Procedury i wdrożenia

Doprowadzenie systemu do konkretnego stanu: dołączenie do domeny, instalacja sterownika, konfiguracja usługi. Kroki, weryfikacja, wycofanie zmian.

- Instalacja Rocky Linux 9 z obrazu ISO do pierwszego logowania
- Dołączenie systemu Linux do domeny Active Directory (realmd + SSSD)
- Instalacja sterownika ODBC do MSSQL na RHEL/Rocky
- Przykładowe konfiguracje ODBC dla różnych baz danych
- Instalacja Rocky Linux z własnym nazewnictwem VG i LV
- Standardowa aktualizacja systemu produkcyjnego

Instalacja Rocky Linux 9 z obrazu ISO do pierwszego logowania

□□ Typ: procedura **□□ Dotyczy: Rocky Linux 9, Hyper-V na hv01, instalator Anaconda**

□ Czas: ~45 min

Efekt: nowa maszyna wirtualna na klastrze Hyper-V, z zainstalowanym od zera systemem Rocky Linux 9, zweryfikowanym obrazem instalacyjnym i kontem administratora gotowym do pierwszego logowania.

□□ Skróty użyte na tej stronie (7)

Skrót	Co znaczy
DHCP	Dynamic Host Configuration Protocol: usługa, która sama rozdaje komputerom adresy IP
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
ISO	International Organization for Standardization: organizacja, która wydaje normy techniczne używane na całym świecie
SCSI	Small Computer System Interface: standard podłączania dysków
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
UEFI	Unified Extensible Firmware Interface: nowoczesne oprogramowanie firmowe płyty głównej albo maszyny wirtualnej, następca starszego BIOS-u, obsługuje między innymi Secure Boot
VLAN	Virtual LAN: wydzielona logicznie sieć na tym samym sprzęcie

Zanim zaczniemy

□□ Z ŻYCIA WZIĘTE

> Trzeci tydzień pracy Tomka. Dostaje pierwsze samodzielne zadanie: postawić nowy serwer Rocky Linux 9 pod aplikacje zespołu Oli, od pobrania obrazu do pierwszego logowania. Wcześniej patrzył tylko, jak robi to ktoś inny.

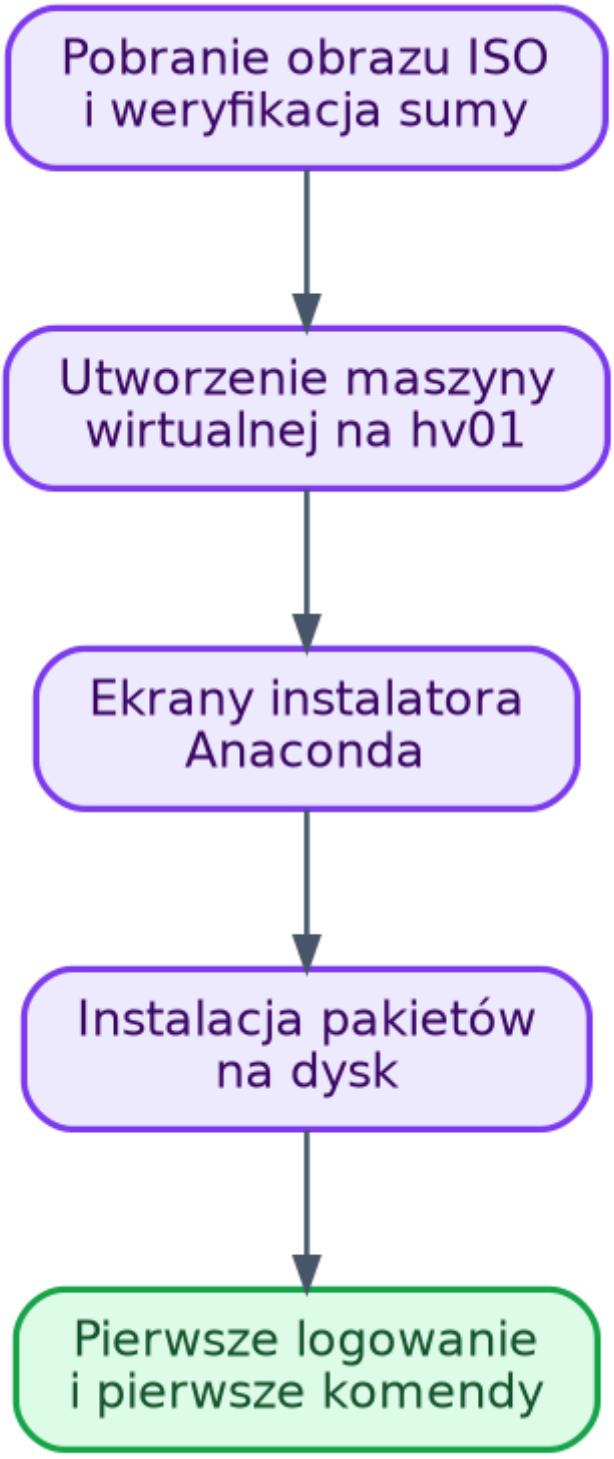
Komendy w tej procedurze wykonujemy w trzech miejscach: w przeglądarce (pobranie obrazu), w konsoli Hyper-V Manager na `hv01` (maszyna wirtualna) i w oknie instalatora Anaconda (sam Linux). Weryfikację sumy kontrolnej robimy na komputerze, z którego pobieramy obraz, więc w zależności od systemu użyjemy `sha256sum` albo `Get-FileHash`.

Wymaganie	Uwaga
Dostęp do Hyper-V Manager na `hv01` lub `hv02`	konto w grupie administratorów Hyper-V
Wolne miejsce na klastrze	licząc dysk systemowy i dysk pod dane
Adres IP i nazwa hosta uzgodnione wcześniej	zgodnie z konwencją nazw serwerów Zielonej
Dostęp do sieci z VLAN serwerów	do pobrania pakietów po instalacji

Konwencja partycjonowania: ta strona pokazuje domyślne, automatyczne partycjonowanie, wystarczające pod proste role. Szczegółowe partycjonowanie w konwencji Zielonej, czyli osobna grupa wolumenów na system, na `/var` i na dane (`vgOS`/`lvOS`, `vgVAR`/`lvVAR` i osobna grupa pod dane), opisuje strona Instalacja Rocky z osobnymi grupami wolumenów. Gdy nowy serwer ma trzymać bazę danych albo duży wolumen danych, przechodzimy do tamtej strony zaraz po ekranie **Installation Destination** poniżej, zamiast korzystać z automatycznego partycjonowania.

> Układ dysków zapisany podczas instalacji jest praktycznie nieodwracalny bez reinstalacji. Jeśli nowy serwer ma jasno określoną rolę (baza danych, duży magazyn plików), decydujemy o partycjonowaniu przed rozpoczęciem instalacji, nie po niej.

Przebieg



Kolejność etapów instalacji: pobranie obrazu, maszyna wirtualna, instalator, instalacja, pierwsze logowanie (kliknięcie otwiera pełny rozmiar)

1. Pobranie obrazu ISO

Wchodzimy na stronę projektu i pobieramy jeden z dwóch wariantów obrazu:

```
rockylinux.org/download
```

Wariant	Rozmiar	Kiedy wybieramy
Minimal ISO	około 1,7 GB	serwery bez środowiska graficznego, nasz domyślny wybór
DVD ISO	około 10 GB	gdy chcemy mieć od razu pod ręką pełny zestaw pakietów bez internetu

W Zielonej instalujemy niemal zawsze z obrazu Minimal, a brakujące pakiety dokładamy po instalacji przez `dnf`.

2. Weryfikacja sumy kontrolnej pobranego obrazu

Obraz mógł uszkodzić się w trakcie pobierania, więc zanim ruszymy dalej, sprawdzamy jego sumę kontrolną. Na komputerze z Linuksem albo macOS:

```
sha256sum Rocky-9.4-x86_64-minimal.iso
```

Przykład:

```
a3f1c9e8b7d6...  Rocky-9.4-x86_64-minimal.iso
```

Na komputerze z Windows, w PowerShellu:

```
Get-FileHash .\Rocky-9.4-x86_64-minimal.iso -Algorithm SHA256
```

Wynik porównujemy z plikiem `CHECKSUM`, dostępnym na tej samej stronie pobierania, obok wybranego obrazu. Zgadzają się tylko pierwsze i ostatnie kilka znaków przy pobieżnym rzucie okiem, warto porównać całość.

3. Utworzenie maszyny wirtualnej w Hyper-V na hv01

Otwieramy Hyper-V Manager na `hv01`, łączymy się z hostem i uruchamiamy kreator nowej maszyny:

```
Hyper-V Manager → hv01 → Action → New → Virtual Machine
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran kreatora New Virtual Machine Wizard, strona Specify Generation, z zaznaczoną opcją Generation 2.

W Zielonej każda nowa maszyna linuksowa powstaje jako **generacja 2**: obsługuje UEFI i Secure Boot, w przeciwieństwie do starszej generacji 1 opartej na klasycznym BIOS-ie. Po utworzeniu maszyny wchodzimy w jej ustawienia i sprawdzamy szablon rozruchu zabezpieczonego:

```
Ustawienia maszyny wirtualnej → Security → Secure Boot Template → Microsoft UEFI Certificate Authority
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: karta Security w ustawieniach maszyny wirtualnej, z zaznaczonym Secure Boot i wybranym szablonem Microsoft UEFI Certificate Authority.

> Szablon Microsoft UEFI Certificate Authority to jedyny, który podpisuje rozruch Rocky Linuksa (i większości dystrybucji linuksowych) w Hyper-V. Domyślny szablon Windows tu nie zadziała.

Do maszyny podłączamy obraz ISO jako napęd DVD i dokładamy dwa dyski twarde: pierwszy pod system, drugi pod dane.

```
Ustawienia maszyny wirtualnej → SCSI Controller → Hard Drive → Nowy dysk twardy, format VHDX
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: karta ustawień z dwoma podłączonymi dyskami SCSI, opisanymi jako System i Dane, oraz podłączonym napędem DVD ze wskazanym plikiem ISO.

4. Uruchomienie instalatora i wybór języka

Uruchamiamy maszynę i łączymy się z jej konsolą. Pierwszy ekran instalatora Anaconda pyta o język, w którym poprowadzi resztę instalacji:

```
WELCOME TO ROCKY LINUX 9 → Polski → Continue
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran powitalny instalatora z zaznaczonym językiem polskim na liście po lewej stronie.

