Konteneryzacja aplikacji w środowisku Docker

Hello

Łukasz Pieczonka

DevOps @ LU-MEDIA

Agenda

- Czym jest Docker ?
- Budowanie, Uruchamianie oraz Zarządzanie.
- Ćwiczymy

Wprowadzenie

Historia: baremetal

Problemy:

- Powolne wdrażanie
- Duże koszta (licencje/sprzęt/utrzymanie)
- Problemy ze skalowaniem
- Trudne w migracji

Historia: wirtualizacja

Zalety:

- Lepsze zarządzanie zasobami
- Łatwość skalowania
- Jeden bare-metal można podzielić na wiele maszyn wirtualnych

Historia: wirtualizacja

Ograniczenia:

- Każda VM alokuję zasoby tj.
  - CPU, RAM, STORAGE
  - SYSTEM OPERACYJNY
- Im więcej VM, tym więcej zasobów potrzebujemy
- Wiele systemów VM (uruchomionych systemów operacyjnych) to wiele zmarnowanych zasobów
- Brak gwarancji przenośności aplikacji

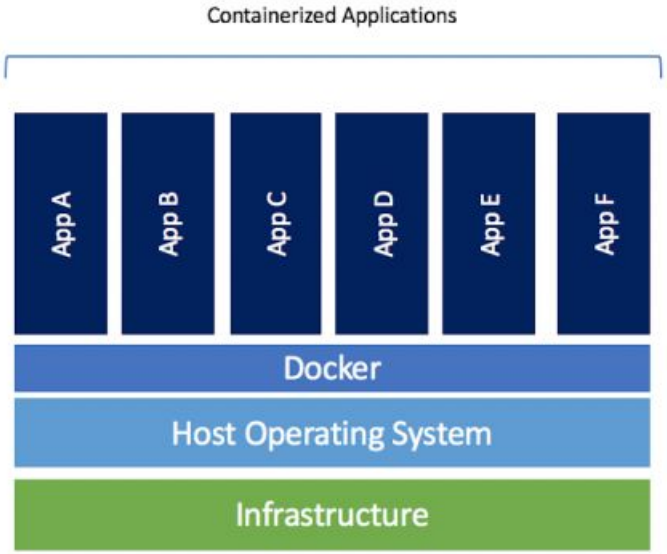
Czym jest Docker?

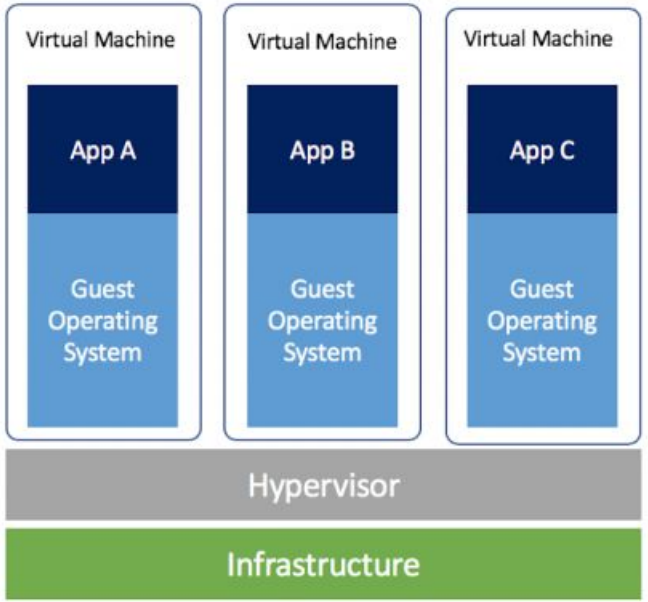
- Zbiór aplikacji służących do obsługi “lekkiej” wirtualizacji.
- System, umożliwiający dostarczanie gotowych do użycia rozwiązań.
- Środowisko, znacząco zmieniające styl tworzenia, zarządzania i dostarczania aplikacji.

Docker to nie VM

- Docker jak i VM Maszyna wirtualna używana jest do zapewniają aplikacji odizolowane środowisko, jednak oba rozwiązania realizują to na swój sposób.

Docker to nie VM





Source: https://www.docker.com/blog/containers-replacing-virtual-machines/

Docker: kontenery

Docker używa kernela systemu operacyjnego hosta do uruchamiania wielu kontenerów.

- Każdy kontener posiada:
  - System plików (rootfs)
  - Procesy
  - Pamięć
  - Urządzenia
  - Porty sieciowe

Docker vs VM

- Kontenery są lżejsze, większa gęstość (APP)
- Mniejsze wykorzystanie zasobów (RAM/CPU)
- Brak konieczności instalacji systemu operacyjnego
- Łatwość w przenoszeniu
- Skalowanie

Docker Engine

Umożliwia tworzenie, dystrybucję i uruchamianie obrazów.

