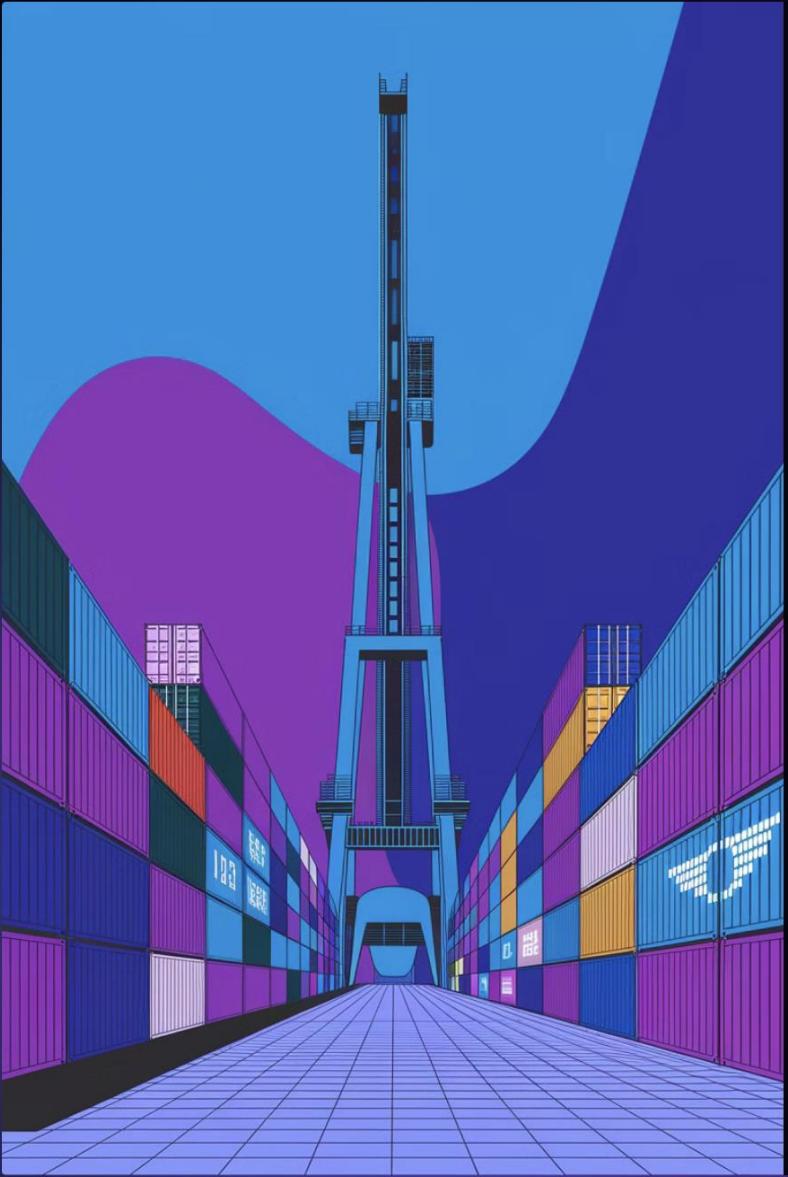


Docker



Docker: Wprowadzenie do konteneryzacji

Docker to platforma do tworzenia, dostarczania i uruchamiania aplikacji w kontenerach. Kontenery pozwalają programistom pakować aplikacje ze wszystkimi niezbędnymi częściami.

Architektura Dockera



Docker Engine

Serce systemu Docker. Odpowiada za tworzenie i zarządzanie kontenerami, obrazami, sieciami i woluminami.



