Monitoring zasilania i łagodne wyłączanie: UPS, NUT, Zabbix

Poziom: zaawansowany

Dotyczy: NUT 2.8, Rocky Linux 9, Zabbix 7.0 LTS, Windows Server 2022 z klastrem Hyper-V, karta sieciowa UPS z SNMP

Czas: ~120 min

Konfigurujemy na `bkp01` program NUT, który odczytuje stan UPS przez SNMP i po trzech minutach pracy na baterii wyłącza serwerownię w ustalonej kolejności, a w Zabbixie na `mon01` dodajemy alarmy o stanie zasilania. Wracamy tu przy nowym UPS, zmianie listy serwerów i po każdym teście wyłączania. Komendy wykonujemy na `bkp01` z uprawnieniami sudo, a polecenia PowerShell na `hv01` w oknie administratora.

Skróty użyte na tej stronie (18)

Skrót	Co znaczy
APC	Angled Physical Contact: polerowanie czoła złącza światłowodowego ściętego pod kątem 8 stopni, które odbija światło w płaszcz, oznaczane kolorem zielonym
EPEL	Extra Packages for Enterprise Linux: dodatkowe repozytorium pakietów dla RHEL i klonów
FSD	Forced ShutDown: stan wymuszonego wyłączenia w NUT, po którym serwer główny rozpoczyna wyłączanie
GSM	Global System for Mobile Communications: standard sieci komórkowej, z którego korzysta modem wysyłający SMS
HCI	Hyper-Converged Infrastructure: infrastruktura, w której serwery łączą obliczenia i dyski w jeden klaster
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
kW	Kilowat: tysiąc watów, miara mocy; prawie cała moc pobrana przez serwery zamienia się w ciepło
LTS	Long Term Support: wersja z długim wsparciem poprawkami

Skrót	Co znaczy
NUT	Network UPS Tools: zestaw programów na Linuksie, który odczytuje stan zasilacza UPS i wyłącza serwery, gdy baterie się kończą
OID	Object Identifier: numer, którym w SNMP wskazujemy jedną wartość urządzenia, na przykład stan baterii UPS
RFC	Request for Comments: oficjalny dokument opisujący standard internetowy, na przykład sposób działania danego protokołu; w ITIL także Request for Change, czyli wniosek o zmianę
S2D	Storage Spaces Direct: programowa macierz Windows Server z dysków wielu serwerów
SMS	Short Message Service: krótka wiadomość tekstowa na telefon, która dociera także bez internetu
SNMP	Simple Network Management Protocol: protokół odczytu stanu urządzeń sieciowych
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
UPS	Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci
UPS-MIB	Standardowa baza obiektów SNMP (RFC 1628) opisująca zasilacze awaryjne UPS, w tym temperaturę baterii
USB	Universal Serial Bus: standard portów i urządzeń podłączanych do komputera, tu pendrive'y i inne nośniki wymienne

Fiszka

□□ ZMIENIA SYSTEM

Dodanie repozytorium EPEL: sudo dnf install epel-release

```
sudo dnf install epel-release
```

Pakiety NUT dla Rocky Linux 9 pochodzą z repozytorium EPEL [1].

□□ **EPEL** = Extra Packages for Enterprise Linux: dodatkowe repozytorium pakietów dla RHEL i klonów · **NUT** = Network UPS Tools: zestaw programów na Linuksie, który odczytuje stan zasilacza UPS i wyłącza serwery, gdy baterie się kończą · **dnf** = Dandified YUM: menedżer pakietów RHEL/Rocky

□□ ZMIENIA SYSTEM

Instalacja NUT i narzędzi SNMP: sudo dnf install nut nut-client net-snmp-utils

```
sudo dnf install nut nut-client net-snmp-utils
```

Pakiet `nut` zawiera sterowniki i serwer `upsd`, `nut-client` program `upsmon` i narzędzie `upsc`, a `net-snmp-utils` polecenie `snmpget`.

NUT = Network UPS Tools: zestaw programów na Linuksie, który odczytuje stan zasilacza UPS i wyłącza serwery, gdy baterie się kończą · **SNMP** = Simple Network Management Protocol: protokół odczytu stanu urządzeń sieciowych · **dnf** = Dandified YUM: menedżer pakietów RHEL/Rocky

TYLKO ODCZYT

Sprawdzenie, czy sterownik SNMP jest zainstalowany: rpm -ql nut

```
rpm -ql nut | grep snmp-ups
```

Komenda wypisuje ścieżkę do sterownika `snmp-ups`. Jeśli nic nie wypisze, szukamy go w pozostałych pakietach NUT poleceniem `dnf search nut`.

SNMP = Simple Network Management Protocol: protokół odczytu stanu urządzeń sieciowych · **NUT** = Network UPS Tools: zestaw programów na Linuksie, który odczytuje stan zasilacza UPS i wyłącza serwery, gdy baterie się kończą · **rpm** = RPM Package Manager: format i narzędzie paczek RHEL

TYLKO ODCZYT

Odczyt stanu baterii z UPS przez SNMP: snmpget

```
snmpget -v2c -c NazwaWspolnoty AdresUPS 1.3.6.1.2.1.33.1.2.1.0
```

(gdzie `NazwaWspolnoty` to ciąg znaków SNMP v2c skonfigurowany w karcie sieciowej UPS, a `AdresUPS` to jej adres IP)

Podajemy OID z UPS-MIB (RFC 1628): `upsBatteryStatus` [2]. Wartość 2 oznacza baterię w normie, 3 niski stan, a 4 wyczerpanie.

Przykład:

```
UPS-MIB::upsBatteryStatus.0 = INTEGER: batteryNormal(2)
```

UPS = Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci · **SNMP** = Simple Network Management Protocol: protokół odczytu stanu urządzeń sieciowych · **IP** = Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection) · **OID** = Object Identifier: numer, którym w SNMP wskazujemy jedną

wartość urządzenia, na przykład stan baterii UPS · **UPS-MIB** = Standardowa baza obiektów SNMP (RFC 1628) opisująca zasilacze awaryjne UPS, w tym temperaturę baterii · **RFC** = Request for Comments: oficjalny dokument opisujący standard internetowy, na przykład sposób działania danego protokołu; w ITIL także Request for Change, czyli wniosek o zmianę · **snmpget** = SNMP get: odczytuje jedną wartość urządzenia przez SNMP po jej numerze OID

□□ TYLKO ODCZYT

Odczyt szacowanego czasu pracy na baterii: snmpget

```
snmpget -v2c -c NazwaWspolnoty AdresUPS 1.3.6.1.2.1.33.1.2.3.0
```

Wartość `upsEstimatedMinutesRemaining` to szacunek UPS w minutach przy bieżącym obciążeniu [2].

