

Docker – dla programistów i nie tylko

Podstawowe informacje o kursie

Potrzebne narzędzia

- Komputer (najlepiej z dostępem administratora) aby zainstalować Dockera (Windows/Mac/Linux)
- Środowisko do edycji kodu (darmowe Visual Studio Code lub może być np. IntelliJ IDEA)

Co musisz wiedzieć aby zacząć

- Podstawowe komendy systemu operacyjnego Linux
- Znajomość obsługi terminala
- Wiedza z zakresu obsługi sieci komputerowych (IP, porty, DNS)
- Aplikacje webowe, REST API

Kod źródłowy

Repozytorium z materiałami: <https://github.com/pnowy/docker-course>

Komendy używane w kursie: <https://github.com/pnowy/docker-course/blob/main/komendy.md>

Dane kontaktowe

Kontakt bezpośrednio przez platformę Udemy

LinkedIn: <https://www.linkedin.com/in/przemeknowak/>

GitHub: <https://github.com/pnowy>

Email: kursdockera@przemeknowak.com

WWW: <https://przemeknowak.com/>

Docker – dla programistów i nie tylko

Agenda - omówienie

Linki

- <https://github.com/pnowy/kubernetes-course/blob/main/agenda.md>
- <https://github.com/pnowy/kubernetes-course/blob/main/komendy.md>
- <https://ossinsight.io/analyze/kubernetes/kubernetes>

Docker – dla programistów i nie tylko

Agenda - omówienie

Docker – dla programistów i nie tylko

Docker - wprowadzenie

Docker

- Docker - otwarte oprogramowanie służące jako „platforma dla programistów i administratorów do tworzenia, wdrażania i uruchamiania aplikacji rozproszonych” ([Wikipedia](#))

Infrastruktura - zmiany

Mainframe → PC (1990)



Maszyny fizyczne → Wirtualizacja (2000)

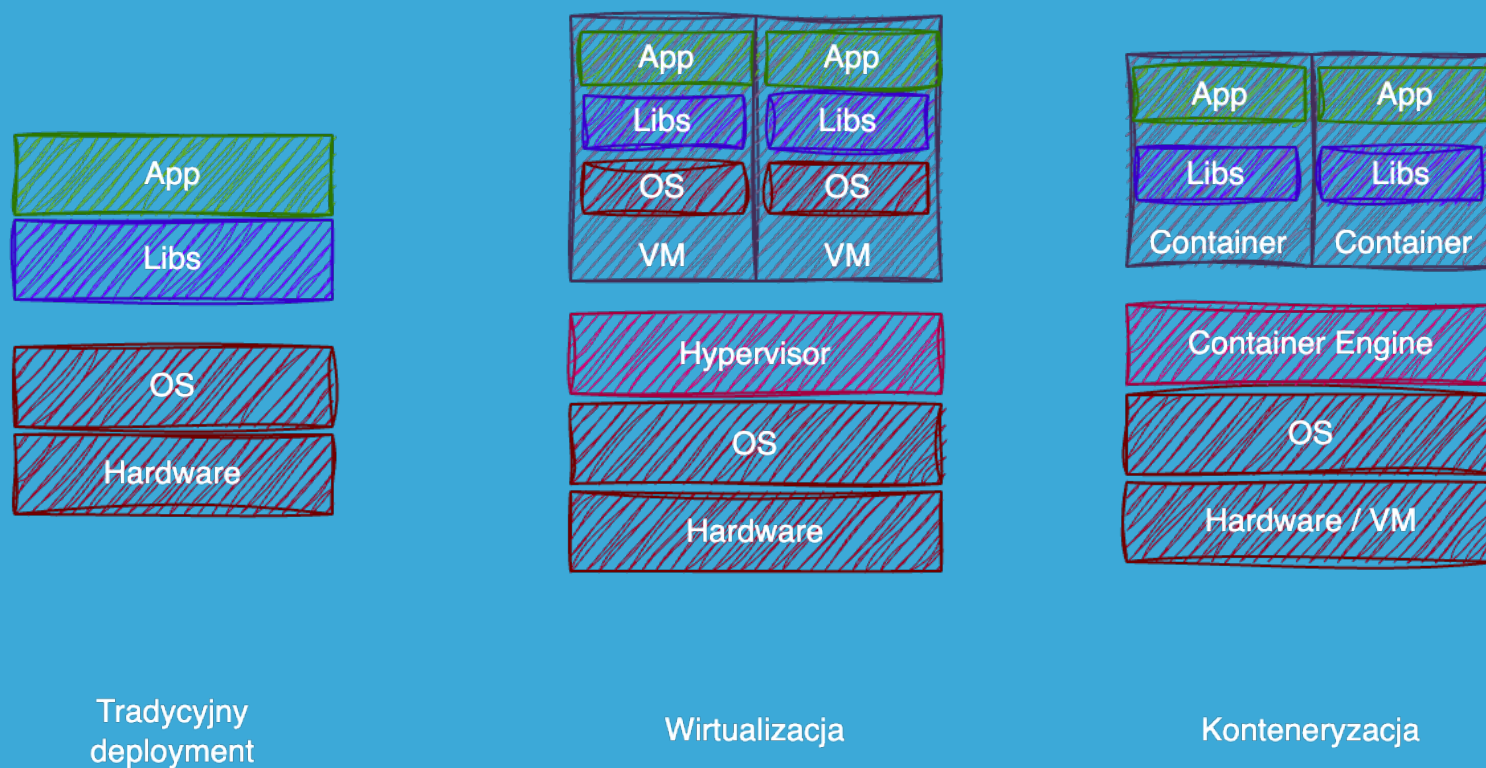


Datacenters → Cloud (2010)

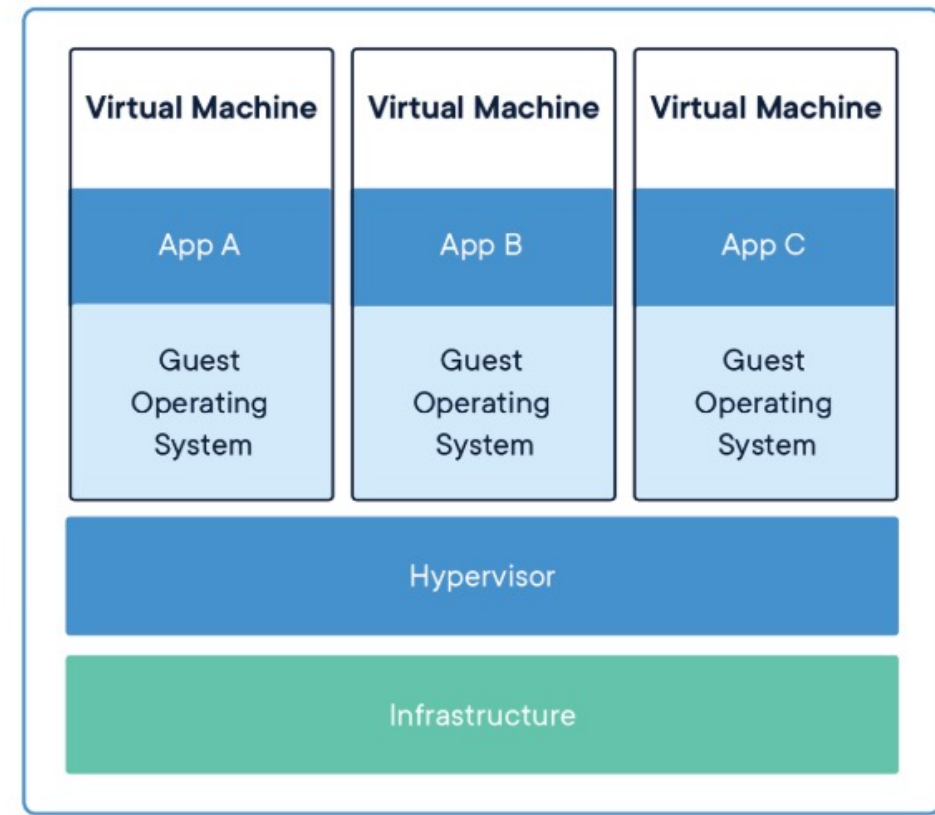
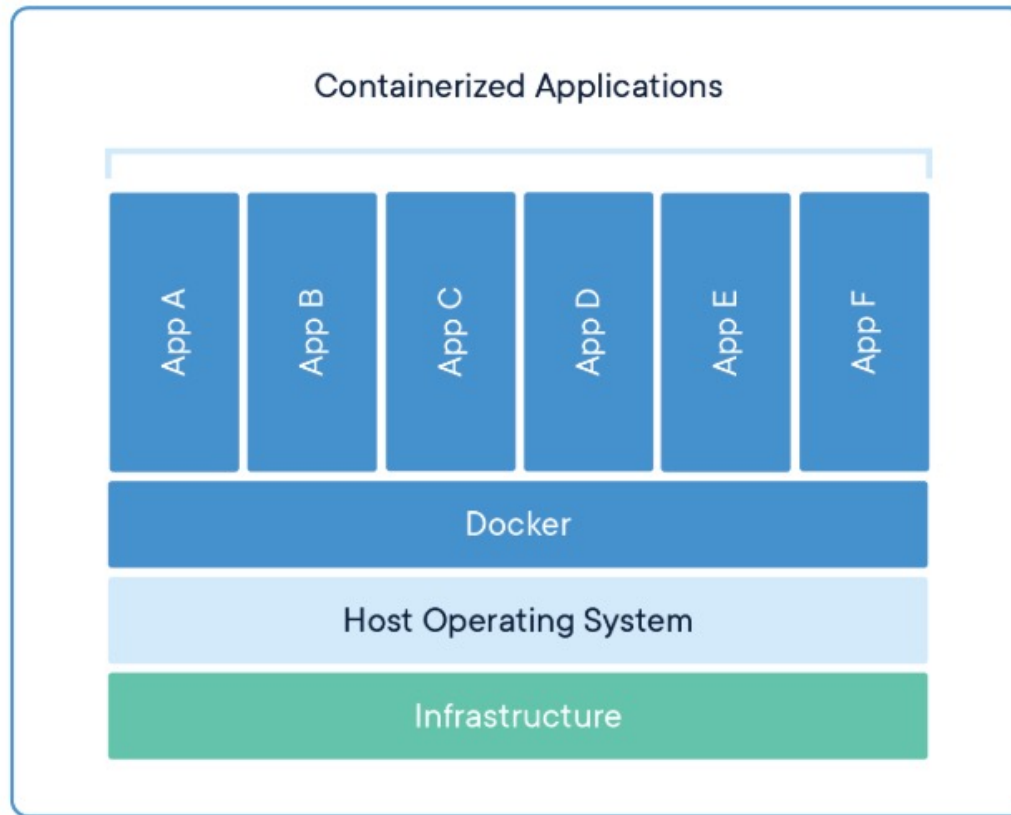


Host → Containers (2014)

Infrastruktura - zmiany



Infrastruktura - zmiany



Docker – dla programistów i nie tylko

Docker – rys historyczny

Docker-a – historia projektu

2010

- Powstanie dotCloud

Docker-a – historia projektu

2010

- Powstanie dotCloud

2013

- Pierwsza prezentacja Dockera
- Projekt Open-Source



Docker-a – historia projektu

2010

- Powstanie dotCloud

2013

- Pierwsza prezentacja Dockera
- Projekt Open-Source

2014

- Wydanie Docker 1.0



Docker-a – historia projektu

2010

- Powstanie dotCloud

2013

- Pierwsza prezentacja Dockera
- Projekt Open-Source

2014

- Wydanie Docker 1.0

2015

- Kubernetes > Docker Swarm



Docker-a – historia projektu

2010

- Powstanie dotCloud

2013

- Pierwsza prezentacja Dockera
- Projekt Open-Source

2014

- Wydanie Docker 1.0

2015

- Kubernetes > Docker Swarm

> 2019

- Rozwój Docker-a głównie jako narzędzie deweloperskiego
- Kubernetes do zarządzania kontenerami w dużej skali

