

Co to jest Docker i jak działa?

Streszczenie referatu

Dominik Błażko 19325

Spis treści

1. Wprowadzenie
 - Czym jest Docker i jakie są jego zalety?
 - Czym są kontenery i jak się różnią od maszyn wirtualnych?
2. Przykład użycia Dockera
 - Jak stworzyć plik Dockerfile do budowy własnego obrazu?
 - Jak uruchomić aplikację frontend-ową i backend-ową w osobnych kontenerach?
3. Podsumowanie
 - Jakie są główne korzyści z używania Dockera w rozwoju i wdrażaniu aplikacji?
 - Jakie są ograniczenia i wyzwania związane z Dockerem?
 - Gdzie szukać więcej informacji o Dockerze?
 - Bibliografia

1. Wprowadzenie

Czym jest Docker i jakie są jego zalety?

Docker jest otwartym oprogramowaniem typu open source, służącym do wirtualizacji programów na poziomie systemu operacyjnego użytkownika.

Konteneryzacja to izolowanie i przechowywanie programu w kontenerze, który jest relatywnie lekki i przenośny.

Dockera można uruchomić na urządzeniach z systemem opartym na jądrze Linux'a.

Przenośność jest główną zaletą Dockera, umożliwiającą łatwe przeniesienie kontenera między różnymi środowiskami i maszynami.

Lekkość Docker'a wynika z jego różnicy w stosunku do maszyny wirtualnej, nie wymaga dodatkowego systemu operacyjnego, pracuje na przydzielonej części jądra systemu hosta.

Docker zapewnia skalowalność poprzez łatwe uruchamianie i zatrzymywanie kontenerów w zależności od potrzeb klienta aplikacji lub obciążenia.

Czym są kontenery i jak się różnią od maszyn wirtualnych?

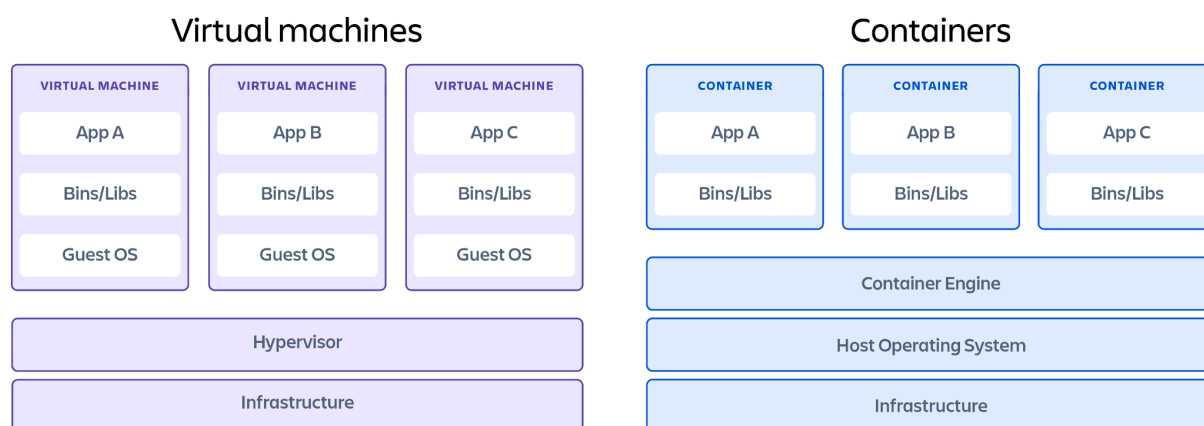
Kontener w Dockerze to sandbox'owy proces, izolowany od innych procesów na komputerze użytkownika za pomocą przestrzeni nazw i grup kontrolnych z jądra systemu operacyjnego.

Docker Desktop lub CLI umożliwiają tworzenie, uruchamianie, zatrzymywanie, przenoszenie i usuwanie kontenerów.

Kontener Dockera jest odizolowany od innych kontenerów i uruchamia własne oprogramowanie, pliki binarne i konfiguracje.

Maszyna wirtualna to emulator kompletnego komputera, działający na poziomie sprzętowym, posiadający własny system operacyjny i zasoby.

Różnice między wirtualizacją maszyn a konteneryzacją to m.in. mniejszy rozmiar i szybsza praca kontenerów, wynikające z ich mniejszej objętości na dysku i szybszego uruchamiania.



Rys.1 Różnica między wirtualną maszyną a kontenerem

źródło: <https://www.atlassian.com/pl/microservices/cloud-computing/containers-vs-vms>

2. Przykład użycia Dockera

Jak stworzyć plik Dockerfile do budowy własnego obrazu?

