Poznajemy Nginx i instalacja

Opowieść, instalacja z pakietu i w Dockerze.

- Nginx: serwer, który powstał, żeby obsłużyć dziesięć tysięcy połączeń
- Instalacja Nginx na Rocky Linux 9 i w Dockerze

Nginx: serwer, który powstał, żeby obsłużyć dziesięć tysięcy połączeń

Poziom: podstawy **Dotyczy: Nginx, cała książka** **Czas: ~15 min**

Ta strona otwiera książkę o Nginksie: skąd wziął się serwer, który dziś stoi przed niemal każdą aplikacją webową w Zielonej, jaki problem miał rozwiązać na starcie i co czeka na kolejnych stronach.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
C10K	problem C10K, dziesięć tysięcy jednoczesnych połączeń: pytanie postawione w 1999 roku, jak jeden serwer ma obsłużyć taki ruch bez klękania pod własnym ciężarem, od którego zaczyna się historia Nginksa
FTP	File Transfer Protocol: stary protokół przesyłania plików, bez szyfrowania
HTTP	Hypertext Transfer Protocol: protokół stron WWW
IT	Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy
TLS	Transport Layer Security: szyfrowanie połączeń, np. w HTTPS
WebSocket	stałe, dwukierunkowe połączenie przeglądarki z serwerem, przez które Home Assistant na żywo odświeża panele; serwer pośredniczący musi przekazywać nagłówki Upgrade i Connection
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Skąd się wzięło

Pod koniec lat dziewięćdziesiątych popularne wtedy serwery WWW, głównie Apache, uruchamiały osobny proces albo wątek systemowy na każde przychodzące połączenie. Przy paruset jednoczesnych użytkownikach to jeszcze działało, ale rosnące portale internetowe zaczęły przyjmować tysiące połączeń naraz, a każdy dodatkowy proces zjadał pamięć i czas procesora na

samo przełączanie między nimi.

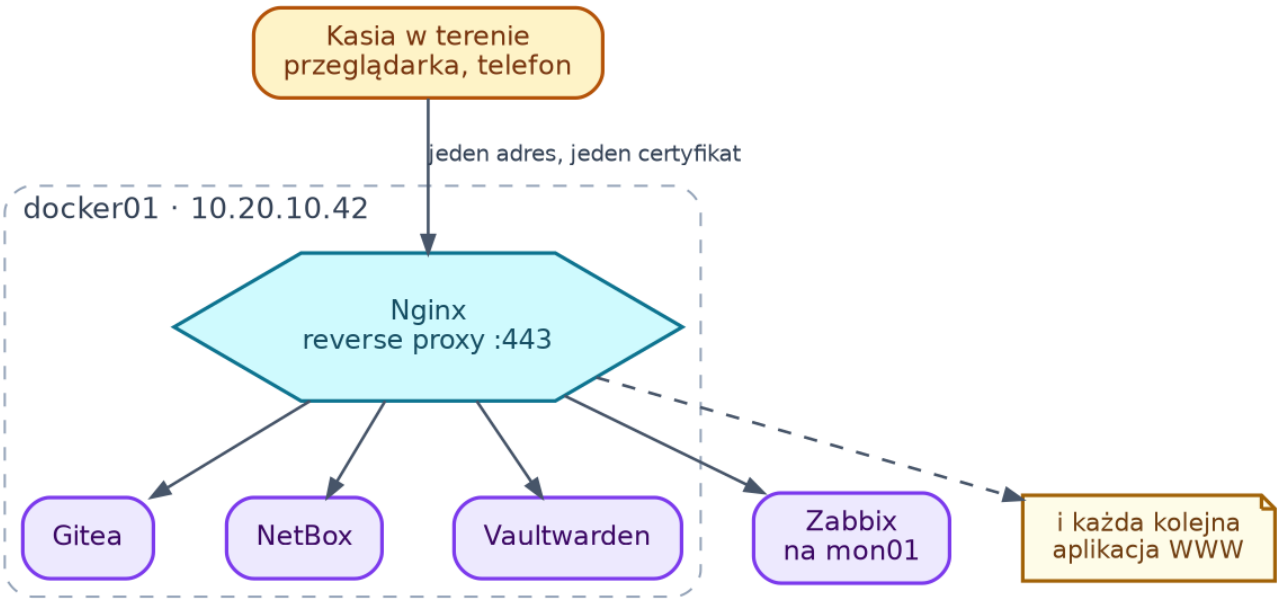
> **Pojęcie: problem C10K.** Nazwa ukuta w 1999 roku przez inżyniera Dana Kegela, opisująca pytanie, jak jeden serwer ma obsłużyć dziesięć tysięcy jednoczesnych połączeń bez klękania pod własnym ciężarem. Kegel sformułował ją po tym, jak zobaczył serwer FTP `cdrom.com` obsługujący dziesięć tysięcy klientów naraz na jednym łączu gigabitowym. Nazwa przylgnęła do całej fali nowych serwerów zdarzeniowych, które zamiast procesu na połączenie trzymały niewielką liczbę procesów obsługujących tysiące połączeń jednocześnie.