Pierwsza prezentacja Docker-a



<https://www.youtube.com/watch?v=wW9CAH9nSLs>

Docker – aktualnie

- Docker jako narzędzie deweloperskie (Docker Desktop)
 - Obrazy budowane przy użyciu Dockera mogą być używane przez Kubernetes!
 - Docker przyczynił się do powstania OCI (Open Container Initiative) – standardu dla obrazów i kontenerów dzięki czemu aktualnie mamy wiele różnych narzędzi
- DockerHub – miejsce dla większości obrazów open-source
- Docker Scout – narzędzie do skanowania obrazów pod kątem bezpieczeństwa
- Docker Build Cloud – budowanie obrazów w chmurze

Docker – zalety

- Spójność pomiędzy platformami – rozwiązanie na problem „u mnie działa”
- Lepsze wykorzystanie infrastruktury (mniejsze koszty)
- Elastyczność, szybsze wrożenia
- Zwiększone bezpieczeństwo

Docker – wady

- Nowe narzędzie, którego trzeba się nauczyć
- Bezpieczeństwo przy nieumiejętnym wykorzystaniu narzędzia
- Nadal występujący narzut wydajnościowy

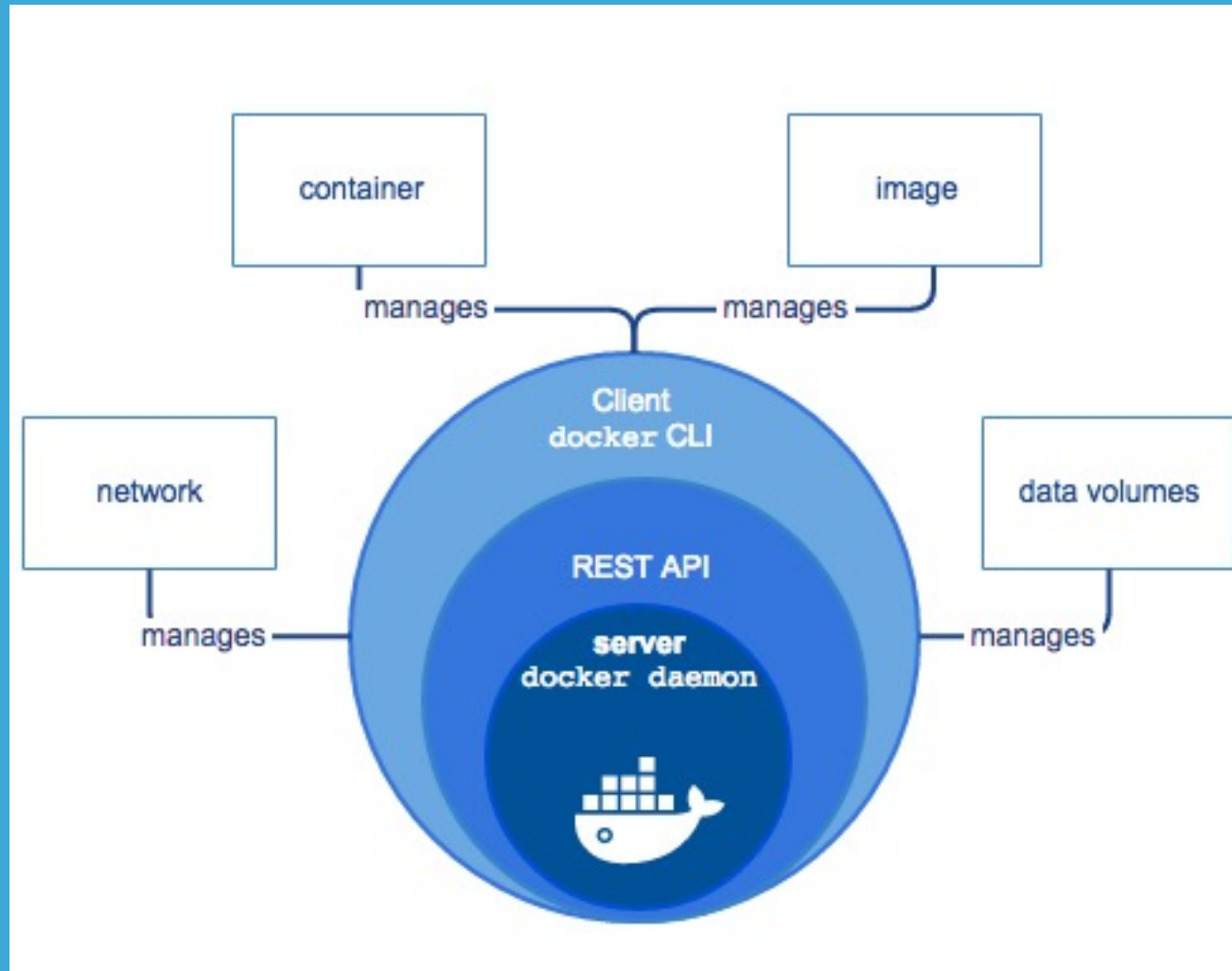
Docker – przyszłość

- Roadmap: <https://github.com/docker/roadmap>

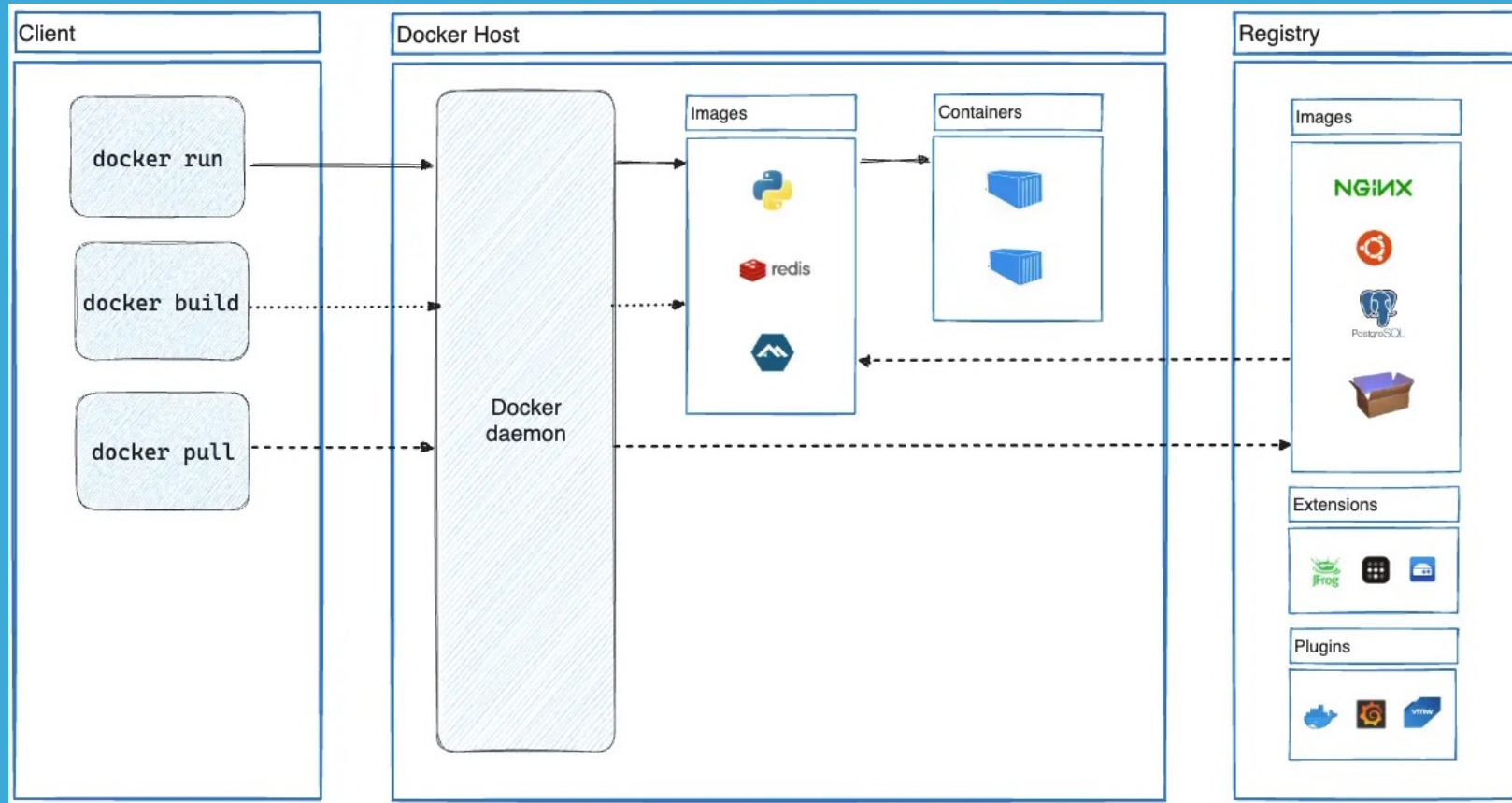
Docker – dla programistów i nie tylko

Docker – architektura

Docker – architektura



Docker – architektura



Docker Desktop - features



Docker Engine

Powerful container runtime

The Docker Engine powers your containerized applications with high performance and reliability. It provides the core technology for building and running containers, ensuring efficient and scalable operations.



Docker CLI

Flexible command-line interface

The Docker CLI offers a robust command-line tool for precise control over your containers. Execute complex commands, automate tasks, and integrate Docker seamlessly into your workflows.



Docker Compose

Streamlined multi-container management

Docker Compose simplifies the process of managing multi-container applications. Define and run complex setups with a single configuration file, making it easier to deploy and scale your applications.



Docker Build

Simplified container building

Docker Build is a powerful tool within Docker Desktop that simplifies the process of creating container images. It enables you to package and build your code to ship it anywhere while integrating seamlessly into your development pipeline.



Docker Kubernetes

Built-in container orchestration

Docker Kubernetes provides built-in Kubernetes support within Docker Desktop, allowing you to orchestrate and manage containers efficiently. Supporting both multi-node clusters and developer-selected versions, Docker Kubernetes simplifies deploying, scaling, testing, and managing containerized applications locally without needing an external cluster.



Volume Management

Effective data management

Docker Volumes provides a robust solution for managing and sharing container data. This feature allows you to easily and securely manage volumes for backup, sharing, or migration purposes, enhancing data management and portability.



Synchronized File Shares

Seamless data synchronization

Synchronized File Shares enable real-time sharing and synchronization of files between your host and containers. This feature ensures that file updates are instantly reflected on the host and container, improving collaboration and consistency.



Docker Debug

Advanced troubleshooting tools

Docker Debug provides comprehensive tools for diagnosing and resolving issues within your containers and images. This CLI command lets you create and work with slim containers that would otherwise be difficult to debug.



Hardened Docker Desktop

Enhanced container isolation

Hardened Docker Desktop includes advanced security features to safeguard your development environment. With enhanced container isolation, registry and image access management, and compliance with industry standard, you can confidently build and deploy secure applications.



VDI Support

Virtual desktop integration

VDI Support allows Docker to seamlessly integrate with virtual desktop infrastructure (VDI) environments. This feature ensures that Docker runs smoothly on virtualized desktops, providing a consistent experience regardless of where you access your containers.



