Helion

<packt>

Zabbix 7 Receptury

Nowe funkcje projektowania, wdrażania i optymalizacji monitoringu infrastruktury IT

Wydanie III

Nathan Liefting

Brian van Baekel

Tytuł oryginału: Zabbix 7 IT Infrastructure Monitoring Cookbook: Explore the new features of Zabbix 7 for designing, building, and maintaining your Zabbix setup, 3rd Edition

Tłumaczenie: Krzysztof Sawka

ISBN: 978-83-289-2476-5

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/zab7r3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

Rozdział 4

Praca z wyzwalaczami i alertami

Na co zdałoby się gromadzenie tych wszystkich danych w Zabbiksie bez jakiejś formy powiadamiania administratora w przypadku pojawienia się nieprawidłowości? Możemy oczywiście używać Zabbiksa jedynie do zbierania danych i przeglądać je ręcznie, ale staje się on znacznie przydatniejszy, gdy zaczniemy wysyłać powiadomienia do użytkowników. W ten sposób nie musimy cały czas siedzieć we frontendzie Zabbix, lecz możemy pozwolić wyzwalaczom i alertom wykonywać za nas ciężką pracę, a sami wchodzić do frontendu tylko wtedy, gdy to konieczne.

W wersji Zabbix 7 występuje nowa składnia wyrażeń wyzwalaczy w porównaniu do wydania 5. Składnia ta występuje już od wersji 5.4, dlatego jeżeli miałeś dłuższą przerwę, to może być Twój pierwszy kontakt z nią. Jeżeli miałeś do czynienia jedynie ze starszymi wydaniami Zabbiksa, pamiętaj, że do nowej składni trzeba się przyzwyczaić. Począwszy od wersji 5.4, składnia ta wygląda tak samo w nowszych wydaniach.

Nauczymy się konfigurować skuteczne wyzwalacze za pomocą nowego formatu wyrażeń, a także korzystać z alertów w następujących recepturach:

- konfigurowanie wyzwalaczy,
- konfigurowanie zaawansowanych wyzwalaczy,
- konfigurowanie alertów,
- utrzymywanie alertów w dobrym stanie,
- dostosowywanie alertów.

Wymogi techniczne

W tym rozdziale będzie nam potrzebny serwer Zabbix, na przykład używany w poprzednim rozdziale:

- Serwer Zabbix zainstalowany w dowolnej dystrybucji Linuksa. Będziemy tu korzystać z serwera skonfigurowanego w rozdziale 1.
- Baza danych MariaDB dostosowana do pracy z serwerem Zabbix.

- Serwer NGINX lub Apache obsługujący frontend Zabbix.
- Będziemy również potrzebować hosta do monitorowania, na którym będziemy tworzyć interesujące wyzwalacze.

Konfigurowanie wyzwalaczy

Wyzwalacze (ang. *triggers*) stanowią ważny element Zabbiksa, ponieważ dzięki nim dowiadujemy się, co się dzieje z naszymi danymi. Chcemy, żeby wyzwalacz się uruchamiał, gdy wartości danych przekraczają określony próg lub gdy otrzymujemy wyznaczoną wartość.

Zatem zacznijmy konfigurować ciekawe wyzwalacze. Istnieje mnóstwo różnych ustawień definiujących wyzwalacze, ale po przeczytaniu tej receptury powinieneś być w stanie samodzielnie tworzyć najistotniejsze wyzwalacze. Wprowadźmy Twoje doświadczenia z wyzwalaczami na wyższy poziom.

Przygotowanie

W tej recepturze będzie nam potrzebny gotowy serwer Zabbix, a także host Linuksa. Będę używał hosta `lar-book-agent_simple` z poprzedniego rozdziału, ponieważ mamy już tam przygotowane pewne pozycje.

Będziemy także potrzebować jeszcze jednego hosta monitorowanego przez agenta Zabbix z opracowanym szablonem. Wykorzystamy jedną z pozycji znajdujących się na tym hoście do stworzenia wyzwalacza. Będzie to mianowicie host `lar-book-agent_passive` z poprzedniego rozdziału.

Na tym hoście już znajdują się pewne gotowe wyzwalacze, ale rozwiniemy je jeszcze bardziej, aby przekazywały nam dokładniejsze informacje.

Wykonanie

W tej sekcji utworzymy trzy wyzwalacze monitorujące zmiany stanu. Zacznijmy od pierwszego wyzwalacza.

Pierwszy wyzwalacz — monitorowanie usługi SSH

Stwórzmy prosty wyzwalacz na hoście `lar-book-agent_simple`. Stworzyliśmy tu już proste sprawdzenie o nazwie `Check if port 22 is available`, ale nie mamy jeszcze żadnego mechanizmu powiadamiającego nas o dostępności portu.

1. Najpierw otwórz stronę *Data collection*/*Hosts*, następnie wybierz hosta i przejdź do zakładki *Triggers*. To tutaj znajdziesz istniejące wyzwalacze i będziesz tworzyć nowe. Aby utworzyć nowy wyzwalacz, kliknij przycisk *Create trigger* w prawym górnym rogu ekranu.

2. Stwórz nowy wyzwalacz za pomocą informacji zawartych na rysunku 4.1.

Rysunek 4.1. Strona tworzenia wyzwalacza — usługa nieosiągalna

3. Kliknij przycisk *Add*, aby zakończyć tworzenie wyzwalacza. Otrzymasz wyzwalacz, który będzie uruchamiany, gdy port *SSH* (*Secure Shell*) stanie się niedostępny.
4. Przetestujmy jego działanie, otwierając wiersz poleceń na hoście, gdzie zablokujemy dostęp serwera Zabbix do portu `22`. Dodamy regułę `iptables`, blokującą ruch przychodzący do portu 22 (SSH):

```
iptables -A INPUT -p tcp -i ens192 -s 10.16.16.152 --destination-port
↪22 -j DROP
```

5. Zmień interfejs `ens192` na swój własny, jak również podaj swój adres IP w miejsce adresu `10.16.16.152`. Odpowiednie informacje uzyskasz po wpisaniu komendy:

```
ip addr
```

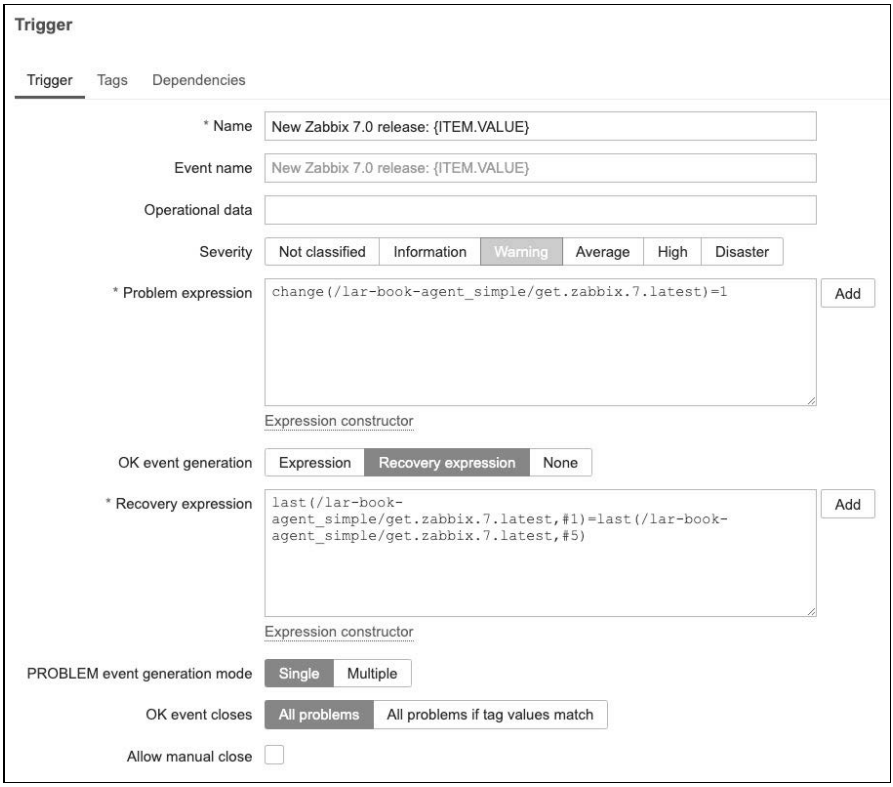
6. Teraz po kliknięciu kategorii *Dashboards* w pasku bocznym i odczekaniu krótkiej chwili powinieneś zobaczyć taki komunikat, jak na rysunku 4.2.

Rysunek 4.2. Problemy wyświetlane na pulpicie — niedziałający port 22

Drugi wyzwalacz — wyzwalanie w momencie wypuszczenia nowej wersji Zabbiksa

Podnieśmy teraz nieco poprzeczkę. W rozdziale 3., w recepturze „Konfigurowanie agenta monitorowania HTTP", utworzyliśmy pozycję sprawdzającą na stronie internetowej Zabbiksa najnowszą wersję oprogramowania. Teraz nawet sami dla siebie chcielibyśmy mieć oko na publikację nowej wersji Zabbiksa:

1. Otwórz stronę *Data collection*/*Hosts* i wybierz hosta `lar-book-agent_simple`.
2. Otwórz teraz *Triggers* i kliknij przycisk *Create trigger*. Stwórz wyzwalacz z ustawieniami zaprezentowanymi na rysunku 4.3.



Rysunek 4.3. Strona tworzenia wyzwalacza — wyzwalacz reagujący na wydanie nowej wersji Zabbiksa 7

3. Kliknij przycisk *Add*, aby zakończyć proces tworzenia wyzwalacza.

Możesz nie ujrzeć w najbliższym czasie komunikatu wysyłanego przez ten wyzwalacz we frontendzie, ale omówię mechanizm jego działania w punkcie „Wyjaśnienie".

Trzeci wyzwalacz — korzystanie z wielu pozycji w ramach jednego wyzwalacza

Do tej pory widzieliśmy wyzwalacze korzystające z jednej pozycji, ale możemy również stosować wiele pozycji w ramach jednego wyzwalacza. Zbudujmy nowy wyzwalacz, wykorzystujący wiele pozycji w jednym wyrażeniu:

1. Otwórz stronę *Data collection/Hosts* i wybierz hosta `lar-book-agent_passive`. Następnie przejdź do sekcji *Triggers* i kliknij przycisk *Create trigger*.
2. Utwórz wyzwalacz z ustawieniami takimi, jak na rysunku 4.4.

