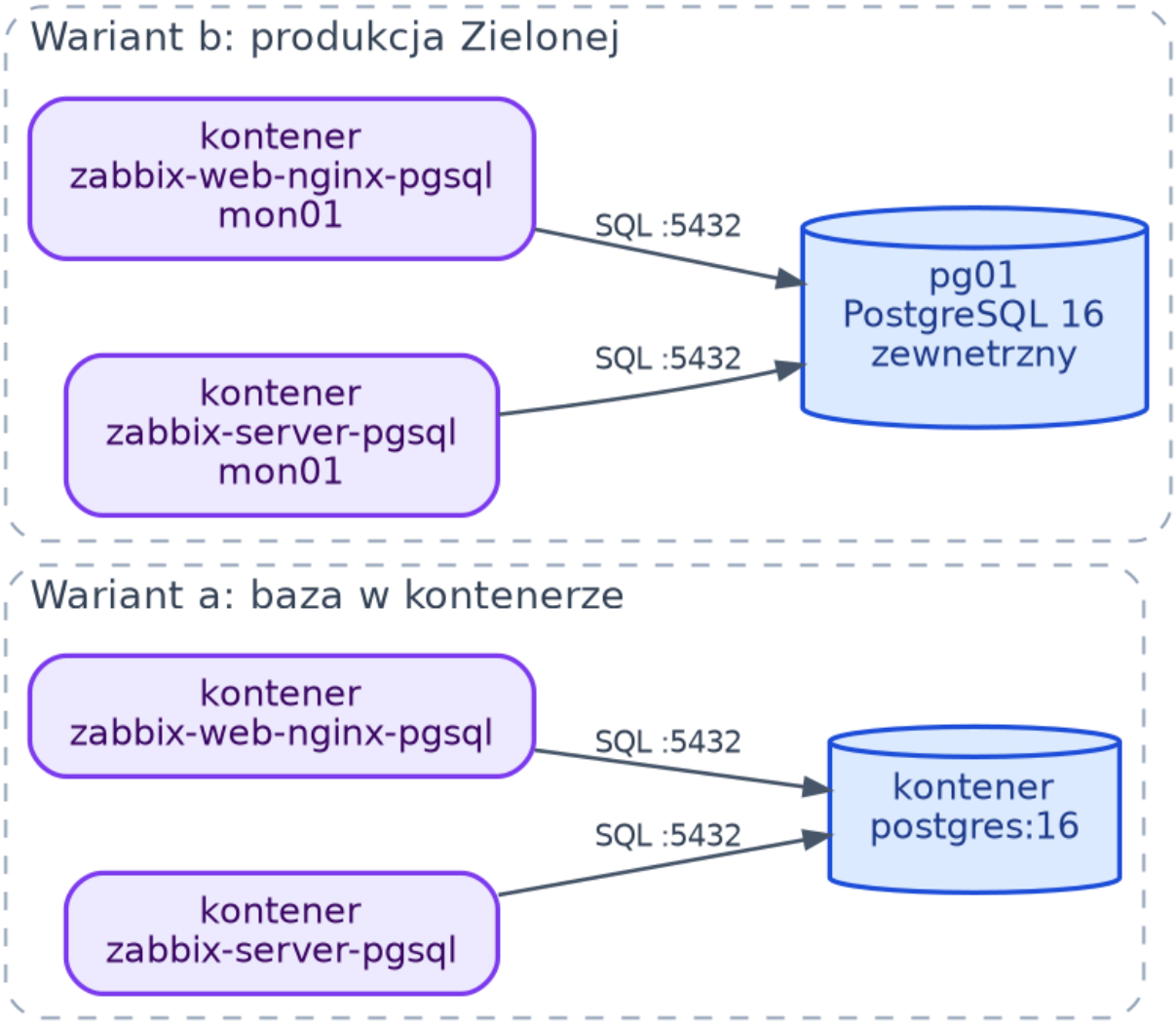
Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo albo root, na maszynie z zainstalowanym Dockerem i wtyczką Compose. Poniżej dwa warianty: laboratoryjny, z własnym kontenerem PostgreSQL obok Zabbixa, oraz produkcyjny Zielonej, z bazą na osobnym serwerze `pg01`. Wybieramy jeden, zależnie od tego, czy mamy już gdzieś działającą bazę PostgreSQL do współdzielenia.

> 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`
Wolny port `80` na hoście dla frontendu	kontener nasłuchuje wewnątrz na `8080`, mapujemy na `80`

Wymaganie	Uwaga
Wolny port `10051` na hoście	port serwera, potrzebny do sprawdzeń aktywnych z innych hostów
Wariant b: osiągalny `pg01` na porcie `5432`	baza tworzona ręcznie przed pierwszym startem

Uważaj: obraz zawsze przypinamy do konkretnej wersji (`7.0.31-ubuntu`), nigdy do `latest`. Aktualizacja to świadoma zmiana numeru w `compose.yaml`, nie przypadkowe pobranie nowszej wersji przy kolejnym `docker compose pull`. Wersja `7.0` to aktualna gałąź LTS Zabbixa; kolejna LTS, `8.0`, w chwili sprawdzania nie miała jeszcze wydania stabilnego, więc trzymamy się `7.0`.