Przykład:

```
UPS-MIB::upsEstimatedMinutesRemaining.0 = INTEGER: 25
```

□□ **UPS** = Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci · **snmpget** = SNMP get: odczytuje jedną wartość urządzenia przez SNMP po jej numerze OID

□□ TYLKO ODCZYT

Odczyt źródła zasilania wyjścia: snmpget

```
snmpget -v2c -c NazwaWspolnoty AdresUPS 1.3.6.1.2.1.33.1.4.1.0
```

Wartość `upsOutputSource`: 3 oznacza pracę z sieci, 4 bypass, a 5 pracę z baterii [2].

□□ **snmpget** = SNMP get: odczytuje jedną wartość urządzenia przez SNMP po jej numerze OID

□□ TYLKO ODCZYT

Plik trybu pracy NUT: /etc/nut/nut.conf

```
MODE=netserver
```

Tryb `netserver` oznacza, że `bkp01` uruchamia serwer `upsd` i udostępnia stan UPS innym serwerom (jeśli dodamy klientów). Gdy klientów nie planujemy, wystarczy `standalone`.

□□ **NUT** = Network UPS Tools: zestaw programów na Linuksie, który odczytuje stan zasilacza UPS i wyłącza serwery, gdy baterie się kończą · **UPS** = Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci

□□ TYLKO ODCZYT

Plik definicji UPS: /etc/nut/ups.conf

```
[ups01]
    driver = snmp-ups
    port = AdresUPS
    desc = "UPS 20 kVA, serwerownia Pamiatkowo"
    snmp_version = v2c
    community = NazwaWspolnoty
    mibs = auto
    pollfreq = 10
```

Sterownik `snmp-ups` odpytuje UPS co `pollfreq` sekund. Domyślnie 30, a my skracamy do 10, żeby wcześniej zauważyć przejście na baterię [3].

□□ **UPS** = Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci

□□ TYLKO ODCZYT

Wariant SNMPv3 pliku ups.conf: /etc/nut/ups.conf

```
[ups01]
    driver = snmp-ups
    port = AdresUPS
    desc = "UPS 20 kVA, serwerownia Pamiatkowo"
    snmp_version = v3
    secLevel = authPriv
    secName = NazwaUzytkownikaSnmp
    authProtocol = SHA
    authPassword = HasloUwierzytelnienia
    privProtocol = AES
    privPassword = HasloSzyfrowania
    mibs = auto
    pollfreq = 10
```

SNMP v3 szyfruje zapytania, a v2c wysyła nazwę wspólnoty otwartym tekstem. Jeśli karta UPS obsługuje v3, wybieramy ten wariant.

SNMP = Simple Network Management Protocol: protokół odczytu stanu urządzeń sieciowych · **UPS** = Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci

ZMIENIA SYSTEM

Uprawnienia plików konfiguracyjnych: chown i chmod

```
sudo chown root:nut /etc/nut/ups.conf /etc/nut/upsd.users /etc/nut/upsmon.conf
sudo chmod 640 /etc/nut/ups.conf /etc/nut/upsd.users /etc/nut/upsmon.conf
```

Pliki zawierają hasła, więc tylko `root` i grupa `nut` mogą je czytać.

chown = change owner: zmień właściciela · **chmod** = change mode: zmień prawa dostępu

TYLKO ODCZYT

Plik nasłuchiwania serwera: /etc/nut/upsd.conf

```
LISTEN 127.0.0.1 3493
```

Serwer `upsd` domyślnie nasłuchuje tylko na adresie lokalnym. Adres sieciowy dodajemy kolejną linią `LISTEN AdresSerweraNut 3493` wyłącznie wtedy, gdy mają się łączyć klienci z innych serwerów.

TYLKO ODCZYT

Plik użytkowników NUT: /etc/nut/upsd.users

```
[monitor]
    password = HasloMonitor
    upsmon primary
```

Użytkownik `monitor` z rolą `primary` może wydać polecenie wyłączenia. Dla klientów na innych serwerach dopisujemy osobnego użytkownika z rolą `secondary`.

NUT = Network UPS Tools: zestaw programów na Linuksie, który odczytuje stan zasilacza UPS i wyłącza serwery, gdy baterie się kończą

□□ ZMIENIA SYSTEM

Plik monitorowania: /etc/nut/upsmon.conf

```
MONITOR ups01@localhost 1 monitor HasloMonitor primary
MINSUPPLIES 1
SHUTDOWNCMD "/usr/local/sbin/wylacz-serwerownie.sh"
NOTIFYCMD /usr/sbin/upssched
POLLFREQ 5
POLLFREQALERT 5
HOSTSYNC 15
DEADTIME 15
POWERDOWNFLAG /etc/killpower
RBWARNTIME 43200
NOCOMMWARNTIME 300
FINALDELAY 5
NOTIFYFLAG ONLINE SYSLOG+EXEC
NOTIFYFLAG ONBATT SYSLOG+EXEC
NOTIFYFLAG LOWBATT SYSLOG+EXEC
NOTIFYFLAG FSD SYSLOG+EXEC
NOTIFYFLAG COMMOK SYSLOG+EXEC
NOTIFYFLAG COMMBAD SYSLOG+EXEC
NOTIFYFLAG SHUTDOWN SYSLOG+EXEC
NOTIFYFLAG REPLBATT SYSLOG+EXEC
NOTIFYFLAG NOCOMM SYSLOG+EXEC
```

`upsmon` wywołuje `SHUTDOWNCMD` na serwerze głównym, gdy UPS jest w stanie krytycznym albo gdy ktoś wymusi wyłączenie poleceniem `upsmon -c fsd`. Każde zdarzenie trafia do `NOTIFYCMD`, czyli do `upssched`, który liczy czas [4].

□□ **UPS** = Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci

□□ TYLKO ODCZYT

Katalog dla gniazda upssched: /etc/tmpfiles.d/upssched.conf

```
d /run/upssched 0750 nut nut -
```

Program `upssched` potrzebuje katalogu, w którym użytkownik `nut` może założyć gniazdo i plik blokady [5].

ZMIENIA SYSTEM

Utworzenie katalogu z pliku tmpfiles: systemd-tmpfiles

```
sudo systemd-tmpfiles --create /etc/tmpfiles.d/upssched.conf
```

systemd-tmpfiles = systemd tmpfiles: tworzy katalogi i pliki tymczasowe według plików w /etc/tmpfiles.d

ZMIENIA SYSTEM

Plik harmonogramu zdarzeń: /etc/nut/upssched.conf

```
CMDSCRIPT /usr/local/sbin/upssched-cmd
PIPEFN /run/upssched/upssched.pipe
LOCKFN /run/upssched/upssched.lock

AT ONBATT * START-TIMER wylaczenie 180
AT ONLINE * CANCEL-TIMER wylaczenie powrot-sieci
AT LOWBATT * EXECUTE wylaczenie-natychmiast
AT COMMBAD * START-TIMER brak-komunikacji 300
AT COMMOK * CANCEL-TIMER brak-komunikacji
AT REPLBATT * EXECUTE wymien-baterie
```

Po 180 sekundach pracy na baterii timer `wylaczenie` uruchamia skrypt. Powrót sieci anuluje timer, a polecenie `powrot-sieci` zadziała tylko wtedy, gdy timer zdążył już minąć [5].

