Docker
konteneryzacja oprogramowania

High Flyers 2023r

Co rozwiązuje konteneryzacja?

Po co korzystać z tej technologii?

- Załóżmy, że stworzyliśmy oprogramowanie które wymaga dużo systemowych bibliotek i nie wiemy czy będzie działać na innych maszynach.
- Aktualizacje systemu mogą w przyszłości zepsuć konfigurację oprogramowania.
- Potrzebujemy szybko i wygodnie uruchomić program na zdalnym serwerze lub innym komputerze.
- Chcielibyśmy zminimalizować wpływ oprogramowania na system operacyjny ze względu na inne programy lub kwestie bezpieczeństwa.

Co rozwiązuje konteneryzacja?

Po co korzystać z tej technologii?

- Rozwiązaniem mogłoby być zamknięcia naszego programu w maszynie wirtualnej i uruchamianie go w ten sposób.
- Taka opcja ma swoje plusy (przykładem może być qubes-os), jednak dla większości przypadków takie rozwiązanie zużywa za dużo mocy obliczeniowej i pamięci.
- Konteneryzacja jest “kompromisem” pomiędzy uruchamianiem oprogramowania bezpośrednio na systemie operacyjnym a maszyną wirtualną.

Jak działa konteneryzacja?

- Omawiany tutaj jest najpopularniejszy system konteneryzacji, czyli Docker jednak inne rozwiązania też istnieją (część ma swoje plusy jak szybszy rozruch).
- Docker działa w oparciu o jądro systemu Linux, czyli jeśli korzystamy z macOS lub Windows, Docker postawi jedną maszynę wirtualną (będzie dzielona między wszystkimi kontenerami).
- Na Linuxie, Docker dzieli jądro systemu z naszym systemem operacyjnym, jednak system plików jest opisany przez kontener. W ten sposób biblioteki i oprogramowanie w kontenerze jest dla niego znane.
- Oprogramowanie działające w kontenerze może współdzielić z systemem operacyjnym wybrane foldery, lub porty sieciowe.

Jak działa konteneryzacja?

- Kontener to instancja obrazu. Na przykład możemy zbudować obraz z bazą danych i uruchomić cztery jej instancje w czterech kontenerach.
- Ponieważ jądro systemu jest dzielone z oprogramowaniem, nie można korzystać z kontenera stworzonego na jednej architekturze procesora na innej.
- Na przykład obraz zbudowany na komputerze stacjonarnym nie może zostać uruchomiony na raspberry pi. Podobnie jako że jest kilka wersji architektury ARM, starsze wersje mogą nie być wspierane.
- Obejściem tego jest zbudowanie obrazu na nowej architekturze. Konieczne jest jednak pamiętanie że jeśli instrukcje są ustawione na biblioteki innej architektury, to rozwiązanie nie zadziała.

Instalacja i pierwsze uruchomienie

- Najlepiej skorzystać z instrukcji dostępnych na stronie Docker. Dla systemów Linux potrzebny jest tylko Docker Engine (chociaż aplikacja daje wygodny podgląd): https://docs.docker.com/engine/install/
  Na systemach Windows i MacOS: https://docs.docker.com/desktop/
- Na systemie Linux ważne jest dodanie grupy docker dla użytkowników (inaczej każda komenda wymaga sudo): https://docs.docker.com/engine/install/linux-postinstall/
- VS Code posiada również wtyczkę pod Docker.
- Po udanej instalacji możemy uruchomić testowy kontener hello-world!

```
$ docker run hello-world
```

Instalacja i pierwsze uruchomienie

```
-> % docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
70f5ac315c5a: Pull complete
Digest: sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Docker Hub

- Na stronie Docker Hub znajduje się baza z dostępnymi obrazami do pobrania.
- Wśród nich znajduje się wiele gotowych środowisk pod wybrane języki programowania jak i bazy danych czy brokery sieciowe.
- Możemy również tworzyć własne obrazy i wrzucać je na hub. W ten sposób możemy je zbudować tylko raz i na innych komputerach tylko je pobierać.
- Przy pobieraniu warto zwrócić na wersje Alpine. Jest to mocno wychudzona dystrybucja Linuxa, dzięki czemu kontenery są mniejsze.

Przykład pierwszy: docker run

- Pierwszy przykład z uruchomieniem w trybie interaktywnym i z dzieleniem folderów.
- Chcemy uruchomić program napisany pod node.js w kontenerze.
- Aby kontener mógł uruchomić program, korzystamy z współdzielenia folderów.
- Wersja pierwsza: docker run -it --entrypoint bash -v ./src:/app node:current
- Wersja alpine: docker run -it -v ./src:/app node:current-alpine3.17 /app/mandelbrot.js

Przykład drugi: docker build

- Przykład z najczęstszym zastosowaniem: tworzymy instrukcje ustawienia aplikacji i uruchamiamy ją potem.
- Chcemy uruchomić serwer napisany pod node.js w kontenerze. Serwer wykorzystuje dodatkowe biblioteki.
- Korzystamy z współdzielenia portów i udostępniania folderów na czas budowy
- Budowanie obrazu: docker build . -t server_example
- Uruchomienie: docker run -p 6969:3000 server_example

Docker compose

- Z kontenerów można stworzyć cały system. Na przykład w jednym kontenerze jest baza danych a w drugim serwer.
- Przy użyciu compose można łączyć kontenery które komunikują się przez wewnętrzną dla nich sieć.
- Pozwala to na szybkie przeładowywanie projektu, i uruchamianie go na serwerze.
- Dodatkowo wzrost bezpieczeństwa przez “schowanie” bazy danych w wewnętrznej sieci.

Do czego korzystamy z Docker w HF?

- Uruchamianie serwerów internetowych.
- Praca robotyków, uruchamianie środowisk deweloperskich (interfejs graficzny w kontenerze jest możliwy!).
- Wykorzystywanie bibliotek które wykorzystują dużo problematycznych rzeczy (mavsdk na raspberry albo ros).
- Ogólna deweloperka, jeśli chcemy łatwo usunąć projekt po zakończeniu pracy.

Co dalej?

- Docker-nvidia pozwala na wykorzystywanie gpu w kontenerze, można korzystać z aplikacji graficznych i używać akceleracji GPU.
- Budowanie obrazów może być wieloetapowe. Na przykład możemy na pełnym ubuntu zbudować projekt w c++, po czym gotową binarke z bibliotekami przenieść do Alpine aby zajmowało mniej miejsca.
- Ustawienie swojego środowiska deweloperskiego tak, że możemy edytować kod na żywo w kontenerze.