Aby stworzyć plik Dockerfile do budowy własnego obrazu, należy wykonać następujące czynności:

Określ bazowy obraz za pomocą dyrektywy "FROM":

Bazowy obraz to obraz Docker, który zostanie wykorzystany jako punkt wyjściowy do budowy Twojego obrazu. Może to być oficjalny obraz z rejestru Docker lub inny dostępny publicznie obraz. Określenie bazowego obrazu w dyrektywie "FROM" odbywa się przez podanie nazwy obrazu i opcjonalnie wersji, np. "FROM ubuntu:latest".

Skopiuj pliki aplikacji do kontenera za pomocą dyrektywy "COPY":

Dyrektywa "COPY" służy do kopiowania plików z lokalnego systemu plików do kontenera Docker. Musisz podać ścieżkę pliku/folderu na swoim hoście oraz ścieżkę docelową w kontenerze, np. "COPY app /usr/src/app". To spowoduje skopiowanie zawartości folderu "app" z Twojego systemu plików do folderu "/usr/src/app" w kontenerze.

Zainstaluj zależności wewnątrz kontenera przy użyciu dyrektywy "RUN":

Dyrektywa "RUN" pozwala na wykonanie poleceń wewnątrz kontenera podczas procesu budowy obrazu. Możesz użyć jej do instalacji zależności, np. narzędzi, bibliotek lub programów potrzebnych do uruchomienia Twojej aplikacji. Przykładowo, jeśli korzystasz z systemu operacyjnego opartego na Debianie, możesz użyć polecenia "RUN apt-get install -y nazwa_pakietu" do instalacji pakietu w kontenerze.

Ustaw port nasłuchu za pomocą dyrektywy "EXPOSE":

Dyrektywa "EXPOSE" informuje Dockera, na którym porcie kontener nasłuchuje na połączenia. Możesz podać numer portu, na którym działa Twoja aplikacja, np. "EXPOSE 80" dla aplikacji działającej na porcie 80.

Opcjonalnie, ustaw zmienne środowiskowe przy użyciu dyrektywy "ENV":

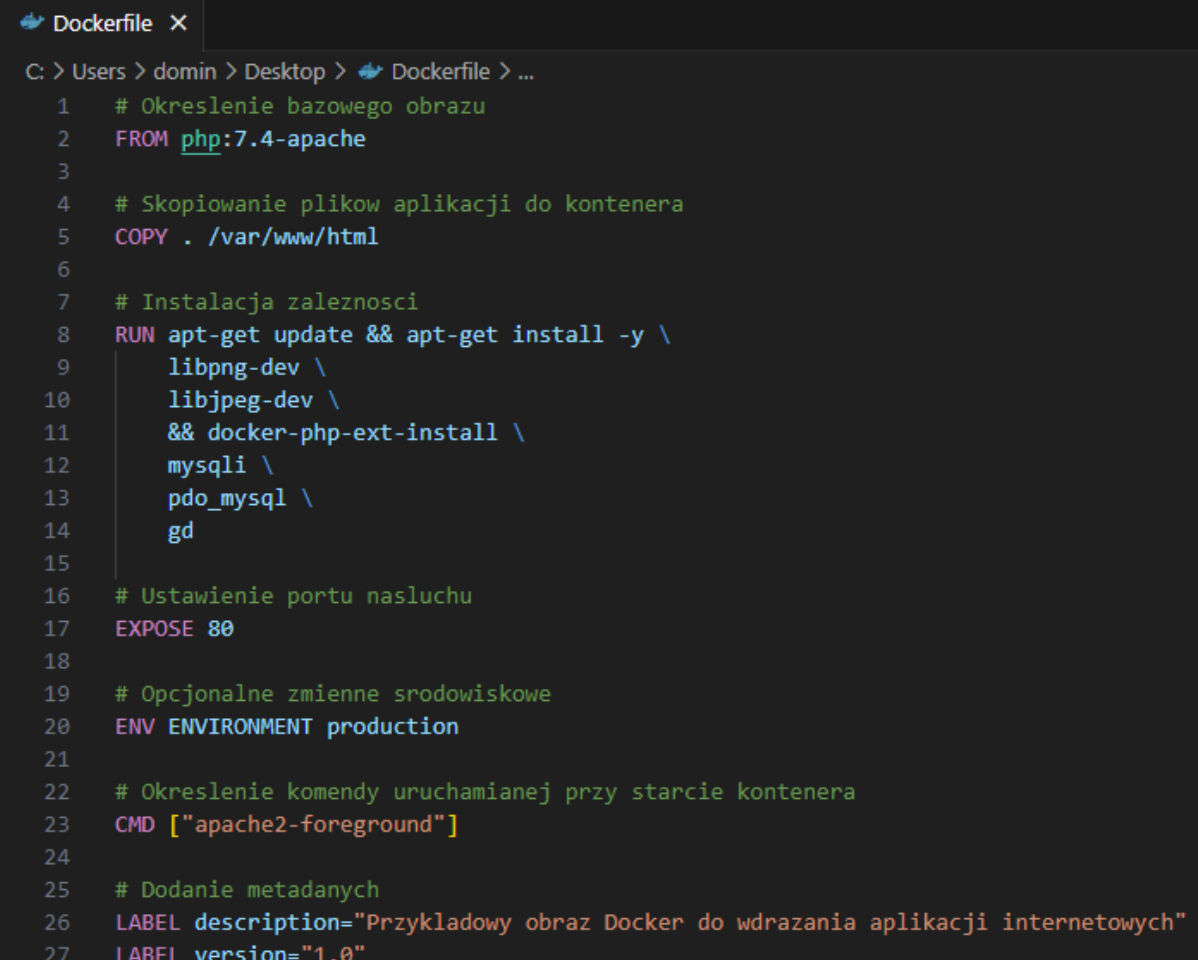
Dyrektywa "ENV" służy do ustawiania zmiennych środowiskowych w kontenerze. Możesz użyć jej do konfiguracji różnych parametrów dla Twojej aplikacji, na przykład ustawienia URL bazy danych. Składnia to "ENV NAZWA_ZMIENNEJ wartość".