Rysunek 4.4. Strona tworzenia wyzwalacza — wyzwalacz reagujący na pakiety przychodzące oraz wychodzące

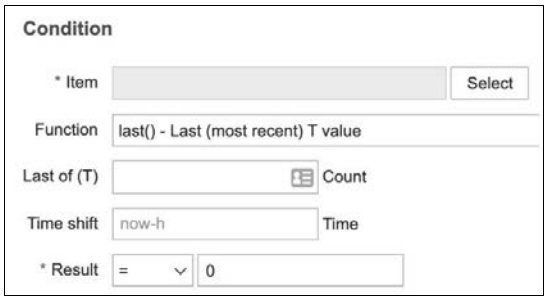
3. Pamiętaj, że u Ciebie może występować inna nazwa interfejsu. W moim przypadku nosi on nazwę `ens192`, dlatego musisz podmienić jego wystąpienia swoją własną nazwą. Za pomocą poniższego polecenia możesz sprawdzić nazwę interfejsu na Twoim hoście:

```
ip addr
```

4. Kliknij przycisk *Add*, aby zakończyć tworzenie wyzwalacza.

Wskazówka

Na stronie tworzenia wyzwalacza kliknij przycisk *Add* obok pola *Expression*, aby dodać warunek i ułatwić sobie tworzenie wyrażenia. Możemy na przykład użyć przycisku *Select*, aby wybrać pozycję z listy. Również bardzo przydatne jest krótkie objaśnienie każdej dostępnej funkcji wyzwalacza, wyświetlane podczas korzystania z listy rozwijanej *Function* (rysunek 4.5).



Rysunek 4.5. Strona tworzenia wyzwalacza

Tak wygląda proces tworzenia wyzwalacza korzystającego z dwóch pozycji.

Wyjaśnienie

Jeśli chcemy stworzyć sprawną platformę monitorowania, musimy dobrze zrozumieć mechanizm tworzenia oraz działania wyzwalaczy. Jest szczególnie ważne, aby prawidłowo konfigurować wyzwalacze, a także je testować po utworzeniu. Wyzwalacze są bardzo istotne, ponieważ odgrywają kluczową rolę w informowaniu nas o stanie monitorowanych elementów. Skonfiguruj wyzwalacze zbyt „luźno", a pewne informacje będą Ci umykać. Skonfiguruj je zbyt „ściśle", a będziesz zasypywany nadmiarem danych.

Istotność wyzwalacza jest definiowana przez poziom zagrożenia, co widać na rysunku 4.6.

Rysunek 4.6. Poziomy wyzwalacza w Zabbiksie

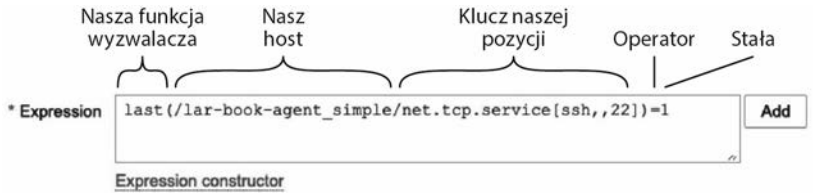
Poziomy te odgrywają ważną rolę w prawidłowym oznaczeniu istotności alertów. Możemy je także filtrować w różnych obszarach frontendu, a nawet w takich elementach, jak akcje.

Teraz wyjaśnimy sobie, dlaczego zbudowaliśmy wyzwalacze w ten, a nie inny sposób.

Pierwszy wyzwalacz — monitorowanie usługi SSH

Jest to bardzo prosty, ale skuteczny wyzwalacz. Gdy zostaje zwrócona wartość `1`, oznaczająca `UP` (działająca usługa), lub `0` dla `DOWN` (niedziałająca usługa), możemy z łatwością tworzyć tego typu wyzwalacze — nie tylko do monitorowania portów logicznych, lecz także każdego elementu powodującego prostą zmianę wartości, na przykład z 0 na 1 lub odwrotnie.

Przyjrzyjmy się teraz naszemu wyrażeniu (rysunek 4.7).



Rysunek 4.7. Wyrażenie wyzwalacza — port 22 (SSH)

Jak widać, wyrażenie składa się z pięciu członów:

- **Funkcja wyzwalacza**. Funkcja wyzwalacza stanowi tę składową wyrażenia, w której definiujemy nasze oczekiwania dotyczące wartości, na przykład tego, czy interesuje nas wyłącznie ostatnia wartość, czy średnia z określonego czasu.
- **Host**. Tutaj definiujemy hosta, wobec którego ma być używany wyzwalacz. W większości przypadków jest to host (lub szablon), na którym pracujemy.
- **Klucz pozycji**. Tutaj określamy klucz pozycji, za pomocą którego będziemy odczytywać wartość (wartości) z hosta i przekazywać je do funkcji wyzwalacza.
- **Operator**. Operator określa mechanizm działania funkcji w porównaniu na przykład do stałej albo innego wyrażenia. Dostępne są przeróżne operatory, co zostało zaprezentowane w tabeli 4.1.
- **Stała**. Jest to rzeczywiście stała (często wartość), za pomocą której wyzwalacz określa, czy powinien znajdować się w stanie `OK` lub `PROBLEM`. Możemy tu wstawiać również makra.

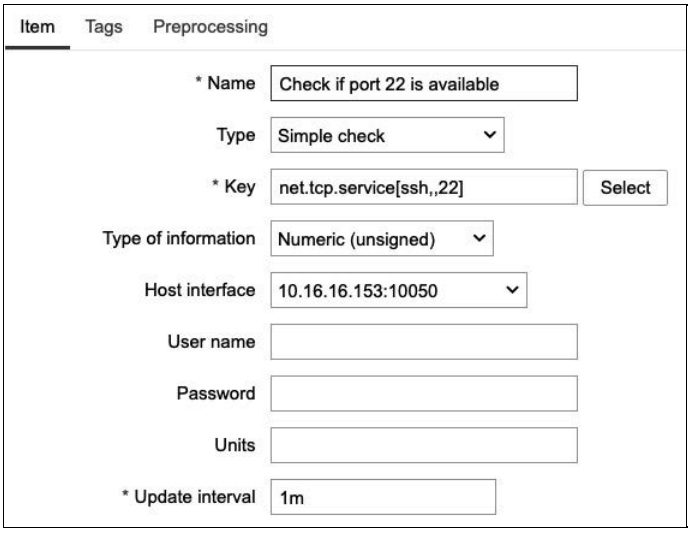
W naszym pierwszym wyzwalaczu zdefiniowaliśmy naszego hosta oraz pozycję określającą status usługi SSH. Przekazujemy przez funkcję wyzwalacza komunikat, że chcemy, aby wyzwalała go ostatnia wartość równa 0.

W przypadku tej pozycji oznaczałoby to uruchomienie wyzwalacza przed upływem minuty, ponieważ zdefiniowaliśmy ją tak jak na rysunku 4.8.

Jeśli spojrzymy na pole *Update interval* na stronie *Item*, to okaże się, że podczas tworzenia tego wyzwalacza oczekujemy wartości równej `0` oraz że wykrycie niedziałającego portu `22` zajmie maksymalnie minutę z powodu interwału równego `1m`.

Tabela 4.1. Przykładowe operatory funkcji wyzwalacza

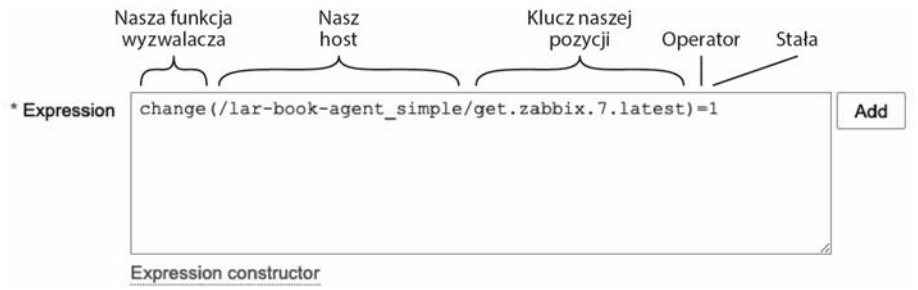
`=`	Równe
`<>`	Nie jest równe
`>`	Większe od
`<`	Mniejsze od
`>=`	Większe lub równe
`<=`	Mniejsze lub równe
`+`	Suma
`-`	Różnica
`/`	Iloraz
`*`	Iloczyn
`and`	Logiczna operacja AND. Używana na przykład do wyboru jednego i drugiego wyrażenia.
`or`	Logiczna operacja OR. Używana na przykład do wyboru jednego lub drugiego wyrażenia.
`not`	Logiczna operacja NOT. Używana na przykład do negowania wyrażenia.



Rysunek 4.8. Strona konfigurowania pozycji — dostępność portu 22

Drugi wyzwalacz — wyzwalanie w momencie wypuszczenia nowej wersji Zabbiksa

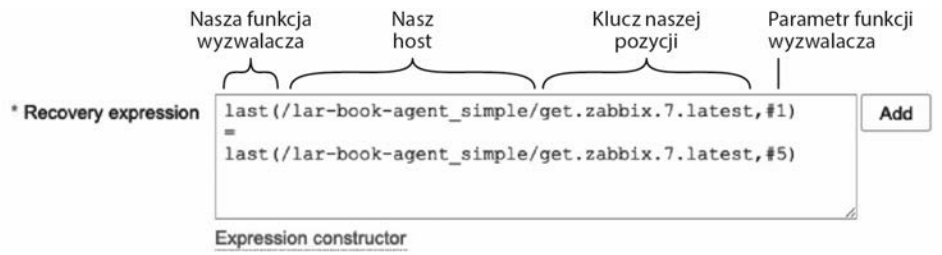
W drugim wyzwalaczu skorzystaliśmy z innego mechanizmu. Przygotowaliśmy wyrażenie nie tylko służące do uruchomienia wyzwalacza w przypadku spełnienia właściwych warunków, lecz także określające warunki potrzebne do przywrócenia stanu domyślnego wyzwalacza. W polu *Problem expression* definiujemy funkcję wyzwalacza, w której ostatnia wartość jest porównywana z wcześniejszą. Wykorzystaliśmy w tym celu funkcję `change` (rysunek 4.9).



Rysunek 4.9. Wyrażenie wyzwalacza — sprawdzanie HTTPS

Zatem nasz wyzwalacz będzie aktywowany w przypadku zmiany najnowszej wersji Zabbiksa. Moglibyśmy sprawić, żeby wyzwalacz spełnił swoje zadanie, gdy tylko wartość bieżąca i poprzednia będą takie same, ale chcę, żeby jeszcze przez chwilę znajdował się w stanie `PROBLEM`.

Dlatego zdefiniowałem także **wyrażenie przywracania** (ang. *recovery expression*). Za jego pomocą sprawiam, że problem zostanie rozwiązany dopiero wtedy, gdy najnowsza oraz piąta w kolejności wartość będą takie same. Widać to na rysunku 4.10.