Używa funkcji kernela linuxa tj.

- Namespaces
- Control groups (cgroups)

CYKL ŻYCIA

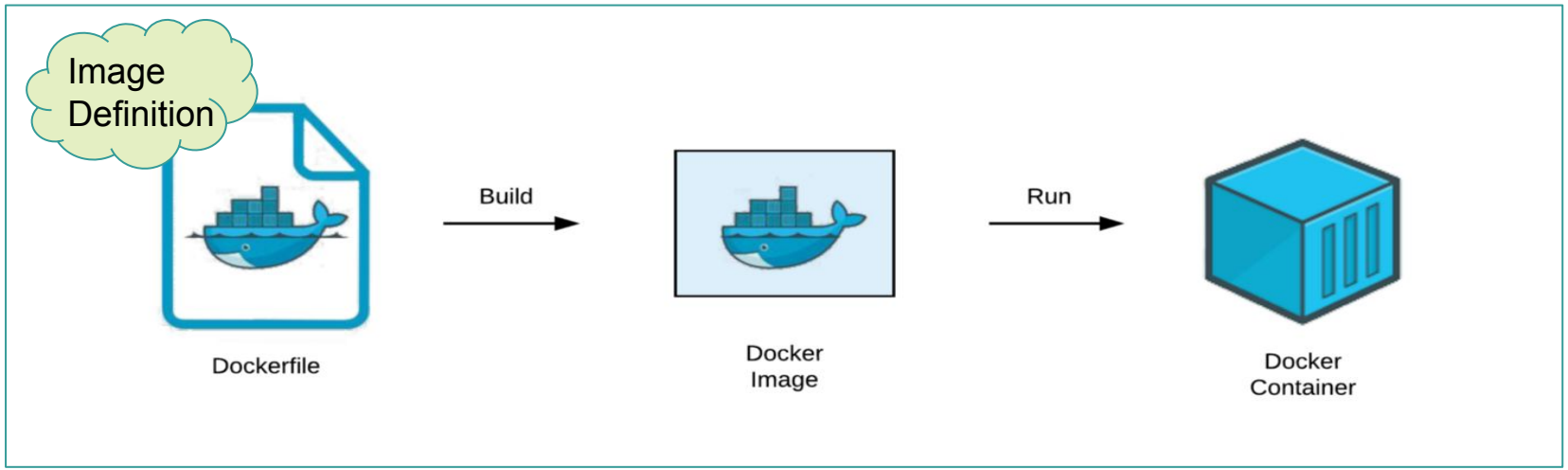


Image
Definition
Dockerfile
Build
Docker
Image
Run
Docker
Container

Dockerfile

Plik Dockerfile to zbiór instrukcji, gdzie zwykle każda z instrukcji dodaje kolejną warstwę do finalnego obrazu. Z reguły w pierwszej warstwie obrazu znajdują się pliki systemowe. Kolejne polecenia takie jak RUN, COPY, ADD tworzą następne warstwy, by finalnie stworzyć kompletny obraz naszej aplikacji.