TYLKO ODCZYT

Skrypt wywoływany przez upssched: /usr/local/sbin/upssched-cmd

```
#!/bin/bash
# Wywolywany przez upssched. Argument to nazwa timera albo polecenia z upssched.conf.
case "$1" in
  wylaczenie)
    logger -t upssched "UPS na baterii od 3 minut: wymuszamy wylaczenie serwerowni"
```

```
    /usr/sbin/upsmon -c fsd
    ;;
  wylaczenie-natychmiast)
    logger -t upssched "UPS zglosil niski stan baterii: wylaczamy natychmiast"
    /usr/sbin/upsmon -c fsd
    ;;
  powrot-sieci)
    logger -t upssched "Siec wrocila juz po rozpoczeciu wylaczania: sprawdzamy stan
serwerowni"
    ;;
  brak-komunikacji)
    logger -t upssched "Brak komunikacji z UPS od 5 minut"
    ;;
  wymien-baterie)
    logger -t upssched "UPS zglasza potrzebe wymiany baterii"
    ;;
  *)
    logger -t upssched "Nieznane polecenie: $1"
    ;;
esac
```

Polecenie `upsmon -c fsd` ustawia stan wymuszonego wyłączenia (FSD), po którym `upsmon` na serwerze głównym wywołuje `SHUTDOWNCMD`.

FSD = Forced ShutDown: stan wymuszonego wyłączenia w NUT, po którym serwer główny rozpoczyna wyłączanie · **logger** = Logger: zapisuje komunikat w dzienniku systemowym · **UPS** = Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci

ZMIENIA SYSTEM

Nadanie uprawnień skryptowi: chmod 755

```
sudo chmod 755 /usr/local/sbin/upssched-cmd
```

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Skrypt wyłączania serwerowni: /usr/local/sbin/wylacz-serwerownie.sh

```
#!/bin/bash
# Uruchamiany przez upsmon (SHUTDOWNCMD) na bkp01. Z parametrem --proba tylko loguje
kroki.
PROBA="${1:-}"
KLUCZE=/etc/nut/klucze
SSH_OPCJE="-o BatchMode=yes -o ConnectTimeout=10 -o StrictHostKeyChecking=yes"
LOG() { logger -t wylacz-serwerownie "$*"; }

wykonaj() {
  if [ "$PROBA" = "--proba" ]; then
    LOG "PROBA: $*"
  else
    "$@"
  fi
}

LOG "Start wylaczania serwerowni"

# 1. Maszyny aplikacyjne na klastrze (skrypt na hv01 czeka, az sie zatrzymaja)
LOG "Faza 1: aplikacje"
wykonaj ssh $SSH_OPCJE -i "$KLUCZE/klucz_aplikacje" svc-wylaczanie@hv01 faza-aplikacje

# 2. Serwery fizyczne poza klastrem: ich bazy zatrzymuja sie przy zamykaniu systemu
for serwer in ora01 hana01 gpu01; do
  LOG "Wylaczamy $serwer"
  wykonaj ssh $SSH_OPCJE -i "$KLUCZE/klucz_wezel" "nazwa_uzytkownika@$serwer" wylacz
done

# 3. Reszta maszyn wirtualnych, kontrolery domeny na koncu, potem Stop-Cluster
LOG "Faza 2: reszta maszyn i klaster"
wykonaj ssh $SSH_OPCJE -i "$KLUCZE/klucz_reszta" svc-wylaczanie@hv01 faza-reszta

# 4. Wezly klastra
for wezel in hv02 hv01; do
  LOG "Wylaczamy wezel $wezel"
  wykonaj ssh $SSH_OPCJE -i "$KLUCZE/klucz_wezel" "svc-wylaczanie@$wezel" wylacz
done

LOG "Wylaczamy bkp01"
```

```
wykonaj /usr/sbin/shutdown -h now
```

Skrypt najpierw zatrzymuje aplikacje, potem serwery fizyczne (polecenie wyłączenia wraca od razu, więc ich bazy zatrzymują się równolegle), potem resztę maszyn i klaster, a na końcu węzły i siebie. Słowa po adresie (`faza-aplikacje`, `wylacz`) są tylko znacznikami: serwery i tak wykonują polecenie wymuszone przez klucz. Przed pierwszym użyciem łączymy się z każdym serwerem ręcznie jako `root` na `bkp01`, żeby zapisać odciski kluczy hostów (opcja `StrictHostKeyChecking=yes`).

□□ ZMIENIA SYSTEM

Nadanie uprawnień skryptowi wyłączania: chmod 750

```
sudo chmod 750 /usr/local/sbin/wylacz-serwerownie.sh
```

□□ **chmod** = change mode: zmień prawa dostępu

□□ TYLKO ODCZYT

Próba skryptu bez wyłączania serwerów: wylacz-serwerownie.sh --proba

```
sudo /usr/local/sbin/wylacz-serwerownie.sh --proba
```

□□ TYLKO ODCZYT

Odczyt dziennika próby: journalctl -t wylacz-serwerownie

```
sudo journalctl -t wylacz-serwerownie --since "10 minutes ago"
```

Przykład:

```
Oct 03 08:12:01 bkp01 wylacz-serwerownie[2210]: Start wylaczania serwerowni
Oct 03 08:12:01 bkp01 wylacz-serwerownie[2211]: Faza 1: aplikacje
Oct 03 08:12:01 bkp01 wylacz-serwerownie[2214]: PROBA: ssh -o BatchMode=yes -o
ConnectTimeout=10 -o StrictHostKeyChecking=yes -i /etc/nut/klucze/klucz_aplikacje svc-
wylaczanie@hv01 faza-aplikacje
Oct 03 08:12:01 bkp01 wylacz-serwerownie[2216]: Wylaczamy ora01
```

journalctl = journal control: czytanie dziennika systemd

ZMIENIA SYSTEM

Skrypt PowerShell na hv01: C:\Skrypty\Wylacz-Klaster.ps1

```
# Uruchamiany na hv01 przez ssh z bkp01 (klucze z wymuszonym poleceniem i parametrem
Faza).
param([Parameter(Mandatory)][ValidateSet('Aplikacje','Reszta')][string]$Faza)
$ErrorActionPreference = 'Continue'
New-Item -ItemType Directory -Force -Path 'C:\Skrypty\logi' | Out-Null
Start-Transcript -Path ("C:\Skrypty\logi\wylaczenie-{0}-{1}.log" -f $Faza, (Get-Date -
Format 'yyyyMMdd-HHmmss'))

# Role maszyn wirtualnych po kolei, z czekaniem do 300 sekund na każdą
foreach ($nazwa in Get-Content ("C:\Skrypty\kolejnosc-{0}.txt" -f $Faza.ToLower())) {
    $grupa = Get-ClusterGroup -Name $nazwa -ErrorAction SilentlyContinue
    if ($grupa -and $grupa.State -eq 'Online') {
        Write-Output ("{0} Zatrzymujemy {1}" -f (Get-Date -Format 'HH:mm:ss'), $nazwa)
        Stop-ClusterGroup -Name $nazwa -Wait 300 | Out-Null
    }
}

if ($Faza -eq 'Reszta') {
    # Czekamy do 2 minut, aż skończą się zadania naprawy dysków S2D
    $koniec = (Get-Date).AddMinutes(2)
    while ((Get-StorageJob | Where-Object { $_.JobState -ne 'Completed' }) -and ((Get-
Date) -lt $koniec)) {
        Start-Sleep -Seconds 10
    }
    # Zatrzymujemy klaster na wszystkich węzłach
    Write-Output ("{0} Stop-Cluster" -f (Get-Date -Format 'HH:mm:ss'))
    Stop-Cluster -Force -Confirm:$false
}
Stop-Transcript
```