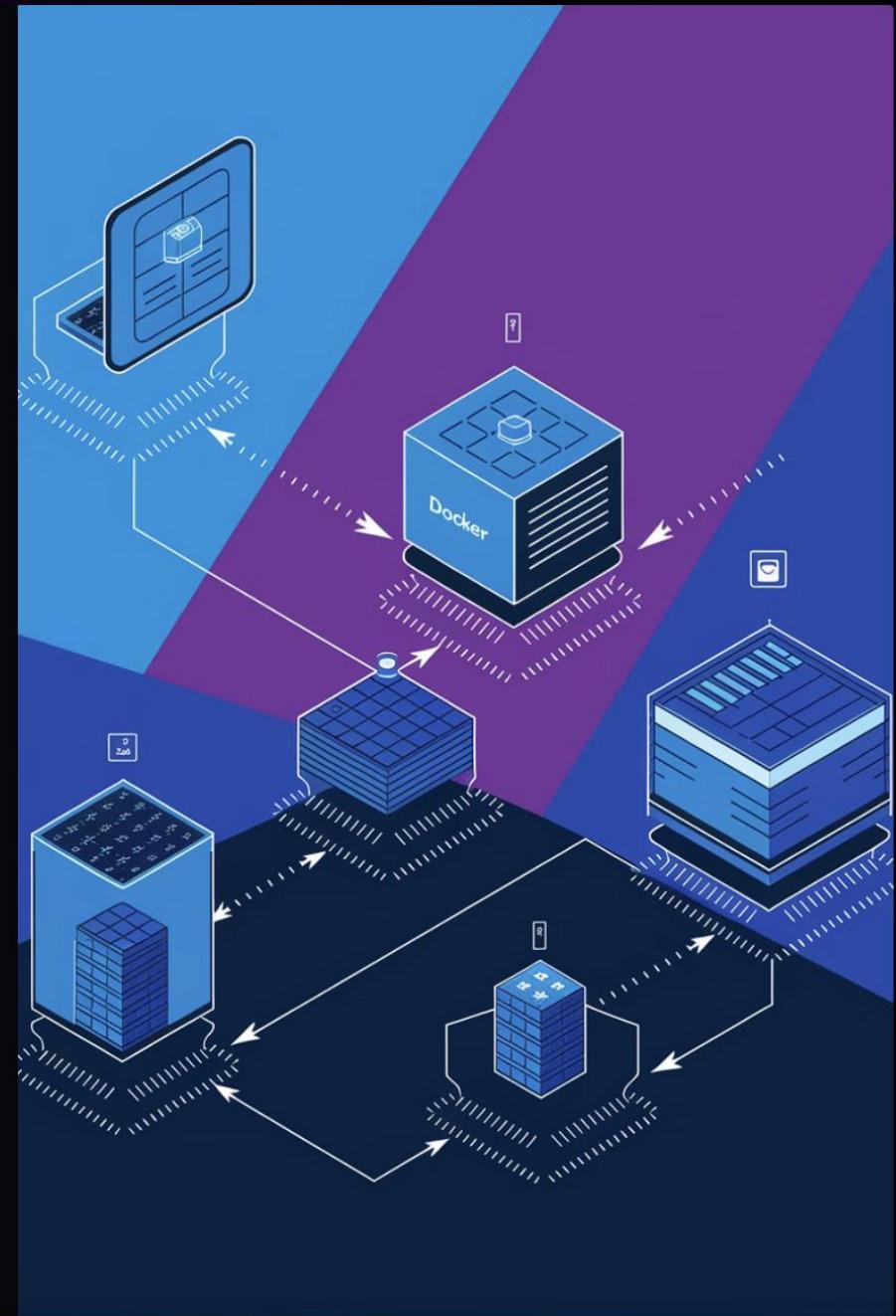
Docker Client

Interfejs wiersza poleceń umożliwiający komunikację z Docker Engine poprzez REST API.



Docker Registry

Repozytorium przechowujące obrazy Docker. Docker Hub to publiczne repozytorium z tysiącami obrazów.





Instalacja Dockera

Wymagania systemowe

Dla Linux: Jądro 3.10 lub nowsze.

Dla Windows: Windows 10

Pro/Enterprise lub nowszy. Dla
macOS: macOS 10.14 lub nowszy.

Linux

Instalacja przez menedżer
pakietów: apt, yum lub dnf.

Dodanie użytkownika do grupy
docker.

Windows i macOS

Instalacja Docker Desktop -
zintegrowanego środowiska z
Docker Engine, CLI, Docker
Compose i Kubernetes.



Podstawowe komponenty Docker'a

1

Obrazy (Images)

Szablony tylko do odczytu zawierające instrukcje do tworzenia kontenerów. Składają się z warstw reprezentujących zmiany w systemie plików.

2

Kontenery (Containers)

Uruchomione instancje obrazów. Izolowane środowiska z własnym procesorem, pamięcią i zasobami sieciowymi.

3

Woluminy (Volumes)

Mechanizm przechowywania danych generowanych i używanych przez kontenery. Dane istnieją niezależnie od życia kontenera.

