

Docker – co piszczy w kontenerach?



Justyna Misiak



Jakub Wójtowicz



Co to jest ?

docker

Docker to zestaw narzędzi open-source, które pozwalają między innymi na umieszczenie aplikacji razem z zależnościami w lekkim, odizolowanym oraz przenośnym środowisku nazywanym kontenerem (container).

Jakie problemy rozwiązuje ?

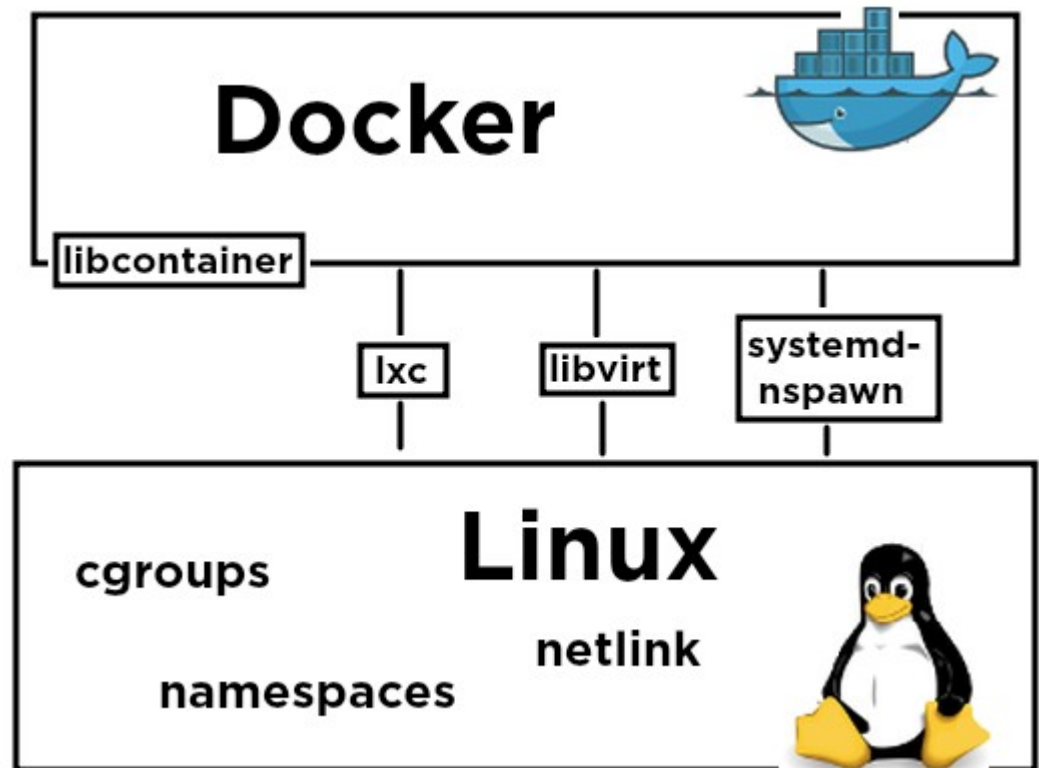
- Trudności w migracji
- Długi czas potrzebny na wdrożenie
- Optymalne zarządzanie zasobami
- Koszty środowiska
- Skalowanie



Architektura

Docker wykorzystuje rozwiązania stosowane w systemach linuxowych:

- grupy kontrolne (**C**ontrol **G**roups)
- Przestrzenie nazw (Namespaces)
- kontenery linuxa (**L**inu**X** **C**ontainer)
- Ujednolicony system plików (**U**nion **F**ile **S**ystem)