5. Klawiatura i strefa czasowa

Na głównym ekranie instalatora (Installation Summary) wchodzimy kolejno w dwie pozycje. Układ klawiatury:

```
Keyboard → dodajemy Polski (programisty) → usuwamy zbędne układy → Done
```

Strefa czasowa:

```
Time & Date → Region: Europe, City: Warsaw → Network Time: włączone → Done
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran Time & Date z wybraną strefą Europe/Warsaw i włączonym pobieraniem czasu z sieci.

6. Źródło instalacji

Sprawdzamy, że instalator widzi podłączony obraz ISO jako źródło pakietów:

```
Installation Source → Local media (wykryte automatycznie) → Done
```

Gdy zamiast tego pojawi się pole do wpisania adresu, oznacza to zwykle problem z podłączeniem napędu DVD w ustawieniach maszyny z kroku 3.

7. Wybór oprogramowania

W Zielonej instalujemy zawsze wariant minimalny, a resztę pakietów dokładamy świadomie po instalacji:

```
Software Selection → Base Environment: Minimal Install → Done
```

ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran Software Selection z zaznaczonym Minimal Install i pustą kolumną dodatków po prawej stronie.

8. Miejsce docelowe

```
Installation Destination → wybrany dysk systemowy → Storage Configuration: Automatic → Done
```

> To jest miejsce, w którym dla serwera pod bazę danych albo duży magazyn plików przechodzimy zamiast tego do procedury Instalacja Rocky z osobnymi grupami wolumenów, zaczynając od terminala instalatora (`CTRL+ALT+F3`) opisanego na tamtej stronie.

Drugi dysk, podłączony pod dane w kroku 3, na tym etapie zostawiamy niesformatowany.

Zamontujemy go później, zgodnie ze stroną Montowanie: mount, /etc/fstab, zasoby sieciowe.

9. Sieć i nazwa hosta

```
Network & Host Name → karta sieciowa: włączona (ON) → Host name: NazwaSerwera.zielona.example
→ Configure...
```

> (gdzie `NazwaSerwera` to ustalona wcześniej nazwa nowego serwera, na przykład `app02`)

W oknie **Configure** ustawiamy adres statyczny zamiast DHCP, zgodnie z pulą adresów VLAN serwerów:

```
IPv4 Settings → Method: Manual → Addresses: adres, maska, brama → Save
```

ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran Network & Host Name z włączoną kartą sieciową, wpisaną nazwą hosta i widocznym adresem IP po lewej stronie.

10. Hasło roota i konto administratora

```
Root Password → hasło zgodne z polityką haseł Zielonej → Done
```

W tym samym miejscu możemy zablokować logowanie roota przez SSH, zostawiając je tylko z konsoli. Zaraz obok tworzymy konto, z którego będziemy pracować na co dzień:

```
User Creation → Full name, User name → Make this user administrator: zaznaczone → Done
```

> Zaznaczenie „Make this user administrator" dodaje konto do grupy `wheel`, czyli daje mu prawo do `sudo` bez ręcznej edycji `/etc/sudoers` po instalacji.

□□ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran User Creation z wypełnionym imieniem i nazwiskiem, nazwą konta i zaznaczonym polem administratora.

11. Rozpoczęcie instalacji

Gdy wszystkie pozycje na ekranie Installation Summary mają odznaczone ostrzeżenia, zatwierdzamy warunki licencji i zaczynamy instalację:

```
License Information → I accept the license agreement → Done → Begin Installation
```

Instalacja pakietów z obrazu Minimal trwa zwykle kilka minut. Po jej zakończeniu wybieramy **Reboot System**, a instalator sam odłącza obraz ISO od maszyny.

12. Pierwsze logowanie i pierwsze komendy

Logujemy się kontem administratora utworzonym w kroku 10 i sprawdzamy podstawowe informacje o nowym serwerze:

```
hostnamectl
```

Przykład:

```
 Static hostname: app02.zielona.example
       Icon name: computer-vm
         Chassis: vm
```

```
      Machine ID: 3f2a1e...
   Operating System: Rocky Linux 9.4 (Blue Onyx)
        Kernel: Linux 5.14.0-427.el9.x86_64
```

```
ip a
```

Sprawdzamy, czy adres z kroku 9 rzeczywiście się przypisał, oraz czy interfejs jest aktywny (`state UP`).

```
sudo dnf update
```

Pierwsza aktualizacja po instalacji, jeszcze przed dołożeniem serwera do jakichkolwiek automatyzacji.

Weryfikacja

```
hostnamectl
```

Sprawdzamy, czy nazwa hosta i wersja systemu zgadzają się z tym, co ustawiliśmy w kroku 9.

```
ip a
```

Sprawdzamy, czy adres IP jest tym samym, który przypisaliśmy w instalatorze, i czy nie ma go przypadkiem na dwóch interfejsach naraz.

```
sudo -l
```

Sprawdzamy, czy konto administratora rzeczywiście ma prawo uruchamiać komendy przez `sudo`.

Przykład:

```
User adm_user may run the following commands on app02:
    (ALL) ALL
```

Wycofanie zmian

Gdy instalacja poszła źle albo maszyna powstała pod złym adresem czy nazwą, prościej jest usunąć ją i zacząć od kroku 3, niż naprawiać każdy ekran po kolei.

```
Hyper-V Manager → wybrana maszyna → Action → Turn Off
```

```
Hyper-V Manager → wybrana maszyna → Action → Delete
```

> Usunięcie maszyny nie usuwa automatycznie plików dysków `.vhdx` z dysku hosta. Sprawdzamy ścieżkę w ustawieniach maszyny (karta Hard Drive) i kasujemy pliki ręcznie, jeśli nie planujemy instalacji od nowa na tych samych dyskach.

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
Suma kontrolna obrazu się nie zgadza	uszkodzone pobieranie	pobieramy obraz ponownie, porównujemy z plikiem CHECKSUM ze strony rockylinux.org
Instalator nie widzi drugiego dysku	dysk pod dane podłączony po uruchomieniu maszyny	wyłączamy maszynę, dodajemy dysk w ustawieniach, uruchamiamy instalator od nowa
Ekran instalatora nie startuje, czarny ekran po starcie VM	brak generacji 2 albo zły szablon Secure Boot	sprawdzamy Generation 2 i szablon Microsoft UEFI Certificate Authority w kroku 3
Installation Source pyta o adres zamiast wykryć obraz	napęd DVD w ustawieniach maszyny nie wskazuje na plik ISO	poprawiamy ścieżkę w ustawieniach SCSI Controller, restartujemy instalator
Brak sieci zaraz po pierwszym logowaniu	karta sieciowa nie została włączona w kroku 9	`nmcli con up NazwaPolaczenia`, sprawdzenie ustawień w `nmtui`

Powiązane: Instalacja Rocky z osobnymi grupami wolumenów · Sieć: adresy IP i trasy · Sudo: uprawnienia administracyjne · Montowanie: mount, /etc/fstab, zasoby sieciowe

Źródła: rockylinux.org/download · dokumentacja instalatora Anaconda · `man 1 hostnamectl` · `man 8 nmcli`

Dołączenie systemu Linux do domeny Active Directory (realmd + SSSD)

Typ: procedura **Dotyczy: RHEL/Rocky 9–10** **Czas: ~30 min**

Efekt: system loguje użytkowników domenowych przez Kerberos, mapuje ich UID/GID przez SSSD, tworzy im katalogi domowe przy pierwszym logowaniu i pozwala wskazanej grupie domenowej używać `sudo`*.*

Skróty użyte na tej stronie (10)

Skrót	Co znaczy
AD	Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)
LDAPS	LDAP szyfrowany TLS, zwykle port 636
PAM	Pluggable Authentication Modules: wymienne moduły logowania w Linuksie, decydują na przykład o sile wymaganego hasła
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SMB	Server Message Block: protokół udostępniania plików i drukarek w sieci Windows, na nim stoi zwykły udział sieciowy
SRV	rekord DNS wskazujący nie adres hosta, tylko usługę i port, pod którym jej szukać, np. kontroler domeny w sieci
SSSD	System Security Services Daemon: usługa na kliencie linuksowym tłumacząca logowanie, sudo i sprawdzanie grup na zapytania do katalogu, czy to FreeIPA, czy Active Directory
UID	User ID: numer użytkownika w Linuksie

Zanim zaczniemy

> **Z ŻYCIA WZIĘTE**
>
> O 9:00 Ola dostaje nowy laptop deweloperski, który ma trafić do domeny ad.zielona.example, żeby mogła logować się swoim kontem firmowym zamiast kontem lokalnym. Tomek uruchamia `realm join`, ale polecenie wisi kilkanaście sekund i kończy się `kinit: Clock skew too great`, bo laptop stał wyłączony w szufladzie kilka tygodni i zegar poszedł w tył. Po ręcznym `chronyc makestep` dołączenie do domeny przechodzi bez dalszych przeszkód.

Wymaganie	Jak sprawdzić
DNS wskazuje na kontrolery domeny (forward i rekordy SRV)	`dig +short _ldap._tcp.NazwaDomeny SRV`
Kontroler domeny odpowiada pod pełną nazwą	`dig +short NazwaKontroleraDC`
Synchronizacja czasu — offset poniżej 5 minut	`chronyc tracking` (pole *System time*)
Porty wychodzące do kontrolera otwarte	`nc -zv NazwaKontroleraDC 88 389 445 464 636 3268`
Nazwa hosta jest pełną, rozwiązywalną nazwą (FQDN)	`hostnamectl`
Mamy konto uprawnione do dołączania komputerów do domeny	konto AD z odpowiednim delegowaniem (`NazwaUseraWDomenie`)

Porty, które musi mieć otwarte firewall między klientem a kontrolerem domeny: **88/tcp+udp** (Kerberos), **389/tcp** (LDAP), **445/tcp** (SMB, używane przez `adcli`), **464/tcp+udp** (zmiana hasła Kerberos), **636/tcp** (LDAPS) i **3268/tcp** (Global Catalog).

Uważaj: dołączenie do domeny nadpisuje `/etc/nsswitch.conf`, `/etc/sssd/sssd.conf` i stos PAM przez `authselect` — na maszynie z aktywnymi sesjami lokalnymi robimy to w oknie serwisowym, bo restart `sssd` zrywa trwające uwierzytelnianie domenowe. Dołączenie tworzy też obiekt komputera w Active Directory; gdy procedura zostanie przerwana w połowie, obiekt może zostać po stronie domeny i wymaga ręcznego usunięcia przez administratora AD.

Przebieg

1. Instalacja pakietów

```
sudo dnf install -y realmd sssd adcli samba-common-tools oddjob oddjob-mkhomedir \
  samba-common krb5-workstation openldap-clients
```

Przykład:

```
Installed:
  adcli-0.9.2-6.el9.x86_64          oddjob-0.34.7-8.el9.x86_64
  oddjob-mkhomedir-0.34.7-8.el9     realmd-0.17.1-11.el9.x86_64
  samba-common-tools-4.19.4-...     sssd-2.9.4-...
Complete!
```

`realmd sssd adcli samba-common-tools oddjob oddjob-mkhomedir` to minimalny zestaw wymagany do dołączenia i zarządzania kontem domenowym; `oddjob-mkhomedir` odpowiada konkretnie za tworzenie katalogów domowych przy pierwszym logowaniu. `krb5-workstation` i `openldap-clients` nie są niezbędne do samego dołączenia, ale przydają się do ręcznej diagnostyki (`kinit`, `ldapsearch`).

2. Rozpoznanie domeny

```
realm discover NazwaDomeny
```

Przykład:

```
ad.zielona.example
  type: kerberos
  realm-name: AD.ZIELONA.EXAMPLE
  domain-name: ad.zielona.example
  configured: no
  server-software: active-directory
  client-software: sssd
  required-package: oddjob-mkhomedir
  required-package: sssd
  required-package: adcli
  required-package: samba-common-tools
```

`realm discover` pyta DNS o rekordy SRV domeny i sprawdza, czy potrzebne pakiety są zainstalowane. `configured: no` to oczekiwany stan przed kolejnym krokiem — system jeszcze nie jest dołączony. Gdy polecenie nie zwraca nic, problem leży w DNS (patrz *Gdy coś nie zadziała*).

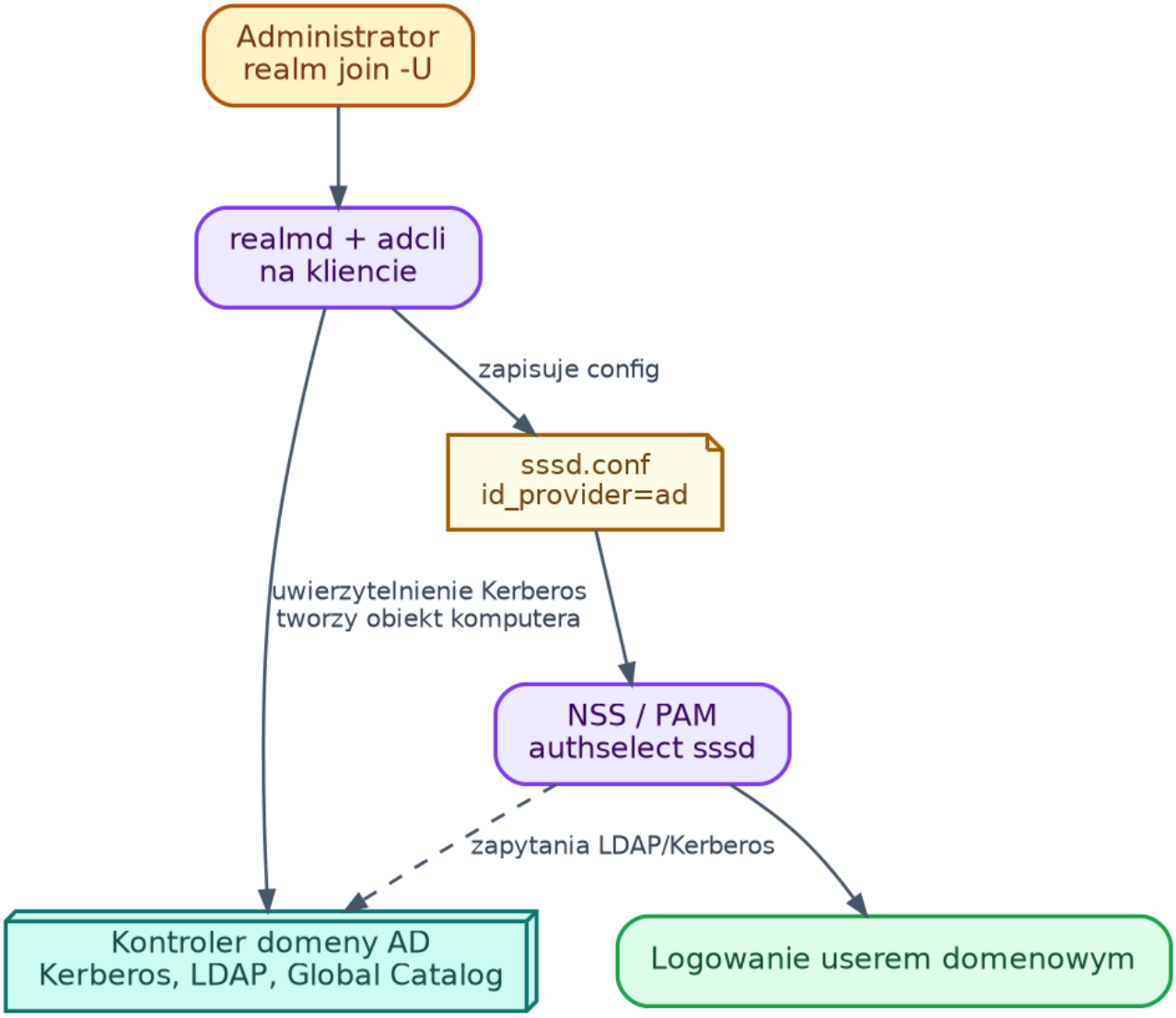
3. Dołączenie do domeny