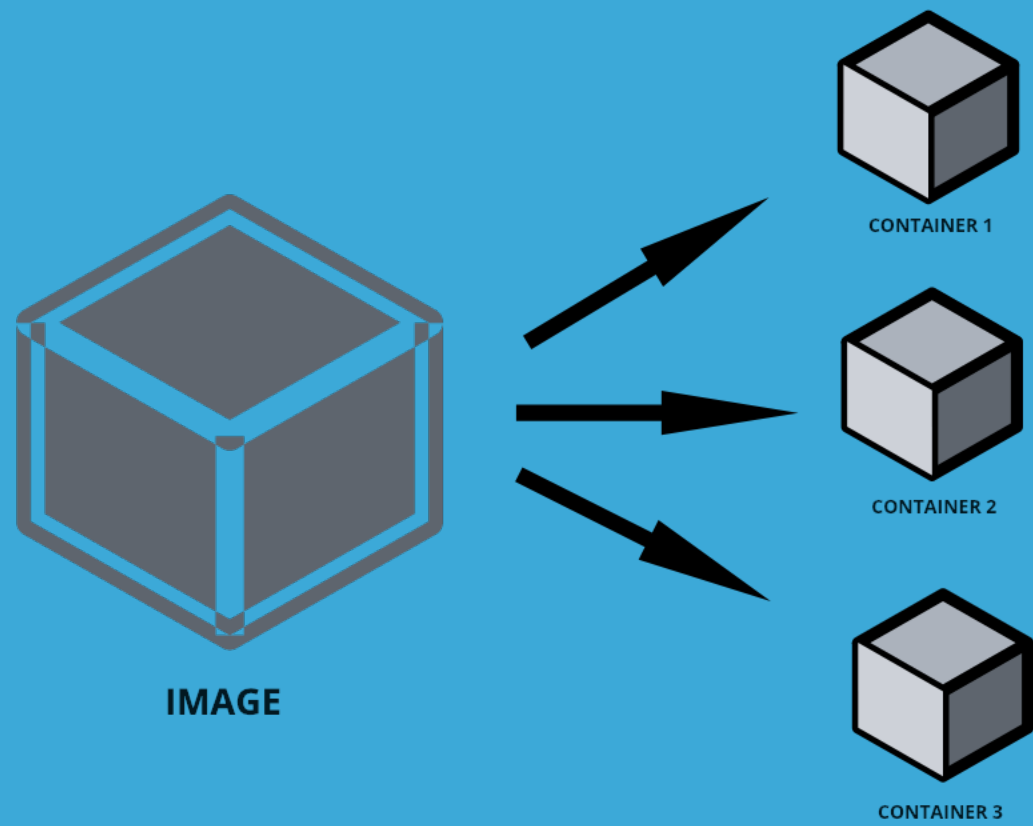
Docker Private Extensions Marketplace

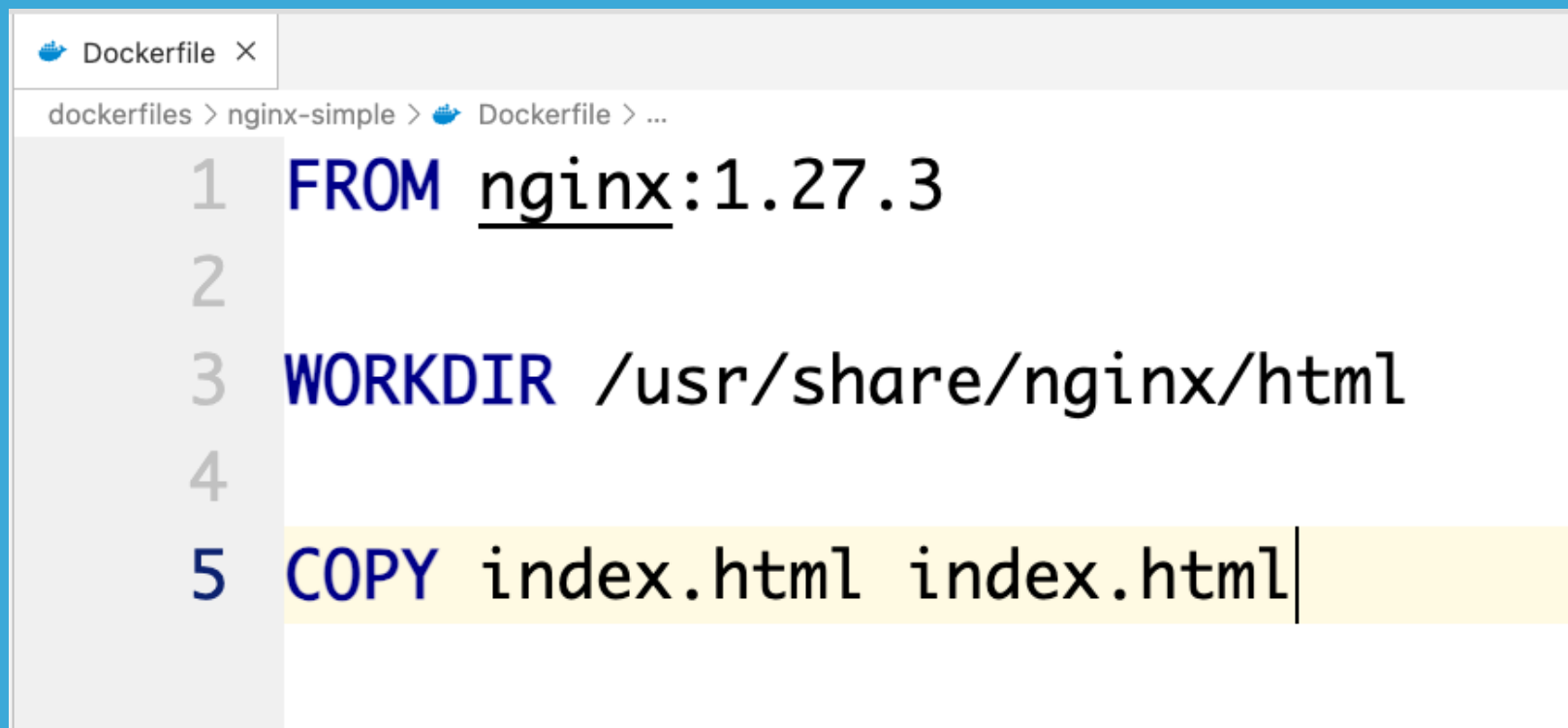
Custom extensions for your needs

The Docker Private Extensions Marketplace offers a curated selection of extensions tailored to your specific requirements. Customize and enhance your Docker environment with specialized tools and integrations available exclusively through the marketplace.

Docker – dla programistów i nie tylko

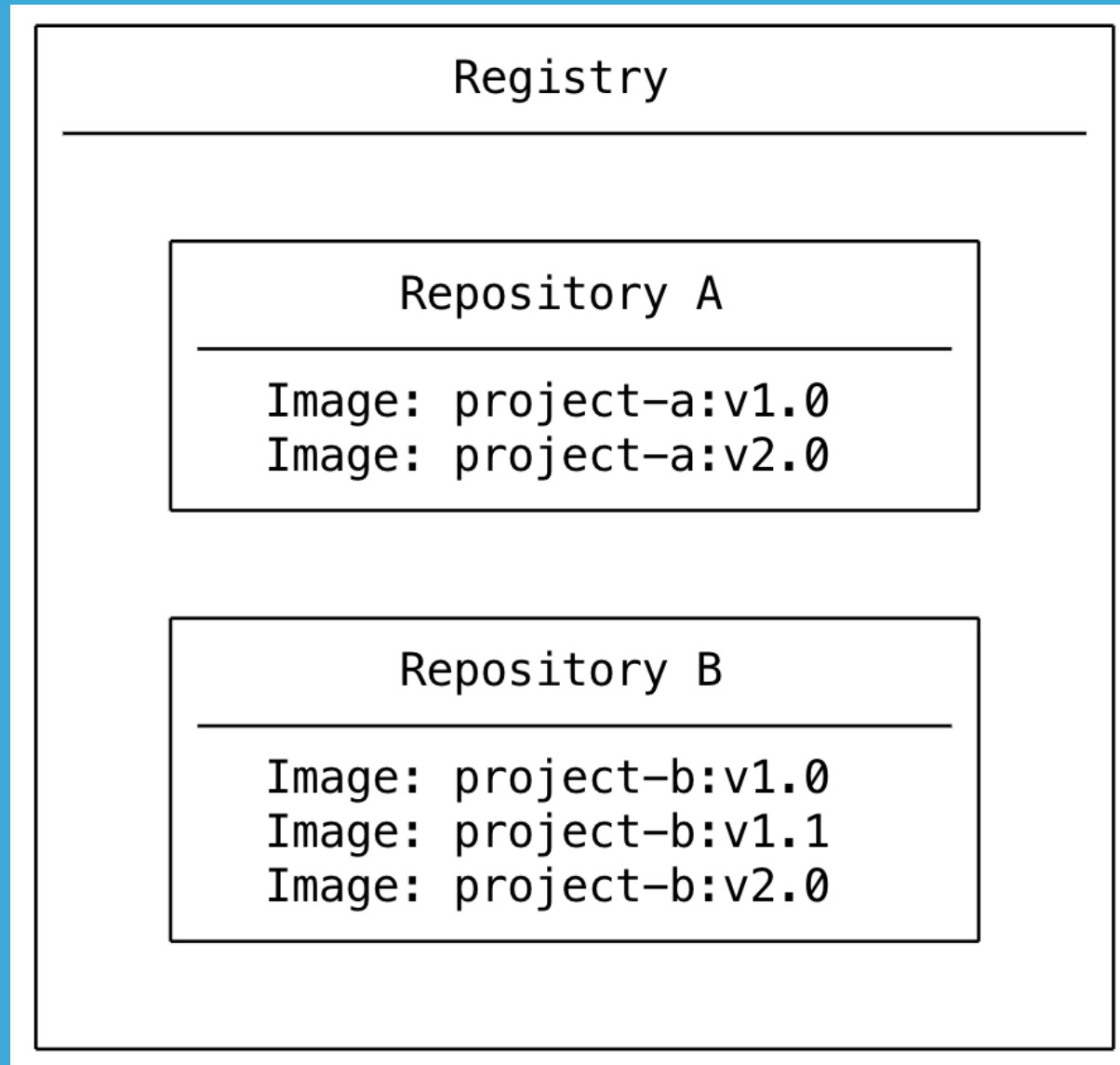
Koncepcje – obrazy, kontenery, rejestr obrazów





The image shows a code editor window titled "Dockerfile" with a close button. The breadcrumb navigation shows the path: "dockerfiles > nginx-simple > Dockerfile > ...". The editor contains five lines of Dockerfile instructions, numbered 1 to 5 on the left. Line 1: "FROM nginx:1.27.3". Line 2: (empty). Line 3: "WORKDIR /usr/share/nginx/html". Line 4: (empty). Line 5: "COPY index.html index.html|". The fifth line is highlighted with a yellow background.

```
1 FROM nginx:1.27.3
2
3 WORKDIR /usr/share/nginx/html
4
5 COPY index.html index.html|
```



Docker – dla programistów i nie tylko

Docker – instalacja Windows

Docker – instalacja Windows

- <https://docs.docker.com/desktop/setup/install/windows-install/>
- <https://docs.docker.com/desktop/features/wsl/>

Docker – dla programistów i nie tylko

Docker – instalacja MacOS

Docker – instalacja MacOS

- <https://docs.docker.com/desktop/setup/install/mac-install/>

Docker – dla programistów i nie tylko

Docker – instalacja Linux

Docker – instalacja Linux

- <https://docs.docker.com/engine/install/ubuntu/>
- <https://docs.docker.com/engine/install/linux-postinstall/>
- <https://docs.docker.com/desktop/setup/install/linux/ubuntu/>

Docker – dla programistów i nie tylko

Docker wprowadzenie – podsumowanie rozdziału

Docker – dla programistów i nie tylko

Komendy - wprowadzenie

Komendy – wprowadzenie

- <https://docs.docker.com/reference/cli/docker/>
- https://docs.docker.com/get-started/docker_cheatsheet.pdf

Docker – dla programistów i nie tylko

Uruchamianie kontenerów (część pierwsza)

Uruchamianie kontenerów

- Obraz i kontener - przypomnienie
- Uruchomienie, zatrzymanie i usunięcie kontenera (nginx)

Docker – dla programistów i nie tylko

Uruchamianie kontenerów (część druga)

Uruchamianie kontenerów

- Uruchomienie, zatrzymanie i usunięcie kontenera (nginx, grafana)
- Sprawdzenie stanu kontenera (logi, procesy)