Wariant a trzyma bazę we własnym kontenerze, wariant b łączy się do PostgreSQL na pg01 (kliknięcie otwiera pełny rozmiar)

Przebieg

Wariant a: laboratorium, Zabbix z własnym kontenerem PostgreSQL

Zakładamy katalogi na dane, zgodnie z konwencją ścieżek:

```
sudo mkdir -p /docker/apps/zabbix/server /docker/apps/zabbix/postgres
sudo mkdir -p /docker/compose/zabbix
```

Plik `/docker/compose/zabbix/.env` z hasłem do bazy, obok pliku `compose.yaml`, nie w nim samym:

```
ZABBIX_DB_PASSWORD=HasloDoBazy
```

Plik `/docker/compose/zabbix/compose.yaml`:

```
services:
  server:
    image: zabbix/zabbix-server-pgsql:7.0.31-ubuntu
    container_name: zabbix-server
    restart: always
    environment:
      - DB_SERVER_HOST=db
      - POSTGRES_DB=zabbix
      - POSTGRES_USER=zabbix
      - POSTGRES_PASSWORD=${ZABBIX_DB_PASSWORD}
    volumes:
      - /docker/apps/zabbix/server:/var/lib/zabbix/export
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "10051:10051"
    depends_on:
      - db

  web:
    image: zabbix/zabbix-web-nginx-pgsql:7.0.31-ubuntu
```

```
    container_name: zabbix-web
    restart: always
    environment:
      - DB_SERVER_HOST=db
      - POSTGRES_DB=zabbix
      - POSTGRES_USER=zabbix
      - POSTGRES_PASSWORD=${ZABBIX_DB_PASSWORD}
      - ZBX_SERVER_HOST=server
      - PHP_TZ=Europe/Warsaw
    ports:
      - "80:8080"
    depends_on:
      - server
      - db

  agent:
    image: zabbix/zabbix-agent2:7.0.31-ubuntu
    container_name: zabbix-agent2
    restart: always
    environment:
      - ZBX_HOSTNAME=mon01
      - ZBX_SERVER_HOST=server
    volumes:
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    depends_on:
      - server

  db:
    image: postgres:16
    container_name: zabbix-postgres
    restart: always
    environment:
      - POSTGRES_USER=zabbix
      - POSTGRES_PASSWORD=${ZABBIX_DB_PASSWORD}
      - POSTGRES_DB=zabbix
    volumes:
      - /docker/apps/zabbix/postgres:/var/lib/postgresql/data
```

Pierwsze uruchomienie:

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

Wariant b: produkcja Zielonej, baza na zewnętrznym pg01

Katalog na dane potrzebny jest tylko dla samego Zabbixa, bez podkatalogu na bazę:

```
sudo mkdir -p /docker/apps/zabbix/server
sudo mkdir -p /docker/compose/zabbix
```

Zanim uruchomimy kontener, zakładamy bazę i użytkownika na `pg01`. Łączymy się tam jako administrator PostgreSQL:

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

W sesji `psql` tworzymy bazę i konto aplikacji:

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

Plik `/docker/compose/zabbix/.env`:

```
ZABBIX_DB_PASSWORD=HasloDoBazy
```

Plik `/docker/compose/zabbix/compose.yaml`: identyczny jak w wariancie a, tylko bez usługi `db` i z `DB_SERVER_HOST=pg01` w usługach `server` i `web` zamiast `DB_SERVER_HOST=db`:

```
services:
  server:
    image: zabbix/zabbix-server-pgsql:7.0.31-ubuntu
    container_name: zabbix-server
    restart: always
    environment:
      - DB_SERVER_HOST=pg01
      - POSTGRES_DB=zabbix
      - POSTGRES_USER=zabbix
```

```
      - POSTGRES_PASSWORD=${ZABBIX_DB_PASSWORD}
    volumes:
      - /docker/apps/zabbix/server:/var/lib/zabbix/export
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "10051:10051"

  web:
    image: zabbix/zabbix-web-nginx-pgsql:7.0.31-ubuntu
    container_name: zabbix-web
    restart: always
    environment:
      - DB_SERVER_HOST=pg01
      - POSTGRES_DB=zabbix
      - POSTGRES_USER=zabbix
      - POSTGRES_PASSWORD=${ZABBIX_DB_PASSWORD}
      - ZBX_SERVER_HOST=server
      - PHP_TZ=Europe/Warsaw
    ports:
      - "80:8080"
    depends_on:
      - server

  agent:
    image: zabbix/zabbix-agent2:7.0.31-ubuntu
    container_name: zabbix-agent2
    restart: always
    environment:
      - ZBX_HOSTNAME=mon01
      - ZBX_SERVER_HOST=server
    volumes:
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    depends_on:
      - server
```

`DB_SERVER_HOST` to jedyna zmienna, która różni się między wariantami: w wariancie a wskazuje na nazwę usługi `db` w tej samej sieci Dockera, w wariancie b na nazwę serwera `pg01` osiągalnego w sieci Zielonej. Pierwsze uruchomienie:

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

Logi pierwszego startu

Sprawdzamy, czy serwer wstał poprawnie i połączył się z bazą:

```
docker compose logs -f server
```

Przykład:

```
zabbix-server  | 2026/09/28 08:11:04 ...include/zbxdbschema.c:... database version 07000000
zabbix-server  | 2026/09/28 08:11:04 ...server.c:... server #0 started [main process]
zabbix-server  | 2026/09/28 08:11:04 ...dbsync.c:... syncing configuration cache
```

Weryfikacja

Sprawdzamy, że kontenery działają:

```
docker compose ps
```

Przykład:

```
NAME             IMAGE                                          STATUS
zabbix-server    zabbix/zabbix-server-pgsql:7.0.31-ubuntu       Up 2 minutes
zabbix-web       zabbix/zabbix-web-nginx-pgsql:7.0.31-ubuntu    Up 2 minutes
zabbix-agent2    zabbix/zabbix-agent2:7.0.31-ubuntu             Up 2 minutes
```

Sprawdzamy, że frontend odpowiada, zanim otworzymy go w przeglądarce:

```
curl -I http://localhost:80
```

Przykład:

```
HTTP/1.1 302 Found
Location: index.php
```

Otwieramy `http://mon01` w przeglądarce i logujemy się domyślnym kontem administratora:

```
Login: Admin
Password: zabbix
```

> Zaraz po pierwszym zalogowaniu zmieniamy hasło konta `Admin`, zanim zrobimy cokolwiek innego: `Users → Users → Admin → Change password`. Domyślne hasło `zabbix` jest publicznie znane z dokumentacji, więc zostawienie go choćby na chwilę to otwarta furtka do pełnej administracji Zabbiksem.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran logowania Zabbixa pod `http://mon01`, a zaraz po nim ekran `Users → Users → Admin` z otwartym formularzem zmiany hasła.

Wycofanie zmian

Zatrzymanie i usunięcie kontenerów, bez kasowania danych z wolumenów:

```
docker compose down
```

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

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

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
Kontener `server` restartuje się w pętli	baza jeszcze nie wystartowała (wariant a) albo `pg01` nieosiągalny (wariant b)	`docker compose logs db` albo sprawdzamy `telnet pg01 5432` z hosta `mon01`
`curl -I http://localhost:80` zwraca `Connection refused`	kontener `web` jeszcze się uruchamia albo port zajęty przez inną usługę	`docker compose logs web`, `ss -tlnp \

Objaw	Przyczyna	Naprawa
Frontend wyświetla błąd połączenia z bazą	inne hasło w `.env` niż w bazie na `pg01`	poprawiamy hasło w `.env` albo w bazie przez `ALTER USER zabbix WITH PASSWORD`
Nowe hosty pokazują status niedostępny mimo działającego serwera	port `10051` niedostępny z zewnątrz kontenera	sprawdzamy mapowanie portu w `compose.yaml` i firewalld na hoście `mon01`

Powiązane: Instalacja serwera Zabbix z pakietów na Rocky Linux 9 · Architektura Zabbixa · Konwencja Dockera w Zielonej

Źródła: Zabbix Docker images · zabbix/zabbix-server-pgsql, Docker Hub · zabbix/zabbix-web-nginx-pgsql, Docker Hub · Zabbix Life Cycle and Release Policy

Instalacja serwera Zabbix z pakietów na Rocky Linux 9

Typ: procedura

Dotyczy: Zabbix 7.0 LTS, pakiety z repozytorium Zabbixa, Rocky Linux 9, PostgreSQL na pg01, nginx i php-fpm (sprawdzone w dokumentacji Zabbixa 28.09.2026)

Czas: ~40 min

Efekt: serwer i frontend Zabbixa jako usługi systemd na Rocky Linux 9, bez Dockera, z bazą na `pg01`, dostępne przez nginx i gotowe do pierwszej konfiguracji. Instalację i konfigurację samego agenta opisuje osobno strona Agent, instalacja i konfiguracja, tutaj jej nie powtarzamy.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
LTS	Long Term Support: wersja z długim wsparciem poprawkami
PHP	PHP: Hypertext Preprocessor, język programowania, w którym napisany jest frontend Zabbixa i wiele innych aplikacji webowych
SELinux	Security-Enhanced Linux: dodatkowa warstwa uprawnień w jądrze RHEL
TCP	Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