Tworzenie obrazów Docker

Dockerfile

Plik tekstowy zawierający instrukcje potrzebne do zbudowania obrazu Docker. Każda instrukcja tworzy nową warstwę w obrazie.

Struktura Dockerfile

- FROM - obraz bazowy
- WORKDIR - katalog roboczy
- COPY/ADD - kopiowanie plików
- RUN - wykonanie poleceń
- EXPOSE - otwieranie portów
- CMD/ENTRYPOINT - polecenie startowe



Polecenia do budowania obrazów

1

docker build

Tworzy obraz z Dockerfile. Użyj flagi -t aby nazwać i oznaczyć obraz:
docker build -t nazwa:tag .

2

docker tag

Oznacza obraz odniesieniem do rejestru: docker tag obraz:tag
repozytorium/obraz:tag

3

docker push

Wysyła obraz do skonfigurowanego rejestru: docker push
repozytorium/obraz:tag



Zarządzanie obrazami

Listowanie obrazów

Polecenie docker images wyświetla wszystkie lokalne obrazy. Użyj docker image ls -a, aby zobaczyć wszystkie warstwy.

Usuwanie nieużywanych obrazów

docker rmi usuwa pojedyncze obrazy. docker image prune usuwa wszystkie nieużywane obrazy.

Optymalizacja rozmiaru obrazów

Używaj wielu etapów budowania (multi-stage builds). Łacz polecenia RUN. Usuwaj niepotrzebne pliki w ramach tej samej warstwy.



Repozytoria Docker

Docker Hub

Publiczne repozytorium obrazów Docker. Zawiera oficjalne obrazy, obrazy organizacji i obrazy społeczności.

Prywatne repozytoria

Docker Registry, Docker Trusted Registry, Amazon ECR, Google Container Registry, Azure Container Registry.

Bezpieczeństwo obrazów

Skanowanie luk bezpieczeństwa. Podpisywanie obrazów. Kontrola dostępu z uwierzytelnianiem i autoryzacją.

Praca z kontenerami - podstawy



Kontenery można uruchamiać w trybie odłączonym (-d), interaktywnym (-it) lub jednorazowym (--rm).

Zarządzanie kontenerami



Regularne monitorowanie kontenerów zapewnia stabilność systemu. Ograniczanie zasobów zapobiega przeciążeniu hosta.

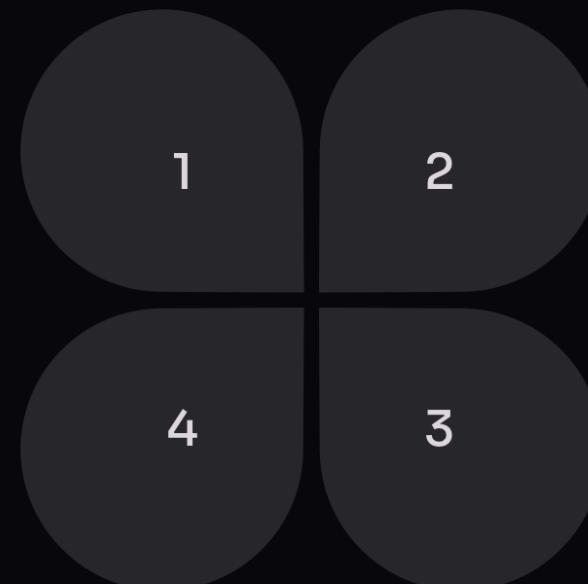
Sieci w Docker

Sieć bridge

Domyślny typ sieci. Izolowana sieć na hoście, używana do komunikacji między kontenerami.

Sieć overlay

Umożliwia komunikację między kontenerami na różnych hostach. Używana w klastrach Docker Swarm.



Sieć host

Usuwa izolację sieciową między kontenerem a hostem. Kontener używa bezpośrednio sieci hosta.

Sieć none

Brak połączenia sieciowego. Używana dla kontenerów nie wymagających sieci.

Przechowywanie danych w kontenerach

1

Woluminy Docker

Najlepsza metoda trwałego przechowywania danych. Zarządzane przez Docker.

2

Bind mounts

Podłączanie katalogów z hosta do kontenera. Świetne podczas rozwoju.

3

tmpfs mounts

Przechowywanie tymczasowe w pamięci. Dane znikają po zatrzymaniu kontenera.