Igor Sysoev, rosyjski programista pracujący wtedy dla Ramblera, jednego z największych rosyjskich portali internetowych, zmagał się z dokładnie tym problemem na co dzień. Zaczął pisać własny serwer HTTP jako projekt poboczny już w 2002 roku, właśnie w odpowiedzi na C10K: architektura zdarzeniowa zamiast procesu na połączenie, mały narzut pamięci na jednoczesne żądanie. W 2004 roku wydał go publicznie pod nazwą Nginx, w skrócie od angielskiego „engine x". Do września 2008 roku sam Rambler obsługiwał nim już 500 milionów żądań dziennie.

Nginx rozchodził się dalej niezależnie od twórcy, a w lipcu 2011 roku Sysoev razem z Maximem Konovalovem założyli firmę Nginx, Inc., żeby sprzedawać wsparcie i produkty komercyjne wokół darmowego serwera. W marcu 2019 roku firmę kupił F5 Networks za 670 milionów dolarów, przejmując rozwój flagowego produktu razem z zespołem, choć sam serwer Nginx pozostał darmowy i otwarty jak wcześniej.

Jak to działa w jednym obrazku

Zamiast tworzyć nowy proces systemowy na każde połączenie, Nginx trzyma niewielką liczbę procesów roboczych, z których każdy obsługuje tysiące połączeń jednocześnie, przełączając się między nimi tylko wtedy, gdy któreś faktycznie ma dane do przeczytania albo wysłania. Ten sam mechanizm pozwala w Zielonej jednemu Nginksowi na `docker01` stać przed wieloma aplikacjami naraz, zamiast wystawiać każdą z nich osobno na świat.



Nginx jako reverse proxy przed Gitea, NetBoksem, Vaultwardenem i Zabbiksem w Zielonej (kliknięcie otwiera pełny rozmiar)

Dlaczego Zielona

> **Z ŻYCIA WZIĘTE**
>
> Wiosna 2022 roku. Zespół IT Zielonej trzyma wtedy Giteę na porcie 3000, a NetBoksa na porcie 8000, Vaultwardena jeszcze nie ma. Każda aplikacja ma własny certyfikat TLS, własny port do zapamiętania i własny wpis w regułach FortiGate. Ola pyta, dlaczego link do repozytorium wygląda jak `https://docker01.zielona.example:3000`, zamiast po prostu `https://git.zielona.example`.

Odpowiedzią był reverse proxy, opisany szerzej na stronie Reverse proxy i równoważenie obciążenia. Jeden Nginx na `docker01` przejął port 443 dla wszystkich aplikacji naraz, każdej przydzielając własną nazwę domenową zamiast numeru portu, i własny certyfikat TLS bez trzymania go osobno w każdej aplikacji. Zespół IT dodaje dziś nową aplikację do Zielonej, dopisując kilka linii do konfiguracji Nginksa, zamiast otwierać kolejny port na zaporze i tłumaczyć użytkownikom nowy adres.