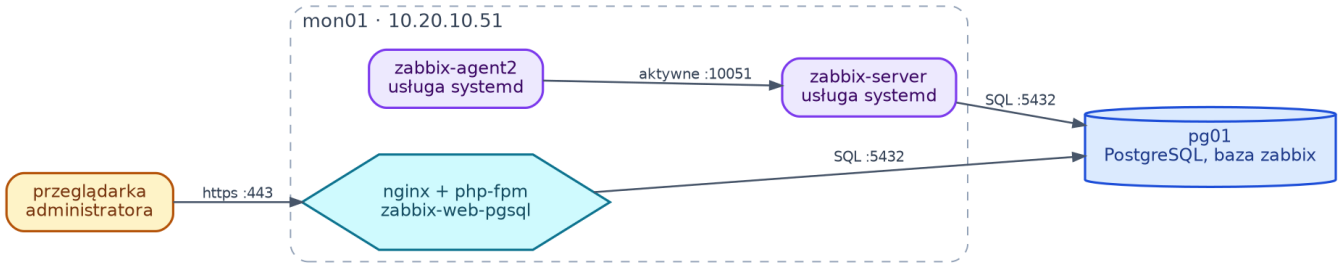
Zanim zaczniemy

Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo albo root, na świeżym Rocky Linux 9. Serwer musi rozwiązywać nazwy DNS i mieć ruch wychodzący do `repo.zabbix.com`, żeby dodać repozytorium.

Wymaganie	Uwaga

Rocky Linux 9 albo RHEL 9	polecenia SELinux i firewalld poniżej dotyczą tej rodziny systemów
Osiągalny `pg01` na porcie `5432`	baza tworzona ręcznie w kroku 2, przed importem schematu
Konto i baza założone w PostgreSQL na `pg01`	patrz krok 2
Ruch wychodzący do `repo.zabbix.com`	pobranie pakietu z definicją repozytorium

Uważaj: ten wariant instaluje frontend za nginx i php-fpm, tak samo jak reszta aplikacji webowych Zielonej. Dokumentacja Zabbixa opisuje też wariant z Apache: jeśli host ma już Apache pod inną aplikacją, sprawdzamy w dokumentacji pakiet `zabbix-web-apache-pgsql` zamiast `zabbix-nginx-conf`, żeby nie utrzymywać dwóch serwerów WWW na jednej maszynie.



Na mon01 stoją nginx z php-fpm, zabbix-server i zabbix-agent2 jako usługi systemd, baza jest na pg01 (kliknięcie otwiera pełny rozmiar)

Przebieg

1. Dodanie repozytorium Zabbixa

```
sudo rpm -Uvh https://repo.zabbix.com/zabbix/7.0/rhel/9/x86_64/zabbix-release-latest-7.0.el9.noarch.rpm
sudo dnf clean all
```

Dodajemy oficjalne repozytorium Zabbixa zamiast wersji z domyślnych repozytoriów dystrybucji, bo ta druga bywa znacznie starsza niż aktualna gałąź LTS.

2. Instalacja pakietów serwera, frontendu i agenta

```
sudo dnf install -y zabbix-server-pgsql zabbix-web-pgsql zabbix-nginx-conf zabbix-sql-scripts
zabbix-selinux-policy zabbix-agent2
```

Pakiet `zabbix-web-pgsql` instaluje wspólne pliki frontendu niezależnie od serwera WWW, `zabbix-nginx-conf` dokłada gotowy plik konfiguracyjny nginksa, a `zabbix-selinux-policy` instaluje dedykowany moduł polityki SELinux dla procesów Zabbixa, zamiast zostawiać je bez etykiety albo w trybie permisywnym.

3. Baza danych: konto i schemat na pg01

Zanim uruchomimy serwer, zakładamy bazę i użytkownika na `pg01`. Łączymy się tam jako administrator PostgreSQL:

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

W sesji `psql` tworzymy bazę i konto aplikacji:

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

Importujemy schemat Zabbixa do świeżo założonej bazy, z hosta `mon01`, gdzie zainstalowaliśmy pakiet `zabbix-sql-scripts`:

```
zcat /usr/share/zabbix-sql-scripts/postgresql/server.sql.gz | psql -h pg01 -U zabbix -d zabbix
```

Komenda prosi o hasło konta `zabbix` założonego chwilę wcześniej. Import trwa od kilku do kilkudziesięciu sekund i nie da się go bezpiecznie przerwać w połowie: przerwany import zostawia bazę w stanie częściowym, z którego jedynym wyjściem jest usunięcie bazy i powtórzenie kroku od nowa.

4. Konfiguracja serwera: zabbix_server.conf

```
sudo sed -i \
  -e 's/^# DBHost=.*/DBHost=pg01/' \
  -e 's/^# DBPassword=.*/DBPassword=HasloDoBazy/' \
  /etc/zabbix/zabbix_server.conf
```

> (gdzie `HasloDoBazy` to hasło konta `zabbix` założonego w kroku 3, identyczne z tym podanym przy `CREATE USER`)

Pojęcie: DBHost. Parametr `zabbix_server.conf` wskazujący adres serwera bazy danych, odpowiednik zmiennej `DB_SERVER_HOST` z wariantu dockerowego opisanego na stronie Instalacja serwera Zabbix w Dockerze: tu wpisujemy `pg01`, bo baza stoi na osobnym serwerze, nie w tym samym kontenerze co serwer Zabbixa.