```
sudo realm join -U NazwaUseraWDomenie NazwaDomeny
```

Przykład:

```
Password for adm_user:
```

`realm join` prosi o hasło konta wskazanego przez `-U`, wykonuje uwierzytelnienie Kerberos, tworzy obiekt komputera w AD i sam dopisuje potrzebne sekcje do `/etc/sssd/sssd.conf` oraz `/etc/nsswitch.conf`. Polecenie kończy się bez żadnego komunikatu przy powodzeniu — brak błędu to sukces.



realmd i adcli uwierzytelniają się w AD, tworzą obiekt komputera i zapisują sssd.conf, a od tej chwili NSS i PAM pytają kontroler domeny przy każdym logowaniu (kliknięcie otwiera pełny rozmiar)

4. Weryfikacja dołączenia

```
realm list
```

Przykład:

```
ad.zielona.example
  type: kerberos
  realm-name: AD.ZIELONA.EXAMPLE
  domain-name: ad.zielona.example
  configured: kerberos-member
  server-software: active-directory
  client-software: sssd
  required-package: oddjob-mkhomedir
  login-formats: %U@ad.zielona.example
  login-policy: allow-realm-logins
```

`configured: kerberos-member` potwierdza, że system jest teraz członkiem domeny. `login-policy: allow-realm-logins` oznacza, że w tej chwili **każdy** użytkownik domenowy może się zalogować — krok 8 to zawęzi.

5. Konfiguracja SSSD dla spójnego logowania

```
sudo vim /etc/sssd/sssd.conf
```

Weryfikujemy, czy sekcje dopisane automatycznie przez `realm join` plus nasze ręczne poprawki dają w efekcie taki plik:

```
[sssd]
domains = NazwaDomeny
config_file_version = 2
services = nss, pam

[domain/NazwaDomeny]
ad_domain = NazwaDomeny
krb5_realm = NAZWADOMENY
realmd_tags = manages-system joined-with-adcli
id_provider = ad
access_provider = ad
ldap_id_mapping = True
```

```
ad_hostname = NazwaKontroleraDC
dyndns_update = true
dyndns_refresh_interval = 43200
dns_update_ptr = true
dyndns_ttl = 3600
dyndns_auth = GSS-TSIG

use_fully_qualified_names = True
fallback_homedir = /home/%u@%d
override_shell = /bin/bash

krb5_store_password_if_offline = True
cache_credentials = True
```

> (gdzie `NazwaDomeny` to pełna nazwa domeny, np. `ad.zielona.example`, a `NAZWADOMENY` to ta sama nazwa wielkimi literami — tak Kerberos zapisuje realm)

`use_fully_qualified_names` wymusza logowanie jako `user@NazwaDomeny`, żeby nie kolidować z kontami lokalnymi o tych samych nazwach. `fallback_homedir` ustala wzorzec katalogu domowego zgodny z tym formatem. `override_shell` wymusza jedną powłokę logowania dla wszystkich userów domenowych, niezależnie od tego, co ma ustawione w atrybucie AD — przydaje się, gdy w AD nikt nigdy nie wypełnił pola powłoki i domyślnie wychodziłby `/bin/false`.

Plik ma prawa `600` i właściciela `root` — jeśli `realm join` je zmieni, przywracamy je przed restartem usługi:

```
sudo chmod 600 /etc/sssd/sssd.conf
```

6. Aktywacja tworzenia katalogów domowych

```
sudo authselect select sssd with-mkhomedir --force
```

Przykład:

```
Profile "sssd" was selected.
The following nsswitch maps are overwritten by the profile:
```

```
- passwd
- group
```

`authselect` podmienia konfigurację PAM i NSS na profil `sssd`, a opcja `with-mkhomedir` dopisuje moduł `pam_oddjob_mkhomedir`, który tworzy katalog domowy przy pierwszym udanym logowaniu. Bez tego kroku logowanie się powiedzie, ale user wyląduje bez katalogu domowego.

7. Restart SSSD

```
sudo systemctl restart sssd
sudo systemctl enable sssd
```

Restart wczytuje nową konfigurację; `enable` gwarantuje, że usługa wstanie po reboocie. Na tym etapie logowanie kontem domenowym już powinno działać, ale jeszcze bez sudo.

8. Nadanie sudo grupie domenowej

```
sudo realm permit -g 'GrupaWdomenie@NazwaDomeny'
echo '%GrupaWdomenie@NazwaDomeny ALL=(ALL) NOPASSWD:ALL' | sudo tee /etc/sudoers.d/ad-linux-admins
sudo chmod 440 /etc/sudoers.d/ad-linux-admins
sudo visudo -c
```

Przykład:

```
/etc/sudoers.d/ad-linux-admins: parsed OK
```

`realm permit -g` zawęża `login-policy` z kroku 4 do konkretnej grupy — od tej chwili logować się mogą tylko jej członkowie, a nie cała domena. Osobny plik w `/etc/sudoers.d/` zamiast edycji `/etc/sudoers` przez `visudo` bezpośrednio ułatwia wycofanie zmiany i nie miesza się z lokalną konfiguracją sudo. `visudo -c` sprawdza składnię przed użyciem — literówka w tym pliku blokuje **każde** wywołanie `sudo` w systemie.

9. Test logowania

```
id NazwaUseraWDomenie@NazwaDomeny
```

Przykład:

```
uid=1458001107(adm_user@ad.zielona.example) gid=1458000513(domain users@ad.zielona.example)
groups=1458000513(domain users@ad.zielona.example),1458001205(linux-admins@ad.zielona.example)
```

```
su - NazwaUseraWDomenie@NazwaDomeny
```

Przykład:

```
Creating home directory for adm_user@ad.zielona.example.
[adm_user@srv01 ~]$
```

`id` potwierdza, że SSSD widzi konto i poprawnie mapuje jego grupy — obecność grupy domenowej na liście grup oznacza, że krok 8 zadziała. Komunikat „Creating home directory" przy pierwszym `su`/logowaniu to dowód, że `oddjob-mkhomedir` z kroku 6 działa.

Weryfikacja

```
id NazwaUseraWDomenie@NazwaDomeny && sudo -l -U NazwaUseraWDomenie@NazwaDomeny 2>/dev/null;
getent passwd NazwaUseraWDomenie@NazwaDomeny
```

Przykład:

```
uid=1458001107(adm_user@ad.zielona.example) gid=1458000513(domain users@ad.zielona.example)
groups=...
adm_user@ad.zielona.example:*:1458001107:1458000513:adm_user:/home/adm_user@ad.zielona.example
:/bin/bash
```

`getent passwd`, czytający pełny wpis (nie tylko `id`), potwierdza, że NSS poprawnie sięga do SSSD, a nie tylko PAM podczas logowania. Katalog domowy z kroku 9 (`ls -ld /home/NazwaUseraWDomenie@NazwaDomeny`) powinien istnieć i należeć do tego użytkownika.

Wycofanie zmian

```
sudo realm leave NazwaDomeny
sudo rm -f /etc/sudoers.d/ad-linux-admins
sudo authselect select sssd --force
sudo systemctl restart sssd
```

`realm leave` usuwa system z domeny lokalnie i, jeśli konto użyte do wykonania polecenia ma odpowiednie uprawnienia, kasuje też obiekt komputera po stronie AD — w przeciwnym razie zostaje on osierocony i administrator domeny musi go usunąć ręcznie z konsoli AD. Sekcje domeny w `/etc/sssd/sssd.conf` zostają w pliku po `realm leave`; jeśli maszyna ma już nigdy nie wracać do domeny, usuwamy je ręcznie i przywracamy `authselect select sssd` bez `with-mkhomedir` (albo `authselect select minimal`, jeśli w ogóle nie ma być SSSD).

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
`realm discover` nie zwraca nic albo `realm: Couldn't find hostname`	Brak rekordów SRV domeny w DNS albo klient pyta zły serwer DNS	`dig +short _ldap._tcp.NazwaDomeny SRV`; poprawiamy `/etc/resolv.conf` albo konfigurację DHCP, żeby wskazywała na DNS kontrolerów domeny
`realm join` kończy się `kinit: Clock skew too great`	Rozjazd czasu między klientem a kontrolerem domeny powyżej 5 minut (limit Kerberosa)	`chronyc tracking`; jeśli offset duży, `sudo chronyc makestep` albo poprawiamy `/etc/chrony.conf` i czekamy na synchronizację
`realm join` kończy się `Necessary packages are not installed`	Brakuje jednego z pakietów wymaganych przez `realm discover` (najczęściej `oddjob-mkhomedir` albo `adcli`)	Instalujemy brakujący pakiet z komunikatu błędu, powtarzamy `realm join`
Logowanie działa, ale użytkownik ląduje bez katalogu domowego / `No directory, logging in with HOME=/`	`authselect` nie ma aktywnego profilu `with-mkhomedir`, albo `oddjobd` nie działa	`authselect current` sprawdza aktywny profil; `sudo systemctl status oddjobd`; powtarzamy krok 6 z `--force`

Powiązane: Uwierzytelnianie PAM i authselect · SSSD: mapowanie ID i cache poświadczeń · sudo i /etc/sudoers.d

Źródła: `man 8 realm` · `man 5 sssd.conf` · `man 8 authselect` · Red Hat Enterprise Linux — *Integrating RHEL Systems Directly with Windows Active Directory*

Instalacja sterownika ODBC do MSSQL na RHEL/Rocky

Typ: procedura **Dotyczy: RHEL/Rocky 7–9 (i pochodne)** **Czas: ~15 min**

Efekt: system ma zainstalowany sterownik ODBC 18 do Microsoft SQL Server, narzędzia `sqlcmd`/`bcp` dostępne w `PATH`, i zweryfikowane połączenie testowe do bazy.

Skróty użyte na tej stronie (8)

Skrót	Co znaczy
B2B	business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych
CA	Certificate Authority: urząd certyfikacji, który podpisuje certyfikaty
DSN	Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC
MSSQL	Microsoft SQL Server: silnik bazy danych Microsoftu, na nim w Zielonej stoi księgowość Optima
ODBC	Open Database Connectivity: uniwersalny sposób łączenia się z różnymi bazami danych tym samym interfejsem
PHP	PHP: Hypertext Preprocessor, język programowania, w którym napisany jest frontend Zabbixa i wiele innych aplikacji webowych
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Zanim zaczniemy

> **Z ŻYCIA WZIĘTE**
>
> O 10:00 Ola wdraża na app01 nową integrację B2B, która musi łączyć się z bazą MSSQL na sql01. Pierwsze zapytanie przez `sqlcmd` kończy się błędem `SSL Provider: certificate verify failed`, bo sql01 ma jeszcze certyfikat self-signed sprzed migracji z testów. Tomek dopisuje `TrustServerCertificate=Yes` do DSN i integracja rusza jeszcze tego samego dnia.

Wymaganie	Jak sprawdzić
Wersja RHEL/Rocky to 7, 8 lub 9	`grep VERSION_ID /etc/os-release`
Dostęp do internetu (albo mirrora z pakietami Microsoftu)	`curl -sSI https://packages.microsoft.com`
Sieć do serwera MSSQL na porcie 1433	`nc -zv NazwaSerwera 1433`
Uprawnienia sudo do instalacji pakietów i repozytoriów	`sudo -l`
Lokalizacja systemu ustawiona na UTF-8 (sterownik tego wymaga)	`locale \

Uważaj: instalacja rejestruje w systemie repozytorium pakietów Microsoftu (`/etc/yum.repos.d/`) — to trwała zmiana źródeł, nie tylko jednorazowa instalacja. Flaga `ACCEPT_EULA=Y` oznacza automatyczną, bezobsługową akceptację licencji `msodbcsql18` — warto to zawczasu ustalić z organizacją, zanim komenda pójdzie na serwer produkcyjny. Sterownik w wersji 18 domyślnie **wymusza szyfrowane połączenie** (inaczej niż wersja 17) — bez `TrustServerCertificate=yes` lub ważnego certyfikatu połączenie do serwera z certyfikatem self-signed zakończy się błędem.

Przebieg

1. Sprawdzenie wersji i pobranie repozytorium Microsoftu

