

Kontenery w systemie rozproszonym

Od niezgodnych środowisk do wymiennej usługi

mgr inż. Jakub Woźniak

Zarządzanie Systemami Rozproszonymi · Semestr zimowy 2026/27

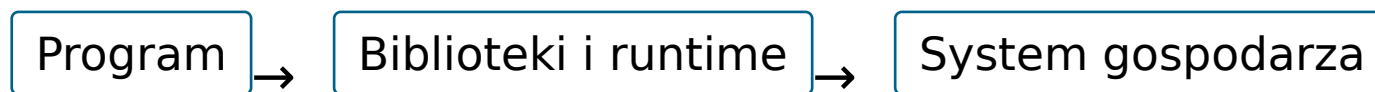
Przypadek

Worker systemu zamówień działa na pierwszym serwerze. Na drugim uruchamia się z tego samego commita, ale korzysta z innej wersji biblioteki i kończy pracę błędem.

Ten sam kod źródłowy $\neq$ to samo działające środowisko

- Wdrożenie zależy nie tylko od programu, lecz także od jego otoczenia.
- Celem jest ograniczenie liczby różnic między miejscami uruchomienia.

Worker nawiązuje do systemu zamówień z wykładu 1.



- Proces potrzebuje określonych bibliotek, narzędzi i ustawień.
- Gdy są przygotowywane osobno na każdym serwerze, łatwo o różnice niewidoczne w repozytorium źródeł.
- Potrzebna jest jednostka dostarczenia obejmująca program i jego zależności, którą można przenieść między węzłami.

OCI — Image Format Specification

Podejście	Co przenosimy	Konsekwencja
Sam program	Kod lub plik wykonywalny	Środowisko trzeba przygotować na hoście
Maszyna wirtualna	System gościa i aplikacja	Osobne jądro wzmacnia izolację
Kontener	Aplikacja i środowisko użytkowe	Wspólne jądro hosta

- Kontener pozwala powielać środowisko aplikacji bez dołączania osobnego systemu gościa do każdej instancji.
- Ta oszczędność wynika z decyzji o współdzieleniu jądra; wpływa też na granicę izolacji.

Wewnątrz

Proces widzi własne drzewo plików, przestrzeń procesów i sieć.

Na hoście

Proces nadal wykonuje wywołania do jądra gospodarza i zużywa jego zasoby.

Kontener = proces + izolowany widok otoczenia + kontrola zasobów

- Nie jest to osobny komputer ani osobne jądro systemu operacyjnego.
- Na Docker Desktop hostem kontenerów Linux może być maszyna wirtualna.

Docker — Engine security

Mechanizm	Po co jest	Czego nie rozwiązuje
Namespaces	Osobny widok procesów i sieci	Błędu współdzielonego jądra
cgroups	Kontrola zużycia zasobów	Poprawności programu
Uprawnienia	Proces bez roota ogranicza skutki błędu	Błędnej konfiguracji usługi

Wniosek

Izolacja pomaga uruchamiać wiele usług na jednym węźle, ale nie zastępuje kontroli uprawnień, aktualizacji hosta ani projektowania odporności.

Jeden obraz



Wiele niezależnych instancji

- Obraz opisuje program i jego zależności jako artefakt.
- Kontener jest konkretnym uruchomieniem obrazu: ma proces, konfigurację uruchomienia i własną warstwę zapisywalną.
- Jeden obraz może służyć do tworzenia wielu instancji. Nowa wersja obrazu nie zmienia procesów już działających.

Po co ten podział?

Umożliwia zastępowanie instancji bez ręcznej zmiany ich zawartości.

Idea	Uzasadnienie
Dockerfile	Jawny przepis utworzenia artefaktu
Warstwy	Współdzielenie niezmiennych składników obrazów
Pamięć podręczna	Szybsze ponowne budowanie po częściowej zmianie

- Te mechanizmy ułatwiają dystrybucję i rozwój aplikacji.
- Sam przepis nie gwarantuje identycznego wyniku, jeżeli obraz bazowy lub pobierane zależności mogą się zmieniać.

Docker — Cache invalidation



- Obraz ogranicza różnice w programie i jego zależnościach.
- Zgodność hosta, adres kolejki, dostępność bazy i dane pozostają właściwościami środowiska uruchomienia.
- Dwie instancje tego samego obrazu mogą więc zachowywać się inaczej.

Granica obietnicy

Przenośność artefaktu ułatwia wdrożenie; nie oznacza przenośności całego działającego systemu.

Proces i pliki tymczasowe

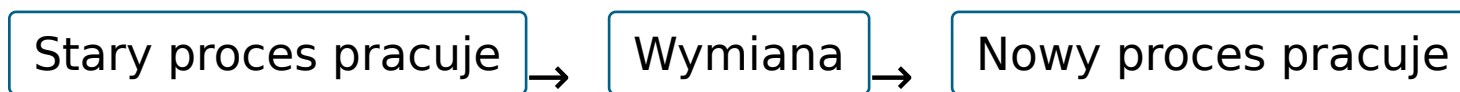
Można je zastąpić razem z kontenerem.

Zamówienia i załączniki

Muszą przetrwać wymianę instancji i awarię węzła.

- Warstwa zapisywalna kontenera znika po jego usunięciu.
- Wolumen może przetrwać usunięcie kontenera, lecz lokalny wolumen nadal wiąże dane z węzłem. Trwałość wymaga osobnej architektury i kopii zapasowej.

Docker — Volumes



- Kontener może zniknąć podczas obsługi żądania lub wiadomości.
- Możliwość ponownego dostarczenia pracy chroni przed jej utratą, lecz może prowadzić do powtórnego wykonania.
- Dlatego bezpieczna wymiana zależy od zachowania aplikacji: obsługi zakończenia, trwałego zapisu i idempotencji.

Przypadek workera rozwija problem ponawiania z wykładu 1.

Daje	Nie daje automatycznie
Przenośny artefakt	Identycznego zachowania w każdym środowisku
Izolację procesów	Pełnej granicy bezpieczeństwa
Wymienialną instancję	Nieprzerwanej obsługi żądań
Możliwość montowania danych	Kopii zapasowej i odtworzenia

Sens tej granicy

Kontenery rozwiązują problem pakowania i uruchamiania procesu. Własności całej usługi wynikają z projektu aplikacji i infrastruktury.

- Obraz i kontener dotyczą pojedynczego artefaktu oraz instancji.
- Przy wielu węzłach pojawiają się nowe pytania: gdzie uruchomić instancję, jak skierować do niej ruch i co zrobić po awarii węzła?
- Platforma kontenerowa może zarządzać rozmieszczeniem, wykrywaniem usług i zastępowaniem instancji. Nie zna jednak znaczenia danych biznesowych.

Kontener → instancja aplikacji → usługa działająca na wielu węzłach

Następny wykład: Kubernetes i rekonsyliacja stanu usługi.

Na początku

Ten sam kod zależał od bibliotek przygotowanych osobno na serwerach.

Po konteneryzacji

Program i zależności są przenoszone jako obraz. Instancje można zastępować.

- Różnice między węzłami zostały ograniczone, ale nie zniknęły: pozostają jądro hosta, konfiguracja, sieć i dostęp do danych.
- O niezawodności przetwarzania zamówień nadal decydują trwałość danych i zachowanie workera podczas przerwania pracy.

Kontener upraszcza dostarczenie programu; nie zastępuje projektu usługi.

- [OCI — Image Format Specification](#)
- [Docker — Engine security](#)
- [Docker — Cache invalidation](#)
- [Docker — Volumes](#)

Przykład workera służy wyjaśnieniu granic i zastosowań kontenerów.