5. Konfiguracja frontendu: nginx i php-fpm

```
sudo sed -i \
  -e 's/^#\s*listen\s\+8080;/        listen          80;/' \
  -e 's/^#\s*server_name\s\+example.com;/        server_name     zabbix.zielona.example;/' \
  /etc/nginx/conf.d/zabbix.conf
```

Plik `/etc/nginx/conf.d/zabbix.conf` to symlink na `/etc/zabbix/nginx.conf`, dołożony przez pakiet `zabbix-nginx-conf` z domyślnie zakomentowanymi liniami `listen` i `server_name`. Bez ich odkomentowania nginx w ogóle nie wystawi frontendu na żadnym porcie.

Strefę czasową dla PHP dopisujemy do konfiguracji php-fpm dedykowanej Zabbiksowi:

```
echo 'php_value[date.timezone] = Europe/Warsaw' | sudo tee -a /etc/php-fpm.d/zabbix.conf
```

6. Uruchomienie usług

```
sudo systemctl enable --now zabbix-server nginx php-fpm
```

Agenta na tym samym hoście włączamy dokładnie tak samo jak na każdym innym monitorowanym serwerze, zgodnie ze stroną Agent, instalacja i konfiguracja: `sudo systemctl enable --now zabbix-agent2`.

7. SELinux

Pakiet `zabbix-selinux-policy` pokrywa większość typowych sytuacji, ale połączenie serwera i frontendu z bazą danych na innym hoście wymaga dodatkowo dwóch flag logicznych, wyłączonych domyślnie.

```
sudo setsebool -P zabbix_can_network on
sudo setsebool -P httpd_can_network_connect_db on
```

> **Pojęcie: flaga logiczna SELinux (boolean).** Przełącznik włączający albo wyłączający fragment polityki SELinux bez pisania własnych reguł od zera. `zabbix_can_network` pozwala procesowi `zabbix-server` łączyć się po sieci na dowolny port TCP, co jest potrzebne właśnie przy bazie na zdalnym `pg01`; `httpd_can_network_connect_db` daje tę samą zgodę procesom serwera WWW i PHP obsługującym frontend.

8. Firewalld

```
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-port=10051/tcp
sudo firewall-cmd --reload
```

Port `10051` otwieramy, jeśli inne hosty (proxy Zabbixa albo agenci przy sprawdzeniach aktywnych) mają się łączyć do tego serwera; usługa `http` otwiera standardowy port `80` dla frontendu.

Weryfikacja

```
sudo systemctl status zabbix-server nginx php-fpm
```

Przykład:

```
● zabbix-server.service - Zabbix Server
     Active: active (running) since Mon 2026-09-28 09:02:11 UTC; 12s ago
● nginx.service - The nginx HTTP and reverse proxy server
     Active: active (running) since Mon 2026-09-28 09:02:11 UTC; 12s ago
● php-fpm.service - The PHP FastCGI Process Manager
     Active: active (running) since Mon 2026-09-28 09:02:11 UTC; 12s ago
```

Sprawdzamy log serwera pod kątem udanego połączenia z bazą:

```
sudo tail -n 20 /var/log/zabbix/zabbix_server.log
```

Przykład:

```
 4102:20260928:090211.442 current database version (mandatory/optional): 07000000/07000000
 4102:20260928:090211.442 server #0 started [main process]
```

Otwieramy `http://mon01` w przeglądarce i logujemy się domyślnym kontem administratora:

```
Login: Admin
Password: zabbix
```

> Zaraz po pierwszym zalogowaniu zmieniamy hasło konta `Admin`: `Users → Users → Admin → Change password`. Domyślne hasło jest publicznie znane z dokumentacji Zabbixa.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: kreator pierwszej konfiguracji Zabbixa pod `http://mon01` (ekran „Check of pre-requisites") z widocznymi zielonymi znacznikami przy wszystkich wymaganiach PHP.

Wycofanie zmian

```
sudo systemctl disable --now zabbix-server nginx php-fpm
sudo dnf remove -y zabbix-server-pgsql zabbix-web-pgsql zabbix-nginx-conf zabbix-sql-scripts
zabbix-selinux-policy
sudo rm -f /etc/nginx/conf.d/zabbix.conf /etc/php-fpm.d/zabbix.conf
```

Bazę na `pg01` usuwamy osobno, tylko gdy naprawdę chcemy zacząć od zera:

```
DROP DATABASE zabbix;
DROP USER zabbix;
```

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
`zabbix_server.log` pokazuje błąd połączenia z bazą	zły `DBHost` / `DBPassword` w `zabbix_server.conf` albo brak reguły `zabbix_can_network`	sprawdzamy `zabbix_server.conf`, `sudo ausearch -m avc -c zabbix_server --start recent`, włączamy `zabbix_can_network`
Frontend zwraca błąd 502 z nginksa	php-fpm nie działa albo gniazdo w złej ścieżce	`systemctl status php-fpm`, `sudo journalctl -u php-fpm -n 50`
Frontend ładuje się, ale zawiesza się na kroku połączenia z bazą	`httpd_can_network_connect_db` wyłączone, SELinux blokuje php-fpm	`sudo setsebool -P httpd_can_network_connect_db on`
`curl -I http://localhost` zwraca `Connection refused`	linie `listen` / `server_name` w `/etc/nginx/conf.d/zabbix.conf` wciąż zakomentowane	odkomentowujemy obie linie, `sudo systemctl restart nginx`

