Agent Zabbixa: instalacja i konfiguracja

□□ Poziom: podstawy **□□ Dotyczy: Zabbix agent2 6.0 LTS, RHEL/Rocky 9, Debian/Ubuntu**

□ Czas: ~20 min

Zanim dodajemy nowy host do monitoringu, instalujemy i konfigurujemy na nim agenta, a dopiero potem sprawdzamy z serwera, czy odpowiada.

□□ Skróty użyte na tej stronie (4)

Skrót	Co znaczy
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
LTS	Long Term Support: wersja z długim wsparciem poprawkami
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SELinux	Security-Enhanced Linux: dodatkowa warstwa uprawnień w jądrze RHEL

Fiszka

□□ TYLKO ODCZYT

Dodanie repozytorium Zabbixa: RHEL/Rocky

```
sudo rpm -Uvh https://repo.zabbix.com/zabbix/6.0/rhel/9/x86_64/zabbix-release-6.0-4.el9.noarch.rpm
```

Dodajemy oficjalne repozytorium Zabbixa, bo wersja w standardowych repozytoriach dystrybucji jest zwykle znacznie starsza niż aktualna gałąź LTS.

□□ **RHEL** = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa · **LTS** = Long Term Support: wersja z długim wsparciem poprawkami · **rpm** = RPM Package Manager: format i narzędzie paczek RHEL

□□ ZMIENIA SYSTEM

Instalacja agenta z repozytorium: RHEL/Rocky

```
sudo dnf install -y zabbix-agent2 zabbix-agent2-plugin-postgresql
```

Instalujemy sam agent oraz — opcjonalnie — pakiet wtyczki, jeśli host ma monitorowaną bazę PostgreSQL kontem monitorującym, temat rozwijamy na stronie Monitorowanie typowych usług.

□□ **RHEL** = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa · **dnf** = Dandified YUM: menedżer pakietów RHEL/Rocky

□□ ZMIENIA SYSTEM

Dodanie repozytorium i instalacja agenta: Debian/Ubuntu

```
wget https://repo.zabbix.com/zabbix/6.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.0-4+ubuntu22.04_all.deb
sudo dpkg -i zabbix-release_6.0-4+ubuntu22.04_all.deb
sudo apt update
sudo apt install -y zabbix-agent2
```

Sekwencja ma sens tylko wklejona w całości — pobranie pakietu z definicją repozytorium, jego instalacja, odświeżenie listy pakietów, a dopiero potem instalacja właściwego agenta.

□□ **wget** = World Wide Web get: pobieranie plików · **dpkg** = Debian package: niskopoziomowe narzędzie instalujące paczki .deb w Debianie i Ubuntu · **apt** = Advanced Package Tool: menedżer pakietów Debiana i Ubuntu

□□ PLIK KONFIGURACYJNY

Podstawowa konfiguracja: zabbix_agentd.conf

```
Server=10.20.10.51
ServerActive=10.20.10.51
Hostname=NazwaHosta
HostMetadata=linux
Include=/etc/zabbix/zabbix_agent2.d/*.conf
```

(gdzie `10.20.10.51` to adres serwera Zabbixa albo lokalnego proxy, a `NazwaHosta` musi dokładnie odpowiadać nazwie hosta wpisanej w konfiguracji Zabbixa — wielkość liter ma znaczenie!)

Pojęcie: Server. Lista adresów, z których agent przyjmuje połączenia przychodzące przy sprawdzeniach pasywnych — jeśli połączenie przyjdzie z adresu spoza tej listy, agent je odrzuci niezależnie od tego, czy klucz istnieje.

Pojęcie: ServerActive. Adres (albo lista adresów, rozdzielona przecinkami), do którego agent sam się łączy przy sprawdzeniach aktywnych, żeby pobrać listę elementów do zbierania i odesłać wyniki.

Pojęcie: Hostname. Nazwa, pod jaką agent przedstawia się serwerowi przy sprawdzeniach aktywnych — musi być identyczna z polem „Nazwa hosta widoczna" (visible name jest inną, kosmetyczną wartością) w konfiguracji hosta we frontendzie, inaczej serwer nie skojarzy przychodzących danych z żadnym hostem.

Pojęcie: HostMetadata. Dowolny tekst opisujący hosta, wykorzystywany przy rejestracji automatycznej do dopasowania reguły — najczęściej wpisujemy tu rolę hosta albo system operacyjny, żeby reguła rejestracji mogła przypisać go do właściwej grupy i szablonu bez ręcznej interwencji.