```
if ! [[ "7 8 9" == *"$(grep VERSION_ID /etc/os-release | cut -d '"' -f 2 | cut -d '.' -f 1)"* ]]; then
    echo "RHEL $(grep VERSION_ID /etc/os-release | cut -d '"' -f 2 | cut -d '.' -f 1) is not currently supported."
    exit 1
fi
curl -sSL -O https://packages.microsoft.com/config/rhel/$(grep VERSION_ID /etc/os-release | cut -d '"' -f 2 | cut -d '.' -f 1)/packages-microsoft-prod.rpm
sudo dnf install -y ./packages-microsoft-prod.rpm
rm -f packages-microsoft-prod.rpm
```

Przykład:

```
Installed:
  packages-microsoft-prod-1.1-...noarch
Complete!
```

Blok sprawdza numer głównej wersji RHEL/Rocky w `/etc/os-release`, przerywa jeśli nie jest wspierana (7/8/9), a następnie pobiera i instaluje pakiet, który dopisuje repozytorium Microsoftu do systemu. Plik `.rpm` po instalacji jest już zbędny, stąd `rm`.

2. Usunięcie pakietów konfliktujących

```
sudo dnf remove -y unixODBC-utf16 unixODBC-utf16-devel
```

Przykład:

```
No match for argument: unixODBC-utf16
No match for argument: unixODBC-utf16-devel
Nothing to do.
```

Wariant `unixODBC-utf16` nie współpracuje ze sterownikiem `msodbcsql18` — komenda usuwa go, jeśli był kiedyś zainstalowany. Komunikat „No match" na świeżym systemie jest normalny i nie oznacza błędu.

3. Instalacja sterownika ODBC

```
sudo ACCEPT_EULA=Y dnf install -y msodbcsql18
```

Przykład:

```
Installed:
  msodbcsql18-18.4.1.1-1.x86_64
Complete!
```

`ACCEPT_EULA=Y` jest wymagane przez pakiet — bez tej zmiennej instalacja przerywa się z prośbą o ręczną akceptację licencji. Pakiet instaluje bibliotekę sterownika oraz sam rejestruje ją w `/etc/odbcinst.ini`.

4. Instalacja narzędzi wiersza poleceń (opcjonalnie)

```
sudo ACCEPT_EULA=Y dnf install -y mssql-tools18 unixODBC-devel
echo 'export PATH="$PATH:/opt/mssql-tools18/bin"' >> ~/.bashrc
source ~/.bashrc
```

Przykład:

```
Installed:
  mssql-tools18-18.4.1.1-1.x86_64   unixODBC-devel-2.3.11-...
```

`mssql-tools18` dostarcza `sqlcmd` i `bcp`, przydatne do szybkich testów i importu/eksportu danych bez pisania kodu. `unixODBC-devel` daje nagłówki potrzebne przy kompilowaniu aplikacji korzystających z ODBC (np. rozszerzenia PHP czy Pythona). Wpis w `.bashrc` dotyczy tylko bieżącego użytkownika — dla wszystkich kont dodajemy plik w `/etc/profile.d/`.

5. Sprawdzenie rejestracji sterownika

```
odbcinst -j
odbcinst -q -d
```

Przykład:

```
unixODBC 2.3.11
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
USER DATA SOURCES..: /home/adm_user/.odbc.ini


[ODBC Driver 18 for SQL Server]
```

`odbcinst -j` pokazuje, z których plików `unixODBC` w ogóle czyta konfigurację — przydatne, gdy w systemie jest więcej niż jedna instalacja `unixODBC` i nie wiadomo, który plik jest brany pod uwagę. `odbcinst -q -d` wypisuje zainstalowane sterowniki; brak wpisu `ODBC Driver 18 for SQL Server` oznacza, że krok 3 się nie powiódł albo zapisał wpis w innym pliku `odbcinst.ini`.

6. Test połączenia przez sqlcmd

```
sqlcmd -S NazwaSerwera -U nazwa_usera -P 'HasloTestowe' -C -Q "SELECT @@VERSION"
```

> (gdzie `NazwaSerwera` to adres serwera MSSQL, a `nazwa_usera` to login SQL uprawniony do bazy)

Przykład:

```
------------------------------------------------------------------------
Microsoft SQL Server 2022 (RTM) - 16.0.1000.6 (X64)
□Oct 8 2022 05:58:25
□Copyright (C) 2022 Microsoft Corporation
□Standard Edition (64-bit) on Linux (Ubuntu 20.04.5 LTS)


(1 rows affected)
```

Flaga `-C` mówi klientowi, żeby zaufał certyfikatowi serwera bez jego weryfikacji — odpowiednik `TrustServerCertificate=yes` w pliku DSN. Bez niej, przy serwerze z certyfikatem self-signed, `sqlcmd` w wersji 18 kończy się błędem `SSL Provider: certificate verify failed`, bo — inaczej niż w wersji 17 — szyfrowanie i weryfikacja certyfikatu są domyślnie włączone.

To jest najczęstsza pułapka migracji z wersji 17 na 18: skrypty i pliki DSN pisane pod starszy sterownik, bez świadomie ustawionej flagi szyfrowania, łączyły się bez przeszkód, bo `Encrypt` domyślnie było wyłączone. Wersja 18 odwraca to domyślne zachowanie — `Encrypt=yes` jest teraz założeniem, a `TrustServerCertificate=yes` (albo instalacja właściwego certyfikatu CA po stronie serwera) to jedyne dwa sposoby, żeby połączenie z self-signed certyfikatem w ogóle przeszło.

7. Konfiguracja pliku źródeł danych

```
sudo tee -a /etc/odbc.ini > /dev/null <<'EOF'


# DSN dla Microsoft SQL Server (MSSQL) z ODBC Driver 18 i self-signed cert
[DSN_MSSQL]
Driver = ODBC Driver 18 for SQL Server
Server = sql01.ad.zielona.example
Port = 1433
Database = your_database
TrustServerCertificate = Yes
# Trusted_Connection=Yes  # NIE aktywuj, jeśli nie masz Kerberosa
EOF
```

Sekcja `[DSN_MSSQL]` staje się nazwą, pod którą aplikacje (np. `isql`, sterowniki PHP/Python, Zabbix) odwołują się do tego połączenia, bez konieczności powtarzania serwera i portu w każdej konfiguracji. `TrustServerCertificate = Yes` to trwały odpowiednik flagi `-C` z poprzedniego kroku — potrzebny właśnie z powodu wymuszonego szyfrowania w wersji 18 sterownika. Login i hasło celowo nie są tu zapisane — podaje je aplikacja łącząca się przez DSN (np. przez makra Zabbixa) albo użytkownik przy każdym połączeniu.

Weryfikacja

```
isql -v DSN_MSSQL nazwa_usera 'HasloTestowe'
```

Przykład:

```
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

`isql` (z pakietu `unixODBC`, nie mylić z narzędziem `isql` firmy Sybase) łączy się przez zdefiniowany DSN, a nie bezpośrednio przez parametry sieciowe — to potwierdza, że cały łańcuch sterownik → `odbcinst.ini` → `odbc.ini` działa razem, a nie tylko sam sterownik. `quit` kończy sesję.



Aplikacja odwołuje się do DSN w odbc.ini, ten wskazuje sterownik zarejestrowany w odbcinst.ini, a dopiero biblioteka sterownika nawiązuje właściwe połączenie z serwerem MSSQL (kliknięcie otwiera pełny rozmiar)

Wycofanie zmian

```
sudo dnf remove -y msodbcsql18 mssql-tools18 unixODBC-devel
sudo sed -i '/mssql-tools18/d' ~/.bashrc
sudo rm -f /etc/yum.repos.d/mssql-release.repo /etc/yum.repos.d/packages-microsoft-com-prod.repo
```

Usunięcie pakietów samo czyści wpisy dodane do `/etc/odbcinst.ini` (menedżer pakietów pilnuje tego pliku wspólnie z `unixODBC`). Wpis DSN dodany ręcznie w kroku 7 do `/etc/odbc.ini` **nie** znika automatycznie — sekcję `[DSN_MSSQL]` usuwamy ręcznie, jeśli nie ma być używana. Plik repozytorium Microsoftu w `/etc/yum.repos.d/` też trzeba skasować osobno, jeśli system nie ma go już nigdy używać.

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
`SSL Provider: certificate verify failed` przy `sqlcmd` lub `isql`	Serwer MSSQL ma certyfikat self-signed, a sterownik 18 domyślnie go weryfikuje	Dodajemy `-C` do `sqlcmd` albo `TrustServerCertificate = Yes` w `odbc.ini`; docelowo lepiej zainstalować ważny certyfikat po stronie serwera
`RHEL X is not currently supported` w kroku 1	Wersja systemu spoza listy 7/8/9 w skrypcie repozytorium	Sprawdzamy na stronie Microsoft Learn, czy dana wersja RHEL/Rocky jest już wspierana, albo dodajemy ją ręcznie do warunku
`sqlcmd: command not found` mimo instalacji `mssql-tools18`	`PATH` nie zawiera `/opt/mssql-tools18/bin` w bieżącej sesji	`source ~/.bashrc` albo wylogowanie i ponowne zalogowanie; sprawdzamy `echo $PATH`
`odbcinst -q -d` nie pokazuje `ODBC Driver 18 for SQL Server`	Instalacja `msodbcsql18` nie powiodła się albo wpis trafił do innego pliku `odbcinst.ini` niż ten czytany przez `unixODBC`	Sprawdzamy `odbcinst -j`, która ścieżka jest aktywna; przeinstalowujemy pakiet (`dnf reinstall msodbcsql18`)
`dnf install` kończy się konfliktem z `unixODBC-utf16`	Wcześniej zainstalowany wariant biblioteki UTF-16 blokuje instalację `msodbcsql18`, który wymaga standardowego `unixODBC`	Wykonujemy krok 2 (`dnf remove unixODBC-utf16 unixODBC-utf16-devel`) przed ponowną próbą instalacji sterownika
Błąd `Login failed for user` mimo poprawnego hasła	Konto SQL nie ma uprawnień na docelowej bazie albo login jest w trybie tylko-Windows (bez uwierzytelniania SQL)	Sprawdzamy po stronie serwera, czy tryb uwierzytelniania to *Mixed Mode* i czy login ma przypisaną rolę na bazie z `Database` w DSN

Powiązane: Przykładowe konfiguracje ODBC dla różnych baz danych

Źródła: Microsoft Learn — *Install the Microsoft ODBC Driver for SQL Server on Linux* · `man odbcinst` · `man isql`

Przykładowe konfiguracje ODBC dla różnych baz danych

□□ Typ: referencja

□□ Dotyczy: RHEL/Rocky 9–10 (unixODBC, dowolna dystrybucja z tym samym pakietem)

Gotowe bloki `/etc/odbc.ini` dla najczęściej spotykanych baz danych. Kopiujemy sekcję, podmieniamy adres serwera, nazwę bazy i — jeśli trzeba — nazwę sterownika na tę zarejestrowaną w naszym `/etc/odbcinst.ini`.