Uruchamianie kontenerów

`docker container run / docker run`

`--publish / -p` (mapowanie portów)

`--detach / -d` (uruchomienie kontenera w tle)

`--name` (nazwa kontenera)

```
docker run --publish 8080:80 --detach --name=mynginx nginx
```

```
docker container run -p 8080:80 -d --name=mynginx nginx
```

Uruchamianie kontenerów

```
docker container ls / docker ps
```

--all / -a (wyświetlenie wszystkich kontenerów, również nieaktywnych)

```
docker container start <container-id/name>
```

```
docker container stop <container-id/name>
```

Uruchamianie kontenerów

`docker container top <container-id/nazwa>`

`docker container logs /docker logs <container-id/nazwa>`

`--follow / -f` (śledzenie logów na bieżąco)

Uruchamianie kontenerów

`docker container rm` / `docker rm` / `docker container remove`

`--force` / `-f` – wymuszenie usunięcia kontenera

Docker – dla programistów i nie tylko

Proces uruchamiania kontenera

Proces uruchamiania kontenera

```
docker run --publish 8080:80 --detach --name=mynginx nginx
```

Proces uruchamiania kontenera

```
nginx == nginx:latest
```

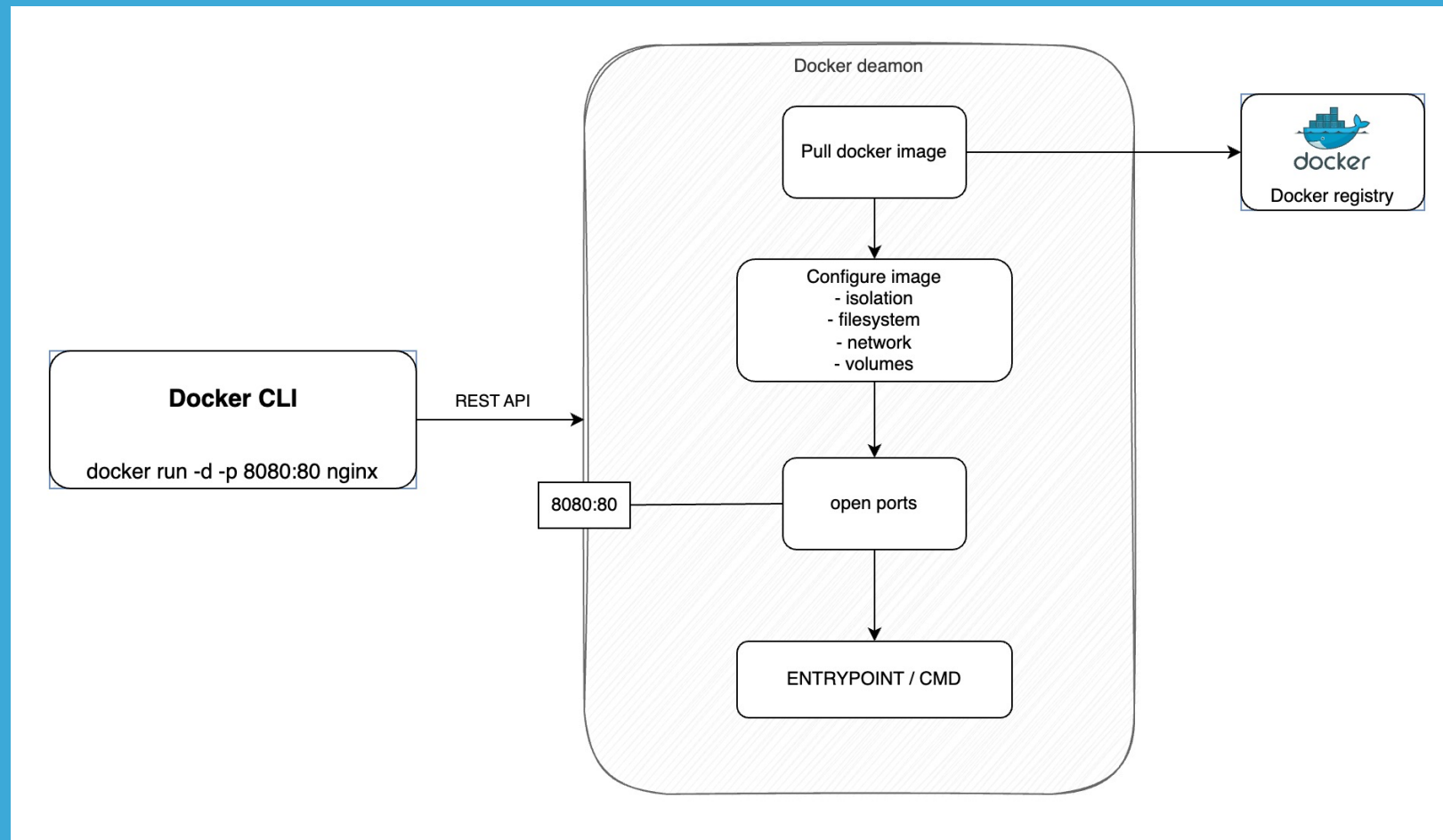
```
nginx:1.27.3
```

```
nginx:1.27
```

```
nginx:1.27.3-alpine
```

```
...
```

Proces uruchamiania kontenera



Proces uruchamiania kontenera

- Zakończenie pracy kontenera
 - Proces w nim uruchomiony zakończy pracę
 - Użytkownik ręcznie zatrzyma kontener (docker stop lub docker kill)
 - Opcja --rm dla polecenia docker run, która powoduje automatyczne usunięcie kontenera po jego zakończeniu

```
docker run --rm -p 8080:80 --name=auto_clean_nginx nginx
```

Proces uruchamiania kontenera

- Komenda np. `docker container run -p 8080:80 -d nginx`
- Docker CLI -> REST API -> docker daemon
- Pobranie obrazu z registry
- Utworzenie kontenera:
 - Stworzenie izolowanego środowiska
 - Konfiguracja warstwy zapisu danych (filesystem)
 - Konfiguracja sieci
 - Montowanie wolumenów
- Mapowanie portów
- Uruchomienie domyślnego procesu kontenera

Docker – dla programistów i nie tylko

Uruchamianie kontenerów (część 3)

Uruchamianie kontenerów (część 3)

- Grafana: <https://grafana.com/>
- Grafana – obraz: <https://hub.docker.com/r/grafana/grafana>
- httpbin: <https://httpbin.org/>, <https://github.com/postmanlabs/httpbin>
- httpbin - obraz: <https://hub.docker.com/r/kong/httpbin>

Docker – dla programistów i nie tylko

Monitorowanie kontenerów

Monitorowanie kontenerów

- Procesy w kontenerze
- Jak sprawdzić konfigurację kontenera
- Statystyki wydajności kontenera

Monitorowanie kontenerów

- `docker container top` - aktualnie uruchomione procesy w kontenerze
- `docker container inspect` - szczegółowe informacje na temat kontenera
- `docker container stats` - zużycie zasobów przez kontener w czasie rzeczywistym

Docker – dla programistów i nie tylko

Kontenery - terminal

Kontenery - terminal

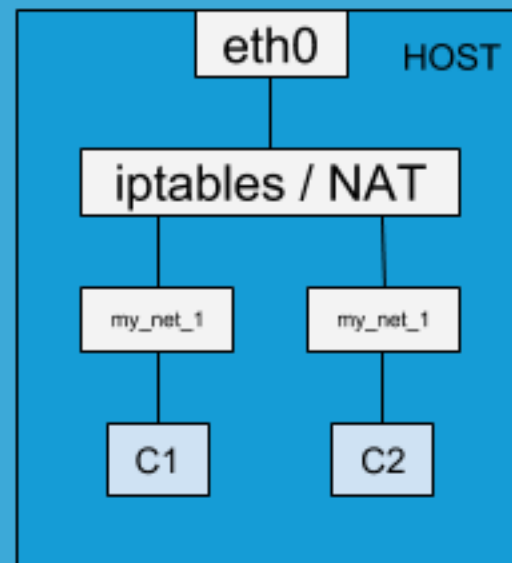
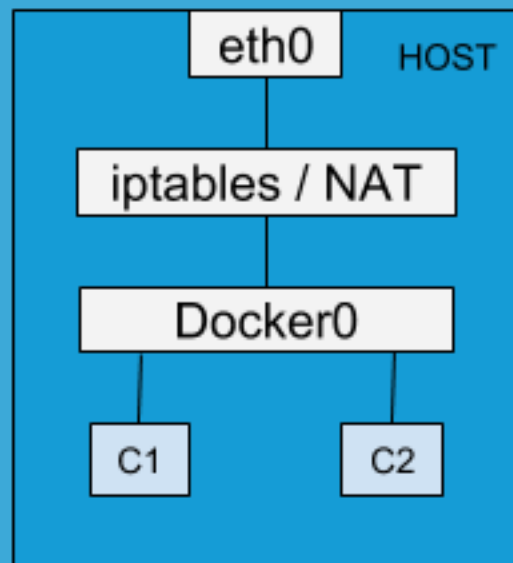
- SSH i kontenery – jak dostać się do terminala i uruchamiać komendy bezpośrednio na kontenerze
- Do czego służą opcje `-i` oraz `-t` przy uruchamianiu kontenera

Docker – dla programistów i nie tylko

Docker - sieci

bridge (domyślny rodzaj sieci)

- Każdy kontener podłączany jest domyślnie do sieci bridge
- Każdy kontener w sieci może komunikować się z innym kontenerem w tej sieci bez konieczności używania `--publish (-p)`
- Ruch sieciowy kontenera podlega NAT-owaniu (translacja adresów sieciowych)



Docker - sieci

- Dobrą praktyką jest tworzyć nową sieć dla grupy aplikacji
- Możliwość tworzenia własnych sieci (np. `my_api_network`, `my_backend_services`, etc.)
- Możliwość podłączenia kontenera do więcej niż jednej wirtualnej sieci
- Rozszerzenia służące do obsługi nowych funkcjonalności

host

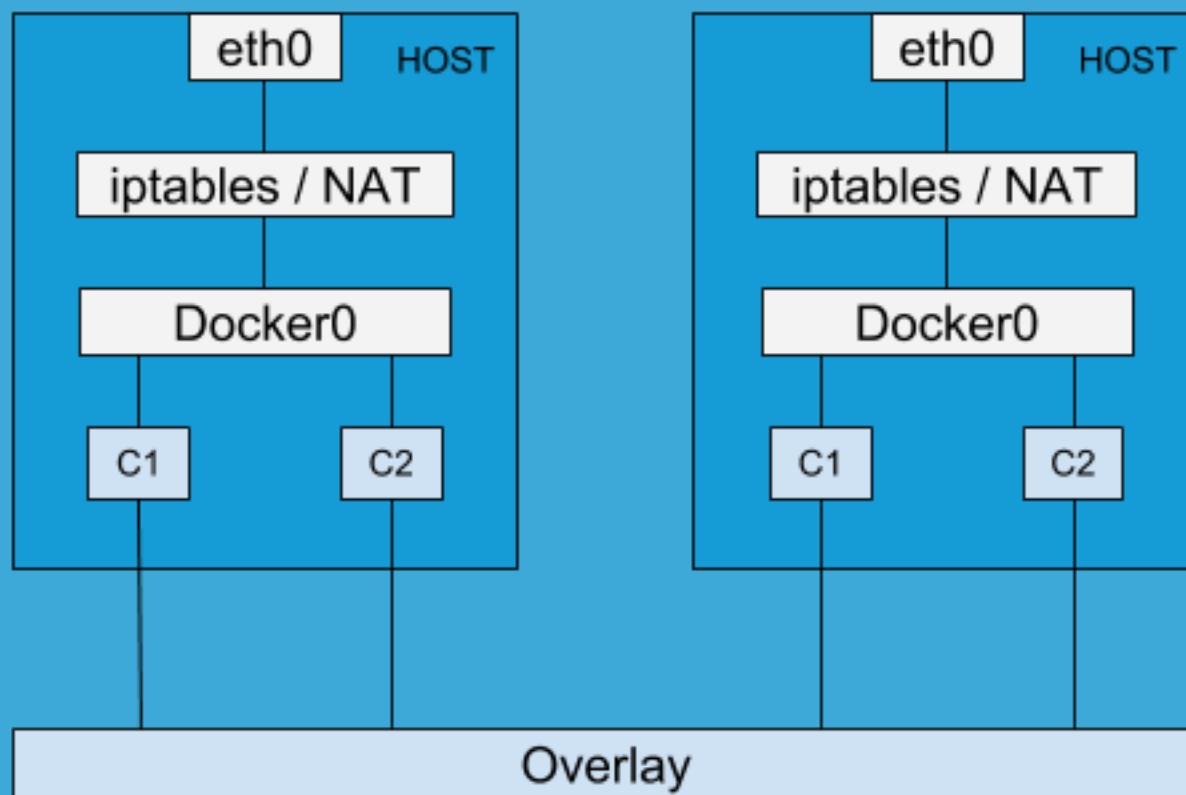
- Kontener działa w sieci host-a
- Wszystkie interfejsy sieciowe hosta są dostępne dla kontenera
- Lepsza wydajność
- Kontener dostępny poprzez IP hosta
- Mniejsze bezpieczeństwo