Pojęcie: Include. Katalog albo pojedynczy plik dociągany do głównej konfiguracji — trzymamy tu osobne pliki na wtyczki i `UserParameter`, żeby nie edytować głównego `zabbix_agent2.conf` przy każdej drobnej zmianie.

□□ TYLKO ODCZYT

Rejestracja automatyczna hosta: reguła po stronie serwera

Rejestracja automatyczna (autoregistration) nie jest ustawieniem agenta — to reguła zdefiniowana we frontendzie serwera, dopasowująca nowe, nieznane jeszcze hosty po wartości `HostMetadata` albo po adresie IP i automatycznie tworząca dla nich wpis hosta z przypisaną grupą i szablonem. Agent musi mieć jedynie poprawnie ustawiony `ServerActive` i unikalny `Hostname` — resztą zajmuje się reguła po stronie serwera, więc nowy host trafia do monitoringu bez ręcznego klikania we frontendzie.

□□ **IP** = Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)

□□ ZMIENIA SYSTEM

Uruchomienie i włączenie w autostarcie: systemd

```
sudo systemctl enable --now zabbix-agent2
```

Włączamy usługę na start systemu i uruchamiamy ją od razu, w jednej komendzie.

□□ **systemctl** = system control: sterowanie usługami systemd

□□ TYLKO ODCZYT

Test z serwera: zabbix_get

```
zabbix_get -s AdresHosta -k agent.ping
```

Wykonujemy z serwera albo proxy zapytanie identyczne z tym, jakie realnie wysyła Zabbix przy sprawdzeniu pasywnym — to jest pierwsza komenda diagnostyczna, gdy host we frontendzie pokazuje status niedostępny.

Przykład:

```
1
```

> **Pojęcie: agent.ping.** Najprostszy klucz agenta, zwracający zawsze wartość `1`, jeśli agent w ogóle odpowiada — nie sprawdza niczego poza samym faktem, że proces żyje i odpowiada na porcie 10050, więc jest pierwszym, a nie jedynym testem.

□□ **zabbix_get** = ręczne odpytanie agenta Zabbixa

□□ TYLKO ODCZYT

Test lokalny na hoście: zabbix_agent2 -t

```
zabbix_agent2 -t agent.ping
```

Wykonujemy sprawdzenie lokalnie, z poziomu samego hosta, z pominięciem sieci — jeśli ten test przechodzi, a `zabbix_get` z serwera nie, problem leży w sieci albo w firewallu, a nie w samym agencie.

Przykład:

```
agent.ping                                [t|1]
```

zabbix_agent2 = sam proces/binarka agenta Zabbixa drugiej generacji, wywoływana też ręcznie do testów

TYLKO ODCZYT

Sprawdzenie logu agenta

```
sudo tail -n 50 /var/log/zabbix/zabbix_agent2.log
```

Log agenta pokazuje błędy startowe (np. zły format pliku konfiguracyjnego), nieudane próby połączenia do serwera przy sprawdzeniach aktywnych, oraz odrzucone połączenia przychodzące spoza listy `Server`.

Przykład:

```
24601:20260922:081512.442 cannot send list of active checks: [111] Connection refused
```

tail = pokaż koniec pliku, z flagą -f śledzi plik na żywo

Z życia wzięte

Z ŻYCIA WZIĘTE

O 10:05 nowy host `docker02` uparcie pokazuje status niedostępny, mimo że Tomek widzi proces agenta działający w `systemctl status`. Testuje lokalnie `zabbix_agent2 -t agent.ping` i dostaje `1`, więc problem musi leżeć między siecią a serwerem. Sprawdza `zabbix_get` z serwera i dostaje timeout, a po chwili w logu firewalla widzi, że reguła na porcie 10050 została dodana dla złego adresu źródłowego, bo skopiowano ją z innego hosta bez zmiany IP.