□□ Skróty użyte na tej stronie (8)

Skrót	Co znaczy
ASE	Adaptive Server Enterprise: nazwa silnika Sybase od 1996 roku, dziś rozwijanego przez SAP jako SAP ASE
DSN	Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC
MSSQL	Microsoft SQL Server: silnik bazy danych Microsoftu, na nim w Zielonej stoi księgowość Optima
ODBC	Open Database Connectivity: uniwersalny sposób łączenia się z różnymi bazami danych tym samym interfejsem
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SAP	Systeme, Anwendungen und Produkte in der Datenverarbeitung: niemiecki producent oprogramowania klasy ERP, właściciel silnika HANA i przejętego wcześniej Sybase ASE
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL
TNS	Transparent Network Substrate: warstwa sieciowa Oracle, przez którą klient znajduje i łączy się z listenerem bazy

Fiszka

□□ TYLKO ODCZYT

Sprawdzenie zainstalowanych sterowników i lokalizacji plików: odbcinst -j

```
odbcinst -j
```

Przykład:

```
unixODBC 2.3.11
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
USER DATA SOURCES..: /home/adm_user/.odbc.ini
```

□□ TYLKO ODCZYT

Wylistowanie zarejestrowanych sterowników: odbcinst -q -d

```
odbcinst -q -d
```

Przykład:

```
[ODBC Driver 18 for SQL Server]
[PostgreSQL Unicode]
[MariaDB ODBC 3.1 Driver]
```

Nazwa w nawiasach kwadratowych to dokładna wartość, którą trzeba wpisać w pole `Driver` DSN — literówka albo inna wielkość liter i `unixODBC` zgłosi `Data source name not found and no default driver specified`, mimo że sterownik fizycznie jest zainstalowany. Na Debianie/Ubuntu i SLES ścieżki plików bywają inne (np. `/etc/unixODBC/odbcinst.ini`) — `odbcinst -j` zawsze pokazuje te faktycznie aktywne na danej maszynie, niezależnie od dystrybucji.

□□ **DSN** = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC · **SLES** = SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL

□□ PLIK KONFIGURACYJNY

DSN dla Microsoft SQL Server

```
# DSN dla Microsoft SQL Server (MSSQL) z ODBC Driver 18 i self-signed cert
[DSN_MSSQL]
Driver = ODBC Driver 18 for SQL Server
Server = sql01.ad.zielona.example
Port = 1433
Database = your_database
TrustServerCertificate = Yes
# Trusted_Connection=Yes  # NIE aktywuj, jeśli nie masz Kerberosa
```

`Driver` musi dosłownie odpowiadać nazwie sekcji w `odbcinst.ini` (sprawdzamy przez `odbcinst -q -d`). `TrustServerCertificate = Yes` jest konieczne od wersji 18 sterownika, bo domyślnie wymusza on szyfrowane i zweryfikowane połączenie — bez tego wpisu, przy certyfikacie self-signed, połączenie odrzuci błąd weryfikacji. `Trusted_Connection` (uwierzytelnianie zintegrowane przez Kerberos) ma sens tylko na maszynie dołączonej do domeny z ważnym biletem — w innym wypadku zostawiamy ją zakomentowaną i podajemy login/hasło przy połączeniu.

DSN = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC · **SQL** = Structured Query Language: język zapytań do relacyjnych baz danych · **ODBC** = Open Database Connectivity: uniwersalny sposób łączenia się z różnymi bazami danych tym samym interfejsem

PLIK KONFIGURACYJNY

DSN dla PostgreSQL

```
# DSN dla PostgreSQL
[DSN_POSTGRES]
Driver = PostgreSQL Unicode
Servername = pg01.ad.zielona.example
Port = 5432
Database = your_pgdb
```

Pakiet dostarczający ten sterownik to zwykle `postgresql-odbc` (RHEL/Rocky) albo `odbc-postgresql` (Debian/Ubuntu). Pole nazywa się `Servername`, nie `Server` jak w bloku MSSQL — niekonsekwencja pochodzi z tego, że każdy producent sterownika sam decyduje o nazwach pól czytanych ze swojej sekcji `odbc.ini`; `unixODBC` tylko przekazuje je dalej. Warianty sterownika bez końcówki `Unicode` (np. `PostgreSQL ANSI`) obsługują tylko kodowania jednobajtowe — dla nowych wdrożeń zawsze wybieramy wariant Unicode.

□□ **DSN** = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC · **RHEL** = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa · **MSSQL** = Microsoft SQL Server: silnik bazy danych Microsoftu, na nim w Zielonej stoi księgowość Optima · **ANSI** = American National Standards Institute: tu chodzi o kody sterujące kolorem i pozycją kursora w terminalu

□□ PLIK KONFIGURACYJNY

DSN dla MySQL / MariaDB

```
# DSN dla MariaDB/MySQL
[DSN_MARIA]
Driver = MariaDB ODBC 3.1 Driver
Server = my01.ad.zielona.example
Port = 3306
Database = your_mariadb
# UID i PWD ustawia aplikacja łącząca się przez makra/zmienne środowiskowe
```

Sterownik z pakietu `mariadb-connector-odbc` obsługuje zarówno MariaDB, jak i MySQL — nazwa `MariaDB ODBC 3.1 Driver` w `odbcinst.ini` zostaje taka sama nawet przy połączeniu do serwera MySQL. Numer `3.1` w nazwie to wersja samego sterownika, nie serwera bazy danych, i bywa inny w zależności od repozytorium, z którego pakiet pochodzi — zawsze sprawdzamy dokładną nazwę przez `odbcinst -q -d` zamiast kopiować ją z dokumentacji.

□□ **DSN** = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC · **ODBC** = Open Database Connectivity: uniwersalny sposób łączenia się z różnymi bazami danych tym samym interfejsem

□□ PLIK KONFIGURACYJNY

DSN dla Oracle

```
# DSN dla Oracle
[DSN_ORACLE]
Driver = Oracle in OraClient12Home1
Server = ora01.ad.zielona.example
Port = 1521
Database = ORCL
```

Sterownik Oracle Instant Client zwykle **nie** rejestruje się automatycznie w `odbcinst.ini` — sekcję dopisujemy ręcznie po instalacji klienta, podając ścieżkę do `libsqora.so.12.1` (albo

odpowiedniej wersji). Nazwa sterownika w polu `Driver` (tu `Oracle in OraClient12Home1`) to dowolna etykieta wybrana przy rejestracji, nie stała nazwa narzucona przez Oracle — musi tylko zgadzać się z nazwą sekcji w `odbcinst.ini`. `Database` w kontekście Oracle to zwykle identyfikator usługi albo nazwa TNS, nie nazwa pojedynczej bazy jak w MSSQL czy PostgreSQL.

DSN = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC · **TNS** = Transparent Network Substrate: warstwa sieciowa Oracle, przez którą klient znajduje i łączy się z listenerem bazy · **MSSQL** = Microsoft SQL Server: silnik bazy danych Microsoftu, na nim w Zielonej stoi księgowość Optima

PLIK KONFIGURACYJNY

DSN dla Sybase

```
# DSN dla Sybase
[DSN_SYBASE]
Driver = Adaptive Server Enterprise
Server = syb01.ad.zielona.example
Port = 5000
Database = your_sybase_db
```

„Adaptive Server Enterprise" to nazwa handlowa silnika Sybase (dziś SAP ASE) i jednocześnie typowa nazwa sekcji sterownika w `odbcinst.ini` po instalacji pakietu klienckiego SAP. Port `5000` jest tu przykładowy — Sybase/ASE nie ma jednego domyślnego portu narzuconego odgórnie tak jak MSSQL (1433) czy PostgreSQL (5432); rzeczywisty numer ustala administrator bazy przy instalacji serwera i trzeba go potwierdzić, zanim ten blok zadziała.

DSN = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC · **SAP** = Systeme, Anwendungen und Produkte in der Datenverarbeitung: niemiecki producent oprogramowania klasy ERP, właściciel silnika HANA i przejętego wcześniej Sybase ASE · **ASE** = Adaptive Server Enterprise: nazwa silnika Sybase od 1996 roku, dziś rozwijanego przez SAP jako SAP ASE · **MSSQL** = Microsoft SQL Server: silnik bazy danych Microsoftu, na nim w Zielonej stoi księgowość Optima

ZMIENIA SYSTEM

Włączenie śledzenia zapytań ODBC: export ODBCSYSINI + odbcinst -q -d

```
export ODBCSYSINI=/etc
odbcinst -q -d
```

Gdy DSN „powinien działać", ale połączenie i tak się nie udaje, `unixODBC` potrafi zapisywać każde wywołanie warstwy ODBC do pliku śledzenia — włączamy to sekcją `[ODBC]` z kluczem `Trace = Yes` i `TraceFile = /tmp/odbc.trace` w `/etc/odbcinst.ini` albo `~/.odbc.ini`. Plik śledzenia pokazuje dokładnie, na którym wywołaniu (`SQLConnect`, `SQLDriverConnect`...) połączenie się załamuje — przydatne, gdy komunikat błędu z samej aplikacji jest zbyt ogólny, żeby cokolwiek z niego wywnioskować.

ODBC = Open Database Connectivity: uniwersalny sposób łączenia się z różnymi bazami danych tym samym interfejsem · **DSN** = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC · **export** = ustaw zmienną środowiskową widoczną dla uruchamianych programów

TYLKO ODCZYT

Szybki test dowolnego DSN: isql

```
isql -v NazwaDSN nazwa_usera haslo
```

Przykład:

```
+---------------------------------------+
| Connected!                            |
+---------------------------------------+
SQL>
```

`isql` z pakietu `unixODBC` działa identycznie dla każdego z powyższych DSN — nie trzeba osobnego narzędzia na bazę. `Connected!` potwierdza, że sterownik, wpis w `odbcinst.ini` i wpis w `odbc.ini` są ze sobą spójne; błąd na tym etapie prawie zawsze wskazuje literówkę w nazwie sterownika (patrz wyżej) albo brak sieci do serwera bazy, nie problem z samą konfiguracją ODBC.

DSN = Data Source Name: nazwa gotowego połączenia do bazy danych zapisanego w konfiguracji ODBC · **ODBC** = Open Database Connectivity: uniwersalny sposób łączenia się z różnymi bazami danych tym samym interfejsem · **isql** = interactive SQL: klient Sybase

Z życia wzięte

Z ŻYCIA WZIĘTE

> O 11:00 Ola prosi o szybkie sprawdzenie połączenia do trzech różnych baz naraz przed testami nowego raportu ETL. Tomek kopiuje gotowe bloki DSN z tej strony, podmienia tylko adresy serwerów na pg01, my01 i ora01, i łączy się do każdej przez `isql` zamiast pisać osobny skrypt testowy dla każdego silnika. Wszystkie trzy `Connected!` pojawiają się w niecałe pięć minut.

Wymagane pakiety

```
unixODBC
unixODBC-devel
```

Sterownik konkretnej bazy (np. `msodbcsql18`, `postgresql-odbc`, `mariadb-connector-odbc`) instalujemy osobno — pakiety `unixODBC`/`unixODBC-devel` dostarczają tylko warstwę pośredniczącą (menedżer sterowników), nie same sterowniki.

`odbcinst.ini` kontra `odbc.ini`

Ta strona nie tłumaczy krok po kroku, jak zainstalować sterownik — do tego służy osobna procedura instalacji ODBC do MSSQL, linkowana na końcu. Tu zebrane są gotowe bloki konfiguracji do skopiowania, gdy sterownik jest już zainstalowany, a brakuje tylko wpisu w `odbc.ini`.