none

- Brak sieci
- Gdy nie potrzebujemy obsługi sieciowej

overlay

- Sieć do obsługi wielu host-ów



ipvlan

- Daje użytkownikom pełną kontrolę nad adresacją zarówno IPv4 jak i IPv6
- Pozwala na współdzielenie adresu MAC z hostem, ale zapewni własny adres IP dla każdego kontenera
- Przydatne w środowiskach gdzie administratorzy ograniczają liczbę adresów MAC
- IPvlan to nowoczesne podejście do sprawdzonych technik wirtualizacji sieci. Implementacje w systemie Linux są wyjątkowo lekkie, ponieważ zamiast tradycyjnego mostu Linuksa do izolacji (bridge), interfejsy IPvlan są bezpośrednio przypisywane do interfejsu Ethernet lub jego podinterfejsu.

macvlan

- Pozwala przypisać adres MAC do kontenera
- Pozwala symulować fizyczne sprzęty w sieci
- Często używana ze starszymi rozwiązaniami które oczekują bezpośredniego połączenia z siecią fizyczną (zamiast stosu sieciowego Docker-a)

Linki

- [Sieci – oficjalna dokumentacja](#)

Docker – dla programistów i nie tylko

Sieci - zarządzanie

Sieci zarządzanie

- Listing sieci / inspekcja sieci
- Tworzenie sieci
- Podłączanie kontenerów do sieci

Docker – dla programistów i nie tylko

Sieci - DNS

Sieci - DNS

- <https://docs.docker.com/engine/network/drivers/bridge/#differences-between-user-defined-bridges-and-the-default-bridge>

Docker – dla programistów i nie tylko

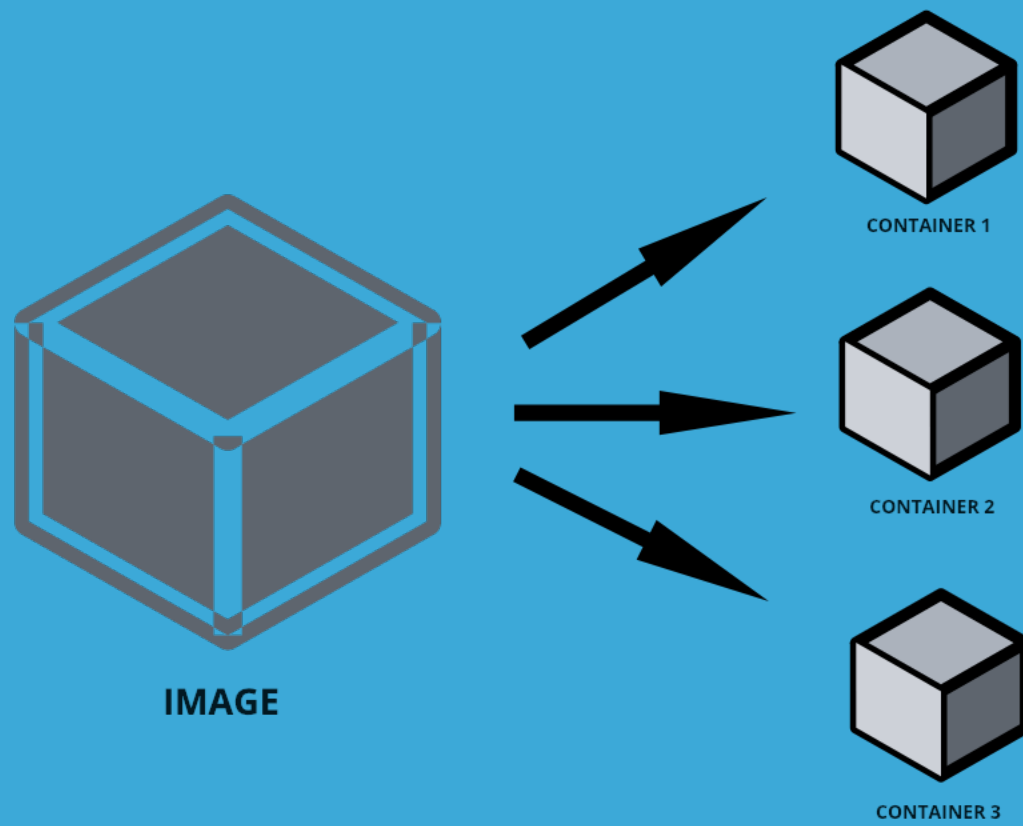
Obrazy - wprowadzenie

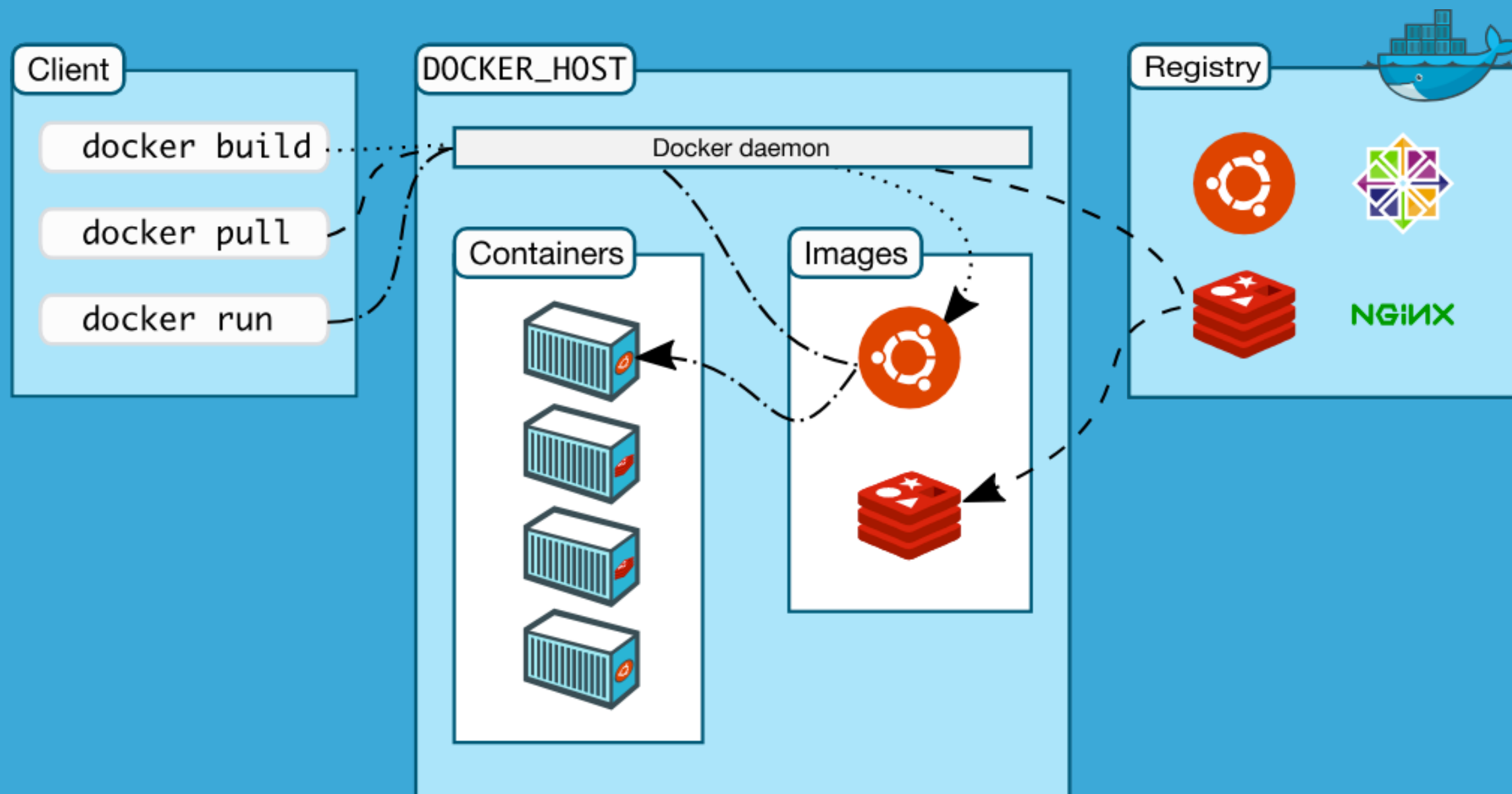
Obrazy - wprowadzenie

- Czy jest obraz i do czego służą obrazy
- Docker Hub (hub.docker.com)
- Obrazy – wersjonowanie (tagowanie)

Czym jest obraz

- Zawiera wszystkie binaria i niezbędne zależności naszej aplikacji
- Metadane o obrazie oraz sposobie jego uruchomienia
- Nie jest to kompletny system operacyjny (brak jądra oraz modułów jądra) – źródłem jest w tym wypadku HOST





Linki

- <https://github.com/docker-library/official-images>

Docker – dla programistów i nie tylko

Obrazy - warstwy

Obrazy - warstwy

- Czym jest warstwa (layer)
- docker image history
- docker image inspect

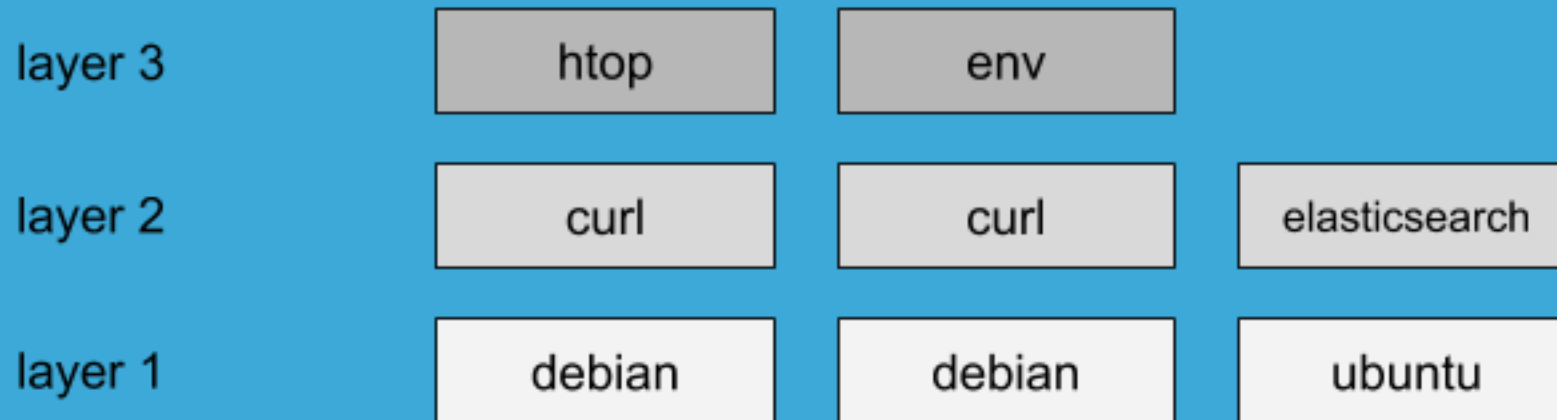
Czym jest warstwa (ang. layer)

