Docker: kontenery, które zmieniły serwerownie

Poziom: podstawy **Dotyczy: Docker Engine 24+** **Czas: ~15 min**

Ta strona otwiera książkę o Dockerze: skąd wziął się pomysł zamykania aplikacji w izolowanych pudełkach, dlaczego Zielona w ogóle postawiła osobny serwer pod kontenery i co czeka na kolejnych stronach.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
LXC	Linux Containers: wczesny mechanizm izolacji procesów w jądrze Linuksa, na którym pierwotnie oparto Dockera

Skąd się wzięło

> **Pojęcie: kontener.** Sposób uruchamiania aplikacji w izolowanym środowisku, które dzieli jądro systemu operacyjnego z hostem, ale ma własny system plików, własne procesy i własną sieć, tak jakby stało na osobnej maszynie. W odróżnieniu od maszyny wirtualnej kontener nie uruchamia własnego systemu operacyjnego: pożycza go od hosta.

Pomysł na izolowane pudełko do transportu jest starszy niż informatyka. 26 kwietnia 1956 roku Amerykanin Malcolm McLean załadował na przerobiony tankowiec SS Ideal-X pięćdziesiąt osiem identycznych metalowych skrzyń i wysłał je z Newark do Houston. Przed tym dniem towar na statek trafiał sztuka po sztuce, ręcznie przekładany przez dokerów z ciężarówki na pokład; po tym dniu wystarczyło przełożyć całą skrzynię dźwigiem, niezależnie od tego, co było w środku. Standardowy rozmiar i standardowe zaczepy sprawiły, że ten sam kontener przechodził bez przeładunku ze statku na ciężarówkę, a z ciężarówki na pociąg.

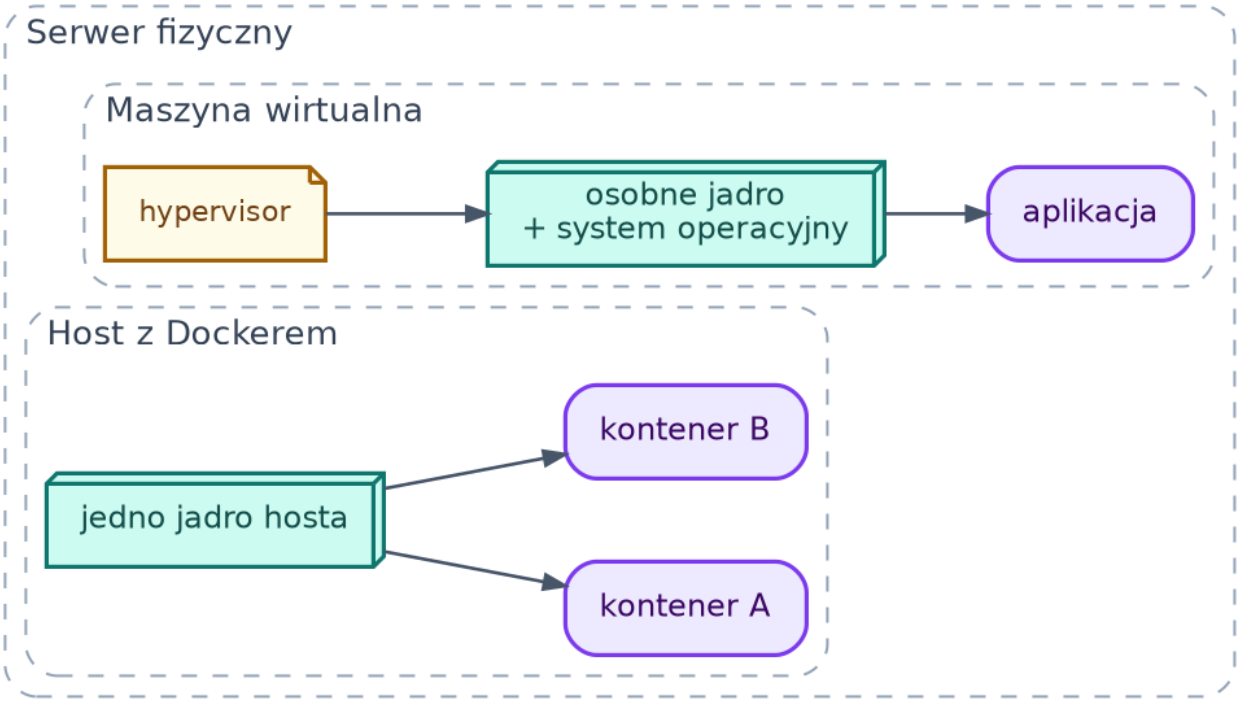
W świecie serwerów podobny problem, jak uruchomić wiele aplikacji na jednej maszynie, nie pozwalając im się nawzajem zaśmiecać, rozwiązywano stopniowo. W 1979 roku, przy pracach nad siódmą wersją Uniksa, pojawił się `chroot`: mechanizm zamykający proces w wybranym podkatalogu, który z jego punktu widzenia wygląda jak cały dysk. W 2000 roku system FreeBSD

rozwinął ten pomysł do jails, dokładając izolację sieci i procesów, nie tylko systemu plików. W 2008 roku w jądrze Linuksa pojawił się LXC, korzystający z dwóch nowych mechanizmów jądra: przestrzeni nazw i grup kontrolnych, które w końcu pozwoliły odizolować od siebie nie tylko pliki, ale i widok na procesy, sieć czy użytkowników.

