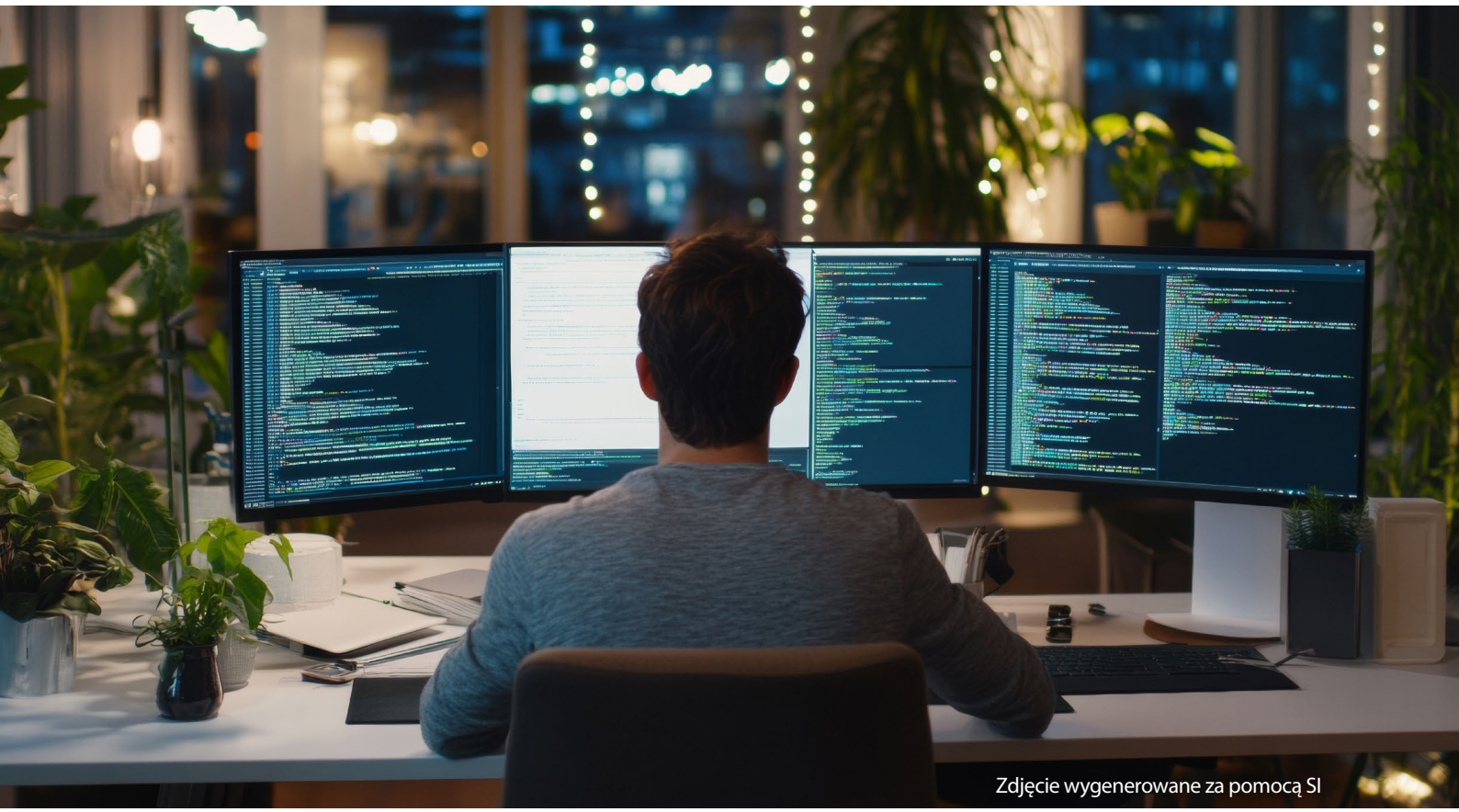
Subiektywny poradnik administratora

cz. II



Zdjęcie wygenerowane za pomocą SI

Na łamach „Domeny" opisywałem już OpenVPN i Zabbixa, z których korzystam w codziennej pracy. Postanowiłem pójść za ciosem i przygotować cykl artykułów o narzędziach open source, które mogą być przydatne w każdej firmie. Chcę w ten sposób przekonać do wybrania otwartoźródłowej ścieżki administrowania infrastrukturą (serwerami VPS, kontenerami, usługami).