- Pliki generowane w związku z wykonaniem konkretnej komendy
- Komendy – Dockerfile lub docker image commit
- Mogą być reużywane przez wiele obrazów (cache)

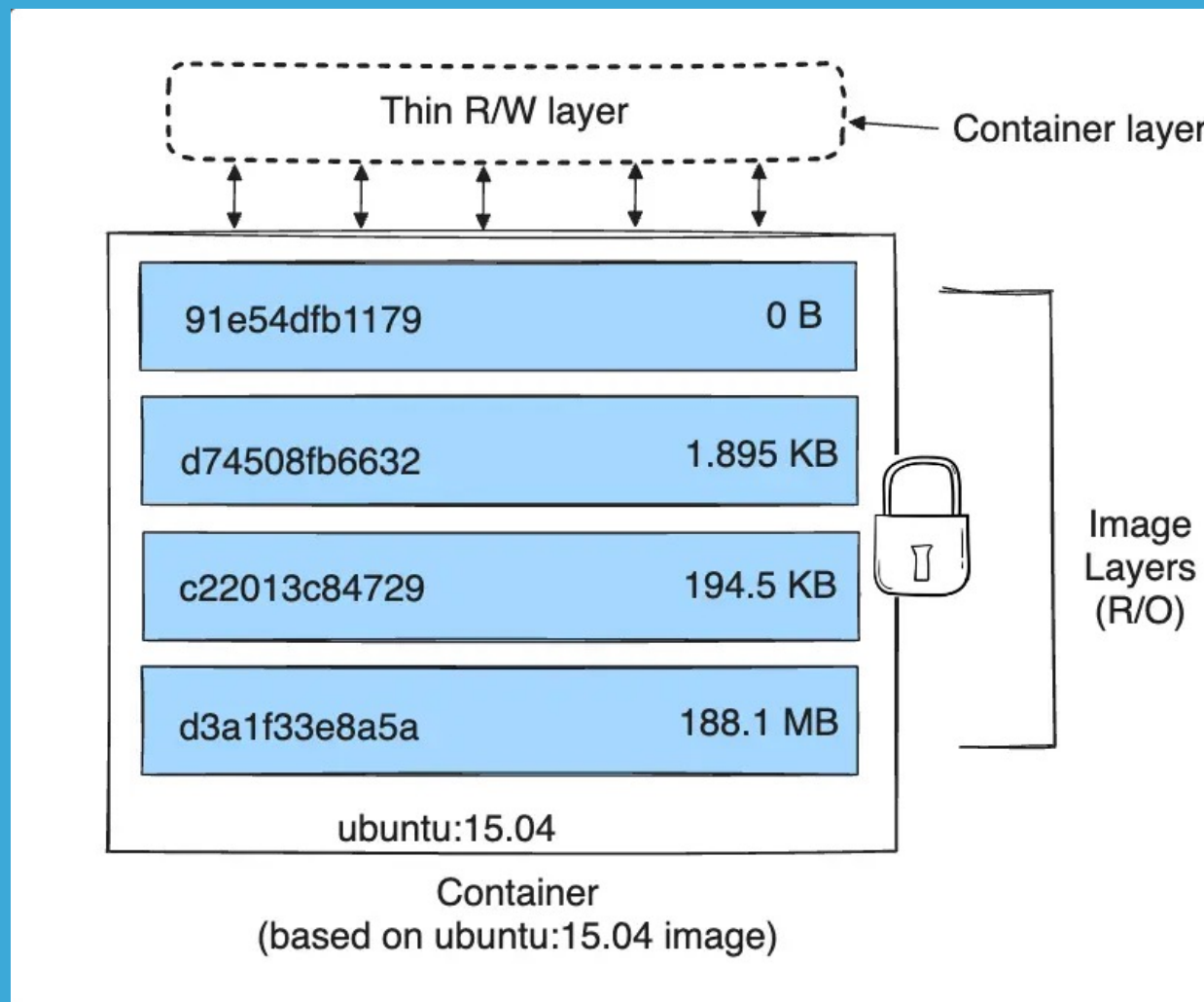
Warstwy - zalety

- wykorzystanie warstw z lokalnej pamięci - Docker nie pobiera tej samej warstwy ponownie, oszczędza to transfer danych i przyspiesza pobieranie
- równoczesne pobieranie warstw - przyspiesza cały proces
- współdzielenie warstw między obrazami - jeśli dwa obrazy korzystają z tej samej warstwy bazowej nie trzeba jej pobierać drugi raz co daje oszczędność miejsca i pozwala cache-ować obrazy

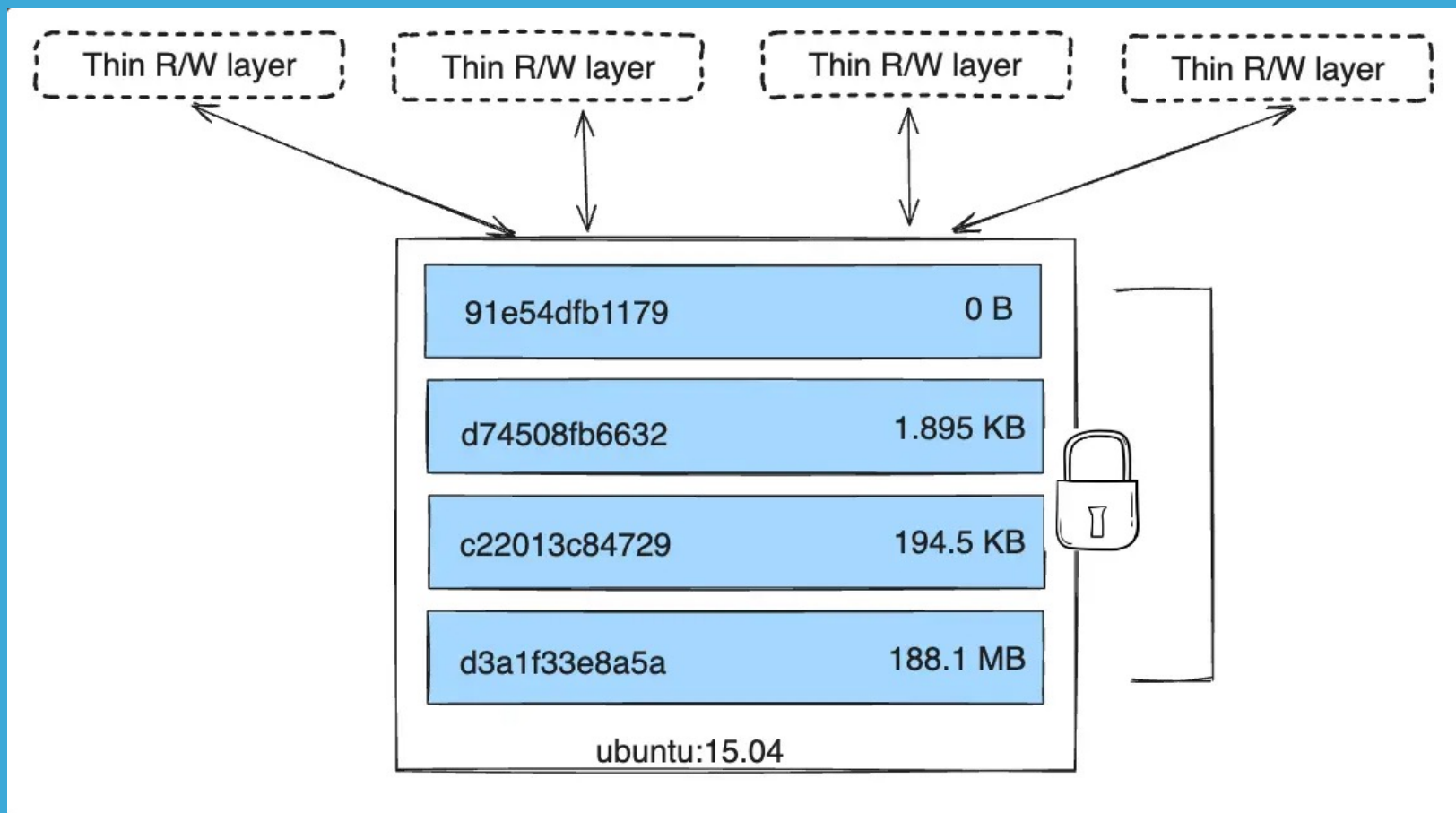
Obrazy - warstwy



Obrazy - warstwy



Obrazy - warstwy



Linki

- Docker – oficjalna dokumentacja (obrazy i warstwy)

Docker – dla programistów i nie tylko

Obrazy - tagowanie

Obrazy - tagowanie

- Tagowanie obrazów
- Zapis obrazów do Docker Hub
- `docker container commit`

Docker – dla programistów i nie tylko

Obrazy – Dockerfile (część teoretyczna)

Linki

- [Docker – oficjalna dokumentacja Dockerfile](#)

Docker – dla programistów i nie tylko

Obrazy – Dockerfile (część praktyczna)

Docker – dla programistów i nie tylko

Własny obraz - nginx

Docker – dla programistów i nie tylko

Obrazy – dystrybucje przystosowane do kontenerów

Alpine Linux

- Zalety
 - Mały rozmiar (~ 10MB)
 - Bezpieczeństwo
 - Przystosowany do małych systemów
- Wady
 - Kompatybilność (musl libc zamiast GNU C Library glibc)

Link do artykułu: <https://martinheinz.dev/blog/92>

Debian Slim

- Zalety
 - Większa kompatybilność niż Alpine (obsługa glibc)
 - Stabilność i długoterminowe wsparcie (LTS)
 - Bogate repozytoria. pakietów
- Wady
 - Większy rozmiar (~ 130MB)

Ubuntu

- Zalety
 - Duża społeczność użytkowników i dokumentacja
 - Regularne aktualizacje i wsparcie LTS
- Wady
 - Większy rozmiar (~ 130MB)

Google Distroless

- Zalety
 - Minimalna ilość pakietów – większe bezpieczeństwo
 - Optymalizacja pod konkretne języki i technologie (np. distroless/nodejs, distroless/java)
- Wady
 - Bardzo mały rozmiar wersji podstawowej (< 10MB)
 - Brak powłoki

Red Hat Universal Base Image (UBI)

- Zalety
 - Długoterminowe wsparcie
 - Certyfikowany do użytku w środowisku korporacyjnym
- Wady
 - Mniej popularny w środowiskach open-source
 - Większy rozmiar (~ 140MB dla wersji minimal)

Wolfi

- Zalety
 - Zaprojektowany pod kątem bezpieczeństwa
 - Certyfikowany do użytku w środowisku korporacyjnym
- Wady
 - Mniej popularny w środowiskach open-source
 - Większy rozmiar (~ 140MB dla wersji minimal)

