Docker: Przewodnik po Podstawowych Poleceniach i Integracji z Frameworkami

Kacper Renkel

September 5, 2024

Contents

1 Wprowadzenie do Dockera

Docker to platforma, która umożliwia deweloperom i administratorom systemów tworzenie, wdrażanie i uruchamianie aplikacji w kontenerach. Kontenery pozwalaja na spójne środowisko uruchomieniowe niezależnie od infrastruktury.

2 Podstawowe Polecenia Dockera

2.1 Zarzadzanie Obrazami

- **docker pull** - Pobieranie obrazu z Docker Hub.
- **docker images** - Wyświetlanie listy lokalnie dostepnych obrazów.
- **docker rmi** - Usuwanie obrazu.

2.2 Zarzadzanie Kontenerami

- **docker run** - Tworzenie i uruchamianie nowego kontenera.
- **docker ps** - Wyświetlanie uruchomionych kontenerów.
- **docker stop** - Zatrzymywanie kontenera.
- **docker rm** - Usuwanie zatrzymanego kontenera.

2.3 Zarzadzanie Sieciami

- **docker network create** - Tworzenie nowej sieci.
- **docker network ls** - Wyświetlanie listy sieci.
- **docker network inspect** - Wyświetlanie szczegółów sieci.
- **docker network rm** - Usuwanie sieci.

2.4 Zarzadzanie Woluminami

- **docker volume create** - Tworzenie nowego wolumenu.
- **docker volume ls** - Wyświetlanie listy woluminów.
- **docker volume inspect** - Wyświetlanie szczegółów wolumenu.
- **docker volume rm** - Usuwanie wolumenu.

3 Zarzadzanie Sieciami w Dockerze

Docker umożliwia tworzenie i zarzadzanie różnymi typami sieci, w tym bridge, host i overlay.

3.1 Tworzenie Sieci

Polecenie `docker network create` służy do tworzenia nowej sieci. Na przykład:

```
docker network create my_network
```

3.2 Inspekcja Sieci

Aby zobaczyć szczegóły sieci:

```
docker network inspect my_network
```

3.3 Usuwanie Sieci

Usuwanie sieci za pomoca:

```
docker network rm my_network
```

4 Uruchamianie Terminala Kontenera

Aby uruchomić interaktywna sesje w kontenerze:

```
docker exec -it <container_id> /bin/bash
```

5 Polecenia używane w Dockerfile

Dockerfile jest plikiem tekstowym, który zawiera wszystkie instrukcje potrzebne do zbudowania obrazu Dockera.

- **FROM** - Określa bazowy obraz, na którym bedzie budowany nowy obraz. Może być z Docker Hub lub z innego rejestru.
- **RUN** - Uruchamia komendy w czasie budowania obrazu. Służy do instalowania zależności i konfiguracji systemu.
- **CMD** - Definiuje domyślne polecenie do uruchomienia po starcie kontenera. Może być nadpisane przez `docker run`.
- **EXPOSE** - Deklaruje otwarty port, na którym aplikacja bedzie dostepna.
- **ENV** - Ustawia zmienne środowiskowe wewnatrz obrazu.
- **COPY** i **ADD** - Kopiuje pliki do obrazu. **COPY** służy do kopiowania plików z hosta, a **ADD** obsługuje także pobieranie z URL.
- **WORKDIR** - Ustawia katalog roboczy kontenera, w którym beda wykonywane wszystkie nastepne komendy.
- **ENTRYPOINT** - Ustawia domyślne polecenie, które nie może być nadpisane. Czesto używane z **CMD**.

- **ARG** - Definiuje zmienne używane tylko w trakcie budowania obrazu.
- **VOLUME** - Definiuje punkt montowania woluminu.
- **LABEL** - Służy do dodawania metadanych do obrazu, takich jak wersje, informacje o autorze.
- **USER** - Określa użytkownika, w kontekście którego beda wykonywane polecenia w kontenerze.
- **ONBUILD** - Definiuje instrukcje, które maja być uruchamiane w czasie budowy obrazu pochodnego.