Adam Jurkiewicz

administrator sieci i serwerów Linux od ponad 25 lat. Programista Pythona, zwariowany nauczyciel młodzieży. Zdecydowany zwolennik oprogramowania open source i systemów Linux od 1993 r., których od ponad dwóch dekad używa w codziennej pracy. Członek zarządu Sekcji Informatyki Szkolnej przy PTI oraz członek oddziału mazowieckiego PTI. Dostępny w sieciach społecznościowych:
https://www.linkedin.com/in/adam-jurkiewicz-python-linux/
https://linux.social/@adam_jurkiewicz
https://jurkiewicz.chat

Po lekturze poprzedniej części poradnika mamy przygotowane środowisko: Proxmox (narzędzie do wirtualizacji systemów) oraz podstawowy kontener z systemem Linux. Teraz zajmiemy się kolejnymi narzędziami: Ansible oraz Observium.

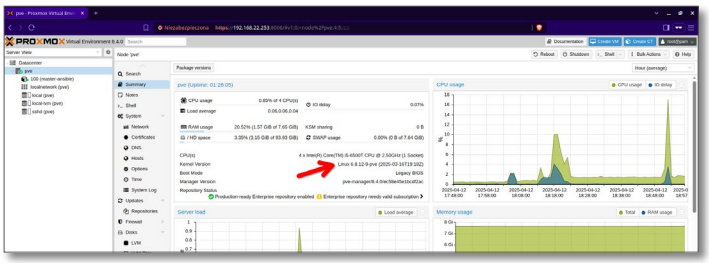
Ansible to narzędzie do automatyzacji IT, które umożliwia łatwe zarządzanie infrastrukturą i konfiguracją systemów. Działa bez potrzeby instalowania agentów na urządzeniach docelowych, wykorzystując protokoły SSH lub PowerShell do wykonywania zadań. Dzięki niemu będziemy mogli automatyzować pewne zadania, a więc umożliwiać zarządzanie różnymi serwerami z jednego miejsca.

Observium z kolei to wszechstronne narzędzie open source do monitorowania sieci, zarządzania konfiguracją urządzeń sieciowych czy automatyzacją zadań sieciowych.

Rozpocznę od przygotowania kontenera, który będzie zawierał Ansible. Sam proces przygotowania kontenera możemy pominąć, opisywałem go poprzednio, jednak przy tej okazji zwracam uwagę na pewien ważny szczegół – wersję jądra systemu operacyjnego.

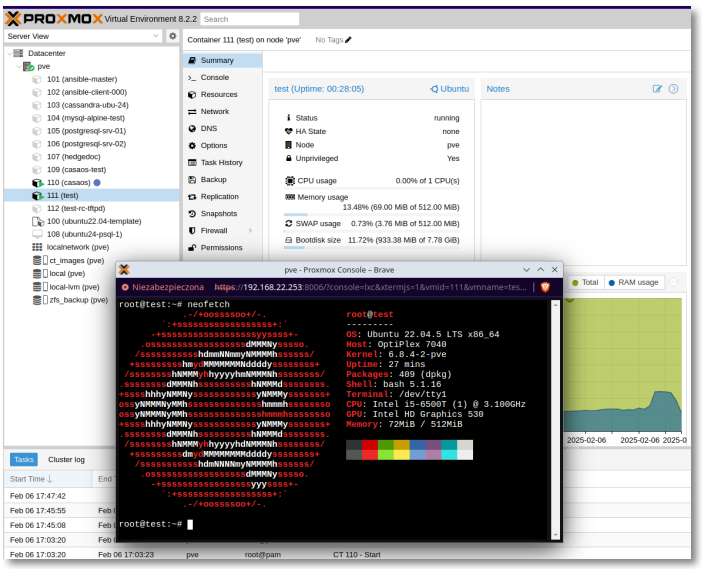
> *Jądro systemu operacyjnego (ang. kernel) – podstawowa część systemu operacyjnego, która jest odpowiedzialna za wszystkie jego zadania.*
>
> *Patrz: https://pl.wikipedia.org/wiki/J%C4%85dro_systemu_operacyjnego*

