

# docker

**Tomasz Adamczyk**

**Nr albumu 243217**

**Informatyka 3. rok, 2017/2018**

# Czym jest Docker?

Docker to platforma dla programistów oraz administratorów służąca do tworzenia, wdrażania i uruchamiania aplikacji rozproszonych.

Pozwala umieścić aplikację oraz jej zależności w lekkim, przenośnym, wirtualnym kontenerze, który można uruchomić na prawie każdym serwerze.

Dzięki temu zyskujemy gwarancję, że oprogramowanie będzie działać zawsze tak samo, niezależnie od architektury systemu oraz środowiska uruchomieniowego.

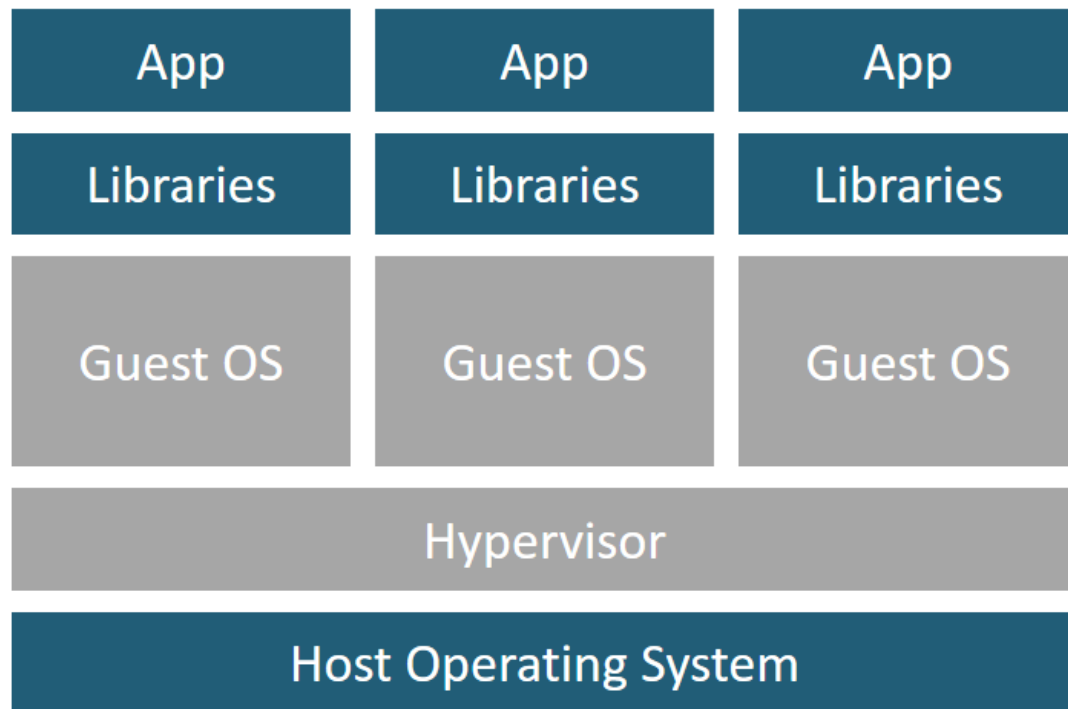
# Docker versus VM

Na pierwszy rzut mogłoby się wydawać, że kontenery i wirtualne maszyny pełnią tę samą rolę i realizują podobne zadania. Istnieją jednak istotne różnice pomiędzy Dockerem a Virtual Machines:

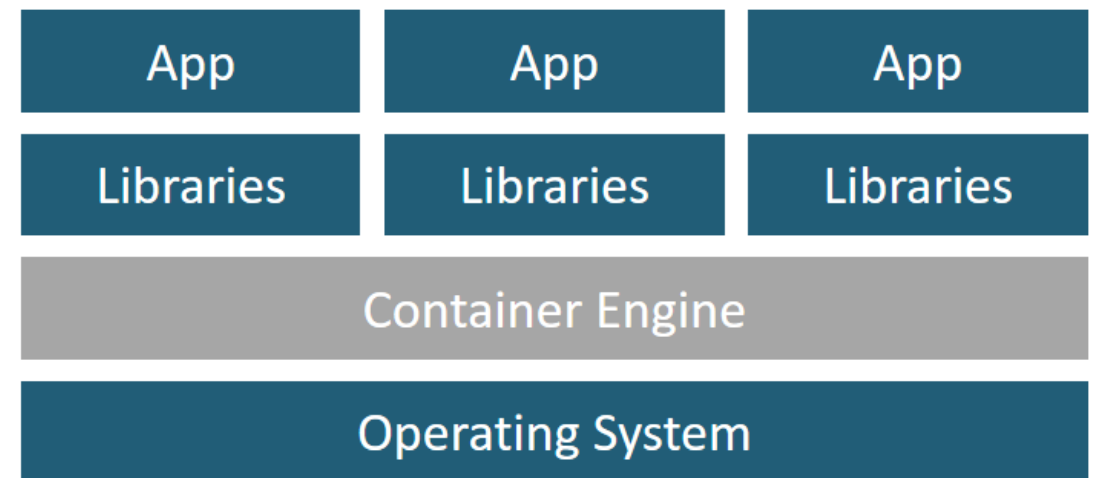
- Kontenery są lżejsze;
- Brak konieczności instalacji OS w kontenerach;
- Kontenery mają mniejsze wymagania względem zasobów PC;
- Większa przenośność kontenerów.