6 Polecenia używane w docker-compose

'docker-compose.yml' jest plikiem definiujacym konfiguracje wielu kontenerów dla aplikacji wielousługowych.

- **version** - Wersja używanego API Compose, np. '3.8'.
- **services** - Definiuje liste usług (kontenerów) w aplikacji.
- **image** - Określa obraz do uruchomienia kontenera.
- **build** - Budowanie obrazów z Dockerfile.
- **volumes** - Montowanie woluminów. Pozwala na współdzielenie plików miedzy hostem a kontenerem.
- **networks** - Konfiguracja sieci miedzy kontenerami.
- **environment** - Ustawianie zmiennych środowiskowych w kontenerze.
- **ports** - Mapowanie portów miedzy hostem a kontenerem.
- **depends_on** - Określa kolejność uruchamiania kontenerów.
- **command** - Zmienia domyślne polecenie uruchamiane w kontenerze.
- **restart** - Definiuje polityke restartu kontenera, np. `always`, `on-failure`.
- **links** - Tworzenie połaczeń miedzy kontenerami.
- **extra_hosts** - Dodawanie wpisów do pliku /etc/hosts w kontenerze.
- **secrets** - Ustawia tajemnice używane przez kontenery, np. hasła.
- **configs** - Zarzadza konfiguracjami do użycia w kontenerach.

7 Automatyczne przeładowanie kontenera po zmianach w plikach

Aby osiagnać automatyczne przeładowanie kontenerów w Dockerze po zmianach w plikach, można skorzystać z mechanizmu woluminów Dockera. Woluminy pozwalaja na synchronizacje plików miedzy systemem plików hosta a kontenerem, dzieki czemu zmiany w kodzie źródłowym na hoście sa od razu odzwierciedlane w kontenerze. Dzieki temu, po odpowiednim skonfigurowaniu kontenerów, zmiany w plikach beda automatycznie przeładowywać aplikacje wewnatrz kontenera.

8 Przykłady konfiguracji dla różnych frameworków

Poniżej znajduja sie przykłady konfiguracji plików Dockerfile i docker-compose.yml dla różnych popularnych frameworków i jezyków.

8.1 Node.js i NestJS

Dockerfile:

```
FROM node:14
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "app.js"]
```

docker-compose.yml:

```
version: '3'
services:
  node-app:
    build: .
    ports:
      - "3000:3000"
    volumes:
      - .:/app
```

8.2 Angular, Vue.js, React

Dockerfile dla Angulara:

```
FROM node:14
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 4200
CMD ["npm", "start"]
```

docker-compose.yml:

```
version: '3'
services:
  frontend:
    build: .
    ports:
      - "4200:4200"
    volumes:
      - .:/app
```

8.3 Spring Boot

Dockerfile:

```
FROM openjdk:11
WORKDIR /app
COPY . .
RUN ./mvnw package
EXPOSE 8080
CMD ["java", "-jar", "target/myapp.jar"]
```

docker-compose.yml:

```
version: '3'
services:
  spring-app:
    build: .
    ports:
      - "8080:8080"
    volumes:
      - .:/app
```

8.4 .NET Core

Dockerfile:

```
FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS base
WORKDIR /app
COPY . .
EXPOSE 5000
CMD ["dotnet", "myapp.dll"]
```

docker-compose.yml:

```
version: '3'
services:
  dotnet-app:
    build: .
    ports:
      - "5000:5000"
    volumes:
```

- .:/app

9 Podsumowanie

Docker jest potężnym narzedziem, które umożliwia łatwe tworzenie, wdrażanie i uruchamianie aplikacji w kontenerach. Dzieki użyciu Dockerfile i docker-compose, można zautomatyzować procesy budowy i uruchamiania aplikacji oraz zarzadzać ich środowiskiem. Integracja z popularnymi frameworkami pozwala na łatwe tworzenie przenośnych aplikacji, które moga być uruchamiane w różnych środowiskach bez zmian w kodzie.