Powiązane: Instalacja serwera Zabbix w Dockerze · Agent, instalacja i konfiguracja · Utrzymanie serwera Zabbix

Źródła: dokumentacja Zabbix 7.0, rozdział „Installation from packages" i „Install frontend" · `man semanage` · `man firewall-cmd`

Zabbix: logowanie kontem z Active Directory

Typ: procedura **Dotyczy: Zabbix 7.0 LTS, Active Directory na dc01/dc02**

Czas: ~35 min

Efekt: logowanie do Zabbixa kontem domenowym z Active Directory, z automatycznym założeniem konta i nadaniem roli osobom z grupy `GG_Zabbix_Admin`, bez ręcznego zakładania kont lokalnych dla każdej nowej osoby.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
JIT	Just In Time: mechanizm zakładający albo aktualizujący konto użytkownika automatycznie w chwili pierwszego udanego logowania, zamiast wymagać ręcznego założenia konta z wyprzedzeniem
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)
LDAPS	LDAP szyfrowany TLS, zwykle port 636
LTS	Long Term Support: wersja z długim wsparciem poprawkami

Zanim zaczniemy

Z ŻYCIA WZIĘTE

Tomek dostaje dostęp do Zabbixa drugiego tygodnia pracy. Zamiast czekać, aż ktoś założy mu konto lokalne w bazie Zabbixa i wyśle hasło mailem, loguje się od razu tym samym kontem domenowym, którym loguje się do komputera i do poczty, a Zabbix sam rozpoznaje po jego grupie w Active Directory, że powinien dostać rolę administratora.

Pojęcie: LDAP. Protokół odpytywania katalogu z kontami, grupami i innymi obiektami organizacyjnymi. Active Directory na `dc01` i `dc02` udostępnia swoje dane właśnie przez LDAP : Zabbix, zamiast trzymać osobną bazę haseł dla każdego użytkownika, o każdym logowaniu

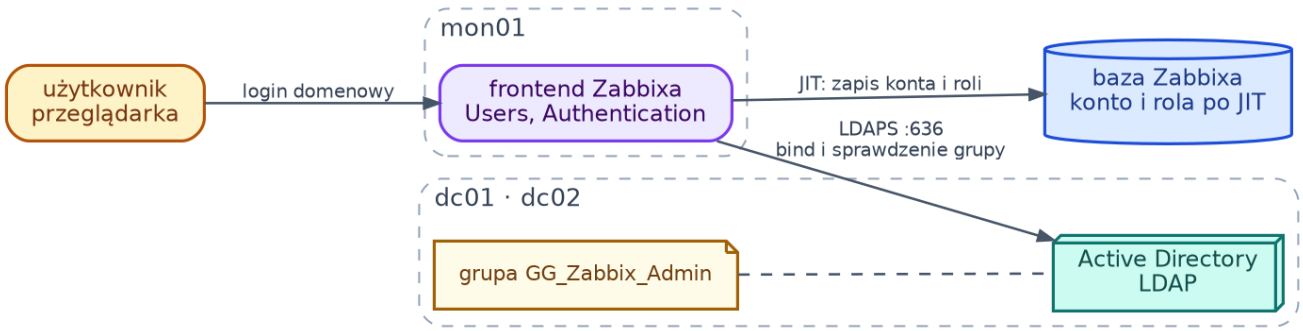
pyta katalog, czy takie konto istnieje, czy podane hasło jest poprawne, i do jakich grup to konto należy.

Pojęcie: provisioning JIT (just in time). Mechanizm w Zabbiksie, sprawdzony w dokumentacji dla wersji 7.0, automatycznie zakładający albo aktualizujący konto użytkownika Zabbixa w chwili pierwszego udanego logowania przez LDAP, na podstawie grup, do których to konto należy w katalogu. Bez JIT provisioningu administrator musiałby ręcznie założyć w Zabbiksie konto dla każdej osoby, zanim ta osoba w ogóle mogłaby się zalogować, nawet jeśli LDAP samo uwierzytelnianie hasła już obsługuje.

Konfigurację wykonujemy zalogowani do frontendu Zabbixa kontem `Admin` z uprawnieniami Super Admin. Ścieżki klikania w interfejsie zapisujemy jako osobne bloki tekstu, zrzuty ekranu dostarcza autor po konfiguracji.

Wymaganie	Uwaga
Konto techniczne w AD do bind LDAP	zakładamy w kroku 1, prawo tylko do odczytu katalogu
Grupa `GG_Zabbix_Admin` w AD	zakładamy w kroku 1, osoby w tej grupie dostają rolę Super Admin
LDAPS na porcie 636 osiągalny z `mon01` do `dc01`/`dc02`	sprawdzamy zaporą, zanim zaczniemy konfigurację w Zabbiksie
Certyfikat firmowego CA zaufany przez system operacyjny `mon01`	bez niego połączenie LDAPS kończy się błędem certyfikatu

Uważaj: konto techniczne wskazane przy bind LDAP powinno mieć wyłącznie prawo odczytu katalogu, na tej samej zasadzie najmniejszych uprawnień, na jakiej opieramy konta monitorujące bazy danych opisane na stronie Monitorowanie typowych usług. Zabbix nie potrzebuje żadnego prawa zapisu w Active Directory do samego uwierzytelniania użytkowników.