Agent 1 kontra agent 2

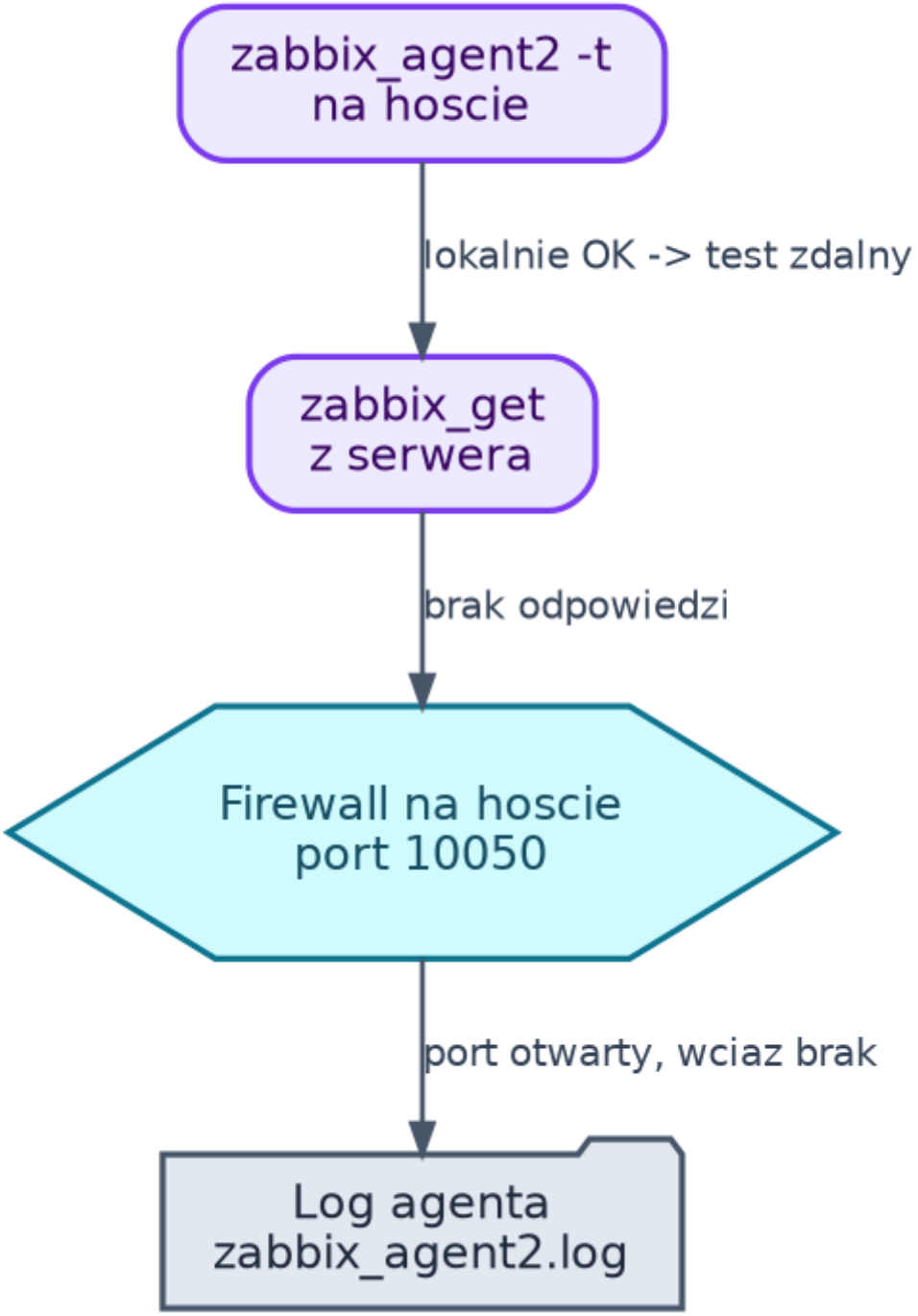
> **Pojęcie: zabbix-agent (agent 1).** Klasyczny agent napisany w C, obsługujący wszystkie standardowe klucze i moduły ładowane (`LoadModule`). Jest jednowątkowy na sprawdzenie — każde sprawdzenie typu `log[]` czy `vfs.file.cksum` na dużym pliku potrafi na chwilę zablokować kolejne sprawdzenia tego samego procesu roboczego.

> **Pojęcie: zabbix-agent2 (agent 2).** Nowsza implementacja w Go, kompatybilna wstecz z kluczami agenta 1, ale działająca w modelu asynchronicznym i rozszerzona o wtyczki (pluginy) do popularnych usług — m.in. gotowe monitorowanie MySQL, PostgreSQL, Docker bez pisania własnych `UserParameter`. Od Zabbixa 6.0 agent2 jest wariantem domyślnie zalecanym do nowych instalacji, a plik konfiguracyjny i większość parametrów pokrywają się z agentem 1 jeden do jednego.

W dalszej części strony posługujemy się nazwą `zabbix-agent2`, ale każda z poniższych fiszek działa analogicznie dla klasycznego `zabbix-agent`, poza sekcją o wtyczkach.

Firewall i SELinux — dwa najczęstsze powody niedostępności hosta

Zanim zaczniemy szukać błędu w samej konfiguracji Zabbixa, sprawdzamy te dwa elementy systemu operacyjnego — w praktyce odpowiadają za większość zgłoszeń „host niedostępny" przy pierwszym podłączaniu nowej maszyny.