Spójrzmy teraz na system Proxmox po instalacji:



Zawiera jądro w wersji **6.8.12-9-pve** – jądro z serii 6.8.

Poniżej kolejny zrzut ekranowy – tym razem jest to konsola kontenera, który będzie głównym kontrolerem Ansible, w nim uruchomione narzędzie Fastfetch, które wyświetla wiele informacji o systemie opracyjnym. Spójrzmy na wersję jądra (kernel) **6.8.12-9-pve** – jest identyczna z wersją Proxmoksa. Po prostu kontener używa systemu operacyjnego hosta, a więc Proxmoksa. Ważne, byśmy o tym pamiętali – takie rozwiązanie nie zawsze będzie idealne, lecz w 99% przypadków wystarczające.



Instalacja

W systemie Ubuntu (a taki wybrałem do stworzenia kontenera) instalacja sprowadza się do dwóch poleceń:

```
apt-add-repository --yes --update ppa:ansible/ansible

apt install ansible
```

Tylko tyle? Naprawdę? TAK – choć trudno w to uwierzyć. Oczywiście w tzw. międzyczasie system zapyta się, czy instalować brakujące zależności, więc polecam odpowiedzieć Y (yes) lub T (tak) – w zależności od skonfigurowanych locales.

Locales – ustawienia regionalne, które określają, jak programy i interfejs użytkownika dostosowują się do konkretnego języka i regionu użytkownika.

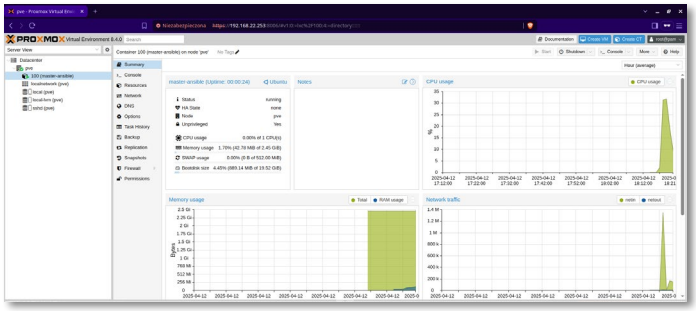
By sprawdzić, czy oprogramowanie działa, wydajemy polecenie:

```
ansible –version
```

W ten sposób mamy zainstalowane środowisko wykonawcze. Nie będę tu szczegółowo opisywał sposobów konfiguracji innych maszyn, którymi będziemy zarządzać – zapew-

ne za jakiś czas i tak takie instrukcje straciłyby aktualność, wolę chętne osoby odesłać do strony dokumentacji projektu – znajdziecie tam zawsze najbardziej aktualne informacje.

https://docs.ansible.com/



Wstępna konfiguracja za nami. Możemy sprawdzić, czy jesteśmy w stanie z poziomu Ansible kontrolera zapukać (moduł ping w Ansible wykonuje podobną operację jak ping systemowy) do observium-host. Wynik działania komendy nie wymaga chyba komentarza.

```
root@master-ansible:~# ansible -m ping all
observium | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
root@master-ansible:~#
```