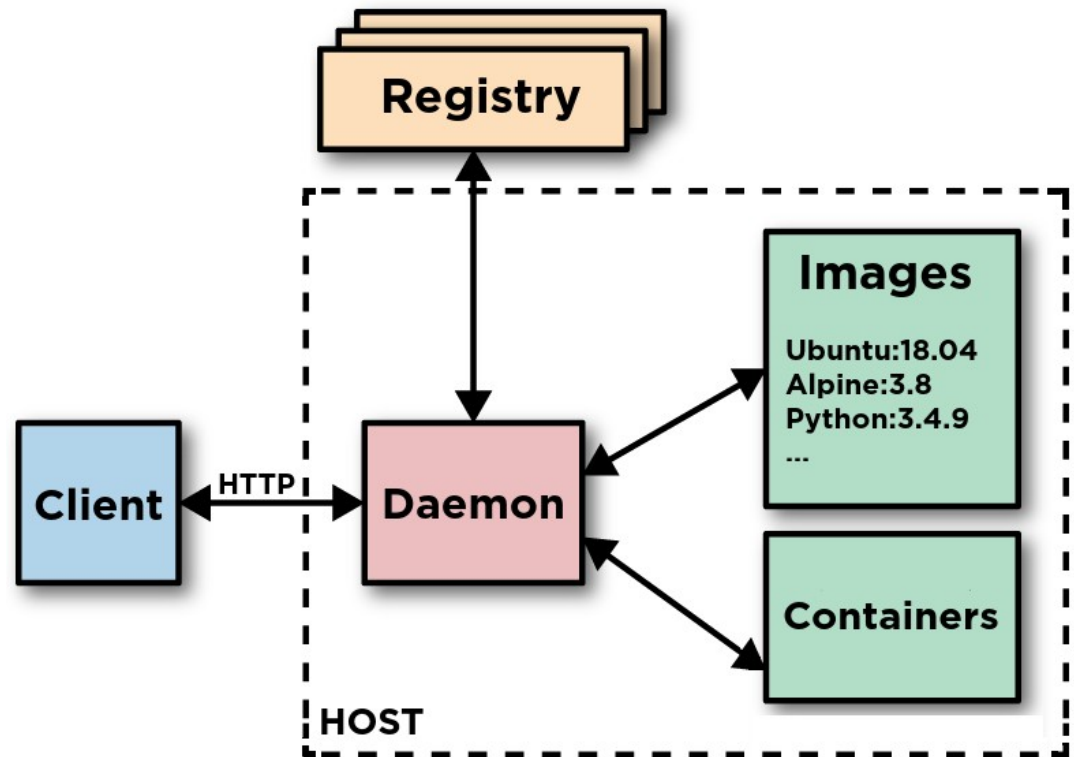


Podstawowe komponenty

Demon Dockera (Daemon) - jest odpowiedzialny za zarządzanie obrazami i kontenerami, np.: pobiera obrazy z repozytorium, buduje obrazy, uruchamia kontenery.

Klient Dockera (Client) - pozwala na wydawanie poleceń demonowi. Komunikacja pomiędzy klientem, a demonem odbywa się za pomocą protokołu HTTP. Lokalnie wykorzystywane są gniazda UNIX, a w przypadku komunikacji zdalnej gniazda TCP.


Repozytorium obrazów (Registry) - przechowuje obrazy (images) kontenerów Dockera. Główne repozytorium (**docker hub**) zawiera oficjalne obrazy oraz udostępnione przez innych użytkowników. Możliwe jest uruchomienie prywatnego repozytorium. Wszystkie lokalnie dostępne obrazy można wyświetlić za pomocą polecenia: **docker images**



Oficjalne repozytorium obrazów Docker Hub


Search - Docker Hub x

Secure | <https://hub.docker.com/search/?isAutomated=0&isOfficial=0&page=1&pullCount=0&q=ubuntu&starCount=0>

 ubuntu

Explore Help [Sign up](#) [Sign in](#)

Repositories (53183)

All			
	ubuntu official	8.7K STARS	10M+ PULLS
	ubuntu-debootstrap official	40 STARS	5M+ PULLS
	ubuntu-upstart official	92 STARS	1M+ PULLS
	rastasheep/ubuntu-sshd public automated build	181 STARS	1M+ PULLS

Uruchomienie kontenera

Komenda:

`docker run`

Składnia:

`docker run [opcje] [obraz:wersja] [komenda] [args]`

Przykład:

`docker run ubuntu:latest echo "Witajcie"`

```
misiajk@misiajk:~$ docker run ubuntu:latest echo "Witaj świecie"
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
473ede7ed136: Already exists
c46b5fa4d940: Already exists
93ae3df89c92: Already exists
6b1eed27cade: Already exists
Digest: sha256:29934af957c53004d7fb6340139880d23fb1952505a15d69a03af0d1418878cb
Status: Downloaded newer image for ubuntu:latest
Witaj świecie
```

Podstawowe komendy

`docker run -it [obraz:wersja] [komenda]`

`docker images`

`docker image rm [ID obrazu]`

`docker ps`

`docker pull [obraz:wersja]`

`docker container exec -it [kontener ID] [komenda]`

`docker container ls`

Zadanie 1.