Diagnostykę zaczynamy od testu lokalnego, dopiero potem sprawdzamy zdalnie, firewall i log agenta (kliknięcie otwiera pełny rozmiar)

Firewalld — otwarcie portu 10050 dla sprawdzeń pasywnych:

```
sudo firewall-cmd --permanent --add-port=10050/tcp
sudo firewall-cmd --reload
```

Wykonujemy to na hoście monitorowanym, jeśli w ogóle korzystamy ze sprawdzeń pasywnych — przy wyłącznie aktywnych sprawdzeniach ten port nie musi być otwarty, bo połączenie idzie w drugą stronę.

SELinux — sprawdzenie, czy blokuje agenta:

```
sudo ausearch -m avc -c zabbix_agentd --start recent
```

Wyszukujemy w logu audytu odmowy SELinuksa dotyczące procesu agenta z ostatniego okresu — typowym objawem jest agent działający lokalnie bez problemu, ale niepotrafiący odczytać pliku spoza domyślnie dozwolonych ścieżek (np. własnego `UserParameter` odwołującego się do logu aplikacji w niestandardowej lokalizacji).

Przykład:

```
type=AVC msg=audit(1758525312.221:481): avc:  denied  { read } for  pid=24601
comm="zabbix_agentd" name="app.log" dev="dm-2" scontext=system_u:system_r:zabbix_agent_t:s0
tcontext=unconfined_u:object_r:var_log_t:s0 tclass=file
```

Zamiast wyłączać SELinux globalnie, nadajemy dedykowany kontekst plikowi albo katalogowi:

```
sudo semanage fcontext -a -t zabbix_agent_t "SciezkaDoPliku(/.*)?"
sudo restorecon -Rv SciezkaDoPliku
```

Częste błędy

Wielkość liter w Hostname. `Hostname=DOCKER02` w pliku konfiguracyjnym i `docker02` we frontendzie to dla Zabbixa dwa różne hosty — agent zgłasza się poprawnie, ale serwer nie kojarzy przychodzących danych z żadnym zdefiniowanym hostem, więc historia pozostaje pusta mimo działającego agenta.

Zmiana konfiguracji bez restartu usługi. Agent, w odróżnieniu od niektórych demonów, nie przeładowuje `zabbix_agentd.conf` sam z siebie po edycji — po każdej zmianie pliku wykonujemy `sudo systemctl restart zabbix-agent2`.

Testowanie wyłącznie z serwera, nigdy lokalnie. Wynik `zabbix_get` mówi tylko, że coś nie działa gdzieś między serwerem a agentem — bez testu lokalnego (`zabbix_agent2 -t`) nie da się jednym krokiem odróżnić problemu sieciowego od problemu samego agenta.

□□ Ćwiczenia

>

> Wykonujemy na maszynie testowej z zainstalowanym agentem — najlepiej takiej, na której możemy bezkarnie popsuć plik konfiguracyjny.

Ćwiczenie 1 — diagnostyka niedostępnego hosta

Host pokazuje we frontendzie status niedostępny. `zabbix_agent2 -t agent.ping` lokalnie zwraca `1`. Wskazujemy kolejność sprawdzeń, jaką wykonujemy dalej.

Odpowiedź

Skoro lokalnie agent odpowiada, problem leży między siecią a serwerem: sprawdzamy `zabbix_get -s AdresHosta -k agent.ping` z poziomu serwera, następnie firewall na hoście monitorowanym (`firewall-cmd --list-all`), następnie zgodność `Hostname` w pliku konfiguracyjnym z nazwą hosta we frontendzie co do wielkości liter, na końcu log agenta pod kątem odrzuconych połączeń.

Pytania kontrolne

1. Dlaczego test `zabbix_agent2 -t` lokalnie i test `zabbix_get` z serwera mogą dać różny wynik dla tego samego klucza?

Odpowiedź

Bo omijają różne warstwy — test lokalny sprawdza wyłącznie logikę samego klucza wewnątrz agenta, pomijając sieć i listę `Server`. Test z serwera przechodzi przez sieć, firewall po obu stronach i listę dozwolonych adresów w `Server`, więc może się nie udać z powodów niemających nic wspólnego z samym kluczem.

Powiązane: Architektura Zabbixa: serwer, proxy, agent, frontend · Host, element danych, wyzwalacz, akcja: jak to się składa · Monitorowanie typowych usług

Źródła: dokumentacja Zabbix 6.0, rozdział „Installation" i „Zabbix agent 2" · `man zabbix_agent2` · `man firewall-cmd`

Revision #37
Created 2026-09-25 07:40:31 UTC by Tomasz Konieczny
Updated 2026-10-26 14:09:46 UTC by Tomasz Konieczny