Zarządzanie jednym hostem z drugiego nie jest specjalnym wyczynem, ani też w żaden sposób nie przyspiesza pracy. Wyobraźmy sobie jednak inną sytuację – mamy pod nadzorem nie jeden, nie kilka, lecz kilkadziesiąt lub więcej serwerów i w każdym z nich musimy wydać jedno polecenie, np. instalujące aplikację htop do celów monitorowania systemu. Zatem czeka nas wykonanie kilkudziesięciu ssh do każdego kolejnego serwera, a potem kilkudziesięciu poleceń instalujących. Ile to zajmie czasu (godzinę czy kilka), zależy od naszych palców, prędkości sieci itp. Dzięki Ansible możemy zrobić to za pomocą jednego polecenia, resztę wykona za nas oprogramowanie. My w tym czasie możemy napić się kawy.

```
GNU nano 7.2                    system_init.yml
# Ansible playbook to install packages... comment
---
- hosts: observium

  tasks:

    - name: Update apt repo and cache on all Debian/Ubuntu boxes
      apt: update_cache=yes force_apt_get=yes cache_valid_time=3600

    - name: Update all packages to their latest version
      ansible.builtin.apt:
        name: "*"
        state: latest

    - name: Install UFW and other packages
      apt:
        name: htop, curl, jq, ufw, bashtop, mc, neofetch, neovim, alien, ncftp
        state: present

                        [ Read 20 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo
```

Przygotujemy plik, w którym opiszemy zadania, które mają się wykonać (zauważmy przy okazji ciekawy edytor Nano – czy ktoś pamięta jeszcze poczciwego ChiWritera?):

Właściwie zawartość pliku mówi sama za siebie, lecz dla porządku wypiszę te elementy:

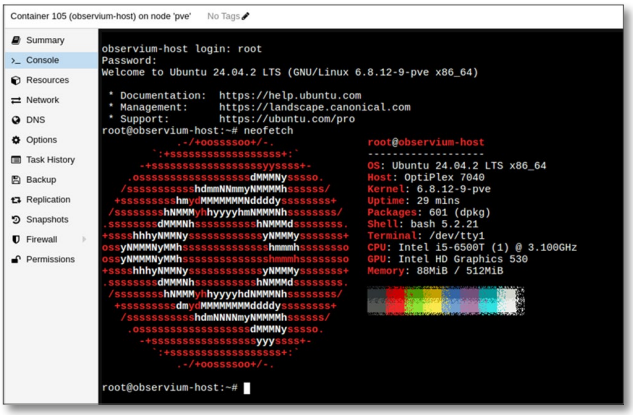
- Aktualizacja cache pakietów,
- Aktualizacja wszystkich zainstalowanych pakietów do najnowszych wersji,
- Instalacja kilku różnych pakietów.

Samo wykonanie takich poleceń wygląda następująco:

```
ansible-playbook nazwa_pliku_z_poleceniami.yml
```

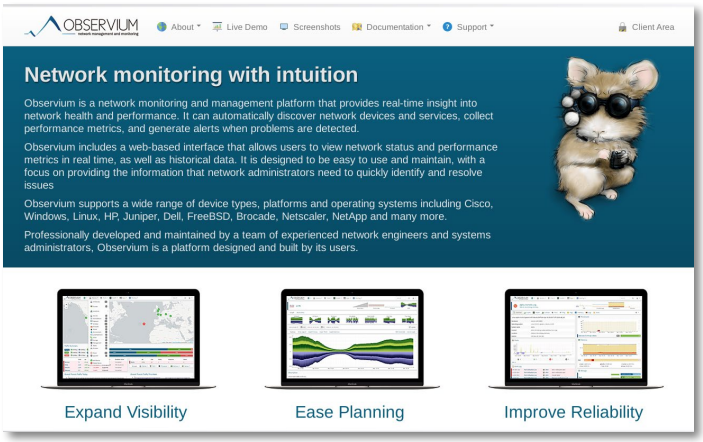
```
root@master-ansible:~/ansible_playbooks# ansible-playbook system_init.yml

PLAY [observium] ***************************************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [observium]

TASK [Update apt repo and cache on all Debian/Ubuntu boxes] ********************************************
changed: [observium]

TASK [Update all packages to their latest version] *****************************************************
changed: [observium]

TASK [Install UFW and other packages] ******************************************************************
changed: [observium]

PLAY RECAP *********************************************************************************************
observium                  : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0
   ignored=0

root@master-ansible:~/ansible_playbooks#
```