Uruchomić

obraz z ubuntu w trybie

“interactive mode”, wywołać komendę uruchamiającą bash i wypisać na ekranie: “Witaj”



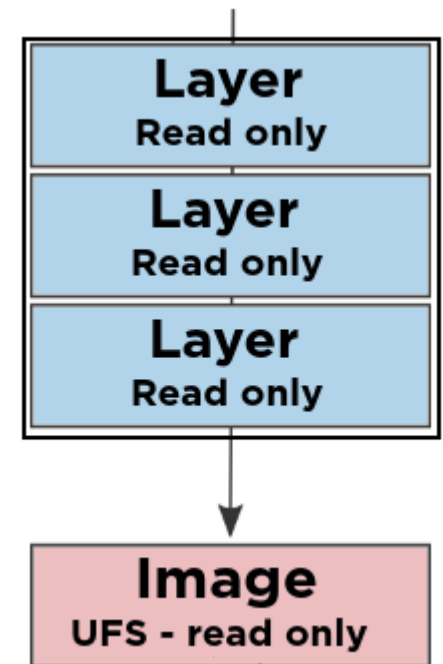
Obrazy (Images)

Obraz jest to ujednolicony zbiór warstw (layers).

Pojedyncza warstwa jest wynikiem wykonania pewnej instrukcji w trakcie budowania obrazu.

Instrukcją może być:

- instalacji pakietu,
- skopiowanie danych,
- ustawienie zmiennej środowiskowej,
- uruchomienie aplikacji



Budujemy obraz

Dockerfile – podstawowe instrukcje:

FROM określa nazwę bazowego obrazu

ADD dodaje pliki z hosta do obrazu

WORKDIR ustalenie katalogu roboczego kontenera

RUN uruchamia komendy, które mają się wykonać na obrazie przed jego uruchomieniem

EXPOSE definiuje porty na których będzie nasłuchiwał kontener

ENV definiuje zmienne globalne

ENTRYPOINT określa parametry dla CMD

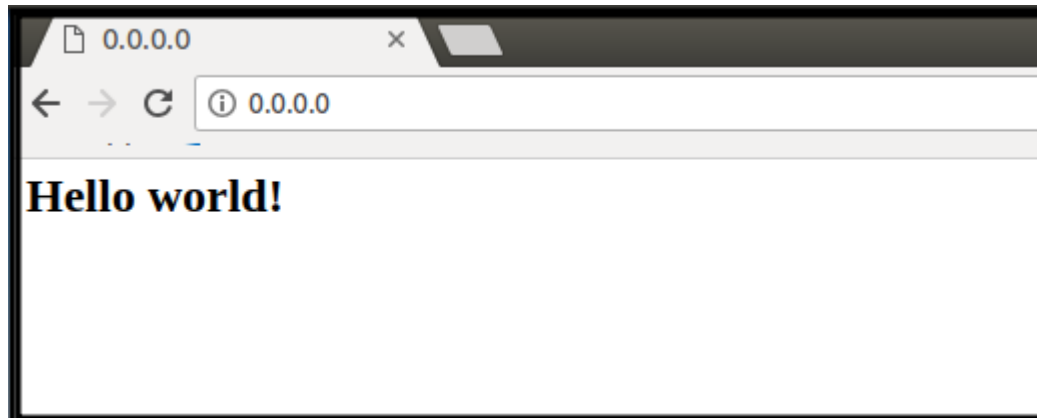
CMD określa komendę, jaka ma zostać wykonana tuż po uruchomieniu kontenera

```
1 # Use an official Ubuntu 18.04 as a parent image
2 FROM ubuntu:18.04
3
4 # Set the working directory to /app
5 WORKDIR /script
6
7 # Copy the current directory contents into the container at /script
8 ADD script.sh /script
9
10 # Update system, install additional packages, do anything you need
11 RUN apt update \
12     && apt install -y figlet toilet \
13     && chmod +x /script/script.sh
14
15 # Set the entrypoint
16 ENTRYPOINT ["/bin/bash"]
17
18 # Run the script when the container launches
19 CMD [ "script.sh" ]
20
```

Budujemy obraz

Zadanie 2.

Bazując na oficjalnym obrazie dockerowym (ubuntu, python) stworzyć obraz, który po uruchomieniu wystartuje serwer http na porcie 80, a po wpisaniu w przeglądarce adresu
http://0.0.0.0:80
wypisze na ekranie "Hello world!"