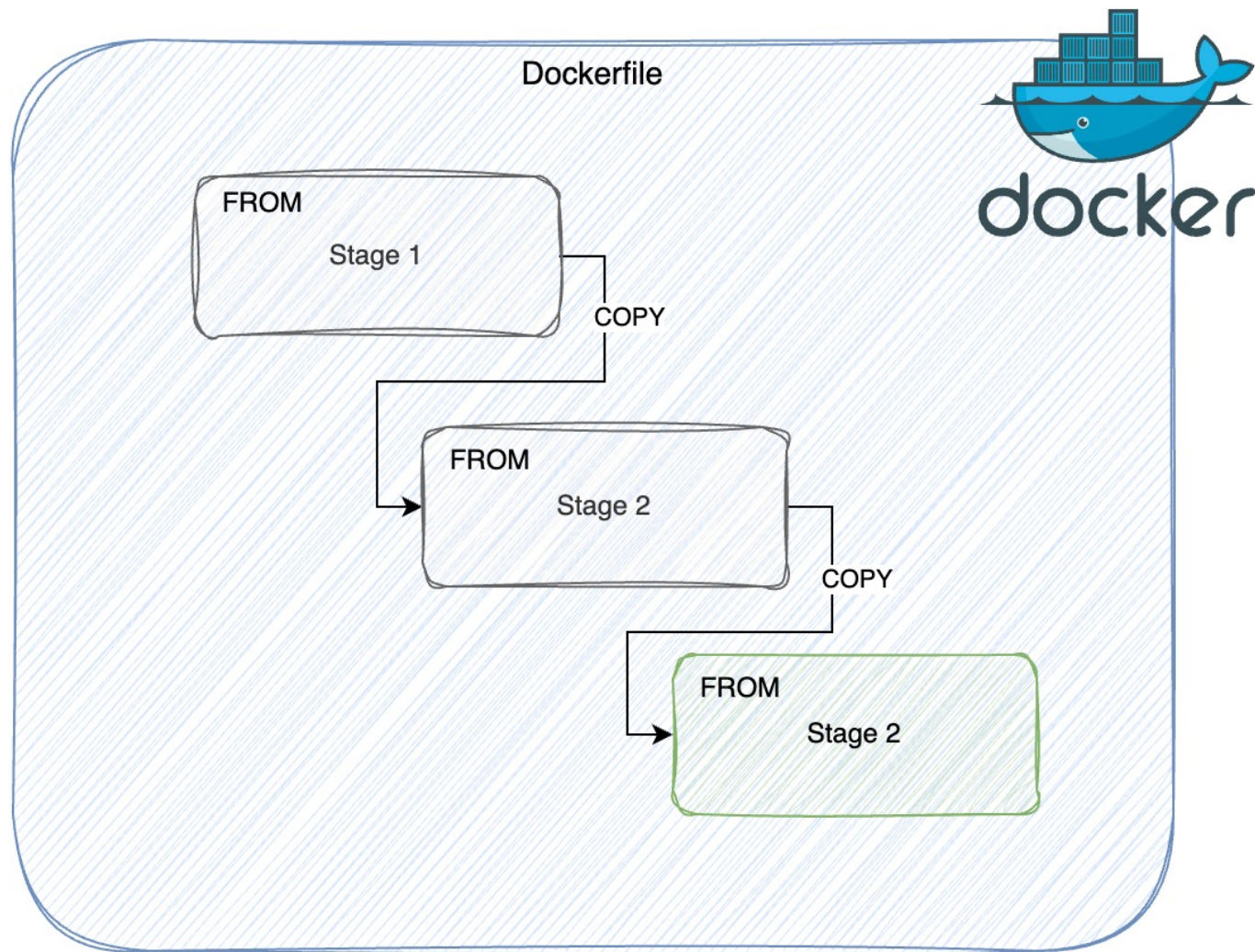
Docker – dla programistów i nie tylko

Multistage build

Multistage build

- [Oficjalna dokumentacja](#)

Multistage build



Docker – dla programistów i nie tylko

Silnik budowania obrazów BuildKit

BuildKit - zalety

- Współbieżność
- Optymalizacja cache
- Izolacja operacji na workerach
- Multiplatform (--platform)
- Frontend(s)
- Rootless

BuildKit

- [Project GitHub](#)
- [Oficjalna dokumentacja Dockera](#)

Docker – dla programistów i nie tylko

Własny – obraz node

Docker – dla programistów i nie tylko

Obrazy - porządki

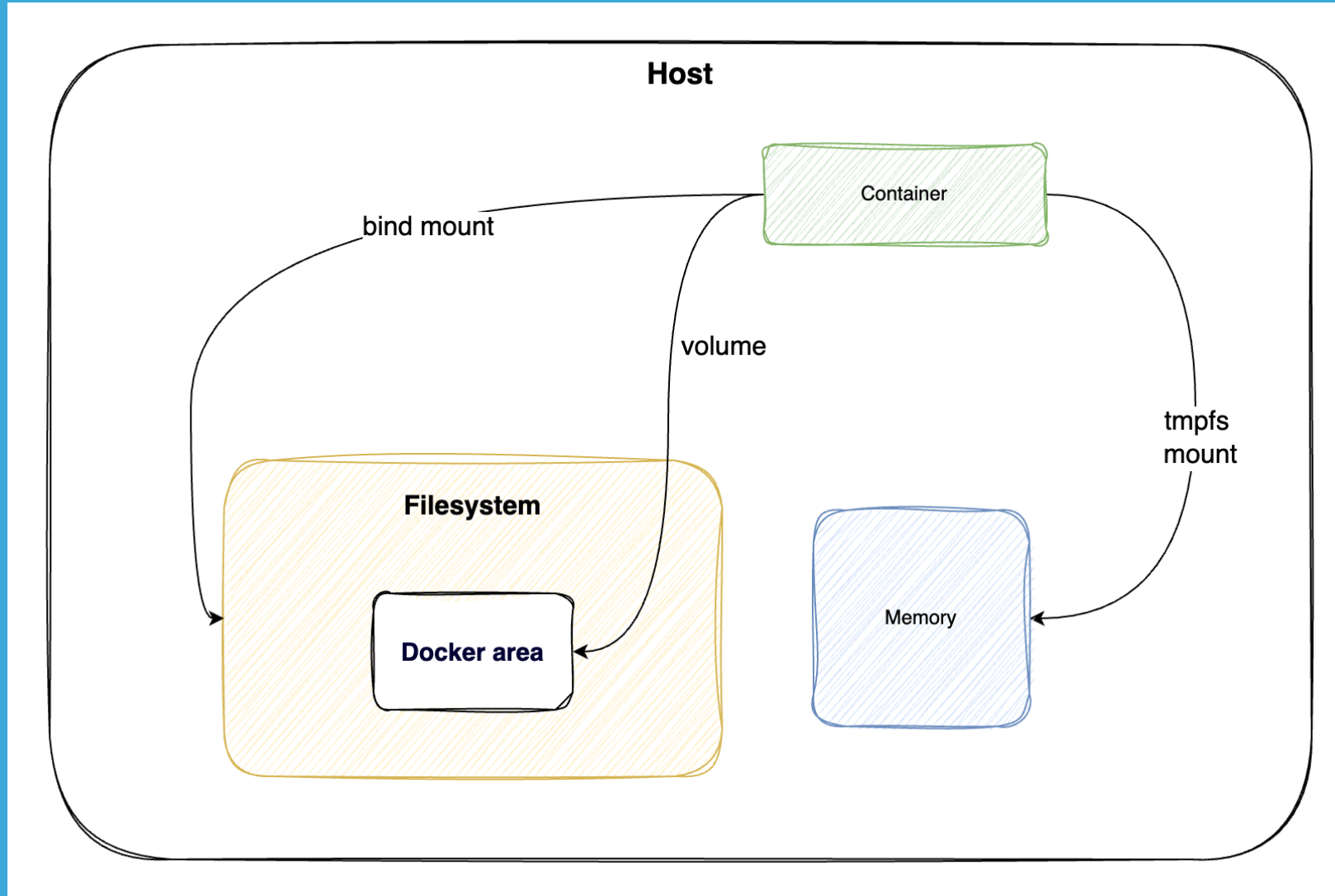
Docker – dla programistów i nie tylko

Przechowywanie danych w kontenerach - wprowadzenie

Przechowywanie danych w kontenerach

CATTLE NOT PETS

Przechowywanie danych w kontenerach



Volumes

Zalety:

- Trwałość – dane przechowywane w wolumenie nie znikają po usunięciu kontenera
- Łatwość zarządzania – można nimi zarządzać za pomocą poleceń Docker CLI (*docker volume create, docker volume ls, docker volume rm*)
- Bezpieczeństwo i izolacja – Docker sam zarządza ich lokalizacją i dostępnością
- Optymalizacja wydajności
- Współdzielenie danych między kontenerami – mogą być montowane do wielu kontenerów jednocześnie

Volumes

tworzenie nowego wolumenu

```
docker volume create <volume-name>
```

uruchomienie kontenera z podłączonym wolumenem

można użyć --mount lub --volume / -v

```
docker run --mount type=volume,src=<volume-name>,dst=<mount-path>
```

```
docker run --volume <volume-name>:<mount-path>
```

Anonymous Volumes

Zalety:

- Automatycznie zarządzane
- Uniknięcie problemów z efemerycznością kontenera

Anonymous Volumes

uruchomienie kontenera z anonimowym wolumenem
`docker run --volume /app/data <volume-name>`

Bind mounts

Zalety :

- Pełna kontrola nad lokalizacją danych
- Łatwy dostęp do plików z poziomu hosta

Wady :

- Pliki przechowywane bezpośrednio na hoście
- Ryzyko przypadkowego usunięcia

Bind mounts

uruchomienie kontenera ze zmapowaną ścieżką
można użyć --mount lub --volume / -v

```
docker run --mount type=bind,src=<host-path>,dst=<container-path>  
docker run --volume <host-path>:<container-path>
```

Tmpfs mounts

- Dane tymczasowe

uruchomienie kontenera z podłączonym tmpfs mount

można użyć --mount lub --volume / -v

```
docker run --mount type=tmpfs,dst=<mount-path>
```

```
docker run --tmpfs <mount-path>
```

Tmpfs mounts

Typ	Wydajność	Zarządzanie przez Dockera	Zastosowanie
Volumes	Wysoka	Tak	Produkcyjne przechowywanie danych, lokalny development
Anonymous volumes	Wysoka	Tak (automatycznie)	Tymczasowe dane pomiędzy restartami kontenera
Bind mounts	Średnia (zależy od system plików)	Nie	Deweloperskie środowiska, dostęp do plików hosta
tmpfs mounts	Bardzo wysoka	Tak	Tymczasowe dane, cache, sesje aplikacji

Docker – dla programistów i nie tylko

Volumes – część praktyczna (1)

Docker – dla programistów i nie tylko

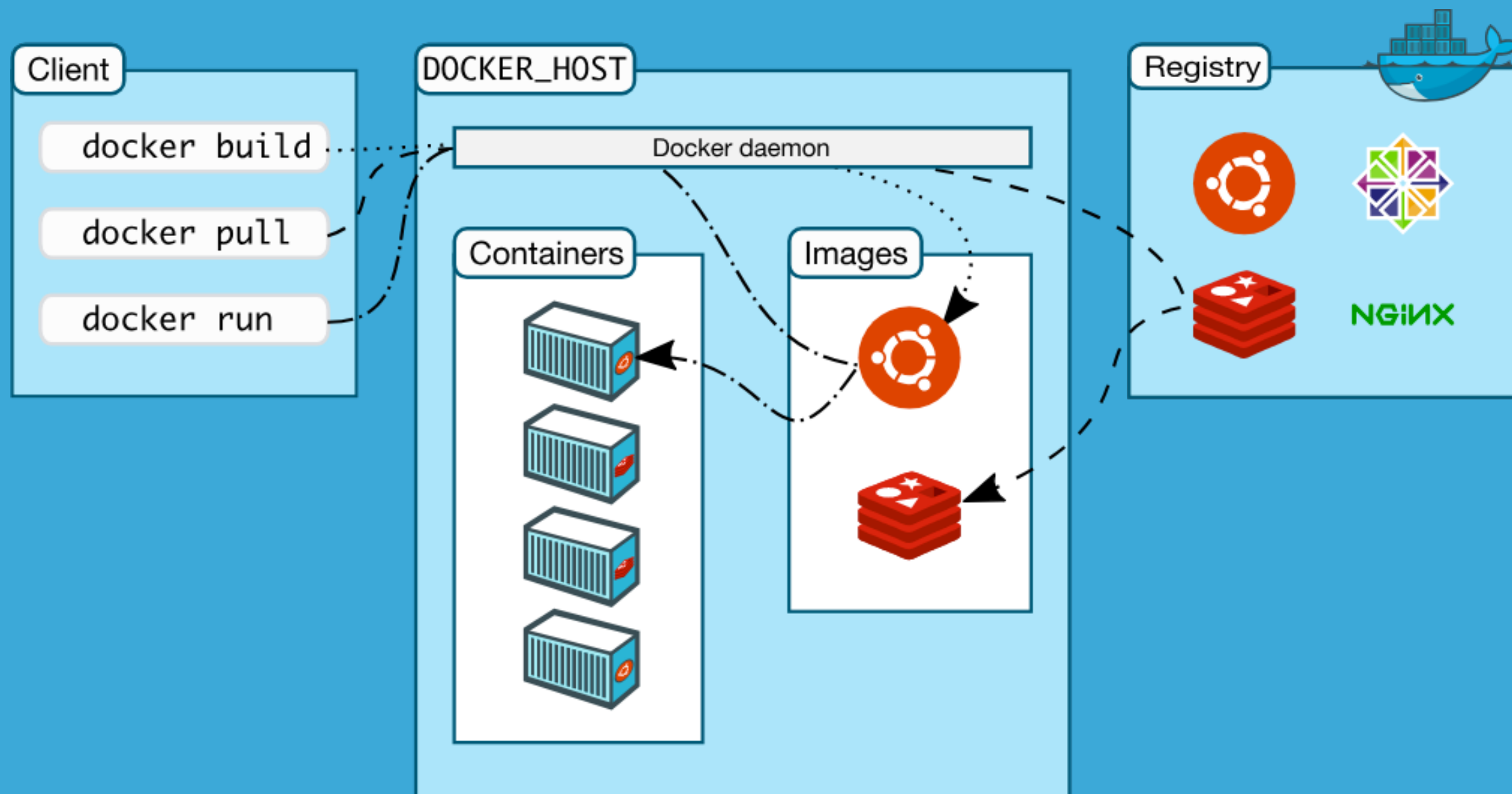
Volumes – część praktyczna (2)