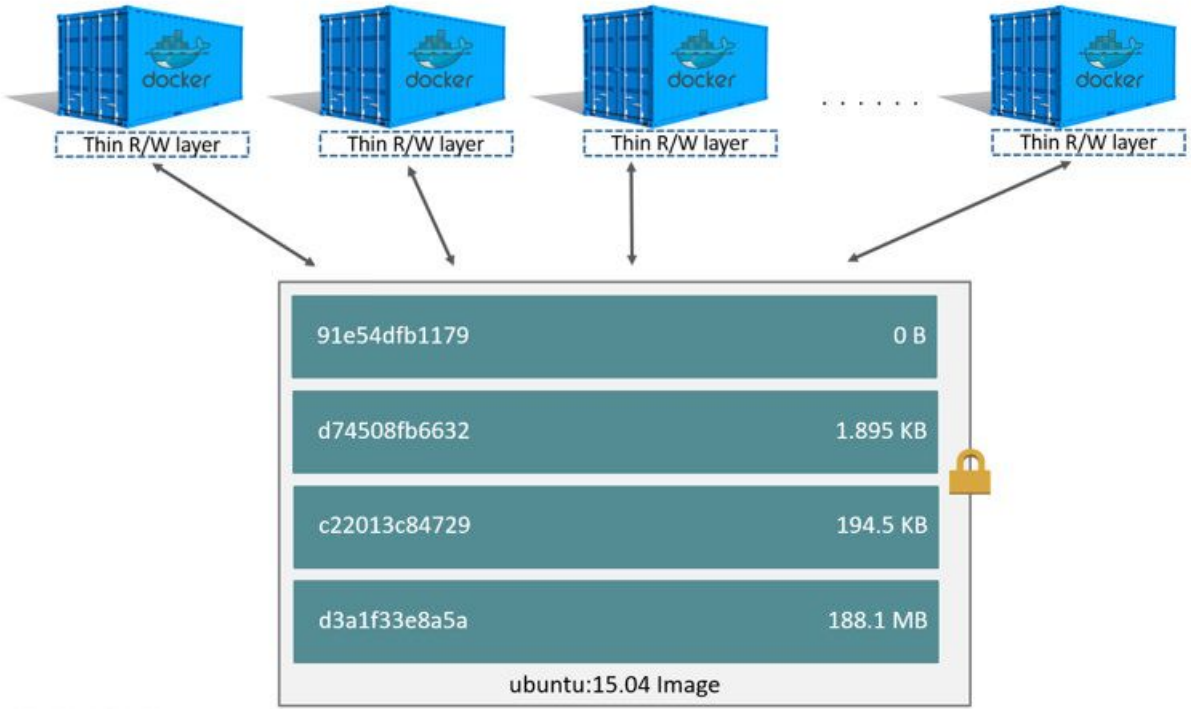
Docker image

Obraz dockera (docker image) jest to szablon/wrorzec używany do zbudowania kontenera. Obrazy są przechowywane lokalnie lub w zdalnym repozytorium (Docker registry) np. Docker Hub https://hub.docker.com. Jeden obraz może być użyty do utworzenia wielu kolejnych kontenerów.

Docker Container(s)

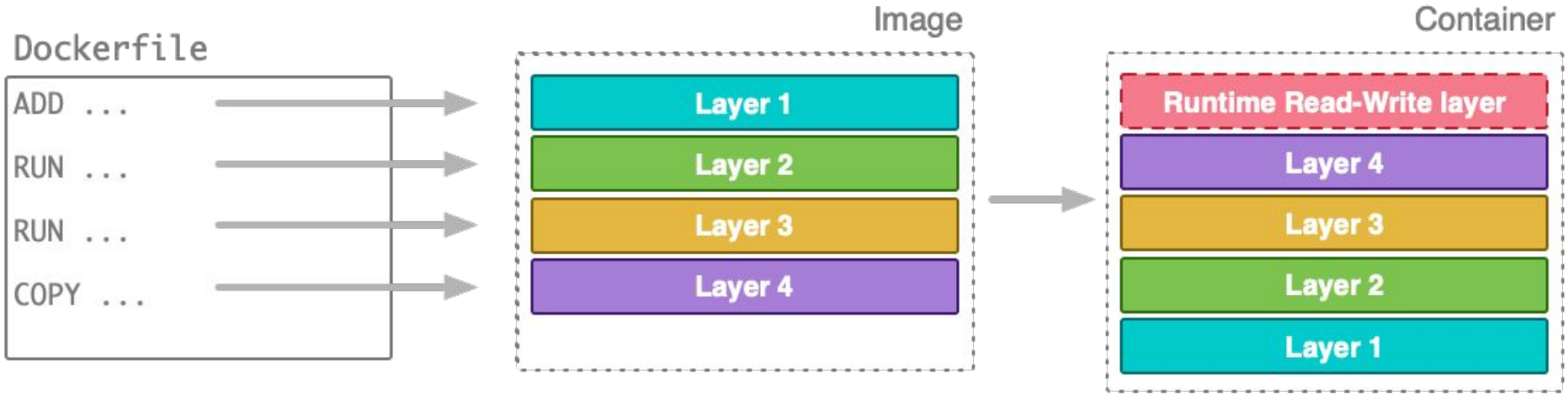
Kontener jest ujednoliconą kolekcją warstw składającą się z warstw tylko do odczytu pochodzących z obrazu kontenera i pojedynczej warstwy do odczytu i zapisu. Warstwa do odczytu i zapisu umożliwia działanie procesów uruchamianych w kontenerze.