Maszyny zatrzymujemy po kolei, bo `Stop-ClusterGroup` z parametrem `-Wait` czeka na zakończenie [6]. Węzły wyłącza dopiero `bkp01` osobnym kluczem, bo po wyłączeniu kontrolerów domeny konta domenowe przestają działać.

TYLKO ODCZYT

Lista aplikacji do wyłączenia najpierw: C:\Skrypty\kolejnosc-aplikacje.txt

```
app01
docker01
ci01
k3s01
```

TYLKO ODCZYT

Lista pozostałych ról w kolejności wyłączania: C:\Skrypty\kolejnosc-reszta.txt

```
sql01
pg01
my01
mongo01
cass01
cass02
cass03
syb01
fs01
rds01
wsus01
log01
ipa01
mon01
dc01
dc02
```

Nazwy muszą być identyczne z nazwami ról klastra. Bazy stoją na początku listy, a kontrolery domeny na samym końcu.

ZMIENIA SYSTEM

Ustawienie zamknięcia systemu gościa przy zatrzymaniu roli: Set-ClusterParameter

```
Get-ClusterResource | Where-Object { $_.ResourceType -eq 'Virtual Machine' } |
    ForEach-Object { $_ | Set-ClusterParameter -Name OfflineAction -Value 2 }
```

Domyślna wartość `OfflineAction` to 1 (zapis stanu maszyny), a wartość 2 oznacza zamknięcie systemu gościa [7]. Bez tej zmiany `Stop-ClusterGroup` zapisuje maszyny w stanie Saved, a nie wyłącza ich normalnie.

□□ TYLKO ODCZYT

Sprawdzenie ustawienia OfflineAction: Get-ClusterParameter

```
Get-ClusterResource | Where-Object { $_.ResourceType -eq 'Virtual Machine' } |
    ForEach-Object { $_ | Get-ClusterParameter -Name OfflineAction } |
    Format-Table ClusterObject, Name, Value
```

Przykład:

```
ClusterObject          Name          Value
-------------          ----          -----
Virtual Machine app01  OfflineAction 2
Virtual Machine sql01  OfflineAction 2
```

□□ TYLKO ODCZYT

Klucz SSH z wymuszonym poleceniem na hv01: administrators_authorized_keys

```
from="AdresBkp01",command="powershell.exe -NoProfile -ExecutionPolicy Bypass -File
C:/Skrypty/Wylacz-Klaster.ps1 -Faza Aplikacje",no-port-forwarding,no-pty ssh-ed25519
KluczAplikacje svc-wylaczanie@bkp01
from="AdresBkp01",command="powershell.exe -NoProfile -ExecutionPolicy Bypass -File
C:/Skrypty/Wylacz-Klaster.ps1 -Faza Reszta",no-port-forwarding,no-pty ssh-ed25519
KluczReszta svc-wylaczanie@bkp01
```

(gdzie `AdresBkp01` to adres IP `bkp01`, a `KluczAplikacje` i `KluczReszta` to zawartość plików `klucz_aplikacje.pub` i `klucz_reszta.pub`)

Plik `C:\ProgramData\ssh\administrators_authorized_keys` na Windows Server służy kontom z uprawnieniami administratora. Każdy klucz działa tylko z adresu `bkp01` i uruchamia jeden skrypt z jednym parametrem.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera · **IP** = Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)

ZMIENIA SYSTEM

Klucz SSH do wyłączania węzła: administrators_authorized_keys

```
from="AdresBkp01",command="shutdown.exe /s /t 5 /c \"UPS: wylaczanie serwerowni\"",no-port-forwarding,no-pty ssh-ed25519 KluczPubliczny svc-wylaczanie@bkp01
```

Ten klucz, `klucz_wezel`, dodajemy na `hv01` i `hv02`. Na `ora01`, `hana01` i `gpu01` wpisujemy go do `authorized_keys` konta wyłączającego z wymuszonym poleceniem `sudo /usr/sbin/shutdown -h now`.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera

ZMIENIA SYSTEM

Generowanie kluczy SSH na bkp01: ssh-keygen

```
sudo mkdir -p /etc/nut/klucze
sudo ssh-keygen -t ed25519 -N "" -f /etc/nut/klucze/klucz_aplikacje
sudo ssh-keygen -t ed25519 -N "" -f /etc/nut/klucze/klucz_reszta
sudo ssh-keygen -t ed25519 -N "" -f /etc/nut/klucze/klucz_wezel
sudo chown -R root:nut /etc/nut/klucze
sudo chmod 750 /etc/nut/klucze
sudo chmod 640 /etc/nut/klucze/klucz_aplikacje /etc/nut/klucze/klucz_reszta /etc/nut/klucze/klucz_wezel
```

Klucze bez hasła są potrzebne, bo nikt nie wpisze hasła w czasie awarii. Dlatego ograniczamy je adresem i wymuszonym poleceniem po stronie serwerów.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera · **mkdir** = make directory: utwórz katalog · **ssh-keygen** = wygeneruj parę kluczy SSH albo sprawdź odcisk klucza · **chown** = change owner: zmień właściciela · **chmod** = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Uruchomienie usług NUT: systemctl enable --now

```
sudo systemctl list-unit-files 'nut*'
sudo systemctl enable --now nut-server.service nut-monitor.service
```

Nazwy jednostek systemd różnią się między wersjami NUT, dlatego najpierw oglądamy listę. W wersji 2.8 sterowniki uruchamiają się razem z usługą serwera [8].

