# 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`

---

Revision #15
Created 2026-09-18 06:57:31 UTC by Tomasz Konieczny
Updated 2026-10-03 19:10:12 UTC by Tomasz Konieczny