Określ komendę uruchamianą przy starcie kontenera za pomocą dyrektywy "CMD":

Dyrektywa "CMD" definiuje polecenie, które zostanie wykonane po uruchomieniu kontenera. Może to być komenda uruchamiająca Twoją aplikację, np. "CMD [nazwa_programu]".

Dodaj metadane, takie jak autor, opis i wersja obrazu za pomocą dyrektywy "LABEL":

Możesz dodać metadane do obrazu, takie jak autor, opis czy wersja, używając dyrektywy "LABEL". Przykładowo, "LABEL autor="Twoje imię i nazwisko"".



```
Dockerfile X
C: > Users > domin > Desktop > Dockerfile > ...
1  # Okreslenie bazowego obrazu
2  FROM php:7.4-apache
3
4  # Skopiowanie plikow aplikacji do kontenera
5  COPY . /var/www/html
6
7  # Instalacja zaleznosci
8  RUN apt-get update && apt-get install -y \
9      libpng-dev \
10     libjpeg-dev \
11     && docker-php-ext-install \
12     mysqli \
13     pdo_mysql \
14     gd
15
16 # Ustawienie portu nasluchu
17 EXPOSE 80
18
19 # Opcjonalne zmienne srodowiskowe
20 ENV ENVIRONMENT production
21
22 # Okreslenie komendy uruchamianej przy starcie kontenera
23 CMD ["apache2-foreground"]
24
25 # Dodanie metadanych
26 LABEL description="Przykladowy obraz Docker do wdrazania aplikacji internetowych"
27 LABEL version="1.0"
```

Rys.2 Przykład pliku Dockerfile, który implementuje opisane dyrektywy

Po przygotowaniu pliku Dockerfile, możemy zbudować obraz Dockera za pomocą komendy docker build. Poniżej przedstawiam pełną procedurę:

1. Przejdź do katalogu, w którym znajduje się plik Dockerfile.
2. Wykonaj polecenie docker build -t nazwa_obrazu:tag ., gdzie "nazwa_obrazu" to dowolna nazwa, którą chcesz nadać nowemu obrazowi, a "tag" to wersja lub etykieta obrazu.
3. Docker przeprowadzi proces budowy obrazu na podstawie zawartości pliku Dockerfile i dostępnych zasobów.
4. Po zakończeniu procesu budowania, otrzymasz nowy obraz Dockera o nazwie i wersji zdefiniowanej w kroku 2.
5. Możesz sprawdzić listę dostępnych obrazów, wykonując polecenie docker images.

6. Teraz, gdy masz zbudowany obraz Dockera, możesz go uruchomić, tworząc i uruchamiając kontener na jego podstawie. Wykonaj polecenie `docker run --name nazwa_kontenera -p port_hosta:port_kontenera nazwa_obrazu:tag`, gdzie "nazwa_kontenera" to dowolna nazwa, którą chcesz nadać kontenerowi, "port_hosta" to port na hoście, a "port_kontenera" to port w kontenerze, na którym działa Twoja aplikacja. Podaj również nazwę obrazu i tag, aby wskazać, który obraz ma być użyty do uruchomienia kontenera.
7. Po wykonaniu tego polecenia kontener zostanie utworzony i uruchomiony na podstawie obrazu Dockera.