Rysunek 4.10. Inne wyrażenie wyzwalacza — sprawdzanie HTTPS za pomocą innej wartości

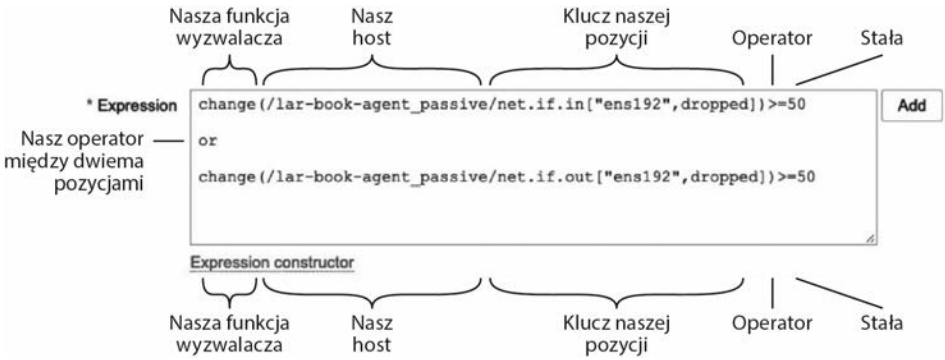
Wyrażenia przywracania bardzo się przydają, gdy chcesz mieć większą kontrolę nad warunkami powrotu wyzwalacza do stanu `OK`.

Wskazówka

Możemy używać wyrażeń przywracania do przedłużania stanu PROBLEM wyzwalacza ponad warunki zdefiniowane w polu *Problem expression*. W ten sposób będziemy wiedzieć, że wciąż znajdujemy się blisko stanu PROBLEM. Za pomocą wyrażenia przywracania określamy, że chcemy, aby wyzwalacz wrócił do stanu OK dopiero po osiągnięciu kolejnej wartości progowej wyznaczonej w tym wyrażeniu. Mechanizm ten działa poprzez oszacowanie obydwu wyrażeń, przy czym wyrażenie problemu musi uzyskać wartość FALSE, a wyrażenie przywracania — wartość TRUE.

Trzeci wyzwalacz — korzystanie z wielu pozycji w ramach jednego wyzwalacza

Trzeci wyzwalacz może wydawać się bardziej skomplikowany z powodu użycia dwóch pozycji, ale w zasadzie mamy do czynienia z taką samą strukturą (rysunek 4.11).



Rysunek 4.11. Wyrażenie wyzwalacza wykorzystujące wiele pozycji

Struktura naszego wyrażenia pozostaje identyczna, z takimi samymi pozycjami funkcji, hosta, klucza pozycji i wartości. Jednak podczas korzystania z wielu pozycji możemy wstawiać między nimi operator or. W ten sposób możemy tak zdefiniować wyzwalacz, żeby w celu przejścia w stan PROBLEM został spełniony warunek którejś z pozycji. W tym przypadku wyzwalacz zostanie aktywowany, gdy wartość którejś z pozycji przekroczy wyznaczony próg.

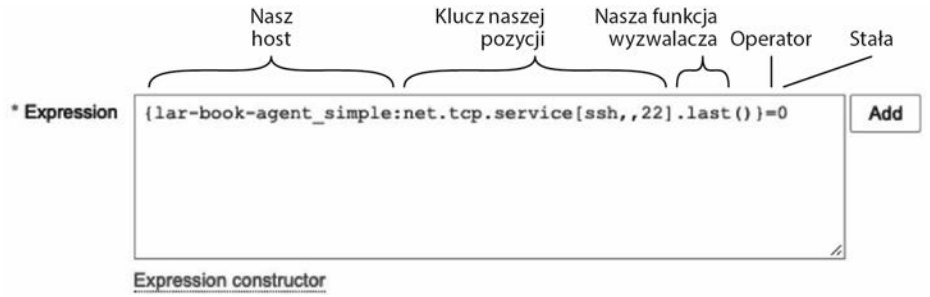
Ważna informacja

W wyrażeniu na rysunku 4.11 widzimy puste wiersze między poszczególnymi wyrażeniami pozycji. Zupełnie to nie przeszkadza w działaniu kodu, a wręcz poprawia czytelność. Korzystaj z tego mądrze podczas tworzenia wyzwalaczy.

Porównanie starej i nowej składni wyrażeń wyzwalaczy

Ten podpunkt może być szczególnie interesujący dla osób, które ostatni raz miały kontakt z Zabbiksem sprzed wersji 5.4. Jak już wspomnieliśmy na początku rozdziału, wprowadzono wielką aktualizację wyrażeń w Zabbiksie. Wyrażenia wyzwalacza działają teraz w odmienny sposób, taki sam jak w pozycjach obliczanych i innych miejscach, dzięki czemu ujednolicono składnię w całej platformie Zabbix.

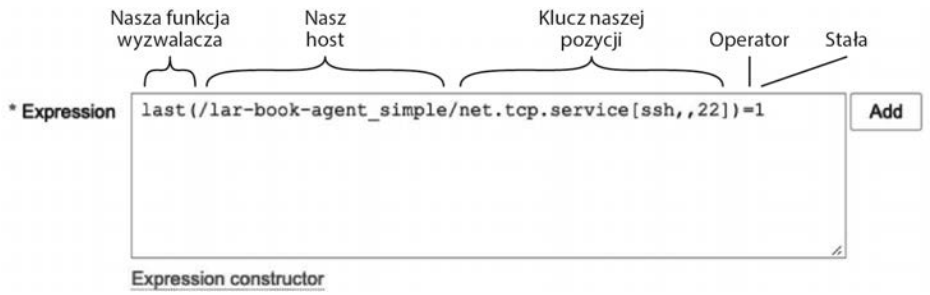
Spójrzmy na starą składnię, używaną do wersji Zabbix 5.2 (rysunek 4.12).



Rysunek 4.12. Stara składnia wyrażeń wyzwalacza

W starej składni wyrażenie zawsze rozpoczynało się od *nawiasu klamrowego*, po którym podawaliśmy nazwę hosta lub szablonu. Między nazwą hosta/szablonu a kluczem pozycji umieszczaliśmy *średnik*. Koniec klucza pozycji oznaczaliśmy *kropką*, ale w samym kluczu również mogą występować kropki. Po niej wstawialiśmy funkcję wyzwalacza i zamykaliśmy całość drugim *nawiasem klamrowym*. Na koniec umieszczaliśmy jeszcze operator i stałą porównywaną do wyrażenia.

Łatwo sobie wyobrazić, że takie wyrażenie szybko mogło stać się zagmatwane, zwłaszcza jeśli używaliśmy kropek w kluczach pozycji. Z kolei nowa składnia została zaprezentowana na rysunku 4.13.



Rysunek 4.13. Nowa składnia wyrażeń wyzwalacza

Nowa składnia wyzwalacza rozpoczyna się od razu funkcją wyzwalacza; od razu wiadomo, jakie jest zadanie tego wyrażenia. Po niej następują *nawias okrągły* i *ukośnik prawy*, po czym podajemy nazwę hosta lub szablonu. Kolejny *ukośnik prawy* służy do rozdzielenia nazwy hosta/szablonu od klucza pozycji. Zamykamy całość *nawiasem okrągłym*, po którym wstawiamy jeszcze operator i porównywaną wartość.

Dzięki umieszczeniu funkcji wyzwalacza na początku łatwo można stwierdzić, jakie zadanie jest wykonywane przez wyrażenie. Umieszczenie nazwy hosta/szablonu między nawiasami, a następnie oddzielenie ich ukośnikami prawymi od klucza pozycji poprawiają czytelność i spójność w trakcie tworzenia wyrażeń. Nie wprowadzamy już również mylących kropek. Zmiana składni jest bardzo dobrze przemyślana, chociaż przyzwyczajenie się do niej wymaga trochę czasu.

To dzięki takim szczegółom całe oprogramowanie daje wrażenie większego profesjonalizmu i okazuje się dobrze przemyślane. Zabbix naprawdę zyskał na tych zmianach.

To nie wszystko

Możemy nie tylko wybierać jedną z pozycji w wyrażeniu wyzwalacza, lecz także zastosować operację logiczną `and`. Dzięki temu możemy sprawić, że wyzwalacz przejdzie do stanu `PROBLEM` jedynie wtedy, gdy wiele pozycji uzyska określoną wartość progową. Wyzwalacze zapewniają pod tym względem olbrzymią swobodę i pozwalają nam szczegółowo określać własne kryteria. Nic tu nie jest predefiniowane — możemy umieszczać w naszych wyrażeniach dowolną liczbę instrukcji `and`, `not` czy `or`, a także funkcji wyzwalacza. Dostosowuj wyzwalacze do swoich potrzeb, a nagle okaże się, że będziesz mógł spać o wiele spokojniej, ponieważ będą Cię one informowały o wszelkich problemach.

Zobacz także

Więcej informacji o wyrażeniach wyzwalacza znajdziesz w dokumentacji Zabbiksa. Zostały tam opisane funkcje pozwalające nam tworzyć idealnie dopasowane wyzwalacze. Szczegóły znajdziesz pod adresem *https://www.zabbix.com/documentation/current/en/manual/config/triggers/expression*.

Konfigurowanie zaawansowanych wyzwalaczy

Wyzwalacze w Zabbiksie stają się coraz bardziej zaawansowane i czasami może być ciężko nadążyć za zmianami. Osoby przesiadające się z wersji Zabbix 5.2 lub starszej do wersji 7 natrafią nie tylko na nową składnię, lecz również na zupełnie nowy zestaw funkcji.

Sprawdźmy, jak się konfiguruje nieco bardziej zaawansowane wyzwalacze w Zabbiksie 7.

Przygotowanie

W tej recepturze będzie nam potrzebny przygotowany serwer Zabbix, a także jeden host monitorowany przez agenta Zabbix z dołączonym szablonem. Wykorzystamy pozycje dostępne na tym hoście do tworzenia wyzwalaczy. Użyjemy tu hosta `lar-book-agent_passive` z poprzedniego rozdziału.

Jeżeli nie masz wspomnianego hosta, wystarczy przygotować nowego hosta z domyślnym szablonem pasywnego monitorowania Linuksa o nazwie *Linux by Zabbix agent*.

Poruszymy tu także bardziej zaawansowane zagadnienia, którymi zajmiemy się w dalszej części książki. Jeśli nie wiesz na przykład, jak korzystać z **wykrywania niskopoziomowego** (ang. *Low-Level Discovery* — LLD), warto najpierw zajrzeć do rozdziału 7., „Automatyczne tworzenie za pomocą narzędzia wykrywania".