Kontenery są z natury ulotne. Używaj woluminów do przechowywania trwałych danych jak bazy danych czy logi.



Projektowanie aplikacji wielokontenerowych

1

Jeden proces

Każdy kontener powinien wykonywać jedną funkcję lub uruchamiać jeden proces.

2

Bezstanowość

Kontenery powinny być bezstanowe, dane zapisuj w zewnętrznych woluminach.

3

Dekompozycja

Rozbij aplikację na mniejsze, wyspecjalizowane komponenty połączone API.

Komunikacja między kontenerami



Docker network

Tworzenie niestandardowej sieci:
`docker network create mojaapp`.
Łączenie kontenerów do sieci: `docker run --network=mojaapp`



Service discovery

Kontenery w tej samej sieci widzą się nawzajem po nazwach. DNS wbudowany w Docker rozwiązuje nazwy kontenerów.



Load balancing

Równoważenie obciążenia przez proxy jak Nginx lub HAProxy. W Docker Swarm używaj usług z wieloma replikami.

Wprowadzenie do Docker Compose

Czym jest Docker Compose?

Narzędzie do definiowania i uruchamiania aplikacji wielokontenerowych. Wykorzystuje plik YAML do konfiguracji usług, sieci i woluminów.

Główne korzyści

- Wszystkie usługi zdefiniowane w jednym miejscu
- Łatwe tworzenie środowisk developerskich
- Uruchamianie wielu kontenerów jednym poleceniem
- Zarządzanie pełnym cyklem życia aplikacji

Podstawowe polecenia Docker Compose

Najważniejsze komendy do zarządzania aplikacjami wielokontenerowymi:

1 docker-compose up

Uruchamia wszystkie zdefiniowane usługi. Flaga -d uruchamia w tle, --build wymusza przebudowanie obrazów, można też uruchomić tylko wybrane usługi podając ich nazwy.

3 docker-compose logs

Wyświetla logi usług. Flaga -f śledzi logi na bieżąco, --tail=[liczba] pokazuje określoną liczbę ostatnich linii.

5 docker-compose exec

Wykonuje polecenie w działającym kontenerze, np. docker-compose exec web bash. Przydatne do debugowania i prac konserwacyjnych.

2 docker-compose down

Zatrzymuje i usuwa kontenery, sieci i domyślne woluminy. Flaga --volumes usuwa wszystkie woluminy, --rmi all usuwa obrazy. Idealne do czyszczenia środowiska.

4 docker-compose ps

Pokazuje stan kontenerów w projekcie - nazwy, status, porty i komendy. Przydatne do szybkiej weryfikacji działających usług.

6 docker-compose build

Buduje lub przebudowuje obrazy usług. Flaga --no-cache ignoruje warstwę cache, wymuszając pełne przebudowanie.

Warto znać również: docker-compose restart (restartuje usługi), config (weryfikuje konfigurację), pull (pobiera najnowsze obrazy), top (pokazuje działające procesy).