Jak uruchomić aplikację frontend-ową i backend-ową w osobnych kontenerach?

Aby uruchomić aplikację frontendową i backendową w osobnych kontenerach, należy wykonać poniższe kroki:

1. Przygotowanie pliku Dockerfile dla frontendu

Pierwszym krokiem jest stworzenie pliku Dockerfile, który będzie zawierał instrukcje dotyczące budowy obrazu kontenera frontendowego. W pliku Dockerfile należy zdefiniować bazowy obraz, który będzie stanowił punkt wyjściowy dla budowy obrazu frontendu. Możemy wybrać istniejący obraz z rejestru Docker, na przykład obraz Node.js, który zawiera niezbędne narzędzia do uruchomienia aplikacji frontendowej. Następnie należy zainstalować zależności, skopiować pliki aplikacji frontendowej do kontenera oraz przeprowadzić ewentualne konfiguracje, takie jak ustawienie zmiennej środowiskowej, w przypadku gdy jest to wymagane przez aplikację. Wszystkie te kroki powinny zostać zdefiniowane w pliku Dockerfile w sposób spójny i klarowny.

2. Budowa obrazu dla frontendu

Po przygotowaniu pliku Dockerfile dla frontendu, należy wykonać polecenie `"docker build"` w katalogu, w którym znajduje się plik Dockerfile. Polecenie to uruchomi proces budowy obrazu kontenera frontendowego na podstawie zdefiniowanych instrukcji w pliku Dockerfile. Podczas tego procesu Docker pobierze niezbędne zależności, skopiuje pliki aplikacji frontendowej i przeprowadzi konfiguracje zdefiniowane w pliku Dockerfile.

```

# Wybierz bazowy obraz, np. Node.js
FROM node:latest

# Utwórz katalog roboczy w kontenerze
WORKDIR /app

# Skopiuj pliki package.json i package-lock.json
COPY package*.json ./

# Zainstaluj zależności aplikacji
RUN npm install

# Skopiuj pozostałe pliki aplikacji
COPY . .

# Ustaw port, na którym będzie nasłuchiwać aplikacja
EXPOSE 3000

# Uruchom aplikację frontendową
CMD ["npm", "start"]

```

Rys. 3 Struktura pliku Dockerfile dla aplikacji frontendowej

```

# Przejdź do katalogu, w którym znajduje się plik Dockerfile dla frontendu
cd [ścieżka-do-katalogu-frontendu]

# Wykonaj polecenie docker build, aby zbudować obraz
docker build -t nazwa-obrazu-frontend .

```

Rys. 4 Budowa obrazu dla aplikacji frontendowej

3. Przygotowanie pliku Dockerfile dla backendu

Analogicznie do aplikacji frontendowej, należy przygotować plik Dockerfile dla backendu, w którym zdefiniujemy instrukcje dotyczące budowy obrazu kontenera backendowego. Podobnie jak wcześniej, należy określić bazowy obraz, instalować niezbędne zależności, skopiować pliki aplikacji backendowej do kontenera i przeprowadzić ewentualne konfiguracje, takie jak ustawienie zmiennych środowiskowych czy konfiguracje sieciowe.

4. Budowa obrazu dla backendu

Po przygotowaniu pliku Dockerfile dla backendu, należy wykonać polecenie "docker build" w odpowiednim katalogu, aby zbudować obraz kontenera backendowego. Proces budowy obrazu polega na wykonaniu zdefiniowanych instrukcji w pliku Dockerfile, w tym instalacji zależności, kopiowania plików aplikacji i konfiguracji.

```

# Wybierz bazowy obraz, np. Node.js
FROM node:latest

# Utwórz katalog roboczy w kontenerze
WORKDIR /app

# Skopiuj pliki package.json i package-lock.json
COPY package*.json ./

# Zainstaluj zależności aplikacji
RUN npm install

# Skopiuj pozostałe pliki aplikacji
COPY . .

# Ustaw port, na którym będzie nasłuchiwać aplikacja
EXPOSE 8000

# Uruchom aplikację backendową
CMD ["npm", "start"]

```

Rys. 5 Struktura pliku Dockerfile dla aplikacji backendowej

```

# Przejdź do katalogu, w którym znajduje się plik Dockerfile dla backendu
cd [ścieżka-do-katalogu-backend]

# Wykonaj polecenie docker build, aby zbudować obraz
docker build -t nazwa-obrazu-backend .