DOCKER KONTENER I WARSTWY



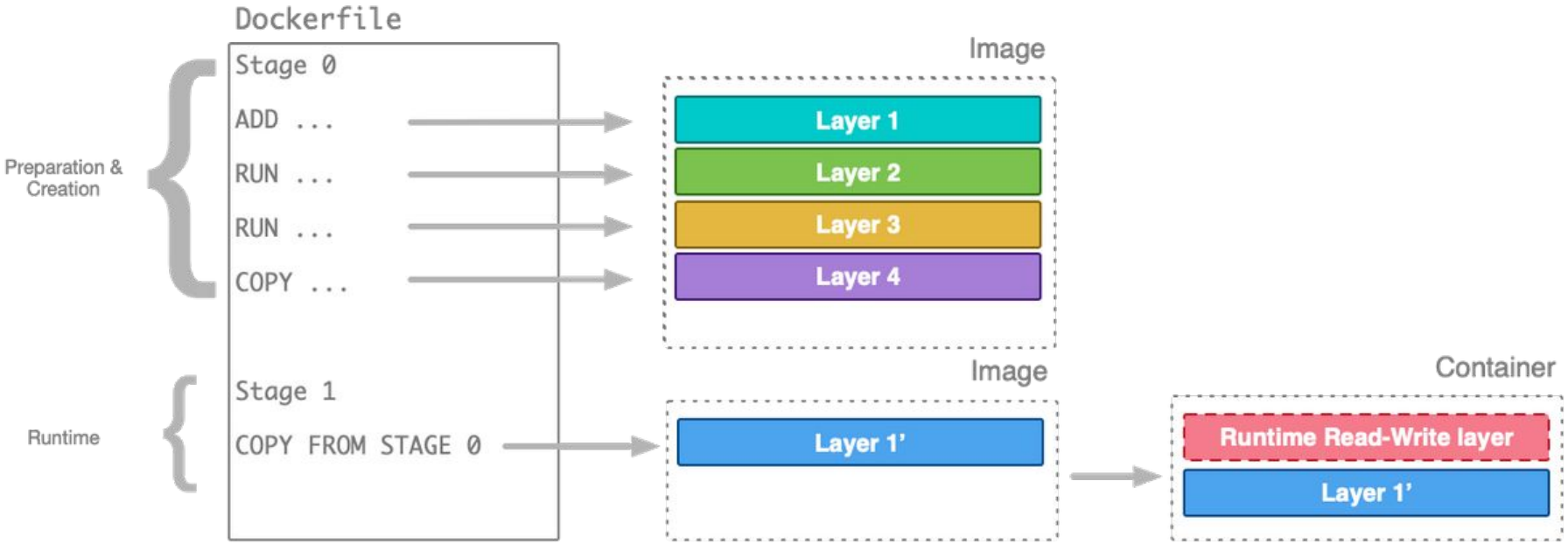
Source: https://docs.docker.com/storage/storagedriver/

DOCKER JEST JAK CEBULA



Source: https://www.metricfire.com/blog/how-to-build-optimal-docker-images/

DOCKER MULTI-STAGE



Source: https://www.metricfire.com/blog/how-to-build-optimal-docker-images/

CO DALEJ?

- Optymalizacja rozmiaru obrazu.
- Agregacja i przetwarzanie logów (zarządzanie logami).
- Jeden kontener == Jeden proces (główny)!
- Optymalizacja (lub mądra organizacja) komunikacja pomiędzy usługami.

CO MUSIMY WIEDZIEĆ?

- “Lekkie” obrazy bazowe (np. Bazujące na Alpine) nie nadają się do każdej usługi.
- Każdy kontener wykorzystuje zasoby systemowe. Pamiętajmy o “write layer”!
- Kolejność operacji w Dockerfile ma (ogromne) znaczenie.
- Kernel ma dostęp do procesów dockera. Jakie to ma znaczenie?

DOBRE PRAKTYKI

- Konfiguracja za pośrednictwem zmiennych środowiskowych.
- Zarządzanie wersją (tagami).
- Nazewnictwo
- Śledzenie i dokumentacja uruchomionych instancji

BUDOWANIE, URUCHAMIANIE I ZARZĄDZANIE

DOCKERFILE

- FROM - obraz bazowy
- COPY - z hosta do obrazu
- RUN - wykonaj w trakcie tworzenia obrazu
- ENTRYPOINT - stała część uruchamianej komendy
- CMD - modyfikowana część komendy
- + ARG, ENV, EXPOSE…

ZADANIE - BUDUJEMY

- Pierwszy obraz

URUCHAMIANIE

- Docker run [--name nazwa kontenera] {nazwa obrazu}
- + volumes: [-v host:kontener]
- + port: [-p host:kontener]
- + network: [--network nazwa sieci]
- + expose: [--expose port]
- + env: [--env nazwa_zmiennej=wartość]

--detach, -d : uruchamia w tle
--interactive, -i : STDIN wymaga aktywnej powłoki np. Sh, Bash itp.

I masa innych: https://docs.docker.com/engine/reference/commandline/run/

Dzieki

Łukasz Pieczonka

lukasz.pieczonka@lu-media.pl

DevOps @ LU-MEDIA