NUT = Network UPS Tools: zestaw programów na Linuksie, który odczytuje stan zasilacza UPS i wyłącza serwery, gdy baterie się kończą · **systemctl** = system control: sterowanie usługami systemd

TYLKO ODCZYT

Ręczny start sterownika: upsdrvctl start

```
sudo upsdrvctl start ups01
```

Jeśli usługi nie wystartowały, ręczny start pokazuje błąd sterownika, na przykład zły adres albo nazwę wspólnoty.

upsdrvctl = UPS driver control: uruchamia i zatrzymuje sterowniki UPS w NUT, także polecenie wyłączenia wyjścia UPS

TYLKO ODCZYT

Odczyt wszystkich wartości UPS przez NUT: upsc

```
upsc ups01@localhost
```

Przykład:

```
battery.charge: 100
battery.runtime: 1518
ups.load: 50
ups.status: OL
```

(wynik skrócony do czterech zmiennych; `OL` oznacza pracę z sieci, `OB` z baterii, a `LB` niski stan baterii)

UPS = Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci · **NUT** = Network UPS Tools: zestaw programów na Linuksie, który odczytuje stan zasilacza UPS i wyłącza serwery, gdy baterie się kończą · **upsc** = UPS client: odczytuje zmienne

UPS z serwera NUT, na przykład ups.status albo battery.charge

TYLKO ODCZYT

Odczyt samego stanu UPS: upsc ups.status

```
upsc ups01@localhost ups.status
```

UPS = Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci · **upsc** = UPS client: odczytuje zmienne UPS z serwera NUT, na przykład ups.status albo battery.charge

ZMIENIA SYSTEM

Reguła firewalld dla klienta NUT na innym serwerze: firewall-cmd

```
sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source
address="AdresKlientaNut" port port="3493" protocol="tcp" accept'
sudo firewall-cmd --reload
```

Tę regułę dodajemy tylko wtedy, gdy na innym serwerze uruchamiamy `upsmon` w roli `secondary`. W układzie opisanym tutaj nie jest potrzebna.

NUT = Network UPS Tools: zestaw programów na Linuksie, który odczytuje stan zasilacza UPS i wyłącza serwery, gdy baterie się kończą · **firewall-cmd** = sterowanie zaporą firewalld

Z życia wzięte

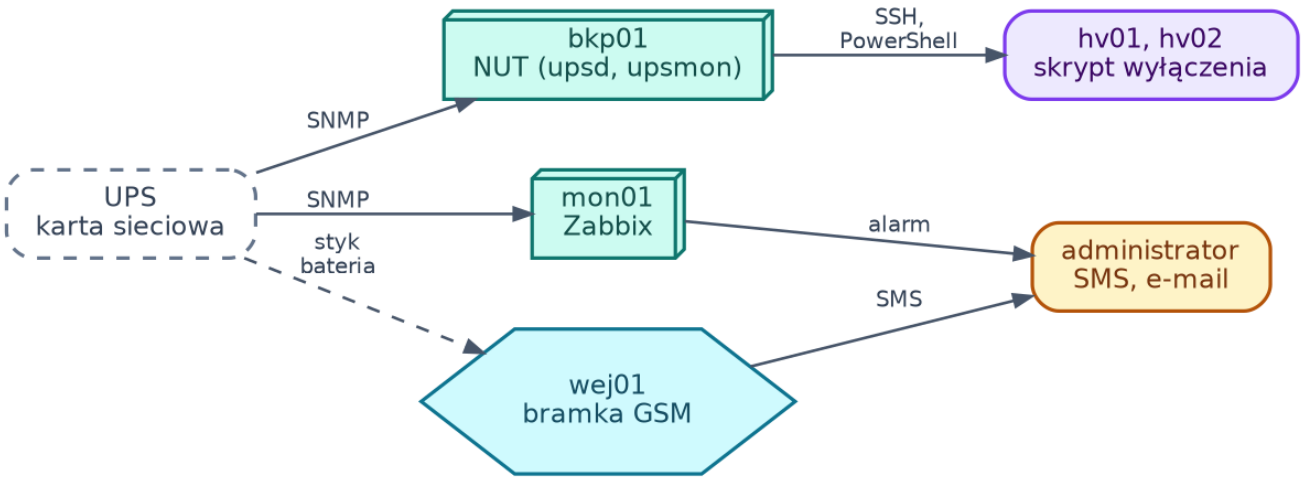
Z ŻYCIA WZIĘTE

Sobota, 7:30, okno serwisowe zaplanowane z wyprzedzeniem, bo pan Zbigniew nie lubi niespodzianek. Baterie są już wymienione i mamy dwa komplety, kopie zapasowe zrobione w piątek wieczorem. Tomek trzyma stoper, wyłącza wyłącznik wejścia UPS, a o 7:33 `upssched` ogłasza wyłączenie. O 7:44 gaśnie ostatni węzeł `hv01`, więc od zaniku sieci minęło 14 minut, a UPS jeszcze długo ma zapas.

Jak to działa

Rozwiń: jak to działa pod spodem

> **Pojęcie: łagodne wyłączanie.** Wyłączenie serwera w kolejności i z zapisaniem danych, zamiast nagłego odcięcia prądu. Bazy kończą transakcje, systemy plików się odmontowują, a po powrocie prądu serwer startuje bez naprawiania.



Rysunek 1. Droga informacji: UPS udostępnia stan przez kartę sieciową. NUT na bkp01 i Zabbix na mon01 odpytują go przez SNMP, a przerywana linia pokazuje niezależny styk „praca na baterii" do wej01, który wysyła SMS nawet gdy mon01 nie działa. (kliknięcie otwiera pełny rozmiar)

> **Pojęcie: SNMP.** Protokół, którym urządzenia sieciowe, w tym UPS, odpowiadają na pytania o swój stan. Każda wartość ma numer, czyli OID. Dla UPS-ów istnieje wspólny słownik wartości, UPS-MIB opisany w RFC 1628 [2].