Frontend Zabbixa łączy się z Active Directory na dc01 po LDAPS, a provisioning JIT zapisuje konto i rolę na podstawie grupy GG_Zabbix_Admin (kliknięcie otwiera pełny rozmiar)

Przebieg

1. Konto techniczne i grupa w Active Directory

Zakładamy konto techniczne do bind LDAP, z hasłem nieulegającym wygaśnięciu i bez prawa jego zmiany przez samo konto:

```
New-ADUser -Name "svc-zabbix-ldap" `
  -SamAccountName "svc-zabbix-ldap" `
  -UserPrincipalName "svc-zabbix-ldap@ad.zielona.example" `
  -Path "OU=KontaTechniczne,DC=ad,DC=zielona,DC=example" `
  -AccountPassword (Read-Host -AsSecureString "Haslo konta") `
  -Enabled $true `
  -PasswordNeverExpires $true `
  -CannotChangePassword $true
```

Zakładamy grupę dla administratorów Zabbixa, zgodnie z konwencją nazw grup domenowych Zielonej opisaną na stronie Struktura OU Zielonej i delegowanie uprawnień:

```
New-ADGroup -Name "GG_Zabbix_Admin" -GroupScope Global -GroupCategory Security -Path
"OU=Grupy,DC=ad,DC=zielona,DC=example"
```

Dodajemy Tomka do grupy:

```
Add-ADGroupMember -Identity "GG_Zabbix_Admin" -Members "tomek.kowalski"
```

2. Dodanie serwera LDAP: Users → Authentication → LDAP settings

```
Users → Authentication → LDAP settings → Servers → Create LDAP server
Nazwa: AD Zielona
```

```
Host: dc01.ad.zielona.example
Port: 636
Base DN: DC=ad,DC=zielona,DC=example
Search attribute: sAMAccountName
Bind DN: ZIELONA\svc-zabbix-ldap
Bind password: (hasło konta z kroku 1)
Zaznaczone: Advanced configuration, Enable connection encryption (LDAPS)
```

> **Pojęcie: Base DN.** Punkt w drzewie katalogu, od którego Zabbix zaczyna szukać kont i grup. U nas to cała domena `ad.zielona.example`, bo konta użytkowników i grupy `GG_*` leżą w różnych jednostkach organizacyjnych pod tym samym korzeniem.

> **Pojęcie: search attribute.** Atrybut LDAP, po którym Zabbix dopasowuje login wpisany na ekranie logowania do konkretnego konta w katalogu. `sAMAccountName` to nazwa konta w starym formacie `ZIELONA\jan.nowak`, dokładnie ta, której pracownicy Zielonej używają już do logowania na stacjach roboczych.

3. Włączenie provisioningu JIT: zakładka Provisioning

```
Users → Authentication → LDAP settings → Servers → AD Zielona → Provisioning
Enable JIT provisioning: zaznaczone
Group configuration: memberOf
Group name attribute: cn
User group membership attribute: memberOf
User name attribute: givenName
User last name attribute: sn
```

> **Pojęcie: memberOf.** Atrybut konta użytkownika w Active Directory, zawierający listę grup, do których to konto należy. Wybranie metody `memberOf` w provisioningu oznacza, że Zabbix odczytuje grupy wprost z tego atrybutu konta, zamiast osobno przeszukiwać obiekty grup w katalogu.

4. Mapowanie grup na role: tabela Group mapping

```
Users → Authentication → LDAP settings → Servers → AD Zielona → Provisioning → Group mapping →
Create group mapping
Group pattern DN: CN=GG_Zabbix_Admin,OU=Grupy,DC=ad,DC=zielona,DC=example
User group: Zabbix administratorzy
Role: Super admin role
```

Powtarzamy dla zwykłych użytkowników, jeśli chcemy, żeby dowolna osoba z domeny mogła się zalogować z ograniczoną rolą:

```
Group pattern DN: CN=Domain Users,CN=Users,DC=ad,DC=zielona,DC=example
User group: Zabbix uzytkownicy
Role: User role
```

Uważaj: kolejność wpisów w tabeli mapowania ma znaczenie tylko wtedy, gdy jedno konto pasuje do kilku wzorców naraz. Osoba należąca jednocześnie do `GG_Zabbix_Admin` i do `Domain Users` dostaje sumę uprawnień z obu dopasowanych grup, nie tylko z pierwszej pasującej reguły.

5. Mapowanie danych kontaktowych: zakładka Media type mapping

```
Users → Authentication → LDAP settings → Servers → AD Zielona → Provisioning → Media type
mapping → Create media type mapping
Media type: Email
Attribute: mail
```

Dzięki temu adres e-mail wpisany w Active Directory trafia od razu do profilu nowo założonego konta Zabbixa, bez ręcznego uzupełniania go po pierwszym logowaniu.