> **Pojęcie: Docker.** Narzędzie, które w 2013 roku zapakowało LXC, a później własny mechanizm izolacji, w prostą komendę i czytelny format opisu aplikacji, dzięki czemu uruchomienie kontenera przestało wymagać znajomości szczegółów jądra Linuksa. Solomon Hykes pracował wtedy nad dotCloud, firmą hostingową wywodzącą się z programu Y Combinator z lata 2010 roku. Docker powstał jako wewnętrzne narzędzie dotCloud do pakowania aplikacji klientów, a w marcu 2013 roku Hykes pokazał je publicznie na konferencji PyCon w Santa Clara i tego samego roku wydał je jako otwarte oprogramowanie. Sama firma dotCloud jako hosting zniknęła z rynku, ale narzędzie, które miało być jej wewnętrznym dodatkiem, zmieniło sposób, w jaki cała branża stawia aplikacje na serwerach.

Jak to działa w jednym obrazku

Maszyna wirtualna i kontener rozwiązują ten sam problem (odizolowanie aplikacji od reszty serwera) na dwóch różnych piętrach. Maszyna wirtualna uruchamia pełny, osobny system operacyjny nad warstwą wirtualizacji sprzętu, więc dwie maszyny wirtualne na jednym serwerze to dwa jądra, dwa zestawy sterowników i dwa razy więcej pamięci zajętej samym systemem, zanim jeszcze wystartuje aplikacja. Kontener dzieli jedno jądro hosta ze wszystkimi innymi kontenerami i uruchamia tylko to, co potrzebne samej aplikacji, więc startuje w sekundach zamiast minut i mieści się w ułamku pamięci maszyny wirtualnej.



Maszyna wirtualna z osobnym jądrem systemowym obok kilku kontenerów dzielących jedno jądro hosta (kliknięcie otwiera pełny rozmiar)

Dlaczego Zielona

> **Z ŻYCIA WZIĘTE**
>
> Zanim Zielona miała `docker01`, każda nowa aplikacja zespołu Oli zaczynała się tym samym zgłoszeniem do nas: „potrzebujemy maszyny". Zamawialiśmy maszynę wirtualną na `hv01`, instalowaliśmy system, aktualizacje, runtime i zależności, a Ola i tak czekała tydzień, zanim cokolwiek w ogóle ruszyło. Gdy trzy miesiące później zespół chciał postawić drugą, mniejszą usługę obok pierwszej, cała procedura zaczynała się od nowa: kolejna maszyna, kolejny tydzień, kolejny system do łatania.

Ta sama aplikacja jako kontener na `docker01` startuje w kilkanaście sekund, bez zamawiania niczego u nas: Ola dostaje dostęp do serwera raz, a każdą kolejną usługę stawia sama, plikiem Compose, który sama pisze i sama rozumie.

Zysk nie jest tylko czasowy. Aktualizacja jednej aplikacji nie wymaga już dotykania systemu operacyjnego pod spodem, bo obraz kontenera niesie swoje zależności ze sobą, a awaria jednej usługi (przepełniony dysk, wyciek pamięci) nie zabiera zasobów pozostałym, dopóki ustawiliśmy

limity opisane na stronie Utrzymanie hosta dockerowego. Zamiast dziesięciu maszyn wirtualnych z osobnym systemem każda, `docker01` trzyma dzisiaj Giteę, NetBoksa, Grayloga, MailCow i Vaultwardena na jednym systemie operacyjnym, każdą aplikację w osobnym, odizolowanym kontenerze.

Mapa tej książki

Strona	Na jakie pytanie odpowiada
Docker: kontenery, które zmieniły serwerownie	Skąd wziął się Docker i dlaczego go używamy
Jak trzymamy aplikacje w Dockerze: konwencje Zielonej	Gdzie w naszym środowisku leżą dane i pliki Compose każdej aplikacji
Instalacja Dockera na Rocky Linux 9 z osobnym wolumenem /docker	Jak postawić hosta dockerowego zgodnie z tą konwencją
Pierwsza aplikacja w Compose krok po kroku	Jak uruchomić swoją pierwszą aplikację od zera
Obraz, kontener, warstwa, wolumen: cztery pojęcia	Czym różni się obraz od kontenera i gdzie żyją dane
Codzienne komendy	Jakich komend Dockera używamy najczęściej
Docker Compose	Jak opisać wiele kontenerów jednym plikiem
Sieci kontenerów i SNAT	Jak kontenery się komunikują i wychodzą do sieci
Utrzymanie hosta dockerowego	Jak dbać o miejsce na dysku, logi i limity zasobów
Kopia zapasowa aplikacji dockerowych	Jak zabezpieczyć dane, gdy coś pójdzie nie tak

Powiązane: Jak trzymamy aplikacje w Dockerze: konwencje Zielonej · Świat przykładów: Zielona

Źródła: en.wikipedia.org/wiki/Docker_(software) · en.wikipedia.org/wiki/Malcom_McLean · en.wikipedia.org/wiki/Chroot · en.wikipedia.org/wiki/LXC

Revision #31
Created 2026-09-27 11:19:54 UTC by Tomasz Konieczny
Updated 2026-10-26 14:09:16 UTC by Tomasz Konieczny