Plik `/etc/odbcinst.ini` to rejestr **sterowników** — wpisuje się do niego menedżer pakietów przy instalacji każdego sterownika, mapując jego nazwę na ścieżkę pliku `.so`. `/etc/odbc.ini` to rejestr **źródeł danych (DSN)** — konkretnych połączeń: adres serwera, port, baza, wybrany sterownik z `odbcinst.ini` po nazwie. Aplikacja łącząca się przez ODBC podaje nazwę DSN (np. `NazwaDSN`), a warstwa `unixODBC` sama odnajduje w `odbcinst.ini`, jakiego sterownika użyć.

DSN **systemowy**, zdefiniowany w `/etc/odbc.ini`, widzi każdy użytkownik systemu i każda usługa działająca pod dowolnym kontem — to właściwe miejsce dla połączeń używanych przez demony (Zabbix, aplikacje serwerowe). DSN **użytkownika**, w pliku `~/.odbc.ini` konkretnego konta, widzi tylko ten użytkownik — przydatny do testów albo połączeń, które nie mają trafić do innych kont na tej samej maszynie. Przy tej samej nazwie DSN w obu plikach wygrywa wpis użytkownika.

Wspólne pola w sekcji DSN

Pole	Znaczenie

`Driver`	nazwa sterownika dokładnie taka, jak w nawiasach `odbcinst -q -d`
`Server` / `Servername`	adres serwera bazy — nazwa pola zależy od konkretnego sterownika
`Port`	port TCP serwera bazy
`Database`	nazwa bazy, schematu domyślnego albo identyfikatora usługi (zależnie od silnika)
`UID` / `PWD`	login i hasło — zwykle **pomijane** w pliku i podawane przez aplikację łączącą się, żeby nie trzymać hasła jawnym tekstem w pliku czytanym przez wszystkich
`TrustServerCertificate`	tylko sterowniki MSSQL — akceptacja certyfikatu self-signed

Pakiety sterowników w zależności od dystrybucji

Nazwa pakietu dostarczającego dany sterownik różni się między rodzinami dystrybucji — sama treść DSN powyżej zostaje bez zmian, zmienia się tylko to, skąd bierze się plik `.so` i wpis w `odbcinst.ini`.

Baza	RHEL/Rocky	Debian/Ubuntu	SLES
MSSQL	`msodbcsql18` (repo Microsoftu)	`msodbcsql18` (repo Microsoftu)	`msodbcsql18` (repo Microsoftu)
PostgreSQL	`postgresql-odbc`	`odbc-postgresql`	`psqlodbc`
MySQL/MariaDB	`mariadb-connector-odbc`	`mariadb-connector-odbc`	`mariadb-connector-odbc`
Oracle	pakiet klienta Instant Client + ręczna rejestracja w `odbcinst.ini`	jak w RHEL	jak w RHEL
Sybase/ASE	pakiet klienta SAP ASE + ręczna rejestracja	jak w RHEL	jak w RHEL

Sterowniki Oracle i Sybase/ASE nie mają gotowego pakietu w standardowych repozytoriach dystrybucji — trafiają na maszynę razem z klientem instalowanym ręcznie od producenta, a wpis w `odbcinst.ini` (ścieżka do biblioteki `.so`, nazwa sterownika) dopisujemy samodzielnie, zwykle instrukcją z dokumentacji instalacyjnej tego klienta.

Powiązane: Instalacja sterownika ODBC do MSSQL na RHEL/Rocky

Źródła: `man odbcinst` · `man odbc.ini` · dokumentacja unixODBC (`/usr/share/doc/unixODBC*`)

Instalacja Rocky Linux z własnym nazewnictwem VG i LV

Typ: procedura **Dotyczy: Rocky Linux 9 / RHEL 9, instalator Anaconda**

Czas: ~20 min

Efekt: system zainstalowany z własnym nazewnictwem grup wolumenów i wolumenów logicznych — osobna grupa na każde przeznaczenie, zamiast domyślnej `rl-root` / `rl-swap` generowanej przez instalator.

Skróty użyte na tej stronie (6)

Skrót	Co znaczy
GRUB	Grand Unified Bootloader: program, który jako pierwszy startuje po włączeniu komputera i podaje jądru, co ma uruchomić
GUI	Graphical User Interface: okienka i myszka zamiast komend
LV	Logical Volume: wolumen logiczny w LVM, na nim leży system plików
LVM	Logical Volume Manager: warstwa w Linuksie składająca dyski w elastyczne wolumeny
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
VG	Volume Group: grupa wolumenów w LVM, pula złożona z dysków

Zanim zaczniemy

Z ŻYCIA WZIĘTE

O 8:30 Tomek instaluje nowy serwer bazy danych, który ma zastąpić starzejący się pg01. Zamiast klikać przez GUI instalatora, wchodzi w terminal Anacondy i ręcznie zakłada osobne grupy wolumenów dla systemu, `/var` i danych PostgreSQL, żeby uniknąć domyślnego `rl-root`, które trzy miesiące później i tak trzeba by przemianować. Dzięki temu powiększenie

> wolumenu z danymi pół roku później to jedna komenda `lvextend`, a nie ryzykowna operacja na wolumenie systemowym.

Dyski muszą być podpięte i widoczne przed uruchomieniem instalatora. Partycjonowanie wykonujemy z sesji terminalowej instalatora, a nie z GUI — dzięki temu mamy pełną kontrolę nad nazwami.

Wymaganie	Uwaga
Nośnik instalacyjny Rocky Linux 9	również przez iLO / iDRAC / konsolę wirtualną
Wszystkie dyski podpięte	dołożenie dysku po instalacji wymaga ponownego partycjonowania
Rozplanowane przeznaczenia	jedna grupa wolumenów na jedno przeznaczenie

Konwencja nazewnicza: `vg<Przeznaczenie>` oraz `lv<Przeznaczenie>`. Nazwa wolumenu logicznego powtarza przeznaczenie grupy, dzięki czemu pełna ścieżka czyta się sama: `/dev/mapper/vgPostgres-lvPostgres`.

Przeznaczenie	Grupa wolumenów	Wolumen logiczny
system operacyjny	`vgOS`	`lvOS`
`/var`	`vgVAR`	`lvVAR`
baza PostgreSQL	`vgPostgres`	`lvPostgres`
baza Oracle	`vgDB`	`lvOracle`

Uważaj: układ zapisany na dysk w trakcie instalacji jest praktycznie nieodwracalny bez reinstalacji. Zmiana nazw po instalacji jest możliwa, ale wymaga przebudowy initramfs i poprawek w GRUB — pomyłka kończy się systemem, który nie startuje.

Przebieg

1. Uruchomienie instalatora i przejście do terminala

Po uruchomieniu instalatora wybieramy odpowiedni język i wciskamy kombinację klawiszy:

```
CTRL + ALT + F3
```

Przeniesie nas to do sesji terminalowej, w której będziemy mogli dowolnie zarządzać dyskami.

2. Weryfikacja wykrytych dysków

Sprawdzamy, jakie dyski widzi system:

```
lsblk
```

Przykład:

```
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   100G  0 disk
sdb      8:16   0   200G  0 disk
sdc      8:32   0   500G  0 disk
sr0     11:0    1   9,5G  0 rom
```

3. Utworzenie wolumenów fizycznych

Na początku musimy utworzyć physical volume poprzez komendę `pvcreate`:

```
pvcreate /dev/sdX
```

> (gdzie `/dev/sdX` to nazwa urządzenia blokowego z wyniku `lsblk`)

Tworzymy resztę wymaganych urządzeń fizycznych:

```
pvcreate /dev/sda
pvcreate /dev/sdb
pvcreate /dev/sdc
```

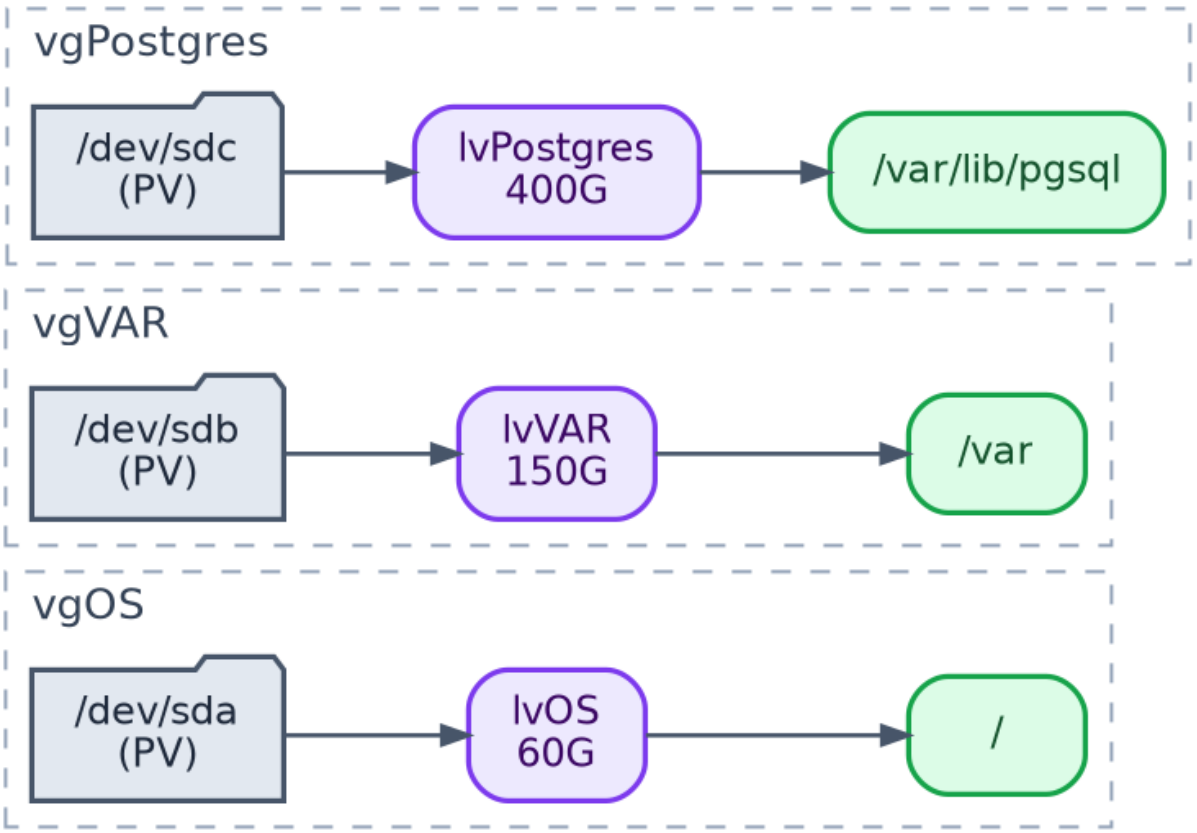
Weryfikujemy:

```
pvs
```

Przykład:

```
  PV         VG Fmt  Attr PSize    PFree
  /dev/sda      lvm2 ---  <100.00g <100.00g
  /dev/sdb      lvm2 ---  <200.00g <200.00g
  /dev/sdc      lvm2 ---  <500.00g <500.00g
```

4. Utworzenie grup wolumenów



Każdy dysk fizyczny staje się osobnym wolumenem fizycznym, trafia do własnej grupy wolumenów, a z niej powstaje jeden wolumen logiczny montowany pod właściwą ścieżką (kliknięcie otwiera pełny rozmiar)

Tworzymy osobną grupę na każde przeznaczenie:

```
vgcreate vgOS /dev/sda
```

```
vgcreate vgVAR /dev/sdb
```

```
vgcreate vgPostgres /dev/sdc
```

Weryfikujemy:

```
vgs
```

Przykład:

```
  VG         #PV #LV #SN Attr   VSize    VFree
  vgOS         1   0   0 wz--n- <100.00g <100.00g
  vgPostgres   1   0   0 wz--n- <500.00g <500.00g
  vgVAR        1   0   0 wz--n- <200.00g <200.00g
```

5. Utworzenie wolumenów logicznych

Wolumen systemowy — zostawiamy zapas w grupie na późniejsze powiększenie:

```
lvcreate -L 60G -n lvOS vgOS
```

Wolumen na `/var`:

```
lvcreate -L 150G -n lvVAR vgVAR
```

Wolumen na dane bazy:

```
lvcreate -L 400G -n lvPostgres vgPostgres
```

> Nie wykorzystujemy `100%FREE` przy zakładaniu. Wolne miejsce w grupie jest tym, co pozwala później powiększyć wolumen jedną komendą, bez dokładania dysku.