Zaawansowane funkcje Docker Compose

Skalowanie usług

```
docker-compose up --scale  
service=5
```

Zarządzanie zależnościami

```
depends_on, healthcheck,  
restart policies
```

Zmienne środowiskowe

```
environment, env_file
```

Sekrety

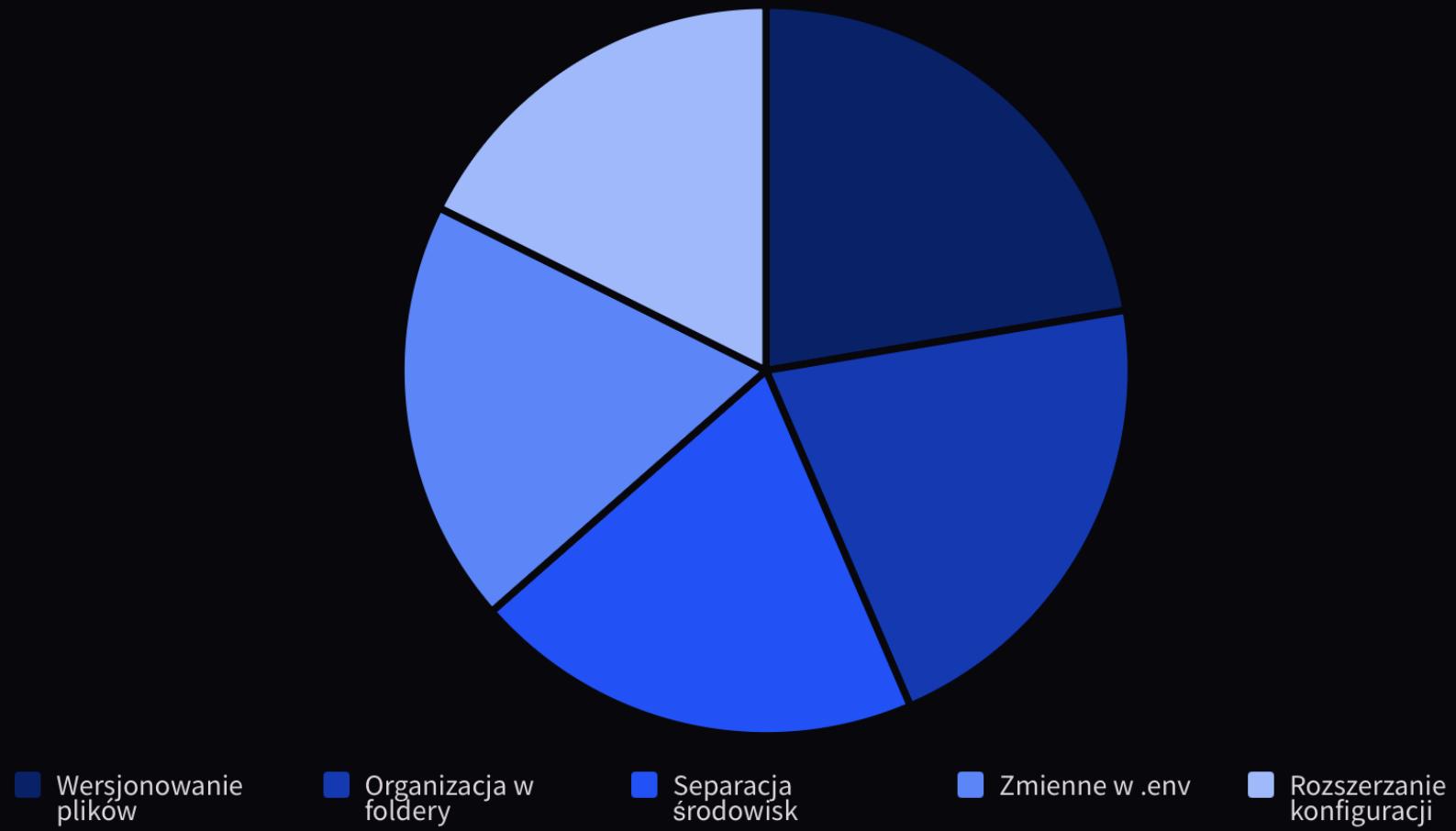
```
secrets, external secrets
```

Sieci niestandardowe

```
networks, zewnętrzne sieci
```



Najlepsze praktyki w Docker Compose



Przechowuj pliki docker-compose.yml w repozytorium. Twórz oddzielne pliki dla różnych środowisk. Używaj plików .env dla zmiennych.



Orkiestracja kontenerów

Docker Swarm

Wbudowane w Docker rozwiązanie do orkiestracji. Łatwe w użyciu, ale z mniejszą funkcjonalnością niż konkurencja.

- Natywna integracja z Docker
- Prosta konfiguracja klastra
- Kompatybilność z docker-compose

Kubernetes

Najpopularniejszy system orkiestracji kontenerów. Potężny, ale z bardziej stromą krzywą uczenia.

- Automatyczne skalowanie
- Zaawansowane zarządzanie siecią
- Rozbudowany ekosystem dodatków



Podsumowanie i przyszłość konteneryzacji

1

Kluczowe zalety Docker

Izolacja aplikacji. Szybkie wdrażanie. Spójne środowiska. Efektywne wykorzystanie zasobów. Elastyczność infrastruktury.

2

Obecne trendy

Konteneryzacja bezserwerowa. Kubernetes jako standard. Bezpieczeństwo kontenerów. DevSecOps. Ciągła integracja i wdrażanie.

3

Dalsze kroki

Eksploracja Kubernetes. Zagadnienia bezpieczeństwa. Monitoring i obserwowanie. Bazy danych w kontenerach. Orkiestracja w chmurze.