Drugi powód jest równie praktyczny: Nginx radzi sobie z ruchem, który same aplikacje obsługują gorzej, jak długie połączenia WebSocket używane przez powiadomienia w Vaultwardenie, albo przekazywanie żądań do kilku instancji tej samej aplikacji naraz przy większym ruchu. Dwuosobowy zespół nie ma czasu uczyć każdej aplikacji z osobna obsługi TLS i równoważenia ruchu, więc robi to

raz, w jednym miejscu, przed nimi wszystkimi.

Mapa tej książki

Strona	Na jakie pytanie odpowiada
Nginx: serwer, który powstał, żeby obsłużyć dziesięć tysięcy połączeń	Skąd wziął się Nginx i po co stoi przed każdą aplikacją w Zielonej
Instalacja Nginx na Rocky Linux 9 i w Dockerze	Jak postawić Nginksa z pakietu albo w kontenerze
Architektura Nginx, procesy, konteksty, pliki konfiguracyjne	Z czego Nginx się składa i co znaczy przeładowanie konfiguracji
Bloki server i location	Który blok obsłuży dane żądanie i w jakiej kolejności
Reverse proxy i równoważenie obciążenia	Jak przekazać ruch do aplikacji i rozłożyć go na kilka serwerów
TLS w Nginx	Jak skonfigurować szyfrowane połączenie i sprawdzić, co serwer faktycznie podaje
Logi i diagnostyka	Gdzie szukać przyczyny błędu 502, 504 albo 413
Nginx: kopia konfiguracji, certyfikatów i bezpieczna aktualizacja	Jak bezpiecznie zaktualizować Nginksa i nie zgubić certyfikatów

Powiązane: Reverse proxy i równoważenie obciążenia · Instalacja Nginx na Rocky Linux 9 i w Dockerze · Konwencja Dockera w Zielonej

Źródła: en.wikipedia.org/wiki/Nginx · en.wikipedia.org/wiki/Igor_Sysoev · en.wikipedia.org/wiki/C10k_problem · blog.nginx.org/blog/celebrating-20-years-of-nginx · thehackernews.com/2019/03/f5-networks-acquires-nginx.html · kegel.com/c10k.html

Instalacja Nginx na Rocky Linux 9 i w Dockerze

Typ: procedura

Dotyczy: Nginx z AppStream Rocky Linux 9 albo z repozytorium nginx.org, oraz obraz nginx:1.30.5 (sprawdzony na Docker Hub 28 września 2026 jako aktualna gałąź stabilna)

Czas: ~30 min

Efekt: działający Nginx gotowy do dalszej konfiguracji jako reverse proxy, w jednym z dwóch wariantów: usługa systemd z pakietu na Rocky Linux 9, albo kontener Docker według konwencji Zielonej. Wybieramy wariant zgodny z tym, co już stoi na docelowym serwerze, obu naraz nie instalujemy na tym samym hoście.

Skróty użyte na tej stronie (5)