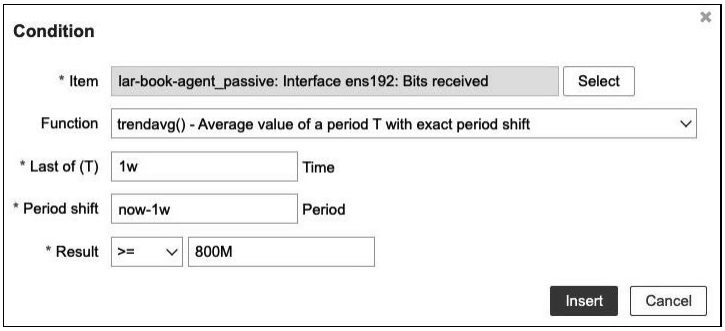
Wykonanie

Przyjrzyjmy się trzem *bardziej zaawansowanym* wyzwalaczom w porównaniu do omówionych w poprzedniej recepturze. Będą to: `trendavg`, analizujący tendencje w danych, `timeleft`, przewidujący przyszłe wartości, a także wyzwalacz realizujący **przesunięcie czasowe** (ang. *time shifting*) w celu porównywania danych z danymi przeszłymi.

Pierwszy zaawansowany wyzwalacz — funkcja trendavg

Najpierw zajmiemy się jednym z nowych elementów wyzwalaczy, mianowicie funkcją obliczania średniej tendencji:

1. Zacznijmy od utworzenia nowego wyzwalacza we frontendzie. Otwórz stronę *Data collection*/*Hosts* i wybierz hosta `lar-book-agent_passive`.
2. Przejdź do zakładki *Triggers* i kliknij przycisk *Create trigger* w prawym górnym rogu ekranu.
3. Kliknij biały przycisk *Add* obok pola *Expression*. Wypełnij dane zgodnie z rysunkiem 4.14.



Rysunek 4.14. Konstruktor wyrażenia wyzwalacza trendavg

4. Kliknij przycisk *Insert* i podaj nazwę. Powinno to wyglądać tak, jak na rysunku 4.15.

Rysunek 4.15. Wypełniony formularz wyzwalacza trendavg

5. Kliknij teraz niebieski przycisk *Add* u dołu ekranu, aby dokończyć tworzenie tego wyzwalacza.

To tyle w kwestii tworzenia tego wyzwalacza. W punkcie „Wyjaśnienie" znajdziesz więcej informacji na jego temat.

Drugi zaawansowany wyzwalacz — funkcja timeleft

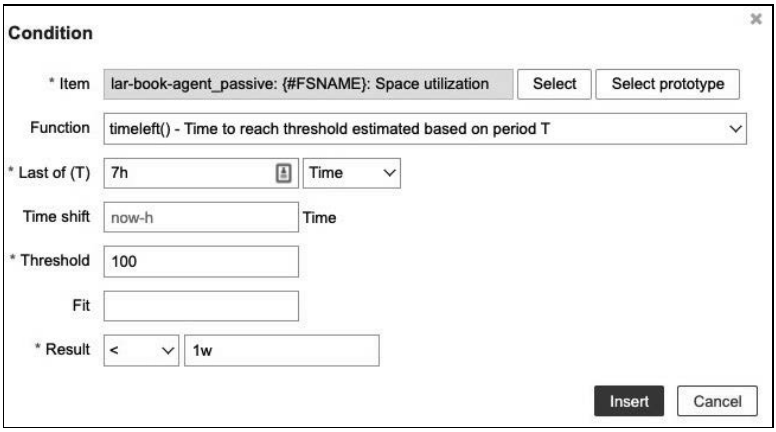
Kolejną interesującą nas funkcją jest `timeleft`, która bardzo się przydaje w takich zadaniach, jak wykorzystanie powierzchni handlowej. Popatrzmy:

1. Utworzymy nowy wyzwalacz we frontendzie Zabbix. Otwórz stronę *Data collection/Hosts* i wybierz hosta `lar-book-agent_passive`.
2. Otwórz okno *Discovery rules* i kliknij przycisk *Trigger prototype* obok *Mounted filesystem discovery*.

Ważna informacja

W tym przypadku tworzymy prototyp wyzwalacza bezpośrednio na hoście za pomocą dostępnej reguły wykrywania szablonu. Jeżeli chcesz wprowadzić tego typu wyzwalacz do każdego hosta za pomocą szablonu, musisz utworzyć go na poziomie szablonu. Reguły wykrywania zostały wyjaśnione dokładniej w rozdziale 7., „Automatyczne tworzenie za pomocą narzędzia wykrywającego".

3. Teraz kliknij *Create trigger prototype*.
4. Kliknij biały przycisk *Add* obok pola *Expression*. Podaj dane zgodne z rysunkiem 4.16.

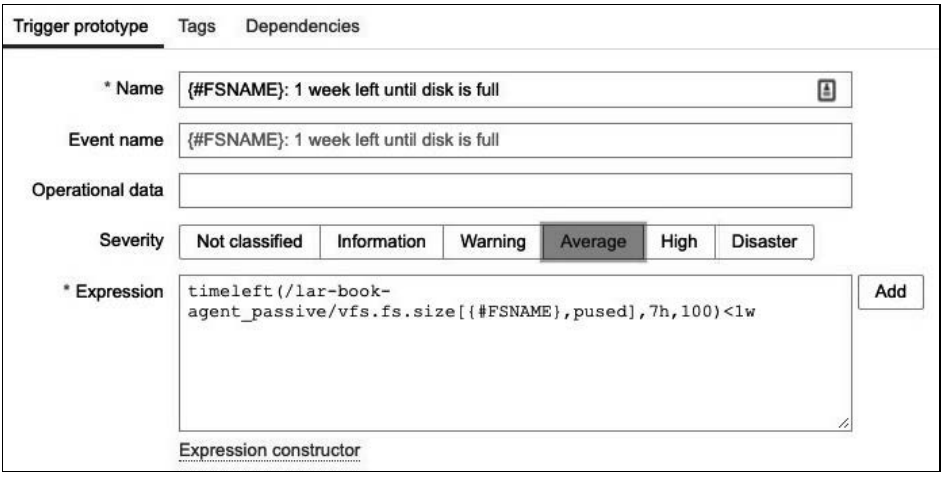


Rysunek 4.16. Konstruktor wyrażenia wyzwalacza timeleft

Ważna informacja

Nie jest zalecane stosowanie krótkich przedziałów w wyzwalaczach predykcyjnych do przewidywania długich przedziałów czasu. Wybierz odpowiedni zestaw danych czasowych w odniesieniu do przewidywanego okresu.

5. Kliknij niebieski przycisk *Insert*. Ukończony wyzwalacz będzie wyglądał tak, jak na rysunku 4.17.



Rysunek 4.17. Wypełniony formularz wyzwalacza timeleft

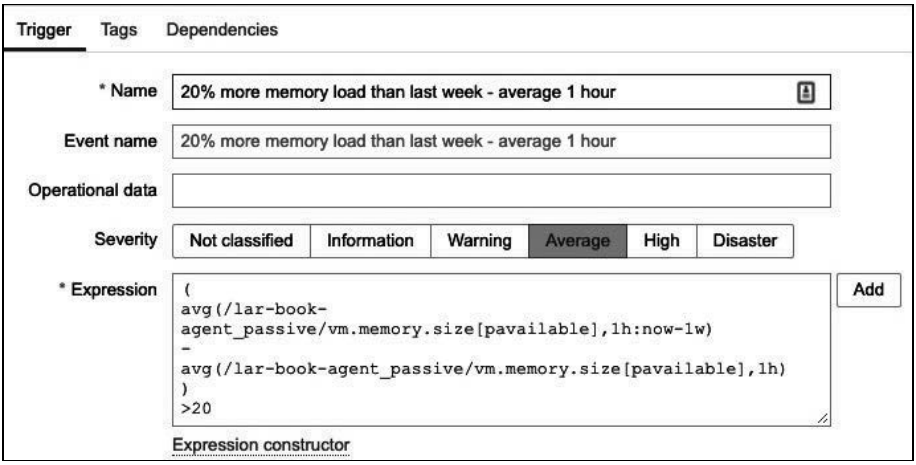
6. Kliknij niebieski przycisk *Add* u dołu ekranu, aby zakończyć konfigurowanie wyzwalacza.

Właśnie zdefiniowaliśmy nowy wyzwalacz zawierający funkcję `timeleft`. Będzie nas powiadamiał, kiedy pozostanie nam tydzień do momentu, gdy skończy się miejsce na dyskach twardych. Więcej informacji na jego temat znajdziesz w punkcie „Wyjaśnienie".

Trzeci zaawansowany wyzwalacz — przesunięcie czasowe za pomocą funkcji matematycznych

Na koniec zajmiemy się przesunięciem czasowym, a w tym przypadku zrobimy to w połączeniu z funkcją matematyczną. Będzie to nieco bardziej skomplikowany przykład, dlatego zachowaj cierpliwość:

1. Otwórz stronę *Data collection*/*Hosts* i wybierz naszego hosta `lar-book-agent_passive`.
2. Przejdź do okna *Triggers* i kliknij niebieski przycisk *Create trigger*.
3. Dodaj wyzwalacz widoczny na rysunku 4.18.



Rysunek 4.18. Wypełniony formularz wyzwalacza średniego przesunięcia czasowego

Jest to wyzwalacz bardzo złożony w konfiguracji, dlatego omówimy go dokładniej w punkcie „Wyjaśnienie".

Wyjaśnienie

Zaawansowane wyzwalacze bywają bardzo skomplikowane. Zaprezentowane tu przez nas to zaledwie czubek góry lodowej. Nie załamuj się ich złożonością, ponieważ znajdziesz w internecie mnóstwo dokumentacji pomagającej w ich skonfigurowaniu: *https://www.zabbix.com/documentation/current/en/manual/config/triggers*.

Opisanie wszystkich przypadków użycia w tej książce graniczyłoby z cudem, dlatego tak dobraliśmy omawiane tu wyzwalacze, żeby pokazać Ci dostępne możliwości. Korzystaj ze zdobytej tu wiedzy we własnych zastosowaniach, pamiętaj jednak, żeby każdy problem rozpatrywać indywidualnie.

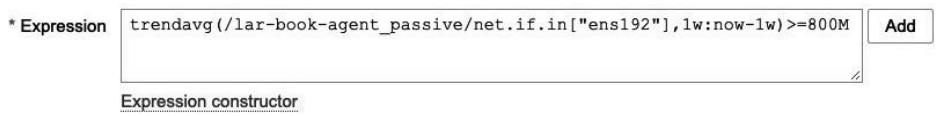
Pierwszy zaawansowany wyzwalacz — funkcja trendavg

Zacznijmy od wyrażenia zajmującego się średnią tendencji. Jest to jedna z niewielu funkcji wyzwalacza, w której wykorzystywane są dane tendencji, a nie historyczne. Przypomnijmy sobie pokrótce, czym są dane tendencji i dane historyczne w Zabbiksie. Dane historyczne to dokładne wartości, które pozycja otrzymuje z monitorowanego hosta. Z kolei dane tendencji to wartości średnia, minimalna i maksymalna z ponad jednej godziny danych obliczonych na podstawie danych historycznych oraz liczby użytych wartości.