Weryfikujemy:

```
lvs
```

Przykład:

```
  LV          VG         Attr       LSize   Pool Origin Data%  Meta%
  lvOS        vgOS       -wi-a-----  60.00g
  lvPostgres  vgPostgres -wi-a----- 400.00g
  lvVAR       vgVAR      -wi-a----- 150.00g
```

6. Powrót do instalatora i przypisanie punktów montowania

Wracamy do instalatora kombinacją:

```
CTRL + ALT + F6
```

Wchodzimy w *Installation Destination* → *Custom* → *Done*. Utworzone wolumeny są już widoczne na liście po lewej stronie. Każdemu przypisujemy punkt montowania:

Wolumen	Punkt montowania	System plików
`vgOS-lvOS`	`/`	xfs
`vgVAR-lvVAR`	`/var`	xfs
`vgPostgres-lvPostgres`	`/var/lib/pgsql`	xfs

Partycję `/boot` i `/boot/efi` zostawiamy poza LVM — instalator utworzy je na dysku systemowym.

7. Instalacja

Zatwierdzamy układ i uruchamiamy instalację. Nazwy grup i wolumenów zostają takie, jakie utworzyliśmy.

Weryfikacja

Po pierwszym uruchomieniu systemu sprawdzamy, czy nazewnictwo się utrzymało:

```
lsblk
```

Przykład:

```
NAME                    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda                       8:0    0   100G  0 disk
├─sda1                    8:1    0     1G  0 part /boot
└─sda2                    8:2    0    99G  0 part
  └─vgOS-lvOS           253:0    0    60G  0 lvm  /
sdb                       8:16   0   200G  0 disk
└─vgVAR-lvVAR           253:1    0   150G  0 lvm  /var
sdc                       8:32   0   500G  0 disk
└─vgPostgres-lvPostgres 253:2    0   400G  0 lvm  /var/lib/pgsql
```

Sprawdzamy wolne miejsce w grupach — to jest zapas na powiększanie:

```
vgs
```

Przykład:

```
  VG         #PV #LV #SN Attr   VSize    VFree
  vgOS         1   1   0 wz--n-  <99.00g  39.00g
  vgPostgres   1   1   0 wz--n- <500.00g 100.00g
  vgVAR        1   1   0 wz--n- <200.00g  50.00g
```

Sprawdzamy montowania:

```
df -h
```

Wycofanie zmian

Jeśli system jest już zainstalowany z domyślnym nazewnictwem, nazwy da się zmienić na żywo — ale procedura dla wolumenu z systemem jest ryzykowna i opisana osobno na stronie o zmianie nazw VG i LV.

Zmiana nazwy grupy, która **nie** zawiera roota ani swapa, jest bezpieczna:

```
vgrename StaraNazwa vgPostgres
```

```
lvrename vgPostgres StaraNazwaLV lvPostgres
```

Poprawiamy wpis w `/etc/fstab` i montujemy:

```
mount -a
```

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
Instalator nie widzi utworzonych grup	partycjonowanie wykonane po wejściu w *Installation Destination*	wychodzimy z okna i wchodzimy ponownie — lista odświeża się przy wejściu
`Device /dev/sdX not found` przy `pvcreate`	dysk niepodpięty albo widoczny pod inną nazwą	`lsblk` i `lsscsi`, sprawdzamy rzeczywistą nazwę

Objaw	Przyczyna	Naprawa
`Device /dev/sdX excluded by a filter`	na dysku jest stara sygnatura LVM lub tablica partycji	`wipefs -a /dev/sdX` i ponownie `pvcreate`
Po instalacji grupa nazywa się `rl`	układ został nadpisany przez automatyczne partycjonowanie	reinstalacja z *Custom*, albo zmiana nazw po instalacji
System nie startuje po zmianie nazw	initramfs i GRUB wskazują na stare nazwy	tryb ratunkowy, `dracut -f` i `grub2-mkconfig`

Powiązane: LVM: dysk fizyczny, grupa wolumenów, wolumen logiczny · Powiększanie przestrzeni dyskowej bez restartu · Zmiana nazwy grupy wolumenów i wolumenu logicznego · Montowanie: mount, /etc/fstab, zasoby sieciowe

Źródła: `man 8 pvcreate` · `man 8 vgcreate` · `man 8 lvcreate` · dokumentacja instalatora Anaconda

Standardowa aktualizacja systemu produkcyjnego

Typ: procedura

Dotyczy: RHEL/Rocky 9–10 (przykłady dla Debian/Ubuntu i SLES w uwagach)

Czas: ~45 min na serwer, plus czas trwania aktualizacji

Efekt: serwer ma zainstalowane bieżące aktualizacje pakietów, jest po restarcie, wszystkie usługi i kontenery wróciły do stanu sprzed prac, a ewidencja serwerów odzwierciedla nowy stan patchowania.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
GRUB	Grand Unified Bootloader: program, który jako pierwszy startuje po włączeniu komputera i podaje jądru, co ma uruchomić
GUI	Graphical User Interface: okienka i myszka zamiast komend
RAM	Random Access Memory: pamięć operacyjna
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera

Zanim zaczniemy

Z ŻYCIA WZIĘTE

Wtorek wieczorem, zaplanowane okno serwisowe na app01. Tomek przechodzi całą procedurę aktualizacji krok po kroku: okno w monitoringu, snapshot maszyny, aktualizacja, restart i weryfikacja usług, zamiast improwizować kolejność, którą łatwo pomylić o dziesiątej wieczorem.

Wymaganie	Jak sprawdzić
Okno serwisowe zatwierdzone i zgłoszone w systemie monitoringu	potwierdzenie zmiany / wpis w kalendarzu zmian
Zgoda właściciela usługi na restart serwera w tym terminie	e-mail albo wpis w zgłoszeniu zmiany
Snapshot maszyny wirtualnej wykonany bezpośrednio przed pracami	konsola wirtualizatora, zakładka snapshotów danej maszyny
Dostęp do konsoli poza SSH (konsola hypervisora / iLO / iDRAC)	zalogowanie się do konsoli i potwierdzenie widoku ekranu logowania

Uważaj: restart w kroku 11 jest punktem bez odwrotu w ramach samej procedury — jeśli aktualizacja jądra albo sterownika ma problem, dowiadujemy się o tym dopiero po restarcie, i wtedy jedyną szybką drogą powrotu jest przywrócenie snapshotu z kroku 6, nie odinstalowanie pakietów na żywym systemie. Snapshot bez zaznaczonej opcji „include virtual machine memory" jest szybszy do wykonania i do przywrócenia — zostawiamy tę opcję zaznaczoną tylko wtedy, gdy mamy konkretny powód, żeby po przywróceniu odtworzyć też stan pamięci RAM w locie.

Przebieg

1. Ustawienie okna serwisowego w systemie monitoringu

Zanim zatrzymamy cokolwiek na serwerze, system monitoringu (Zabbix, Nagios albo ich odpowiednik w naszym środowisku) musi wiedzieć, że nadchodzące alerty z tego hosta są planowane, a nie awarią. W większości wdrożeń to krok wykonywany w konsoli GUI systemu monitoringu (w Zabbiksie: *Data collection → Maintenance*; w Nagiosie: zaplanowane *downtime*), nie z linii poleceń serwera. Jeśli monitoring udostępnia API, krok da się zautomatyzować — przykład dla API zgodnego z Zabbiksem:

```
curl -sk -X POST https://NazwaSerwera/api_jsonrpc.php \
  -H 'Content-Type: application/json-rpc' \
  -d '{"jsonrpc":"2.0","method":"maintenance.create","params":{"name":"Patch serwera 2026-09-18","active_since":1758196800,"active_till":1758204000,"hosts":[{"hostid":"10105"}],"timeperiods":[{"timeperiod_type":0,"period":7200}]},"auth":"TOKEN","id":1}'
```

Przykład:

```
{"jsonrpc":"2.0","result":{"maintenanceids":["17"]},"id":1}
```

Serwer jest teraz oznaczony jako w oknie serwisowym — alerty z niego w najbliższych dwóch godzinach trafią do monitoringu, ale nie wygenerują powiadomień do zespołu dyżurnego.

2. Sprawdzenie, czy monitoring nie zgłasza już alertów

```
systemctl --failed
```

Przykład:

```
0 loaded units listed.
```

Zerowy wynik znaczy, że żadna jednostka systemd nie jest w stanie `failed` przed rozpoczęciem prac — dobry punkt odniesienia, do którego wracamy w kroku 12. Jeśli lista nie jest pusta, wyjaśniamy i naprawiamy te usterki *przed* aktualizacją, żeby po restarcie nie zgadywać, czy problem jest stary, czy nowy.

3. Zapisanie listy uruchomionych kontenerów

```
sudo docker ps
```

Przykład:

```
CONTAINER ID   IMAGE                                          COMMAND
CREATED       STATUS                 PORTS                NAMES
8c210b9a76a3   git.zielona.example/zielona/logstash:1.0.0     "/usr/local/bin/entry…"   3
weeks ago   Up 3 weeks (healthy)   5044/tcp, 9600/tcp   stack_logstash_1
b96065fb9645   git.zielona.example/zielona/kibana:1.0.0       "/bin/tini -- /bin/s…"    3
weeks ago   Up 3 weeks (healthy)   5601/tcp             stack_kibana_1
```

Jeśli maszyna hostuje kontenery, zapisujemy tę listę do pliku — to jest lista, którą po restarcie porównujemy, żeby się upewnić, że każdy kontener wrócił i w tej samej wersji. Kontenery uruchamiane ręcznie (bez `restart: always` w Compose i bez jednostki systemd) nie wstają same po restarcie hosta — o tym w sekcji *Gdy coś nie zadziała*.

```
sudo docker ps > /root/docker-przed-aktualizacja-$(date +%F).txt
```

4. Zatrzymanie agenta bezpieczeństwa i zakomentowanie wpisu w cronie

Agent bezpieczeństwa / antywirus zainstalowany na serwerze (typowo kilka usług naraz, plus wpis w cronie pilnujący, że agent żyje) potrafi blokować pliki w trakcie aktualizacji pakietów albo fałszywie zgłaszać incydent przy podmianie binarek systemowych. Sprawdzamy, czy jest zainstalowany, i jeśli tak — zatrzymujemy go na czas prac:

```
sudo systemctl stop agent-bezpieczenstwa-tapd.service
sudo systemctl stop agent-bezpieczenstwa-espd.service
sudo systemctl stop agent-bezpieczenstwa-ma.service
```

Przykład:

```
● agent-bezpieczenstwa-tapd.service - Threat Prevention Daemon
     Active: inactive (dead) since Thu 2026-09-18 09:02:11 UTC
```

Komentujemy też wpis crona, który regularnie sprawdza kondycję agenta i potrafi go automatycznie ponownie uruchomić — bez tego kroku agent wstanie sam w trakcie okna serwisowego:

```
sudo crontab -e
```

Przykład:

```
# przed:
0,30 * * * * /opt/agent-bezpieczenstwa/scripts/agent checkhealth >/dev/null 2>/dev/null
# po (zakomentowane):
#0,30 * * * * /opt/agent-bezpieczenstwa/scripts/agent checkhealth >/dev/null 2>/dev/null
```

Ten konkretny wpis dotyczy głównie starszych wersji systemu (starszych niż RHEL/Rocky 8) — na nowszych obrazach kontrola kondycji bywa zaszyta w jednostce systemd zamiast w cronie, więc ten krok może nie mieć zastosowania.

5. Sprawdzenie procesów zombie

```
ps -eo pid,ppid,stat,comm | awk '$3 ~ /^Z/'
```

Przykład:

```
    PID    PPID STAT COMMAND
```

Pusty wynik (sama nagłówkowa linia) jest stanem oczekiwanym. Proces zombie sam w sobie nie zablokuje aktualizacji, ale jego obecność często sygnalizuje aplikację, która nie sprząta po swoich procesach potomnych — wart odnotowania przed restartem, żeby po restarcie było wiadomo, czy problem zniknął sam, czy trzeba go zgłosić osobno.