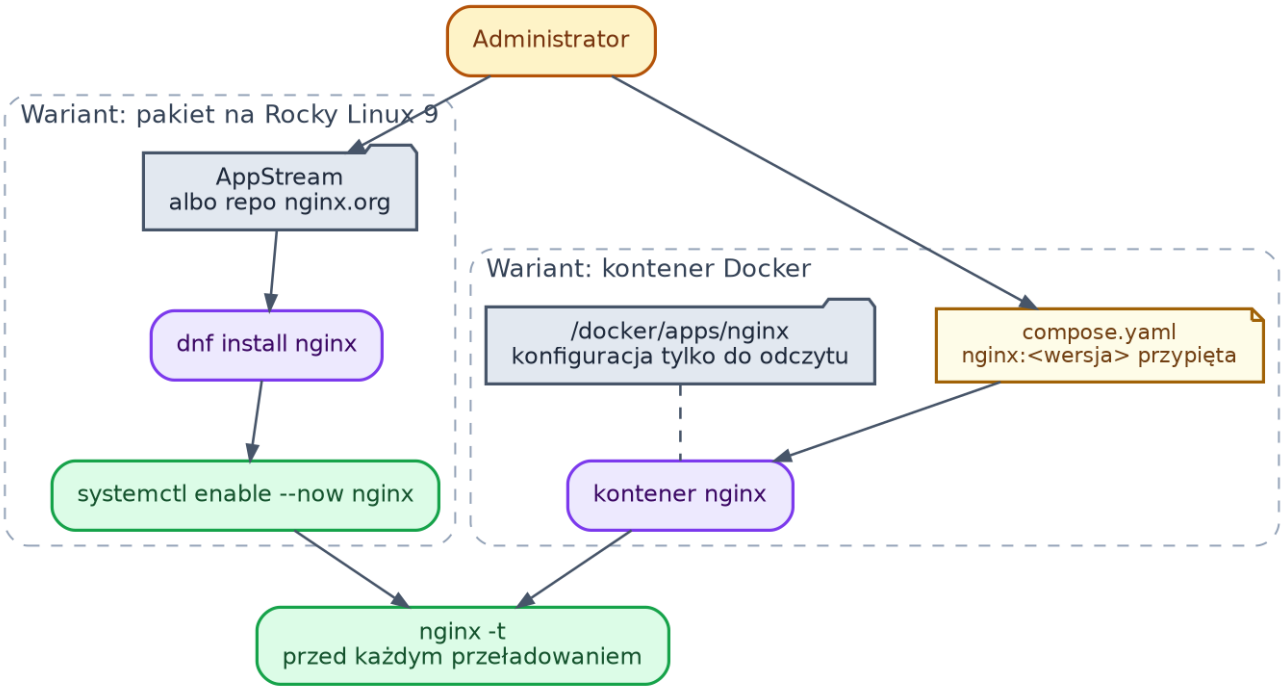
Skrót	Co znaczy
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
OS	Operating System: system operacyjny
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SELinux	Security-Enhanced Linux: dodatkowa warstwa uprawnień w jądrze RHEL
TLS	Transport Layer Security: szyfrowanie połączeń, np. w HTTPS

Zanim zaczniemy

Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo albo root. OS musi rozwiązywać nazwy domeny i mieć ruch wychodzący do repozytorium pakietów albo do rejestru Docker Hub, zależnie od wybranego wariantu.

Wymaganie	Uwaga
Rocky Linux 9 albo RHEL 9 (wariant pakietowy)	polecenia SELinux i firewalld dotyczą tej rodziny systemów
Docker Engine i wtyczka Compose (wariant kontenerowy)	sprawdzamy `docker compose version`
Wolny port 80 i 443 na hoście	żaden inny proces nie może już na nich nasłuchiwać

Wymaganie	Uwaga
Ruch wychodzący do `nginx.org` (jeśli wybieramy ten wariant pakietu)	pobranie pliku repozytorium i pakietów



Instalacja Nginx: wariant pakietowy z AppStream albo nginx.org kończący się usługą systemd, wariant Docker z compose.yaml i wolumenem konfiguracji tylko do odczytu, oba zweryfikowane przez nginx -t (kliknięcie otwiera pełny rozmiar)

> W naszym środowisku dane aplikacji trzymamy w `/docker/apps/<nazwa>`, a plik Compose w `/docker/compose/<nazwa>/compose.yaml`. To konwencja, nie wymóg Dockera: można ją zmienić, byle w całej firmie była jedna. Katalog `/docker` to osobny wolumen `vgDocker/lvDocker`, żeby pełny dysk aplikacji nie zatrzymał systemu.