Spójrzmy, jakie mamy do dyspozycji funkcje wyzwalacza korzystające z danych tendencji:

- `trendavg`. Uzyskuje średnią wartość z danych tendencji dla określonego przedziału czasu.
- `trendmax`. Uzyskuje wartość maksymalną z danych tendencji dla określonego przedziału czasu.
- `trendmin`. Uzyskuje wartość minimalną z danych tendencji dla określonego przedziału czasu.
- `trendcount`. Uzyskuje liczbę wartości z danych tendencji dla określonego przedziału czasu.
- `trendsum`. Uzyskuje sumę danych tendencji dla określonego przedziału czasu.

Jak już wspomniałem, funkcje te wykorzystują dane tendencji. Wartości te są przechowywane w specjalnym obszarze pamięci Zabbiksa, skąd mogą być wykorzystywane przez wyzwalacz. My użyliśmy funkcji `trendavg`. Przypomnijmy sobie, jak to wyglądało (rysunek 4.19).



Rysunek 4.19. Wyrażenie wyzwalacza trendavg

Najpierw podajemy funkcję `trendavg`, a następnie *nazwę hosta/szablonu* i *klucz pozycji*, podobnie jak w poprzedniej recepturze. Nowością jest człon `1w:now-1w`. Określamy tu przedział czasu; w tym przypadku chcemy skorzystać z wartości sprzed tygodnia.

Zgodnie z tym wyrażeniem, jeżeli średnia wartość tendencji sprzed tygodnia przekroczy 800 Mbps, to wyzwalacz przejdzie w stan `PROBLEM`.

Drugi zaawansowany wyzwalacz — funkcja timeleft

Jest to kolejna wielce interesująca funkcja wyzwalacza. Za pomocą funkcji `timeleft` możemy tworzyć wyzwalacze, które są aktywowane tylko wtedy, gdy przewidywane będzie przekroczenie jakiejś wartości progowej w przyszłości. Jest to tak zwany wyzwalacz predykcyjny, ponieważ stara się przewidywać przyszłość na podstawie wcześniejszych danych.

Przyjrzyjmy się ponownie temu wyrażeniu wyzwalacza (rysunek 4.20).

Rysunek 4.20. Wyrażenie wyzwalacza timeleft

Jak widać, początek wyrażenia jest standardowy: *funkcja wyzwalacza*, *nazwa hosta/szablonu* oraz *klucz pozycji*. W tym przypadku powiązaliśmy te elementy z przedziałem czasu, na podstawie którego wyzwalacz predykcyjny ma uzyskać przewidywania. Wartość `7h` oznacza dane historyczne z ostatnich siedmiu godzin. Dodaliśmy do tego wartość progową `100`, dzięki czemu wyzwalacz zostanie aktywowany, jeśli pojawi się oczekiwane maksymalne wykorzystanie pamięci dyskowej. Do pełni szczęścia potrzebujemy jeszcze jednego elementu, czyli spodziewanego rezultatu, w tym przypadku `<1w`.

Podsumowując: to wyrażenie wyzwalacza analizuje *siedem godzin* danych historycznych i jeżeli będzie przewidywać, że wykorzystanie pamięci osiągnie *100%* w ciągu *najbliższego tygodnia*, to wprowadzi wyzwalacz w stan `PROBLEM` i wyśle do Ciebie ostrzeżenie.

> **Wskazówka**
>
> Możesz połączyć funkcję wyzwalacza `timeleft` z innymi funkcjami, aby ograniczyć liczbę wysyłanych powiadomień. Na przykład w przypadku miejsca na dysku możemy przewidywać, że zacznie go brakować w ciągu tygodnia, ale być możesz chcesz się o tym dowiedzieć, gdy zostanie mniej niż 50 GB wolnej przestrzeni. Dodaj wyrażenie zaprezentowane na rysunku 4.21 i gotowe.

```
last(/Linux filesystems by Zabbix agent/vfs.fs.size[{#FSNAME},pused])>90%
and
timeleft(/Linux filesystems by Zabbix agent/vfs.fs.size[{#FSNAME},pused],1h,100)<1d)
```

Rysunek 4.21. Wyrażenie funkcji wyzwalacza timeleft

Trzeci zaawansowany wyzwalacz — przesunięcie czasowe za pomocą funkcji matematycznych

Podczas zajęć dotyczących użytkowania Zabbiksa zawsze muszę poświęcić więcej czasu z uczniami na omówienie wyrażeń wyzwalacza przesunięcia czasowego. Jest to logiczne, ponieważ mamy tu do czynienia z jednym z bardziej skomplikowanych wyrażeń, dodatkowo połączonym z funkcjami matematycznymi.

Przypomnijmy sobie to wyrażenie i rozbijmy je na czynniki pierwsze (rysunek 4.22).

```
1 (
2 avg(/lar-book-agent_passive/vm.memory.size[pavailable],1h:now-1w)
3 -
4 avg(/lar-book-agent_passive/vm.memory.size[pavailable],1h)
5 )
6 >20
```

Rysunek 4.22. Wyrażenie wyzwalacza przesunięcia czasowego

Dla naszej wygody ponumerowałem poszczególne wiersze. Teraz możemy zająć się omówieniem każdego z nich po kolei:

1. Jest to nawias otwierający operację matematyczną, przy czym obydwie pozycje rozdziela operator.
2. Nasza pierwsza pozycja wykorzystująca funkcję przesunięcia czasowego. Pozycja ta będzie pozyskiwać wartość procentową dostępnej pamięci sprzed tygodnia, począwszy dokładnie od tego momentu. Jeżeli bieżący dzień to poniedziałek, 24 listopada, i jest godzina 14:00, to pozycja ta pozyska średnią z jednej godziny z poniedziałku, 17 listopada, między godziną 13:00 a 14:00.
3. Nasz operator matematyczny oznaczający różnicę. Będziemy odejmować wynik pierwszego wyrażenia od wyniku drugiego wyrażenia.
4. To nasza druga pozycja. Nie korzystamy w niej z przesunięcia czasowego. Będzie ona uzyskiwała średnią wartości z zeszłej godziny.
5. Nawias zamykający naszą operację matematyczną.
6. Na końcu pozostają operator i stała. W tym przypadku wyzwalacz będzie aktywowany tylko wtedy, gdy wynik operacji matematycznej będzie większy niż 20.

Skoro już wiemy, co się dzieje w poszczególnych wierszach, zastanówmy się nad mechanizmem działania całego wyrażenia. Wstawimy wartości własnoręcznie i sprawdzimy, czy dane wyrażenie uzyskuje wartość `TRUE` czy `FALSE`. Wartość `TRUE` oznacza, że pojawił się problem, a `FALSE` mówi nam, że wszystko jest w porządku. Zatem wygląda to tak:

```
(Zeszły tydzień – Bieżący tydzień) = Wynik
Jeśli Wynik jest większy niż 20, to wtedy wyrażenie ma wartość TRUE.
Wartość tego wyrażenia: TRUE/FALSE
```

Jeżeli w zeszłym tygodniu było 80% dostępnej pamięci, a w tym jest jej tylko 50%, to dzieje się rzecz następująca:

```
(80 – 50) = 30
Jeśli 30 jest większe niż 20, to wtedy wyrażenie ma wartość TRUE.
Wartość tego wyrażenia: TRUE
```

Spróbujmy jeszcze raz, ale tym razem dla 80% pamięci dostępnej w zeszłym tygodniu i 70% w tym tygodniu:

```
(80 – 70) = 10
Jeśli 10 jest większe niż 20, to wtedy wyrażenie ma wartość TRUE.
Wartość tego wyrażenia: FALSE
```

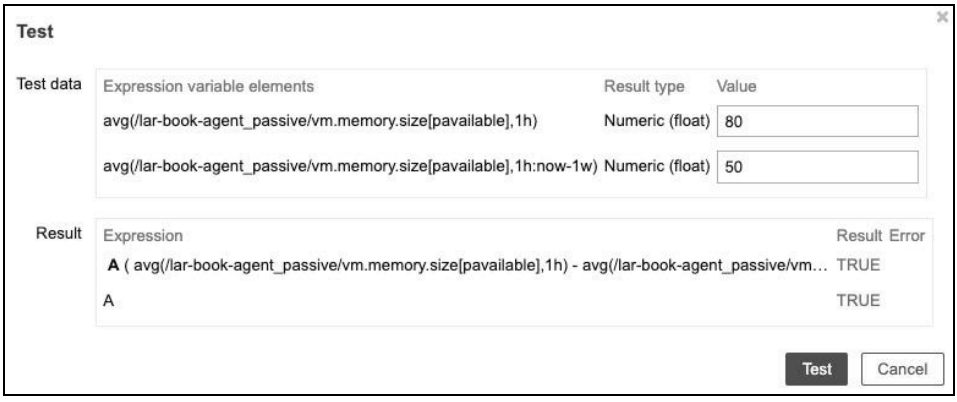
W taki właśnie sposób należy konfigurować wyrażenia przesunięcia czasowego. Możesz skorzystać z notatnika lub dowolnego innego narzędzia, rozpisać wyrażenie w prosty sposób, tak jak powyżej, i przeprowadzić obliczenia.

To nie wszystko

Wyrażenia wyzwalaczy można także testować w samym Zabbiksie. Jeżeli otworzysz stronę *Data Collection/Hosts/Triggers* i wybierzesz któryś z zaawansowanych wyzwalaczy, to będziemy mogli przeprowadzić mały test. Na przykład dla wyzwalacza przesunięcia czasowego możemy otworzyć *Expression constructor* (rysunek 4.23).

Rysunek 4.23. Konstruktor wyrażeń — wyzwalacz przesunięcia czasowego

Możemy tutaj kliknąć przycisk *Test* i wypełnić nasze wartości. Skorzystajmy z naszych przykładowych wartości 80% i 50% (rysunek 4.24).



Rysunek 4.24. Konstruktor wyrażeń — wyzwalacz przesunięcia czasowego, test

Jak widać, dowiemy się z tego testu, czy nasze wyrażenie będzie miało wartość TRUE czy FALSE dla dowolnych wybranych przez nas wartości. Krótko mówiąc, jeśli chcesz się upewnić, że Twoje obliczenia na papierze będą zgodne z tymi w Zabbiksie, przetestuj je za pomocą narzędzia *Expression constructor*.

Konfigurowanie alertów

Wysyłanie alertów stanowi bardzo ważny element architektury Zabbiksa. Gdy konfigurujemy alerty, chcemy, aby odbiorca dowiedział się, co się dzieje. Ważne jest także, żeby nie zalewać ludzi alertami; chcemy korzystać z nich skutecznie.

Dlatego w tej recepturze poznamy podstawy konfigurowania alertów, żeby od początku wiedzieć, jak należy robić to we właściwy sposób.