Uruchamiamy to na kontrolerze, który łączy się do observium-host i wykonuje tam polecenia. W efekcie na naszym systemie observium-host mamy zainstalowane różne ważne aplikacje – możemy to zobaczyć, logując się w konsoli Proxmoksa:



Pora na Observium

Mamy przygotowany system, teraz przejdźmy do Observium. Oto wycinek strony domowej:



Sam system to platforma do monitorowania i zarządzania siecią, oferująca wgląd w czasie rzeczywistym w stan i wydajność infrastruktury sieciowej: wykrywa urządzenia i usługi sieciowe, zbiera metryki wydajności oraz generuje alerty w przypadku wykrycia problemów. Sprawdźmy zatem w praktyce, czy po instalacji „odkryje" inne serwery i urządzenia w mojej domowej sieci.

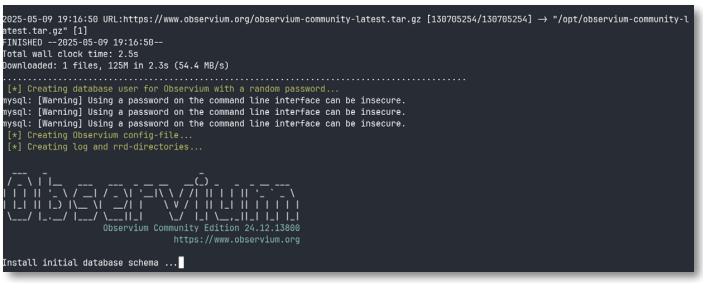
Musimy upewnić się, że mamy pewne niezbędne pakiety zainstalowane w systemie:

```
apt install software-properties-common wget
```

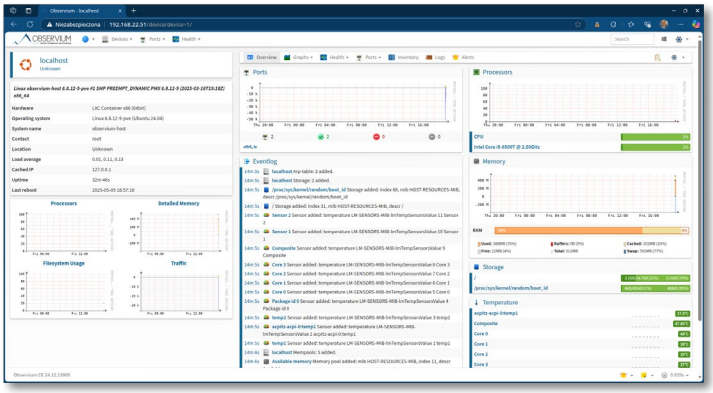
Na stronie dokumentacji https://docs.observium.org/install_debian/ znajdziemy dosyć prostą recepturę instalacyjną – właściwie to trzy polecenia:

- `wget http://www.observium.org/observium_installscript.sh`
- `chmod +x observium_installscript.sh`
- `./observium_installscript.sh`

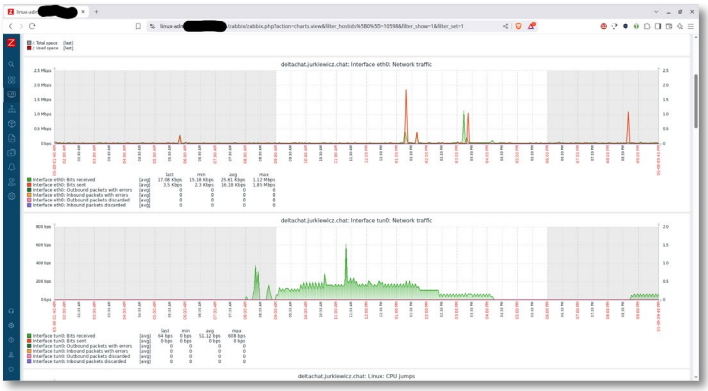
Pobieramy ze strony Observium skrypt instalacyjny, nadajemy mu uprawnienia wykonywania (to w Linuksie coś jak exe w Windows – wybaczcie proszę ten skrót myślowy, może niezbyt idealny, ale powinien dać pogląd na to, co się dzieje) i uruchamiamy. Skrypt wykonuje za nas tzw. brudną robotę, czyli instaluje i konfiguruje niezbędne komponenty.