```

Rys. 6 Budowa obrazu dla aplikacji backendowej

5. Uruchomienie kontenera backendu

Uruchamiamy kontener backendu, używając polecenia "docker run" z odpowiednimi parametrami. W poleceniu należy określić nazwę kontenera, mapowanie portów, tak aby aplikacja backendowa była dostępna z poziomu hosta, oraz połączenie z utworzoną wcześniej siecią. Możemy również przekazać ewentualne zmienne środowiskowe wymagane przez aplikację backendową.

6. Uruchomienie kontenera frontendu

Analogicznie do backendu, uruchamiamy kontener frontendu przy użyciu polecenia "docker run". Podajemy odpowiednie parametry, takie jak nazwa kontenera, mapowanie portów oraz połączenie z tą samą siecią, co backend. Również tutaj możemy przekazać ewentualne zmienne środowiskowe, jeśli są wymagane przez aplikację frontendową.

```
# Uruchom kontener backendowy
docker run -d --name nazwa-kontenera-backend -p 8000:8000 nazwa-obrazu-backend

# Uruchom kontener frontendowy
docker run -d --name nazwa-kontenera-frontend -p 3000:3000 nazwa-obrazu-frontend
```

Rys. 7: Uruchomienie kontenera backendowego i frontendowego

3. Podsumowanie

Jakie są główne korzyści z używania Dockera w rozwoju i wdrażaniu aplikacji?

Docker to narzędzie, które umożliwia rozwój i wdrażanie aplikacji. Zapewnia izolację aplikacji i ich zależności, działając w oddzielnym środowisku. To eliminuje problemy związane z różnicami konfiguracji systemów operacyjnych.

Tworzenie obrazów z kompletnymi projektami ułatwia przenoszenie projektów i wprowadzanie pojedynczych zmian. Docker ułatwia pracę zespołową, umożliwiając wspólne korzystanie z tego samego środowiska. Automatyzacja przygotowania środowiska przez Docker eliminuje problemy związane z modyfikacją składu zespołu.

Gdzie szukać więcej informacji o Dockerze?

W celu poszerzenia wiedzy na temat Docker'a rekomenduje się skoncentrować na oficjalnej dokumentacji dostępnej na stronie "docs.docker.com". Istnieje również aktywna społeczność Docker na stronie głównej, gdzie można znaleźć fora dyskusyjne, blogi, artykuły i materiały szkoleniowe (<https://www.docker.com/community/>).

Dodatkowo warto rozważyć skorzystanie z kursów dotyczących Docker'a dostępnych na platformie Udemy lub skorzystanie z bezpłatnych materiałów edukacyjnych dostępnych na platformie YouTube, gdzie liczni twórcy udostępniają wartościowe treści na temat Docker'a.

Ważne jest, aby pamiętać, że zasoby edukacyjne związane z Dockerem są głównie dostępne online. Konieczne jest sprawdzenie daty umieszczenia materiałów, aby uniknąć korzystania z przestarzałych informacji, ponieważ Docker jest ciągle rozwijany. Należy zwrócić szczególną uwagę na aktualność materiałów dotyczących aplikacji okienkowej Dockera, zmiany w CLI są mniej drastyczne dla użytkownika.

Bibliografia

Docker Documentation. [Online]. Dostępne: <https://docs.docker.com>.

Docker Community. [Online]. Dostępne: <https://www.docker.com/community>.

YouTube. [Online]. Dostępne: <https://www.youtube.com>.

IBM Documentation. [Online]. Dostępne: <https://www.ibm.com>

Artykuł "Monitorowanie i zarządzanie kontenerem Docker jest łatwe dzięki tym 8 narzędziom CLI" [Online]. Dostępne:

<https://newsblog.pl/monitorowanie-i-zarzadzanie-kontenerem-docker-jest-latwe-dzieki-tym-8-narzedziom-cli/>

DockerHub. [Online]. Dostępne: <https://hub.docker.com>

Artykuł "How to Create Docker Image with Dockerfile" [Online]. Dostępne:

<https://phoenixnap.com/kb/>

Artykuł "Docker" [Online]. Dostępne: <https://devopedia.org>

Artykuł "Docker: kontener, podstawowe operacje" [Online]. Dostępne: <https://sii.pl>

Artykuł "Docker on Android   " [Online]. Dostępne:

<https://gist.github.com/FreddieOliveira/efe850df7ff3951cb62d74bd770dce27>