Przygotowanie

Przy tej recepturze będą nam potrzebne tylko dwie rzeczy. Alerty będziemy tworzyć z poziomu serwera Zabbix, a ponadto będziemy korzystać z wyzwalaczy, na przykład utworzonych w poprzedniej recepturze. Posłużą nam one do inicjowania procesu wysyłania alertów, żebyśmy mogli zobaczyć, w jaki sposób serwer Zabbix przekazuje te informacje.

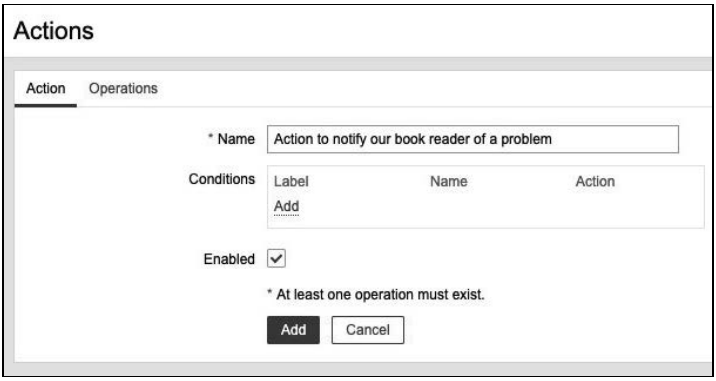
Wykonanie

1. Zacznijmy od skonfigurowania akcji we frontendzie Zabbix. W tym celu otwieramy stronę *Alerts/Actions/Trigger actions*, gdzie powita nas ekran widoczny na rysunku 4.25.

Rysunek 4.25. Strona Trigger actions z jedną akcją wyzwalacza

Widzimy tu już jedną akcję skonfigurowaną do powiadamiania administratorów Zabbiksa o problemach. W wersji Zabbix 7 wiele elementów, takich jak akcje czy media, jest odgórnie zdefiniowanych. W większości przypadków wystarczy je uaktywnić i uzupełnić informacjami.

2. Zdefiniujemy własną akcję, która będzie powiadamiać członków grupy *Zabbix administrators* o nowych wyzwalaczach. Kliknij niebieski przycisk *Create action*, aby zostało wyświetlone okno ukazane na rysunku 4.26.



Rysunek 4.26. Strona tworzenia akcji — powiadamianie czytelnika tej książki

3. Zaznacz tu opcję *Enabled*, aby uaktywnić tę akcję. Nadaj jej czytelną nazwę, dzięki czemu będziesz mógł szybko wybierać swoje akcje.
4. Przejdź teraz do zakładki *Operations* (rysunek 4.27).

Rysunek 4.27. Strona tworzenia akcji — powiadamianie czytelnika tej książki, zakładka Operations

5. Domyślnie zakładka *Operations* jest pusta, dlatego musimy tu przygotować pewne operacje. Stworzymy tu trzy typy operacji — zacznijmy od operacji *Operations* (rysunek 4.28). Kliknij przycisk *Add*.

Rysunek 4.28. Strona ze szczegółami operacji Operations — powiadamianie czytelnika tej książki

6. Mamy tu możliwość dodawania użytkowników i/lub grup użytkowników, którzy mają dostawać alerty. Jeżeli postępowałeś zgodnie z instrukcjami zawartymi w rozdziale 1., „Instalacja Zabbiksa i przygotowanie frontendu", to możesz wybrać grupę *Networking*. W przeciwnym wypadku wystarczy również grupa *Zabbix administrators*.
7. Po kliknięciu niebieskiego przycisku *Add* u dołu formularza wrócimy na stronę *Actions*.
8. Teraz utworzymy następną operację, o nazwie *Recovery operations*. Tworzymy operację widoczną na rysunku 4.29.

Rysunek 4.29. Szczegóły operacji Recovery operations — powiadamianie czytelnika tej książki

9. Operacja ta będzie powiadamiała wszystkich użytkowników, którzy otrzymali alert pochodzący z pierwszej operacji. Wszyscy, którzy otrzymali powiadomienie o stanie `PROBLEM`, zostaną również powiadomieni o przywróceniu normalnego działania. Kliknij przycisk *Add* i kontynuujmy.

   Jeżeli jesteś taki jak ja i zawsze chcesz być na bieżąco, możesz utworzyć powiadomienie z aktualizacją. W ten sposób dowiemy się na przykład, że ktoś już pracuje nad rozwiązaniem problemu. Zazwyczaj wybrałbym do takich alertów inny kanał informacyjny, na przykład wiadomości SMS do alertów o wysokim priorytecie, a kanał Slack czy Teams do pozostałych powiadomień.

10. Kliknij przycisk *Add* na stronie operacji *Update operations* i dodaj konfigurację widoczną na rysunku 4.30.

Rysunek 4.30. Szczegóły operacji Update operations — powiadamianie czytelnika tej książki

11. Zrobimy to samo co w operacji *Recovery operations* i będziemy powiadamiać wszystkich wyznaczonych użytkowników o wszelkich postępach. Po kliknięciu przycisku *Add* kliknij kolejny niebieski przycisk *Add* na ekranie *Actions*, aby zakończyć tworzenie działania.

12. Teraz chcemy utworzyć typ mediów, za pomocą którego użytkownicy będą otrzymywać powiadomienia. Otwórz stronę *Alerts/Media types*, gdzie zobaczysz ekran widoczny na rysunku 4.31.

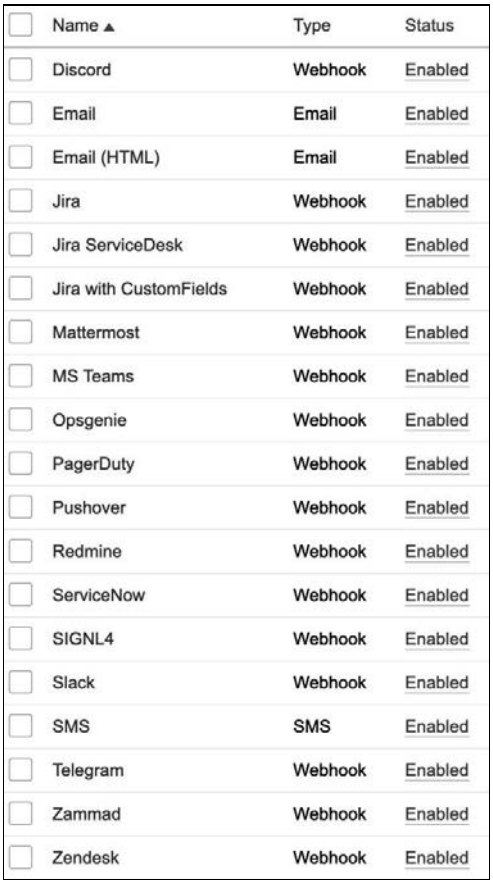
    Jak widać, w wersji Zabbix 7 istnieje całkiem sporo gotowych typów mediów. Widzimy tu Slack, Opsgenie, a nawet Telegram. Zacznijmy jednak od kanału posiadanego niemal przez wszystkich: e-maila.

13. Kliknij typ mediów *Email* i skonfiguruj go do swoich potrzeb (rysunek 4.32).

14. Użyłem tu swoich ustawień z pakietu Office 365, ale powinien zadziałać każdy **protokół SMTP** (ang. *Simple Mail Transfer Protocol*). Wpisz swoje ustawienia SMTP, co powinno sprawić, że będziesz otrzymywać powiadomienia.

15. Sprawdź także następną zakładkę, *Message templates*. Na przykład szablon wiadomości dla generowanego zdarzenia *Problem* został zaprezentowany na rysunku 4.33.

    Konfigurujemy go w taki sposób, aby otrzymywać wiadomości powiadamiające o sytuacji. Można go całkowicie modyfikować, aby przekazywał taki zakres informacji, jaki uznamy za słuszny.



Name ▲	Type	Status
Discord	Webhook	Enabled
Email	Email	Enabled
Email (HTML)	Email	Enabled
Jira	Webhook	Enabled
Jira ServiceDesk	Webhook	Enabled
Jira with CustomFields	Webhook	Enabled
Mattermost	Webhook	Enabled
MS Teams	Webhook	Enabled
Opsgenie	Webhook	Enabled
PagerDuty	Webhook	Enabled
Pushover	Webhook	Enabled
Redmine	Webhook	Enabled
ServiceNow	Webhook	Enabled
SIGNL4	Webhook	Enabled
Slack	Webhook	Enabled
SMS	SMS	Enabled
Telegram	Webhook	Enabled
Zammad	Webhook	Enabled
Zendesk	Webhook	Enabled

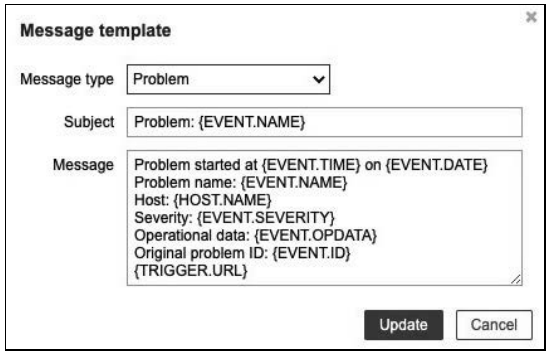
Rysunek 4.31. Strona Media types z gotowymi typami mediów

16. Na razie zostawmy ustawienia domyślne. Przejdźmy jeszcze na stronę *Users/Users* i edytujmy użytkownika w grupie *Zabbix administrators* lub *Networking*. W tym przykładzie użyję konta *Admin* (rysunek 4.34).
17. Przejdź do zakładki *Media* i kliknij przycisk *Add*. Chcemy dodać informacje widoczne na rysunku 4.35, żeby otrzymywać na adres e-mail powiadomienia o wszystkich poziomach wyzwalacza.
18. Kliknij teraz niebieski przycisk *Add*, by zakończyć tworzenie mediów dla tego użytkownika.

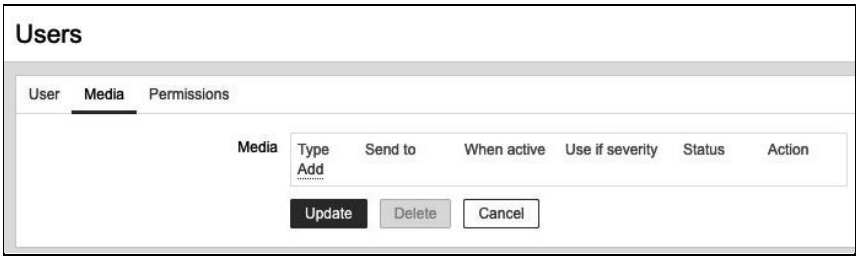
Wyjaśnienie

Tak właśnie konfigurujemy alerty. Będziesz otrzymywać powiadomienia na adres e-mail zgodnie ze schematem zaprezentowanym na rysunku 4.36.