Wskazówki:

`docker run -p` – publish list Publish a container's port(s) to the host

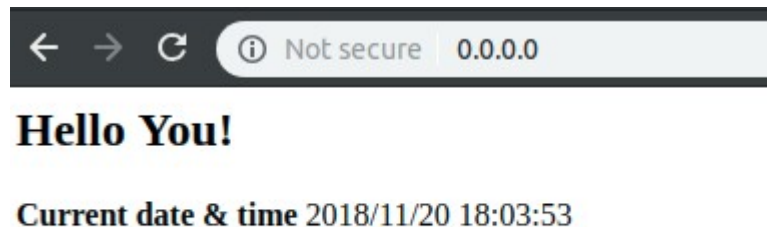
Pomoce:

<https://github.com/justolka/ldi-workshop/blob/master/dummy-server/>

Environment

Zadanie 3.

Bazując na obrazie z zadanie 2. uruchom kontener, który wystartuje serwer http na porcie 80, a po wpisaniu w przeglądarkę adresu <http://0.0.0.0:80> wypisze na ekranie “Hello You!”



Wskazówki:

Użyj zmiennej środowiskowej NAME

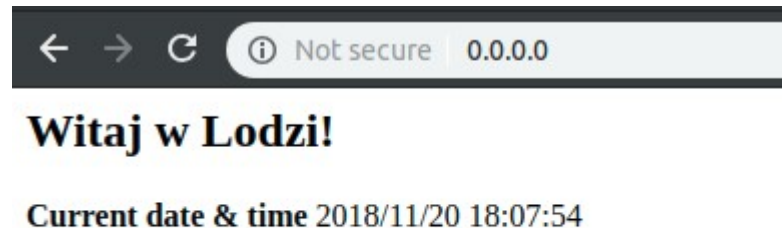
Pomoce:

<https://github.com/justolka/ldi-workshop/blob/master/dummy-server/>

Volume

Zadanie 4.

Bazując na obrazie z zadanie 2. uruchom kontener, który wystartuje serwer http na porcie 80, a po wpisaniu w przeglądarkę adresu <http://0.0.0.0:80> wypisze na ekranie “Witaj w Lodzi!”



Wskazówki:

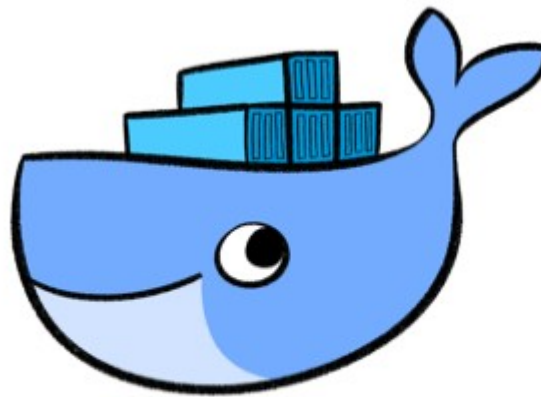
Użyj volume aby nadpisać plik aplikacji w kontenerze.

Pomoce:

<https://github.com/justolka/ldi-workshop/blob/master/dummy-server/>

Docker-compose

- Automatyzacja budowania kontenerów
- Prosta konfiguracja środowiska – 1 polecenie
- Strukturalizowana składnia (YAML)



Wincy kontenerów!

Docker-compose

```
1  version: '3.0'
2  services:
3      web:
4          image: "wojtjk/pythonapp"
5          ports:
6              - "4000:80"
7      redis:
8          image: "redis:alpine"
9          container_name: "redis"
```

Docker-compose

Zadanie 5.

Przygotować docker-compose dla aplikacji dummy_server, a następnie uruchomić ją za pomocą polecenia:

docker-compose up

Zadanie 6.

Rozbudować docker-compose o obraz, który będzie pokazywał licznik odwiedzin naszej strony.

Kod aplikacji:

<https://github.com/justolka/ldi-workshop/tree/master/counter-server>

Więcej poleceń

- **docker inspect** – szczegółowe informacje o kontenerze
- **docker stats** – zużycie zasobów przez kontener
- **docker top** – informacje o procesach kontenera
- **docker history** – informacje o warstwach obrazie
- **docker swarm** – zarządzanie klastrem Docker Swarma
- **docker node** – informacje o maszynach Docker Swarma

What's next?