6. Wykonanie snapshotu maszyny wirtualnej

W konsoli wirtualizatora (vCenter albo odpowiednik) tworzymy snapshot maszyny bezpośrednio przed przystąpieniem do aktualizacji pakietów. Zostawiamy odznaczoną opcję **„include virtual machine memory"**, chyba że mamy konkretny powód, żeby ją zaznaczyć — z pamięcią RAM snapshot trwa znacznie dłużej do wykonania i do ewentualnego przywrócenia, a w praktyce przy aktualizacji pakietów systemowych stan pamięci procesów nie jest tym, co chcemy odtwarzać.

Przykład:

```
Nazwa snapshotu: przed-patchowaniem-2026-09-18
Opis: przed standardową aktualizacją pakietów
Snapshot the VM's memory: [ ] odznaczone
Quiesce guest file system: [x] zaznaczone (jeśli VMware Tools/open-vm-tools działa)
```

To jest ostatni moment, w którym wycofanie zmian jest szybkie i pewne — po tym kroku każda kolejna czynność zakłada, że w razie problemu wracamy właśnie do tego punktu.

7. Wyczyszczenie cache menedżera pakietów

```
sudo dnf clean all
```

Przykład:

```
40 files removed
```

Usuwa pobrane wcześniej metadane i pakiety z lokalnego cache — gwarantuje, że kolejny krok pobierze świeże informacje o repozytoriach, zamiast polegać na czymś, co mogło się zdezaktualizować od ostatniego uruchomienia. Na Debianie/Ubuntu odpowiednikiem jest `sudo apt clean`, na SLES-ie `sudo zypper clean --all`.

8. Sprawdzenie dostępnych aktualizacji bez instalowania

```
sudo dnf check-update
```

Przykład:

```
kernel.x86_64                 5.14.0-427.31.1.el9_4     baseos
openssl.x86_64                1:3.2.2-6.el9_4           baseos
openssl-libs.x86_64           1:3.2.2-6.el9_4           baseos
sudo.x86_64                   1.9.5p2-10.el9_4          baseos
```

Podgląd, co konkretnie zostanie zaktualizowane, zanim cokolwiek się zmieni. Zwracamy uwagę, czy na liście jest `kernel` — jeśli tak, po aktualizacji będzie potrzebny restart, a nie tylko restart pojedynczych usług. Na Debianie/Ubuntu odpowiada temu `sudo apt update && apt list --upgradable`, na SLES-ie `sudo zypper lp` (lista łatek) albo `sudo zypper list-updates`.

9. Sprawdzenie starych kerneli i ich usunięcie, gdy `/boot` jest ciasne

```
df -h /boot
```

Przykład:

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       1014M  920M   34M  91% /boot
```

Przy 91% zapełnienia kolejna instalacja jądra może się nie zmieścić i przerwać aktualizację w połowie z błędem braku miejsca. Pełny opis komend do bezpiecznego usuwania starych jąder (`dnf remove --oldinstallonly`, weryfikacja aktywnego jądra, aktualizacja wpisów GRUB) jest na osobnej stronie — patrz *Powiązane*. Szybki wariant, jeśli `/boot` jest rzeczywiście ciasne:

```
sudo dnf remove --oldinstallonly --setopt installonly_limit=2 kernel
```

Przykład:

```
Dependencies resolved.
Removing: kernel-5.14.0-362.el9.x86_64
```

```
Removed: 1 package
```

10. Wykonanie właściwej aktualizacji

```
sudo dnf upgrade -y
```

Przykład:

```
Upgrading:
 kernel                     5.14.0-427.31.1.el9_4    baseos    68 M
 openssl                    1:3.2.2-6.el9_4          baseos    1.2 M
 openssl-libs               1:3.2.2-6.el9_4          baseos    2.1 M
 sudo                       1.9.5p2-10.el9_4         baseos    1.1 M

Transaction Summary
================================================================================
Upgrade  47 Packages

Complete!
```

Na Debianie/Ubuntu odpowiada temu `sudo apt full-upgrade -y` (nie `apt upgrade`, jeśli aktualizacja obejmuje jądro — patrz strona o `apt`/`dpkg`), na SLES-ie `sudo zypper patch` albo `sudo zypper up`. Jeśli po drodze trzeba było podnieść wersję firmware maszyny wirtualnej (VMware Tools/`open-vm-tools`), robimy to teraz, przy okazji nadchodzącego restartu, żeby nie generować drugiego okna serwisowego tylko dla tej jednej zmiany.

11. Restart serwera

```
sudo reboot
```

Restart aktywuje nowe jądro (jeśli było w aktualizacji) i domyka wszystkie zmiany bibliotek systemowych, których nie da się bezpiecznie podmienić bez zatrzymania procesów, które je trzymają. Czekamy, aż host odpowie na ping i na próbę połączenia SSH, zanim przejdziemy dalej — jeśli host nie wraca w rozsądnym czasie, przechodzimy do konsoli poza SSH z tabeli *Zanim zaczniemy*.

12. Weryfikacja usług, kontenerów, montowań i zapory

```
systemctl --failed
sudo rpm -q kernel | sed "/$(uname -r)/ s/$/ ACTIVE/"
sudo docker ps
mount | grep -v ^tmpfs
sudo firewall-cmd --list-all
```

Przykład:

```
0 loaded units listed.
kernel-5.14.0-362.el9.x86_64
kernel-5.14.0-427.31.1.el9_4.x86_64 ACTIVE
public (active)
  target: default
  ports: 22/tcp 80/tcp 443/tcp
```

Pięć rzeczy naraz: brak nowych jednostek w stanie `failed` względem kroku 2, aktywne jądro to faktycznie to najnowsze z listy, kontenery z kroku 3 wróciły w tej samej liczbie i wersji, punkty montowania z `/etc/fstab` są na miejscu, a zapora wróciła z zapisaną konfiguracją zamiast domyślnej, pustej strefy. Pełny zestaw komend do samodzielnego, dokładniejszego przejścia po tym punkcie jest w sekcji *Weryfikacja* niżej.

13. Włączenie agenta bezpieczeństwa i odkomentowanie wpisu w cronie

```
sudo systemctl start agent-bezpieczenstwa-tapd.service
sudo systemctl start agent-bezpieczenstwa-espd.service
sudo systemctl start agent-bezpieczenstwa-ma.service
sudo crontab -e
```

Przykład:

```
# przywrócone:
0,30 * * * * /opt/agent-bezpieczenstwa/scripts/agent checkhealth >/dev/null 2>/dev/null
```

Serwer wraca do normalnej ochrony na bieżąco, dokładnie w stanie sprzed kroku 4.

14. Zdjęcie okna serwisowego w systemie monitoringu

Ten krok wykonujemy w tym samym miejscu co krok 1 — usuwamy albo kończymy wcześniej utworzone okno serwisowe, żeby monitoring wrócił do normalnego alarmowania na tym hoście.

```
curl -sk -X POST https://NazwaSerwera/api_jsonrpc.php \
  -H 'Content-Type: application/json-rpc' \
  -d '{"jsonrpc":"2.0","method":"maintenance.delete","params":["17"],"auth":"TOKEN","id":2}'
```

Przykład:

```
{"jsonrpc":"2.0","result":{"maintenanceids":["17"]},"id":2}
```

15. Aktualizacja ewidencji serwerów

Aktualizujemy datę i status w ewidencji serwerów/CMDB — bez tego kroku procedura jest technicznie skończona, ale organizacyjnie niewidoczna: następny administrator patrzący na CMDB nie będzie wiedział, że ten serwer już przeszedł to okno patchowania.

Przykład wpisu:

```
Serwer: srv01
Data aktualizacji: 2026-09-18
Status: zaktualizowany
Wersja jądra po aktualizacji: 5.14.0-427.31.1.el9_4
Wykonał: adm_user
```

Weryfikacja

Zestaw komend do odpalenia po zakończeniu wszystkich kroków, jako niezależne od nich potwierdzenie, że system jest w pełni sprawny — przydatny też, gdy wracamy do serwera po godzinie, żeby dopiero wtedy potwierdzić stabilność:

```
systemctl --failed
sudo rpm -q kernel | sed "/$(uname -r)/ s/$/ ACTIVE/"
uptime -s
```

```
sudo docker ps
mount | grep -v ^tmpfs
df -h
sudo firewall-cmd --state
```

Przykład:

```
0 loaded units listed.
kernel-5.14.0-427.31.1.el9_4.x86_64 ACTIVE
2026-09-18 09:18:42
CONTAINER ID   IMAGE                                          STATUS                  NAMES
8c210b9a76a3   git.zielona.example/zielona/logstash:1.0.0     Up 4 minutes (healthy)
stack_logstash_1
running
```

Wynik `uptime -s` musi pokazywać czas *po* wykonanym restarcie z kroku 11 — jeśli pokazuje sprzed prac, restart z jakiegoś powodu nie doszedł do skutku (np. host wznowił się z hibernacji zamiast pełnego rozruchu) i krok 11 powtarzamy.

Wycofanie zmian

Jeśli problem pojawił się przed krokiem 11 (restart), wycofanie jest proste: przerywamy procedurę, nie restartujemy serwera i albo pozostawiamy stary stan (jeśli aktualizacja się nie powiodła i nic nie zainstalowała), albo cofamy pojedynczą transakcję pakietów:

```
sudo dnf history list
sudo dnf history undo ID_TRANSAKCJI
```

Jeśli problem pojawił się **po** restarcie (usługa nie startuje, jądro się nie budzi, aplikacja rzuca błędami po aktualizacji biblioteki) — najszybsza i najpewniejsza droga to przywrócenie snapshotu z kroku 6 w konsoli wirtualizatora, nie ręczne odkręcanie zmian na żywym systemie. Po przywróceniu snapshotu serwer wraca do stanu sprzed jakiejkolwiek zmiany z tej procedury, łącznie z zatrzymanym agentem bezpieczeństwa i zahaszowanym wpisem w cronie — po przywróceniu odtwarzamy je do stanu produkcyjnego (krok 13) albo powtarzamy całą procedurę od nowa po ustaleniu przyczyny problemu.

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa

Serwer nie odpowiada na ping/SSH po restarcie	zbyt długi `fsck` na starcie albo zawieszenie na etapie GRUB/initramfs	Logujemy się przez konsolę poza SSH (tabela *Zanim zaczniemy*) i sprawdzamy ekran boot logu na żywo
`systemctl --failed` pokazuje jednostkę po restarcie, której nie było przed aktualizacją	zmiana ABI biblioteki albo niekompatybilna zmiana formatu pliku konfiguracyjnego w nowej wersji pakietu	`journalctl -u nazwa-usługi --since -10min`, w razie potrzeby `sudo dnf history undo` dla tego konkretnego pakietu
Kontener nie wstał po restarcie hosta	kontener uruchomiony ręcznie (`docker run`), bez `restart: always`/`unless-stopped` w Compose i bez jednostki systemd	`docker-compose up -d` w katalogu z definicją stosu, sprawdzenie logów kontenera
Aktualizacja przerwana błędem braku miejsca na `/boot`	zbyt wiele starych jąder zajmujących partycję rozruchową	Usuwamy stare jądra (krok 9 i osobna strona o czyszczeniu kerneli), powtarzamy `dnf upgrade`
Zapora po restarcie ma inną konfigurację niż przed aktualizacją	`firewalld` wrócił do strefy domyślnej zamiast zapisanej trwale konfiguracji	`firewall-cmd --list-all`, porównujemy z konfiguracją w `/etc/firewalld/`, przywracamy `--permanent` i `--reload`
Agent bezpieczeństwa nie wraca po `systemctl start` po zakończeniu prac	wpis w cronie wciąż zahaszowany, agent nie zdążył się samodzielnie zarejestrować w konsoli centralnej	Odkomentowujemy wpis crona z kroku 4/13, ręcznie uruchamiamy skrypt kontroli kondycji agenta

Powiązane: apt i dpkg: odpowiedniki na Debianie i Ubuntu · zypper: odpowiedniki na SLES i openSUSE · Kernele: lista, usuwanie starych, brak miejsca na /boot · systemd i systemctl — jednostki, zależności, stany · firewalld: strefy, porty, usługi, rich rules

Źródła: OneNote: Proces aktualizacji.txt („[LINUX] Standardowa aktualizacja") · `man 8 dnf` · `man 1 systemctl` · `man 1 firewall-cmd`