6. Włączenie LDAP jako sposobu logowania: zakładka Authentication

```
Users → Authentication → Authentication
Default authentication: LDAP
Deprovisioned users group: Disabled
```

> **Pojęcie: grupa deprowizjonowanych użytkowników.** Grupa Zabbixa, do której provisioning JIT przenosi konto, gdy przy kolejnym logowaniu okazuje się, że dana osoba nie należy już do żadnej zmapowanej grupy w Active Directory, na przykład po odejściu z zespołu Oli. Wskazujemy tu grupę o statusie `Disabled`, żeby takie konto od razu traciło dostęp, zamiast wisieć aktywne z ostatnią przypisaną rolą.

Weryfikacja

Wylogowujemy się z konta `Admin` i logujemy testowo kontem `tomek.kowalski` z hasłem domenowym:

```
Login: tomek.kowalski
Password: (hasło domenowe Tomka)
```

Sprawdzamy we frontendzie, że konto zostało założone automatycznie z właściwą rolą:

```
Users → Users → tomek.kowalski
```

Przykład:

```
Username: tomek.kowalski
User type: Super Admin
User groups: Zabbix administratorzy
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran `Users → Users` z widocznym wierszem `tomek.kowalski`, kolumną `User type` ustawioną na `Super Admin` i kolumną `Groups` pokazującą `Zabbix administratorzy`, założonym automatycznie po pierwszym logowaniu LDAP.

Jeśli chcemy sprawdzić provisioning bez czekania na kolejne logowanie danej osoby, uruchamiamy je ręcznie:

```
Users → Users → tomek.kowalski → Provision now
```

Częste błędy

Objaw	Przyczyna	Co robimy
Logowanie LDAP zwraca „Login name or password is incorrect" dla poprawnego hasła	zły `Bind DN` albo hasło konta technicznego, Zabbix w ogóle nie może przeszukać katalogu	testujemy bind osobno narzędziem `ldapsearch` z hosta `mon01`, poprawiamy dane w konfiguracji serwera LDAP
Konto zakłada się, ale zawsze z rolą `User`, nigdy `Super Admin`	wzorzec `Group pattern DN` w mapowaniu grup nie pasuje do rzeczywistego DN grupy `GG_Zabbix_Admin`	sprawdzamy dokładny DN grupy przez `Get-ADGroup GG_Zabbix_Admin`, poprawiamy wzorzec
Połączenie z LDAP kończy się błędem certyfikatu	certyfikat CA Zielonej niezaufany przez system operacyjny `mon01`	kopiujemy `CA-Zielona.crt` do `/etc/pki/ca-trust/source/anchors/` i wykonujemy `sudo update-ca-trust`, zgodnie ze stroną Certyfikaty SSL, wystawienie, konwersja, weryfikacja
Osoba usunięta z grupy `GG_Zabbix_Admin` nadal ma uprawnienia administratora w Zabbiksie	provisioning JIT uruchamia się dopiero przy kolejnym logowaniu, nie od razu po zmianie w AD	dla pilnych przypadków wyłączamy konto ręcznie w `Users → Users`, zamiast czekać na naturalne wygaśnięcie sesji

> Po przenosinach urzędu według strony Przenosiny urzędu z dc01 bez przerwy: dwa urzędy równolegle i wygaszenie CA-Zielona dokładamy tu w ten sam sposób zaufanie do `Zielona-Root-CA`, a `CA-Zielona.crt` zostaje obok do wygaszenia starego urzędu.

Ćwiczenia

1. Osoba należy jednocześnie do grup `GG_Zabbix_Admin` i `Domain Users`, obie zmapowane w tabeli grup na różne role. Jaką rolę dostanie po zalogowaniu?

Odpowiedź

Dostanie sumę uprawnień z obu dopasowanych grup, nie tylko z pierwszej reguły na liście. W praktyce, jeśli jedna z dopasowanych ról to Super Admin, konto końcowo działa z uprawnieniami Super Admin, bo to najszersza z przypisanych ról.

1. Dlaczego samo włączenie LDAP jako `Default authentication` bez skonfigurowania provisioningu JIT nie wystarczy, żeby nowa osoba mogła się zalogować pierwszy raz?

Odpowiedź

Bez provisioningu JIT LDAP obsługuje wyłącznie sprawdzenie hasła, ale konto użytkownika w bazie Zabbixa musi istnieć, zanim ktokolwiek się nim zaloguje. Administrator musiałby więc ręcznie założyć konto w Zabbiksie dla każdej nowej osoby, dokładnie ten ręczny krok, który provisioning JIT eliminuje.

Powiązane: Instalacja serwera Zabbix z pakietów na Rocky Linux 9 · Struktura OU Zielonej i delegowanie uprawnień · Monitorowanie typowych usług

Źródła: Authentication, dokumentacja Zabbix · LDAP, dokumentacja Zabbix · Just-in-Time user provisioning explained, blog Zabbix