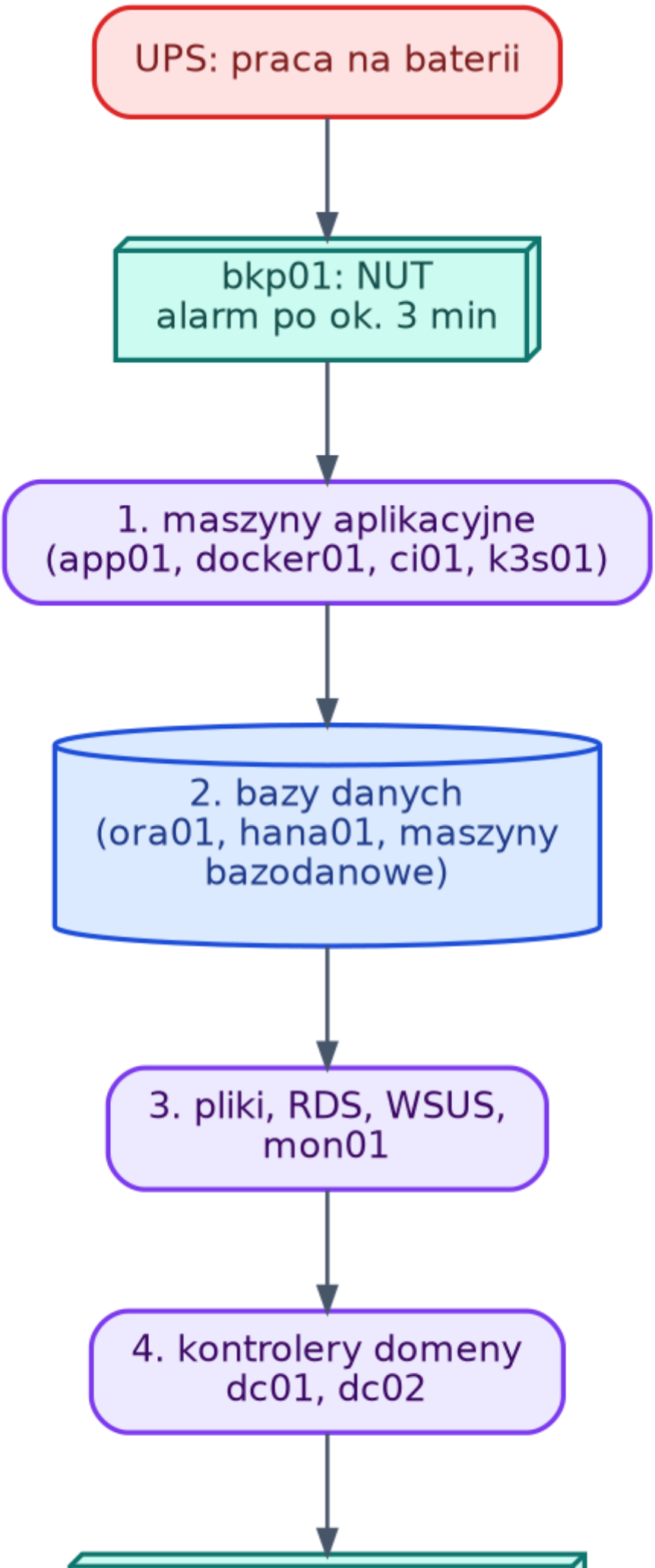
> **Pojęcie: NUT (Network UPS Tools).** Zestaw programów, który odczytuje stan UPS (sterowniki, np. `snmp-ups`), udostępnia go przez sieć (`upsd`) i monitoruje (`upsmon`). Powstał jako projekt wolnego oprogramowania i rozwija się od końca lat dziewięćdziesiątych [9].

Dlaczego NUT na bkp01

NUT musi działać na maszynie, która wyłącza się ostatnia i nie jest na klastrze, który wyłączamy. `mon01` jest maszyną wirtualną na klastrze, więc zgaśnie przed końcem. `bkp01` to fizyczny serwer kopii zapasowych poza klastrem, z Rocky Linux 9, więc może odpytywać UPS, zarządzać wyłączeniem i zgasnąć na końcu.

Kolejność wyłączania

Zasada jest prosta: konsumenci przed dostawcami. Najpierw maszyny aplikacyjne, potem bazy danych, serwery plików i pulpity, na końcu usługi, od których wszystko zależy: kontrolery domeny, a potem klaster i węzły. Serwery fizyczne `ora01`, `hana01` i `gpu01` wyłączamy zaraz po aplikacjach, razem z maszynami bazodanowymi, bo zatrzymanie bazy trwa najdłużej.


UPS: praca na baterii
bkp01: NUT
alarm po ok. 3 min
1. maszyny aplikacyjne
(app01, docker01, ci01, k3s01)
2. bazy danych
(ora01, hana01, maszyny
bazodanowe)
3. pliki, RDS, WSUS,
mon01
4. kontrolery domeny
dc01, dc02

Rysunek 2. Kolejność wyłączania: aplikacje, bazy, usługi pomocnicze, kontrolery domeny, klaster i węzły, na końcu bkp01. Warto zauważyć, że kontrolery domeny gasną prawie ostatnie, bo bez nich logowanie się do reszty przestaje działać. (kliknięcie otwiera pełny rozmiar)

Hyper-V i S2D

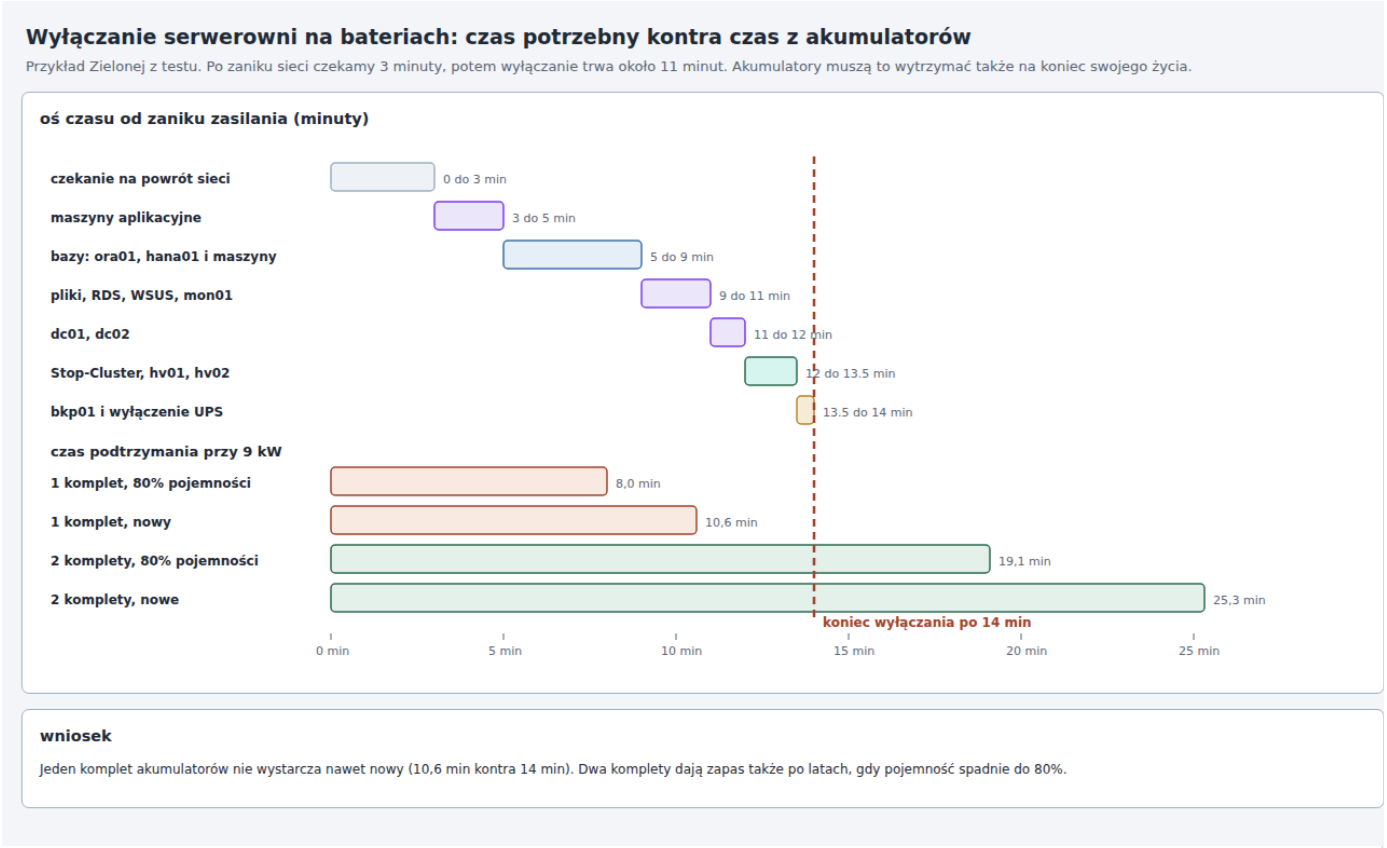
Wyłączenie klastra Hyper-V z S2D ma kilka pułapek. Polecenie `Stop-Cluster` zatrzymuje usługę klastra na wszystkich węzłach, ale domyślnie maszyny wirtualne trafiają w stan Saved zamiast zostać normalnie wyłączone [6][7]. Dlatego ustawiamy `OfflineAction` na 2 i zatrzymujemy role po kolei.