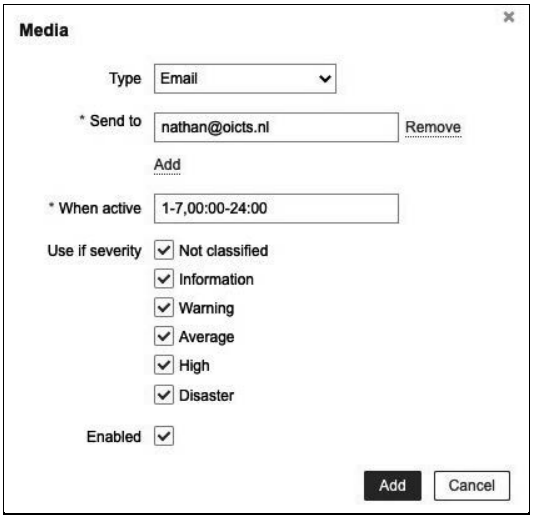
Rysunek 4.32. Strona konfigurowania kanału Email



Rysunek 4.33. Domyślny szablon wiadomości informującej o problemach



Rysunek 4.34. Zakładka Media użytkownika Admin

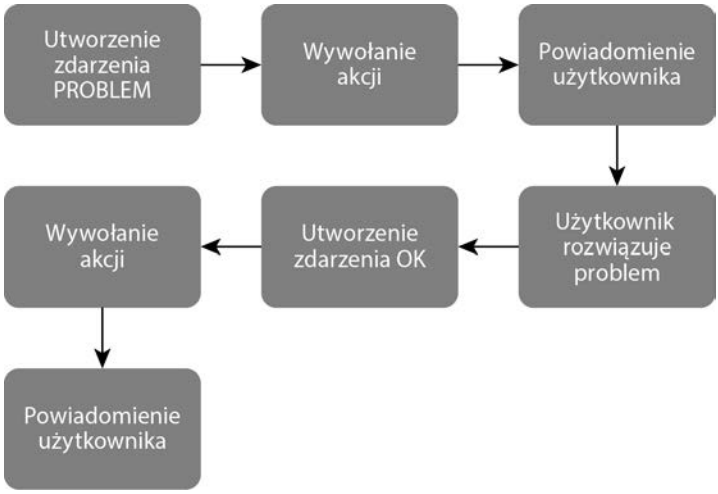


Rysunek 4.35. Strona konfigurowania mediów dla użytkownika Admin

Gdy coś się psuje, nasza konfiguracja wyzwalacza wywołuje w Zabbiksie stan `PROBLEM`. Zdarzenie to wywoła **akcję** (ang. *action*) polegającą na wykorzystaniu danego typu mediów i konfiguracji mediów użytkownika do powiadomienia naszego użytkownika. Użytkownik następnie rozwiązuje problem (na przykład uruchamia ponownie serwer), po czym zostaje wygenerowane zdarzenie `OK`. Zostaje ponownie wywołana akcja, która przesyła odpowiedni komunikat do użytkowników.

Wskazówka

Przed skonfigurowaniem takich alertów przygotuj najpierw dla siebie samego taki przepływ informacji jak na rysunku 4.36 i tylko wyznacz grupy oraz użytkowników, którzy mają otrzymywać powiadomienia. W ten sposób będziesz mógł się przekonać, czy system wysyłania alertów spełnia Twoje wymagania.



Rysunek 4.36. Schemat ukazujący przepływ informacji o problemie w Zabbiksie

To nie wszystko

Istnieje mnóstwo typów mediów i sposobów ich integracji, a my zaledwie ruszyliśmy wierzchołek góry lodowej, przeglądając listę gotowych rozwiązań. Przejrzyj listę gotowych sposobów integracji na stronie *https://www.zabbix.com/integrations* albo stwórz własne rozwiązanie za pomocą skryptów webhook Zabbiksa i innych rozszerzeń.

Utrzymywanie alertów w dobrym stanie

Ważne jest utrzymywanie alertów w dobrym stanie, aby użytkownicy nie otrzymywali ani za dużo, ani za mało powiadomień. W tym celu zmienimy wyzwalacz i typ mediów *Email* na coś odpowiedniejszego.

Przygotowanie

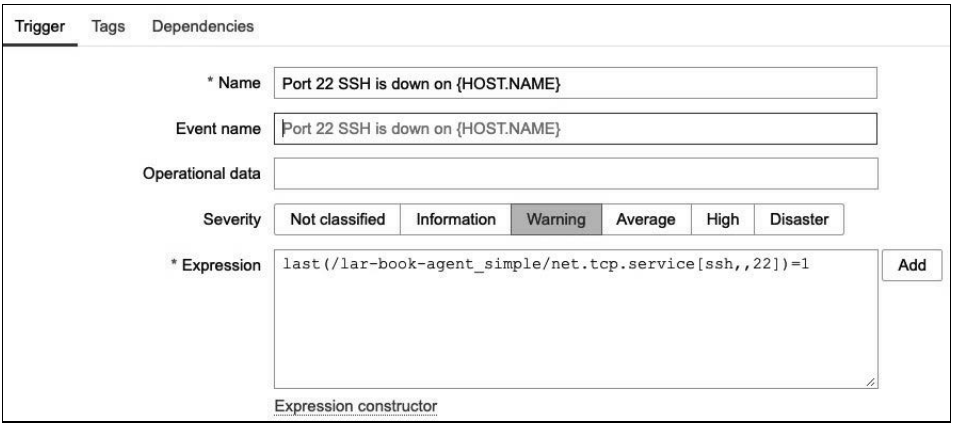
Będziemy używać pierwszego wyzwalacza z pierwszego przepisu, a także domyślnego typu mediów e-mail w Zabbiksie.

Poza tym, oczywiście, będzie nam potrzebny serwer Zabbix.

Wykonanie

Przygotujemy efektywne alerty w następujący sposób:

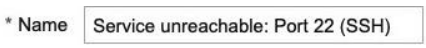
1. Zacznijmy od naszego pierwszego wyzwalacza, utworzonego w recepturze „Konfigurowanie wyzwalaczy". Przejdź do hosta `lar-agent-simple` na stronie *Data collection*/*Hosts* i kliknij jego zakładkę *Trigger*.
2. Niektórzy podają tutaj inną nazwę wyzwalacza, jak widać na rysunku 4.37.



Rysunek 4.37. Pierwszy wyzwalacz z pierwszej receptury w tym rozdziale

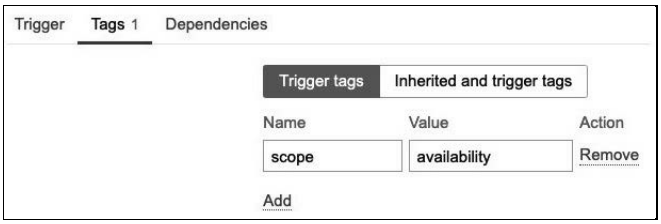
Nawet po użyciu w wyzwalaczu makra `{HOST.NAME}` wciąż pozostaje on prosty, dlatego na szczęście nie trzeba tu wiele zmieniać. Jeżeli użyłeś nazwy hosta w nazwie wyzwalacza, możesz ją zmienić, żeby komunikat był czytelniejszy.

3. Podaj krótką i opisową nazwę wyzwalacza, na przykład tak jak na rysunku 4.38.



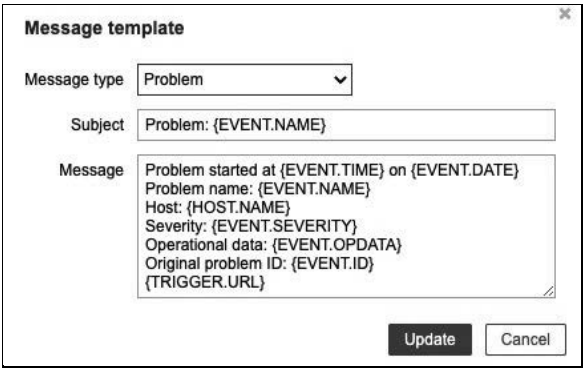
Rysunek 4.38. Nowa nazwa pierwszego wyzwalacza

4. Przejdź teraz do zakładki *Tags* i dodaj znacznik zapewniający uporządkowane nazewnictwo wyzwalaczy. Może wyglądać to tak, jak na rysunku 4.39.



Rysunek 4.39. Znaczniki pierwszego wyzwalacza

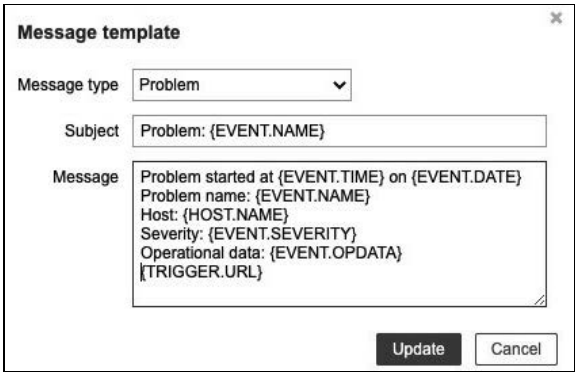
5. Innym doskonałym sposobem utrzymywania porządku jest zmiana komunikatu w mediach. Zmieńmy go na lepiej odpowiadający naszym potrzebom. Otwórz stronę *Alerts*/*Media types* i wybierz nasz typ mediów *Email*.
6. Wybierz *Message templates* i kliknij przycisk *Edit* obok naszego pierwszego problemu. Zostanie wyświetlone okno widoczne na rysunku 4.40.



Rysunek 4.40. Domyślna wiadomość typu mediów Email

Obecnie w Zabbiksie wysyłany jest domyślny komunikat. Jeżeli jednak chcemy to zmienić, możemy utworzyć własną wiadomość. Domyślny komunikat został zaprezentowany na rysunku 4.40.

7. Możemy zmienić wiadomość w typie mediów. Na przykład jeśli nie chcesz wysyłać oryginalnego identyfikatora problemu lub potrzebujesz bardziej zmodyfikowanego powiadomienia, wystarczy usunąć odpowiedni wiersz, co ukazano na rysunku 4.41.

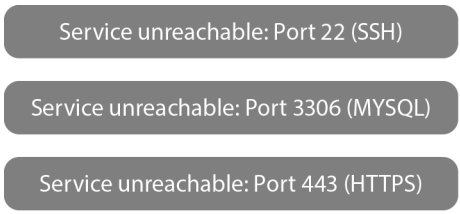


Rysunek 4.41. Niestandardowa wiadomość typu mediów Email

Wyjaśnienie

W tej recepturze wykonaliśmy dwie czynności: zmieniliśmy nazwę wyzwalacza i dodaliśmy do niego znacznik.

Przemyślane, uporządkowane nazewnictwo wyzwalaczy jest istotne dla utrzymania dobrej organizacji środowiska Zabbix. Zmieniliśmy nazwę `Port 22 SSH down on {HOST.NAME}` po to, aby zapewnić naszej architekturze standaryzację, dzięki czemu możemy tworzyć użyteczne struktury, na przykład za pomocą przyszłych wyzwalaczy (rysunek 4.42).