Po kilku minutach możemy logować się za pomocą przeglądarki do działającego systemu i zobaczymy parametry urządzeń, które monitorujemy – na początku jest to serwer, który podczas instalacji dodałem do monitoringu.



System ten jest podobny do bardziej znanego Zabbiksa (https://www.zabbix.com).



Zrzut mojego systemu Zabbix, monitorującego usługi i serwery moje i moich klientów.

Poprosiłem system perplexity.ai o dokonanie szybkie porównania tych systemów – oto efekt (tabela na str. 36).

Observium jest prostszym narzędziem do monitoringu sieci, idealnym dla mniejszych i średnich infrastruktur (najlepiej urządzeń z SNMP), natomiast Zabbix to rozbudowany, skalowalny system monitoringu całej infrastruktury IT – jednak wam pozostawiam decyzję, którego z systemów będziecie używać.

Pamiętajmy również , że to jest …subiektywny poradnik administratora. Pokazuję różne systemy, ciekawe i przydatne (moim zdaniem) w firmie mniejszej lub większej. Przede wszystkim muszą być dostępne do instalacji na własnym serwerze (self-hosted), bo wtedy mamy pełną kontrolę i uniezależniamy się od różnych decyzji firm trzecich.

https://blog.jurkiewicz.tech/zabbix-monitoring-server-also-ansible-command-center-how-to-9c22d254eba7

Funkcja/cecha	Observium	Zabbix
Łatwość użycia	Intuicyjny i prosty interfejs, łatwy w obsłudze, szczególnie dla sieci małych i średnich.	Bardziej rozbudowany i konfigurowalny, ale wymaga większej wiedzy i czasu na naukę.
Skalowalność	Dobrze sprawdza się w małych i średnich sieciach, może mieć ograniczenia przy bardzo dużych środowiskach.	Bardzo skalowalny, nadaje się do monitorowania tysięcy urządzeń, posiada architekturę rozproszoną.
Obsługa urządzeń	Skupiony głównie na urządzeniach sieciowych (Cisco, Juniper, HP itp.).	Obsługuje szeroki zakres urządzeń: serwery, aplikacje, maszyny wirtualne i sieci.
Funkcje alertowania	Podstawowe alerty, bardziej zaawansowane wymagają integracji z zewnętrznymi narzędziami.	Zaawansowane alerty i powiadomienia (email, SMS, push), rozbudowane opcje eskalacji.
Interfejs użytkownika	Webowy, prosty, z automatycznym mapowaniem sieci i wizualizacjami.	Webowy, z możliwością tworzenia spersonalizowanych dashboardów i raportów.
Model licencjonowania	Edycja Community darmowa, pełna wersja wymaga licencji komercyjnej.	Całkowicie open source, bezpłatny dostęp do wszystkich funkcji.
Wsparcie społeczności	Aktywna, ale mniejsza społeczność i mniej dostępnych wtyczek.	Duża i aktywna społeczność, bogata dokumentacja, liczne integracje i szablony.
Automatyczne wykrywanie	Obsługuje protokoły takie jak LLDP, CDP do automatycznego wykrywania urządzeń i mapowania sieci.	Posiada funkcję autowykrywania urządzeń i usług, wspiera monitoring agentowy i bezagentowy.
Zastosowanie	Głównie monitoring sieci i urządzeń sieciowych.	Kompleksowy monitoring infrastruktury IT: sieci, serwerów, aplikacji i usług.
Wydajność i architektura	Prosta architektura, mniej zasobożerna, ale mniej elastyczna w dużych środowiskach.	Wysoka wydajność dzięki modułowej architekturze, możliwość rozdzielenia funkcji na różne serwery.