Procedura Microsoftu dla S2D wymaga też sprawdzenia, że nie trwają zadania naprawy dysków (`Get-StorageJob` zwraca pusty wynik), a potem zatrzymania klastra i wyłączenia węzłów [10]. Po powrocie prądu włączamy węzły, startujemy klaster poleceniem `Start-Cluster`, sprawdzamy dyski wirtualne i zadania naprawy, a dopiero potem uruchamiamy maszyny [10].

Czas

Czas wyłączania mierzymy w teście, a nie zakładamy. U Zielonej trzy minuty czekania i około jedenaście minut samego wyłączania dają 14 minut, a nasze baterie mają dawać co najmniej 15 przy 9 kW dla ogniw z 80 procentami pojemności. Jeśli test pokaże dłuższy czas, skracamy czas czekania, równolegle wyłączamy maszyny albo dokładamy baterie.

> ⚠ Baterie to nie jedyny zegar. Klimatyzacje A i B nie są zasilane z UPS, więc przy zaniku sieci chłodzenie staje od razu, a temperatura na wlotach szaf może dojść do 32 °C szybciej niż w 14 minut (szacunek w księdze Chłodzenie to 5 do 11 minut, do potwierdzenia próbą). Dlatego oprócz wyzwalacza „praca na baterii” planujemy wyzwalacz od temperatury wlotu, skracamy czekanie albo najpierw wyłączamy sprzęt, który grzeje, a nie jest potrzebny (`gpu01`, hosty laboratoryjne). Rachunek opisuje strona Planowanie chłodzenia serwerowni: bilans, zapas N+1, awaria klimatyzacji.



Rysunek 3. Czas wyłączania na tle czasu z baterii. Warto zauważyć, że jeden komplet baterii nie wystarcza nawet nowy, a dwa komplety dają zapas także przy 80 procentach pojemności. (kliknięcie otwiera pełny rozmiar)

Dlaczego czekamy trzy minuty

Większość zaników napięcia trwa kilka sekund albo minut. Trzy minuty (nastawa `START-TIMER wylaczenie 180`) odsiewa krótkie zaniki, więc nie wyłączamy serwerowni z powodu mrugnięcia. Jeśli zaniki w Pamiątkowie są dłuższe, ale rzadko trwają ponad dziesięć minut, wydłużamy czas, ale tylko do wartości, którą baterie wytrzymają razem z czasem wyłączania.

Drugi wariant: UPS z USB i apcupsd

Przy małym UPS podłączonym przez USB do jednego serwera wystarczy sterownik `usbhid-ups` z ustawieniem `port = auto` w `ups.conf`, bez SNMP. W UPS firmy APC można też użyć programu `apcupsd`, który wywodzi się z projektu Pavla Kořenskiego i jest rozwijany przez społeczność [11]. W serwerowni wybieramy jednak NUT z kartą sieciową, bo jeden UPS obsługuje wiele serwerów.

Serwery fizyczne, które nie mają być wyłączane przez `bkp01`, mogą działać jako klienci NUT: uruchamiamy na nich `upsmon` w roli `secondary`, wskazujemy `bkp01` i ustawiamy `SHUTDOWNCMD "/sbin/shutdown -h +0"`. Serwer główny czeka wtedy do `HOSTSYNC` sekund, aż klienci się wylogują, i dopiero potem wyłącza siebie.

Integracja z monitoringiem

Zabbix na `mon01` odpytuje kartę UPS bezpośrednio przez SNMP, niezależnie od NUT. Dzięki temu mamy dwie niezależne ścieżki: NUT wyłącza serwery, a Zabbix alarmuje ludzi. Dla UPS firmy APC Zabbix ma gotowy szablon „APC UPS by SNMP", który zawiera m.in. progi pojemności baterii (50 procent), obciążenia wyjścia (80 procent) i temperatury [12].

Jeśli szablonu dla naszego producenta nie ma, tworzymy elementy danych z OID z UPS-MIB. Host `ups01` ma interfejs SNMP z adresem karty, a elementy danych typu „SNMP agent" odpytują co 30 sekund.

Nazwa elementu	Klucz	OID	Znaczenie
Stan baterii	`ups.battery.status`	`1.3.6.1.2.1.33.1.2.1.0`	2 norma, 3 niski, 4 wyczerpana
Czas na baterii	`ups.seconds.on.battery`	`1.3.6.1.2.1.33.1.2.2.0`	sekundy pracy z baterii
Szacowany czas pracy	`ups.battery.minutes`	`1.3.6.1.2.1.33.1.2.3.0`	minuty do rozładowania
Naładowanie baterii	`ups.battery.charge`	`1.3.6.1.2.1.33.1.2.4.0`	procent
Źródło zasilania wyjścia	`ups.output.source`	`1.3.6.1.2.1.33.1.4.1.0`	3 sieć, 4 bypass, 5 bateria
Obciążenie wyjścia	`ups.output.load`	`1.3.6.1.2.1.33.1.4.4.1.5.1`	procent mocy

Wyzwalacze w składni Zabbix 7.0:

Nazwa	Wyrażenie	Ważność
UPS pracuje na baterii	`last(/ups01/ups.output.source)=5`	wysoka
Niski stan baterii	`last(/ups01/ups.battery.status)>=3`	katastrofa
Bateria poniżej 50 procent	`last(/ups01/ups.battery.charge)<50`	ostrzeżenie
Za krótki czas na baterii (czas dla pełnego naładowania poniżej planowanego)	`last(/ups01/ups.battery.minutes)<20 and last(/ups01/ups.output.source)=3`	ostrzeżenie
Obciążenie powyżej 80 procent	`last(/ups01/ups.output.load)>80`	ostrzeżenie
Brak danych z UPS	`nodata(/ups01/ups.output.source,5m)=1`	wysoka

Ostatnie ostrzeżenie o czasie służy do wykrywania zużytych baterii: przy pełnym naładowaniu UPS powinien zapowiadać co najmniej tyle minut, ile planujemy. Spadek poniżej progu to sygnał do

próby i wymiany baterii.