Przebieg

Wariant A: pakiet na Rocky Linux 9

1. Wybór źródła pakietu: AppStream kontra repozytorium nginx.org

Rocky Linux 9 udostępnia Nginksa jako moduł AppStream, sprawdzamy dostępne strumienie wersji:

```
dnf module list nginx
```

Przykład:

```
Rocky Linux 9 - AppStream
Name    Stream    Profiles           Summary
nginx   1.24 [d]  common [d]         nginx webserver
nginx   1.26      common [d]         nginx webserver
```

Uważaj: strumień oznaczony `[d]` to domyślny, zwykle starszy niż najnowsza wersja stabilna z nginx.org, dostępne strumienie sprawdzamy zawsze komendą wyżej, bo lista rośnie z czasem. Repozytorium `nginx.org` daje dostęp do najnowszej gałęzi stabilnej i mainline niezależnie od cyklu wydawniczego dystrybucji, kosztem jednej dodatkowej rzeczy do aktualizowania poza `dnf update` całego systemu. W Zielonej wybieramy AppStream tam, gdzie wystarcza nam wersja z dystrybucji, a `nginx.org` tam, gdzie zależy nam na najnowszych poprawkach TLS albo konkretnym module.

2. Instalacja z AppStream

```
sudo dnf module enable nginx:1.26 -y
sudo dnf install -y nginx
```

Przykład:

```
Installed:
  nginx-1.26.3-1.el9.x86_64    nginx-filesystem-1.26.3-1.el9.noarch
Complete!
```

3. Instalacja z repozytorium nginx.org (alternatywa dla kroku 2)

Plik `/etc/yum.repos.d/nginx.repo`:

```
[nginx-stable]
name=nginx stable repo
baseurl=https://nginx.org/packages/centos/9/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
```

```
sudo dnf install -y nginx
```

Uważaj: przy pierwszym imporcie klucza `dnf` pokazuje jego odcisk do potwierdzenia, `573B FD6B 3D8F BC64 1079 A6AB ABF5 BD82 7BD9 BF62` dla klucza nginx.org sprawdzonego 28 września 2026 roku, porównujemy go z tym, co podaje strona nginx.org, zanim potwierdzimy instalację.

Wykonujemy tylko jeden z kroków 2 i 3, nie oba naraz na tym samym hoście, drugi pakiet konfliktowałby z pierwszym.

4. Uruchomienie usługi

```
sudo systemctl enable --now nginx
```

Przykład:

```
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service →
/usr/lib/systemd/system/nginx.service.
```

5. Firewalld

```
sudo firewall-cmd --permanent --add-service=http --add-service=https
sudo firewall-cmd --reload
```

Przykład sprawdzenia:

```
sudo firewall-cmd --list-services
```

```
cockpit dhcpv6-client http https ssh
```

6. SELinux: httpd_can_network_connect dla proxy

Domyślna polityka SELinux pozwala Nginksowi serwować pliki statyczne, ale blokuje mu łączenie się po sieci z backendem, gdy zaczynamy używać go jako reverse proxy opisanego na stronie Reverse proxy i równoważenie obciążenia.

```
sudo setsebool -P httpd_can_network_connect on
```

> **Pojęcie: httpd_can_network_connect.** Flaga logiczna SELinux (boolean), wspólna dla Nginksa i Apache, bo obie usługi na RHEL działają w tej samej domenie SELinux `httpd_t`, opisana szerzej przy Zabbiksie na stronie Instalacja serwera Zabbix z pakietów na Rocky

> Linux 9. Bez jej włączenia `proxy_pass` do adresu innego niż `localhost` kończy się błędem 502, mimo poprawnej konfiguracji Nginksa, bo to SELinux, nie sam proces, blokuje połączenie wychodzące.

Wariant B: kontener Docker

7. Katalogi na dane, zgodnie z konwencją ścieżek

```
sudo mkdir -p /docker/apps/nginx/conf.d
sudo mkdir -p /docker/apps/nginx/certs
sudo mkdir -p /docker/compose/nginx
```

Katalog `conf.d` trzyma pliki `server` dla poszczególnych aplikacji, `certs` certyfikaty i klucze TLS, opisane szerzej na stronie TLS w Nginx.