Docker – dla programistów i nie tylko

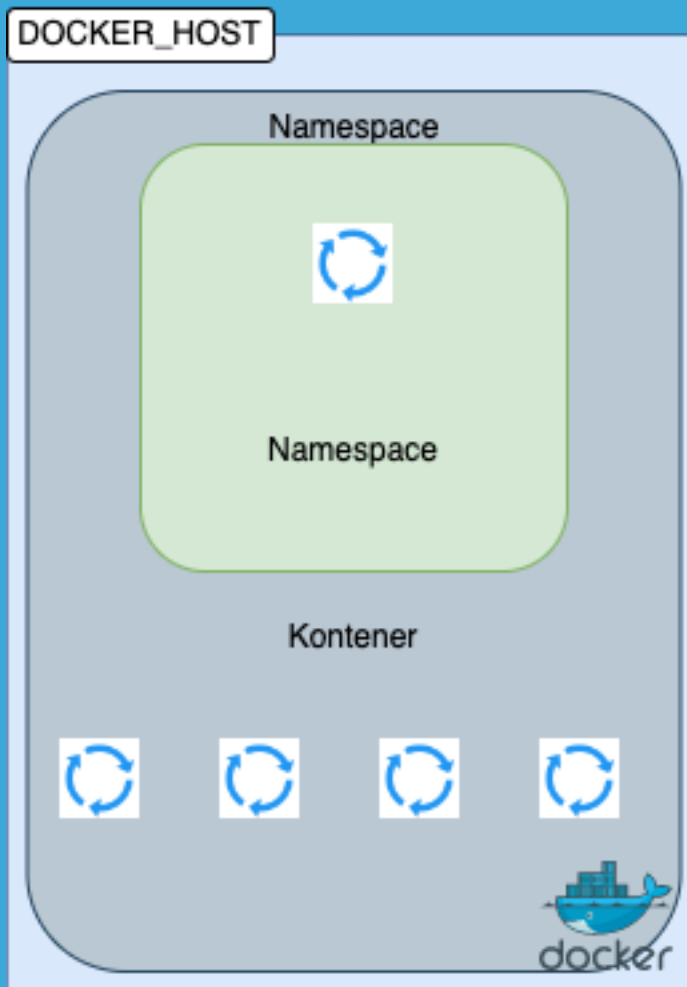
Bind mounts – część praktyczna

Docker – dla programistów i nie tylko

Bezpieczeństwo - wprowadzenie




```
> docker run busybox sleep 9999
```



Linki

- [Docker bezpieczeństwo – oficjalna dokumentacja](#)
- [Przykładowa luka bezpieczeństwa \(przykład 1 – CVE-2022-0492\)](#)
- [Przykładowa luka bezpieczeństwa \(przykład 2 – CVE-2022-0185\)](#)
- [Domyślne capabilities w Dockerze](#)
- [Lista capabilities w Linuxie](#)

Docker – dla programistów i nie tylko

Bezpieczeństwo – komendy i przykłady

Linki

- [Nginx unprivileged](#)

Docker – dla programistów i nie tylko

Bezpieczeństwo – obraz rootless

Docker – dla programistów i nie tylko

Bezpieczeństwo – skanowanie obrazów

Linki

- [Docker Scout – dokumentacja](#)
- [Docker Scout – subskrypcja](#)
- [Docker Scout Demo – kod źródłowy](#)
- [Trivy](#)
- [Trivy - instalacja](#)
- [Snyk – używany do skanowania w starszych wersjach dockera](#)

Docker – dla programistów i nie tylko

Docker compose – wprowadzenie (teoria)

Docker compose

- Narzędzie do definiowania i uruchamiania wielu kontenerów
- Pozwala definiować:
 - Kontenery (i zależności między nimi a także inne aspekty)
 - Sieci
 - Wolumeny
- Definicja w pliku YAML (compose.yaml lub docker-compose.yaml)
- Zarządzanie za pomocą CLI
- Pozwala szybko uzyskać pełne środowisko developerskie

Docker compose YAML file

- Występuje w dwóch wersjach
 - V1 (jako osobne CLI uruchamiane jako docker-compose)
 - V2 (zintegrowane z CLI Docker-a, uruchamiane jako docker compose)
- Komendy
 - `docker compose --help`
 - `docker compose up`
 - `docker compose up -d`
 - `docker compose up --wait`
 - `docker compose down`
 - `docker compose start`
 - `docker compose stop`

Linki

- [Compose – oficjalna dokumentacja](#)
- [Compose file - specyfikacja](#)
- [Migracja z V1 do V2](#)
- [Compose V2 General Availability - blog post](#)
- [Materiały do kursu](#)

Docker – dla programistów i nie tylko

Docker compose – wprowadzenie (praktyka)

Docker – dla programistów i nie tylko

Docker compose – komendy, profile i nadpisywanie plików

Linki

- [Compose – reguły nadpisywania plików](#)

Docker – dla programistów i nie tylko

Docker compose – build

Linki

- Compose – build

Docker – dla programistów i nie tylko

Docker compose – watch

Watch - rodzaje akcji

- **sync** - synchronizacja dla wskazanych plików
- **rebuild** - w momencie wykrycia zmian przebudowa kontenera
- **sync+restart** - synchronizuje pliki i restartuje kontener

Repozytorium do sklonowania:

```
git clone https://github.com/pnowy/compose-avatars.git
```

Linki

- Compose – watch
- Watch – GA release
- Repozytorium

Docker – dla programistów i nie tylko

Docker Hub

Linki

- [DockerHub](#)

Docker – dla programistów i nie tylko

Docker registry - lokalnie

Docker – lokalne registry

- Proxy dla DockerHub
- Większa kontrola i bezpieczeństwo
- Szybsze buildy i deploymenty (CI/CD)
- Cache do często używanych obrazów
- Prywatne obrazy

Docker – dla programistów i nie tylko

Docker registry – pozostałe opcje

Linki

- [GitLab / Registry docs](#)
- [GitHub Container Registry](#)
- [Google Artifact Registry](#)
- [Amazon Elastic Container Registry \(ECR\)](#)
- [Quay.io](#)
- [Jfrog artifactory](#)

Docker – dla programistów i nie tylko

Docker init

Docker - init

- [Oficjalna dokumentacja Docker-a](#)

Docker – dla programistów i nie tylko

Docker i Apple Silicon – jak żyć?

Linki

- [nginx:1.25.3](#)
- [pnowy/toolbox:1.0.0](#)
- [Rosetta](#)
- [Docker desktop – Rosetta](#)
- [Multi-platform images](#)
- [Load parameter issue](#)

Docker – dla programistów i nie tylko

Dobre praktyki

1. Oficjalne (zweryfikowane) obrazy

https://hub.docker.com/_/node

https://hub.docker.com/_/golang

https://hub.docker.com/_/caddy

2. Konkretna wersja obrazu (zamiast 'latest')

```
FROM node
```

```
FROM node:23.8.0-alpine
```

- W przypadku automatyzacji każdy kolejny build może mieć inną wersję obrazu
- Nowa wersja obrazu może wprowadzać zmiany które spowodują, że aplikacja przestanie działać
- 'latest' może zawierać breaking changes

3. Obrazy o jak najmniejszym rozmiarze (np. alpine)

FROM node:18.12.1

FROM node:18.12.1-alpine3.17

node	18.12.1	7b2a09676e2c	9 days ago	991MB
node	18.12.1-alpine3.17	6d7b7852bcd3	2 weeks ago	169MB

- Mniejszy rozmiar oznacza szybsze pobieranie na wszystkich środowiskach
- Mniej zależności oznacza mniej luk bezpieczeństwa

4. Multistage build

Multistage build tam gdzie aplikacja może być uruchomiona na znacznie mniejszym obrazie niż ten, który potrzebny jest do jej zbudowania

```
FROM golang:1.24.0 AS builder
WORKDIR /build
COPY main.go go.mod ./
RUN CGO_ENABLED=0 GOOS=linux go build -o app .
```

```
FROM alpine:3.21.2
WORKDIR /applications
COPY --from=builder /build/app /applications/app
EXPOSE 8080
CMD ["/app"]
```

5. Ograniczone przywileje użytkownika

Dedykowany użytkownik o ograniczonych uprawnieniach zamiast root-a (niektóre obrazy jak node posiadają już takiego użytkownika).

```
...
ARG GID=1000
ARG UID=1000

RUN addgroup --system --gid $GID app \
    && adduser --system --ingroup app --uid $UID app

WORKDIR /applications
RUN chown -R app:app /applications
USER app
...
```

6. Skanowanie obrazów pod kątem bezpieczeństwa

```
docker scout cves node:18.12.1-alpine3.17
```

```
trivy image node:18.12.1-alpine3.17
```

7. Dodatkowy plik .dockerignore

- Ograniczenie rozmiaru obrazów
- Szybsze budowanie
- Mniejszy rozmiar obrazu

8. Optymalizacja cache poprzez odpowiednie budowanie warstw

- Każda warstwa (polecenie) jest cache-owane przez Docker-a
- Podczas budowania nowej wersji obrazu Docker wykorzystuje cache
- Podczas push/pull Docker wykorzystuje cache
- Wydziel te warstwy które się rzadko zmieniają od tych które zmieniają się często i uporządkuj je w Dockerfile wg tej częstotliwości
- Niektóre narzędzia (np. jib) automatycznie przeprowadzają tego typu optymalizacje

Docker – dla programistów i nie tylko

Inne narzędzia

Podman

Kaniko

Lazy Docker

Dive

Slim

Linki

- [Podman](#)
- [Podman desktop](#)
- [Kaniko](#)
- [Lazydocker](#)
- [Dive](#)
- [Slim](#)

Docker – dla programistów i nie tylko

Podsumowanie zdobytej wiedzy

Docker – dla programistów i nie tylko

Co dalej? Gdzie szukać dodatkowy informacji

Linki

- [Awesome Docker](#)
- [Kubernetes](#)
- [Docker – oficjalny blog](#)
- [Kurs – Kubernetes od podstaw – dla programistów i nie tylko](#)