Rysunek 4.42. Schemat struktury utworzonej z wyzwalaczy

Nasze wyzwalacze są jasno opisane i od razu dowiadujemy się z nich, który host, port i usługa nie działają.

Ponadto dodaliśmy znacznik do niedziałającej usługi, dzięki czemu usługa ta będzie teraz wyraźnie wyświetlana oraz zostaniemy poinformowani o sytuacji (rysunek 4.43).

Host	Problem	Duration	Ack	Actions	Tags
lar-book-agent_simple	Service unreachable: Port 22 (SSH)	1s	No		component: tcp services scope: availability

Rysunek 4.43. Uporządkowany wyzwalacz informujący o niedziałającej usłudze

W wersji Zabbix 6 wprowadzono nowe zasady tworzenia znaczników. Spełniliśmy wymogi tych nowych standardów, najpierw tworząc używaną w wyzwalaczu pozycję ze znacznikiem składowej (*component*), a przed chwilą dodając również znacznik zakresu (*scope*). Jeżeli spojrzymy na rysunek 4.43, od razu zauważymy, że problem dotyka dostępności usługi SSH. W zakresie *scope* zazwyczaj umieszczamy jedną z pięciu wartości: `availability` (dostępność), `performance` (wydajność), `notification` (powiadomienie), `security` (zabezpieczenia) i `capacity` (pojemność).

Więcej informacji na temat tych nowych zasad tworzenia znaczników znajdziesz w dokumentacji:

> *https://blog.zabbix.com/tags-in-zabbix-6-0-lts-usage-subfiltersand-guidelines/19565/*

Usunęliśmy również makro `{HOST.NAME}`, jeśli korzystaliśmy z niego wcześniej. Wiemy z kolumny *Host*, który host uaktywnił wyzwalacz, dlatego nie potrzebujemy tego makra.

Nazwy wyzwalaczy powinny być krótkie oraz czytelne, a makra nazwy hosta powinniśmy wstawiać w sekcji *Media* lub używać jedynie pola dostępnego we frontendzie.

Zmieniliśmy tu również akcję. Modyfikowanie komunikatów wysyłanych przez różne typy mediów stanowi doskonały sposób utrzymywania porządku w naszych kanałach nadawczych. Czasami chcemy otrzymywać mniej lub więcej informacji przez określone kanały, a jednym ze sposobów osiągnięcia tego jest zmiana struktury wysyłanych komunikatów.

Możemy również tworzyć niestandardowe wiadomości z poziomu *Action*, co powoduje zmianę wszystkich komunikatów wysyłanych wybranymi kanałami.

To nie wszystko

W tej recepturze próbuję wyjaśnić, że chociaż łatwo jest skonfigurować środowisko Zabbix, to o wiele trudniej przygotować dobrą architekturę monitorowania w ramach Zabbiksa — dotyczy to właściwie każdej platformy monitorowania — bez odpowiedniej dozy planowania. Ostrożnie zaplanuj strukturę wyzwalaczy, zanim zaczniesz tworzyć strukturę systemu monitorowania w Zabbiksie.

Inżynier pracujący na ustrukturyzowanych danych, który poświęci czas na stworzenie przemyślanej struktury monitorowania, zaoszczędzi w przyszłości mnóstwo czasu, ponieważ pozna istotę problemu przed innymi.

Dostosowywanie alertów

Wysyłanie alertów to bardzo przydatna funkcja, pozwalająca zachować porządek w strukturach platformy monitorującej, zwłaszcza w połączeniu z dotychczas opisanymi w tej książce sztuczkami. Czasami jednak potrzebujemy od alertów możliwości wykraczających poza gotowe rozwiązania.

W tej recepturze odrobinę dostosujemy alerty do naszych potrzeb.

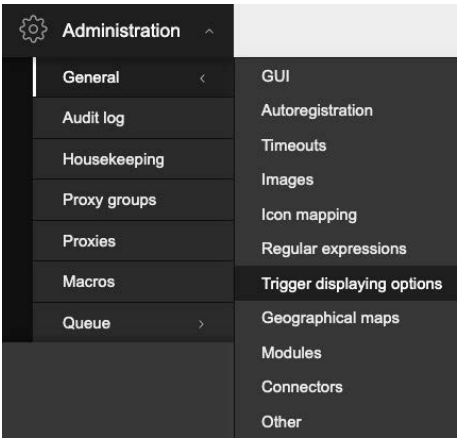
Przygotowanie

W tej recepturze będzie nam potrzebny jedynie nasz serwer Zabbix.

Wykonanie

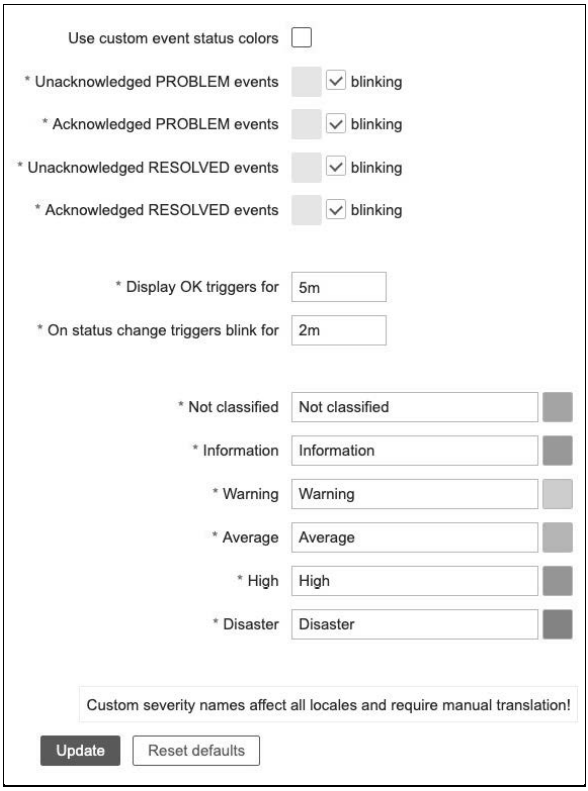
Aby dostosowywać alerty, należy wykonać następujące czynności:

1. Stwórzmy niestandardowe poziomy wyzwalacza, odzwierciedlające potrzeby naszej organizacji. Otwórz stronę *Administration*/*General* i wybierz *Trigger displaying options* w bocznym menu (rysunek 4.44).



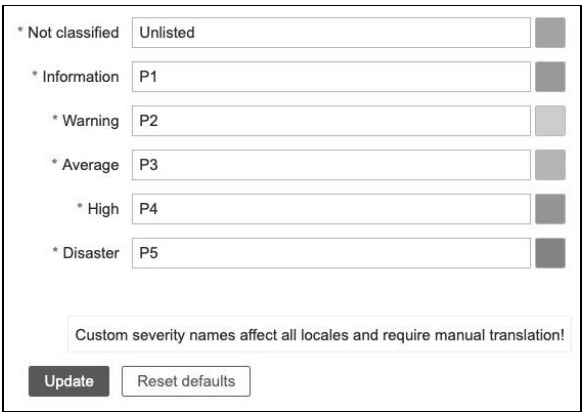
Rysunek 4.44. Opcje Administration/General w bocznym menu

Po kliknięciu przejdziemy na nową stronę. Znajdziemy tam domyślne poziomy wyzwalacza, co widać na rysunku 4.45.



Rysunek 4.45. Strona z domyślnymi poziomami wyzwalacza

2. Możemy tu zmodyfikować te domyślne poziomy wyzwalacza, na przykład tak jak na rysunku 4.46.



Rysunek 4.46. Niestandardowe ustawienia poziomów wyzwalacza

3. Nie zapomnij wcisnąć niebieskiego przycisku *Update* u dołu strony, aby zapisać zmiany.

Wyjaśnienie

Nie wszystkie firmy lubią stosować takie pojęcia jak *High* czy *Disaster* do opisu poziomów zagrożenia; wolą używać określeń typu *P1* czy *P2*. Dzięki możliwości modyfikowania poziomów wyzwalacza możemy bardziej dostosować Zabbiksa do własnych potrzeb i na przykład wprowadzić nomenklaturę stosowaną w innych narzędziach.

Zmiana poziomów wyzwalacza nie jest w żadnym wypadku obowiązkowa, ale stanowi dobry sposób na szybsze przyzwyczajenie się do Zabbiksa, jeśli dotychczas korzystałeś z innych narzędzi.

Skorowidz

P

R

S

T

U

W

Z

Monitoring czy chaos? Prosty wybór! Wybierz Zabbix – pełną kontrolę!

Systemy monitorujące są często bagatelizowane, tymczasem to właśnie dzięki nim możesz uniknąć problemów. Spośród dostępnych na rynku opcji Zabbix jest atrakcyjnym rozwiązaniem: pozwala na monitoring dowolnie dużej infrastruktury składającej się z wielu różnych komponentów i na daleko idącą automatyzację zadań, zapewnia też rozbudowane wizualizacje. W dodatku można z niego korzystać zupełnie za darmo.

Ten praktyczny przewodnik zawiera receptury uwzględniające nowe funkcje środowiska Zabbix. Znajdziesz tu informacje potrzebne do konfiguracji Zabbiksa z wbudowanym trybem wysokiej dostępności. Dowiesz się też, jak korzystać z aprowizacji użytkowników LDAP JIT, implementować funkcję niskopoziomowego wykrywania hostów i tworzyć zaawansowane wyzwalacze. Każda receptura została opracowana z myślą o różnych typach monitorowania i korzystania z serwerów proxy Zabbix. Ponadto nauczysz się modyfikować serwer i bazę danych Zabbix, a także zarządzać nimi za pomocą interfejsu API. Poznasz również rozwiązania problemów, na które możesz natrafić podczas pracy z Zabbiksem.

W książce:

- implementacja infrastruktury w trybie wysokiej dostępności
- szablony monitorowania
- skalowanie środowiska Zabbix
- niestandardowe integracje i interfejsy, a także zaawansowane wyzwalacze i alerty
- zaawansowane operacje zarządzania bazami danych Zabbix
- monitorowanie usług chmurowych, takich jak Amazon Web Services, Azure czy Docker

Nathan Liefting jest konsultantem i trenerem. Ma duże doświadczenie w zarządzaniu sieciami. Obecnie pracuje w Opensource ICT Solutions BV. Tworzy profesjonalne składniki Zabbiksa dla największych firm z całego świata.

Brian van Baekel w 2017 roku uzyskał certyfikat Zabbix Certified Trainer. Rok później założył dwie firmy koncentrujące się głównie na tworzeniu środowisk Zabbix. Jest też właścicielem pięknego kota o imieniu… Zabbix.

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-2476-5
9 788328 924765
Cena: 139,00 zł
packt