Alarmy wysokie i katastrofy wysyłamy jako SMS i telefon do dyżurnego, jak opisuje strona Serwerownia Zielonej: gdzie stoją czujniki i dokąd płyną alarmy. UPS ma też styk „praca na baterii", który wpinamy do modułu `wej01`. Ten ma własny modem GSM, więc wyśle SMS także wtedy, gdy `mon01` już nie działa.

Test wyłączania

Test robimy w oknie serwisowym, po uprzedzeniu firmy i po sprawdzeniu, że kopie zapasowe są świeże. Wykonujemy trzy testy, od najbezpieczniejszego.

1. Próba na sucho: `wylacz-serwerownie.sh --proba` sprawdza, że skrypt działa i widzi wszystkie kroki.
2. Test kluczy: na `hv01` wpisujemy tymczasowo klucz z poleceniem `hostname` zamiast skryptu, sprawdzamy połączenie z `bkp01` i dopiero potem przywracamy właściwe polecenie.
3. Test prawdziwy: wyłączamy wejście UPS w rozdzielnicy (Q10), mierzymy czas i patrzymy w dziennik.

W teście prawdziwym zapisujemy czas rozpoczęcia, czas wyłączenia każdego serwera i czas do ostatniego węzła. Porównujemy go z czasem baterii z tabeli producenta. Po teście włączamy najpierw węzły i klaster, a potem maszyny w odwrotnej kolejności niż przy wyłączaniu: kontrolery domeny, usługi pomocnicze, bazy, aplikacje. Priorytety startu ról ustawia się w klastrze, więc zapisujemy je w dokumentacji.

Bezpieczeństwo i ograniczenia

- Hasła w plikach NUT mają uprawnienia 640, a klucze SSH są ograniczone adresem i wymuszonym poleceniem.
- Na `hv01` i `hv02` musi działać serwer OpenSSH (opcjonalna funkcja Windows Server), a jego dostęp ograniczamy zaporą do adresu `bkp01`.
- Konto `svc-wylaczanie` jest lokalne na `hv01` i `hv02`, a nie domenowe, bo kontrolery domeny wyłączamy przed klastrem.
- Karta SNMP UPS jest w sieci serwerowej z listą dozwolonych adresów (`bkp01`, `mon01`) i bez dostępu z internetu.
- Nazwę wspólnoty SNMP v2c zmieniamy z domyślnej, a gdzie się da, stosujemy SNMP v3.
- NUT i Zabbix nie zastąpią próby: wyłączanie, które nigdy nie było sprawdzone, nie zadziała w dniu awarii.
- W BIOS-ie serwerów ustawiamy powrót do pracy po powrocie zasilania (opcja w rodzaju „Restore on AC Power Loss"), jeśli chcemy, żeby wstały same po awarii.

- Po powrocie prądu serwery nie wstają same w dobrej kolejności, więc dokumentujemy ręczną procedurę uruchamiania.

Częste błędy

Objaw	Przyczyna	Co robimy
`upsc` zwraca „Data stale"	sterownik nie odpowiada na SNMP (zły adres, wspólnota, zapora)	`snmpget` z `bkp01`, a potem `upsdrvctl start ups01`
Wyłączanie nie startuje po trzech minutach	brak katalogu `/run/upssched` albo skrypt bez prawa wykonania	tworzymy katalog (tmpfiles), nadajemy `chmod 755`
Skrypt działa ręcznie, a z `upsmon` nie	możliwa blokada SELinux (domena `nut` może nie mieć prawa uruchomić `ssh`)	`sudo ausearch -m avc -ts recent`, a w razie potrzeby dedykowana reguła lub usługa systemd
Maszyny wirtualne po powrocie są w stanie Saved	`OfflineAction` = 1 (domyślnie)	ustawiamy 2 i powtarzamy test
Klaster nie wstaje po powrocie prądu	węzły włączone w złej kolejności albo trwa naprawa dysków	sprawdzamy `Get-StorageJob`, startujemy klaster z jednego węzła
Serwerownia wyłącza się po krótkim mrugnięciu	brak czekania, `FSD` wysłane od razu	używamy `START-TIMER` z czasem 180 s
Wyłączanie trwa dłużej niż baterie	czas nie był mierzony, baterie zużyte	skracamy czekanie, równolegle wyłączamy maszyny, dokładamy baterie

□□ Ćwiczenia

1. UPS pokazuje `upsOutputSource` równe 5. Co to znaczy i który wyzwalacz zadziała?

Odpowiedź

UPS pracuje z baterii. Zadziała wyzwalacz „UPS pracuje na baterii" o wysokiej ważności, a `upssched` w NUT rozpoczyna odliczanie 180 sekund.

1. Sieć wraca po 90 sekundach od przejścia na baterie. Czy serwerownia się wyłączy?

Odpowiedź

Nie. Zdarzenie ONLINE anuluje timer `wylaczenie`, który miał minąć po 180 sekundach, więc `upsmon -c fsd` nie zostanie wywołane.

1. Baterie dają 12 minut przy 9 kW, a wyłączanie trwa 11 minut plus 3 minuty czekania. Co zmieniamy?

Odpowiedź

Plan wymaga 14 minut, a baterie dają 12, więc nie wystarczy. Możemy skrócić czekanie, równolegle wyłączać maszyny albo dołożyć komplet baterii. Dokładamy też margines na zużycie baterii.

Powiązane: UPS: topologie, dobór i baterie, Planowanie zasilania serwerowni: bilans mocy, zapas, rozbudowa, Odbiór instalacji od wykonawcy: lista kontrolna, pomiary, dokumenty, Serwerownia Zielonej: gdzie stoją czujniki i dokąd płyną alarmy

Źródła:

1. Fedora Packages, pakiet nut dla EPEL 9: packages.fedoraproject.org
2. RFC 1628, UPS Management Information Base: rfc-editor.org
3. NUT, opis sterownika snmp-ups: networkupstools.org
4. NUT, upsmon.conf: networkupstools.org
5. NUT, upssched.conf: networkupstools.org
6. Microsoft Learn, Stop-Cluster i Stop-ClusterGroup: learn.microsoft.com
7. Microsoft, właściwość OfflineAction zasobu maszyny wirtualnej w klastrze: learn.microsoft.com
8. NUT, podręcznik użytkownika: networkupstools.org
9. en.wikipedia.org: Network UPS Tools
10. Microsoft Learn, „What are the steps to shut down / boot the entire system in an Azure Stack HCI configuration?": learn.microsoft.com
11. en.wikipedia.org: apcupsd
12. Zabbix, szablon APC UPS by SNMP: zabbix.com

Revision #15
Created 2026-10-03 11:23:30 UTC by Tomasz Konieczny
Updated 2026-10-26 14:19:45 UTC by Tomasz Konieczny