8. Główny plik konfiguracyjny: /docker/apps/nginx/nginx.conf

```
user  nginx;
worker_processes  auto;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;

    include /etc/nginx/conf.d/*.conf;
}
```

9. Plik Compose: /docker/compose/nginx/compose.yaml

```
services:
  nginx:
    image: nginx:1.30.5
    container_name: nginx
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /docker/apps/nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - /docker/apps/nginx/conf.d:/etc/nginx/conf.d:ro
      - /docker/apps/nginx/certs:/etc/nginx/ssl:ro
```

Obraz przypinamy zawsze do konkretnej wersji, nigdy do `latest`, tak jak przy każdej innej aplikacji w Zielonej. Wszystkie trzy wolumeny z konfiguracją montujemy z flagą `:ro`, czyli tylko do odczytu z poziomu kontenera: konfigurację zmieniamy zawsze na hoście i przeładowujemy Nginksa, nigdy edycją pliku wewnątrz działającego kontenera, która i tak zniknęłaby przy jego następnym odtworzeniu.

10. Pierwsze uruchomienie

```
cd /docker/compose/nginx
docker compose up -d
```

Przykład:

```
[+] Running 2/2
 ✔ Network nginx_default  Created
 ✔ Container nginx        Started
```

Weryfikacja

Sprawdzenie składni przed jakimkolwiek przeładowaniem, w obu wariantach jednakowo ważne:

```
sudo nginx -t
```

Przykład (wariant pakietowy):

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

W wariancie kontenerowym tę samą komendę wykonujemy wewnątrz kontenera:

```
docker compose exec nginx nginx -t
```

Sprawdzenie, że usługa faktycznie odpowiada:

```
curl -I http://localhost
```

Przykład:

```
HTTP/1.1 200 OK
Server: nginx/1.30.5
```

Sprawdzenie procesu (wariant pakietowy) albo kontenera (wariant Docker):

```
sudo systemctl status nginx
```

```
docker compose ps
```

Wycofanie zmian

Wariant pakietowy:

```
sudo systemctl disable --now nginx
sudo dnf remove -y nginx
sudo firewall-cmd --permanent --remove-service=http --remove-service=https
sudo firewall-cmd --reload
```

Wariant Docker, bez kasowania konfiguracji z `/docker/apps/nginx`:

```
docker compose -f /docker/compose/nginx/compose.yaml down
```

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
`dnf install nginx` instaluje bardzo starą wersję	domyślny strumień AppStream `[d]` jest najstarszym dostępnym	`dnf module list nginx`, włączamy nowszy strumień przed instalacją, patrz krok 1
`curl -I http://localhost` zwraca `Connection refused` mimo działającej usługi	firewalld blokuje port	`firewall-cmd --list-services`, dodajemy `http` / `https`
Reverse proxy do backendu na innym hoście zwraca 502, `nginx -t` przechodzi bez błędu	SELinux blokuje połączenie wychodzące procesu Nginksa	`sudo ausearch -m avc -c nginx --start recent`, włączamy `httpd_can_network_connect`, patrz krok 6
W wariancie Docker zmiana w `nginx.conf` na hoście nie widoczna po `docker compose restart`	edytowano plik wewnątrz kontenera zamiast na hoście, zmiana zniknęła przy restarcie	edytujemy wyłącznie pliki w `/docker/apps/nginx` na hoście, wolumeny `:ro` i tak nie pozwalają zapisać z kontenera
Kontener startuje i od razu kończy działanie	błąd składni w `nginx.conf` albo brakujący plik wskazany przez `include`	`docker compose logs nginx`, poprawiamy plik na hoście, `docker compose up -d` ponownie

Powiązane: Architektura Nginx, procesy, konteksty, pliki konfiguracyjne · Reverse proxy i równoważenie obciążenia · Konwencja Dockera w Zielonej · Nginx: kopia konfiguracji, certyfikatów i bezpieczna aktualizacja

Źródła: nginx.org/en/linux_packages.html, sprawdzone 28 września 2026 · docs.redhat.com, „Setting up and configuring NGINX", RHEL 9 · hub.docker.com/_/nginx, sprawdzone 28 września 2026 jako `nginx:1.30.5` · `man 8 setsebool`