# Porównanie architektury VM oraz kontenerów Dockera

## Virtual Machines



## Containers



# Przydatne linki

Instalator Docker

Instalator Docker for Windows

Instalator Docker Compose

Repozytorium obrazów Docker

Instrukcja konfiguracji pliku Dockerfile

Instrukcja konfiguracji pliku docker-compose.yml

# Przygotowanie środowiska

Aby zainstalować Dockera i móc z niego korzystać na swoim komputerze należy pobrać instalator (link na 5 stronie).

Jeśli korzystamy z systemu operacyjnego opartego na jądrze systemu Linux wystarczy sam Docker. W przypadku systemów z rodziny Windows należy również pobrać Docker for Windows (link na 5 stronie).

W momencie posiadania odpowiednich instalatorów należy zainstalować za ich pomocą wymagane oprogramowania.

Jeśli będziemy chcieli umieścić stworzony przez nas obraz systemu w repozytorium obrazów Dockera to musimy założyć konto w platformie (link do repozytorium na 5 stronie).

# Prosty przykład

Używając Dockera możemy stworzyć środowisko zarówno dla prostych aplikacji konsolowych, jak i aplikacji webowych.

Poniżej przedstawię jak stworzyć prostą aplikację konsolową, którą będziemy uruchamiać używając kontenera.



1. Tworzymy nowy katalog JavaApp i przechodzimy do niego
2. Tworzymy dwa pliki: HelloWorld.java oraz Dockerfile

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop
$ mkdir JavaApp

Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop
$ cd JavaApp/

Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp
$ touch HelloWorld.java Dockerfile

Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp
$ ls
Dockerfile  HelloWorld.java
```




3. Otwieramy dowolny edytor tekstowy, w którym nadpiszemy zawartość plików HelloWorld.java oraz Dockerfile.  
Mój wybór padł na Visual Studio Code

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp  
$ code .
```

#### 4. Wpisujemy poniższy kod do pliku HelloWorld.java

```
1 public class HelloWorld {  
2     private static void SayHello() {  
3         System.out.println("Hello World from Docker container!");  
4     }  
5  
6     public static void main(String[] args) {  
7         System.out.println("Say hello from method:");  
8         SayHello();  
9     }  
10 }
```

## 5. Wpisujemy poniższy kod do pliku Dockerfile

 Dockerfile x

```
1 FROM java:8
2 COPY . /var/www/java
3 WORKDIR /var/www/java
4 RUN javac HelloWorld.java
5 CMD ["java", "HelloWorld"]
```

6. Wykonujemy poniższe polecenie, dzięki któremu zbudujemy obraz aplikacji. Wykonując poniższe polecenie następuje budowanie obrazu poprzez kroki zapisane w pliku Dockerfile

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp
$ docker build -t helloworld-java-app .
Sending build context to Docker daemon 3.072kB
Step 1/5 : FROM java:8
---> d23bdf5b1b1b
Step 2/5 : COPY . /var/www/java
---> 9b7da7e9cb1d
Step 3/5 : WORKDIR /var/www/java
Removing intermediate container 7434733a86a5
---> bb579e2d16e8
Step 4/5 : RUN javac HelloWorld.java
---> Running in f890412e35d5
Removing intermediate container f890412e35d5
---> aa1fd8573db0
Step 5/5 : CMD ["java", "HelloWorld"]
---> Running in 8f06928a4d8e
Removing intermediate container 8f06928a4d8e
---> e218e0794f06
Successfully built e218e0794f06
Successfully tagged helloworld-java-app:latest
```

7. Następnie uruchamiamy obraz naszej aplikacji.  
W tym momencie utworzy się nowy kontener naszej aplikacji

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp  
$ docker run helloworld-java-app  
Say hello from method:  
Hello World from Docker container!
```

# Co dalej?

Utworzyliśmy już obraz naszej aplikacji oraz uruchomiliśmy go w jednym kontenerze.

Jeśli będziemy chcieli utworzyć kolejny kontener wykorzystujący obraz naszej aplikacji to wystarczy ponownie wykonać polecenie przedstawione w 7. kroku powyższej instrukcji.

Natomiast jeśli chcemy ponownie uruchomić wcześniej stworzony kontener należy wykonać poniższe polecenia.

1. Wpisujemy poniższe polecenie i szukamy na liście kontenerów wpisu z nazwą obrazu identyczną do nazwy naszej aplikacji. Następnie zapamiętujemy pierwsze trzy znaki CONTAINER ID lub jego całą nazwę

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp
```

```
$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
181e5ff357ac	helloworld-java-app	"java HelloWorld"	8 minutes ago	Exited (0) 4 minutes ago
2caa77e1456a	tadamczyk/dockerized-akteam:latest	"rails server"	8 weeks ago	Exited (1) 8 weeks ago
57cbb57ccb8	8fa51c76e055	"rails s"	2 months ago	Exited (0) 2 months ago



2. Wpisujemy poniższe polecenie, dzięki któremu uruchomimy ponownie stworzony wcześniej kontener. W tym przypadku wykorzystujemy trzy pierwsze znaki CONTAINER ID, czyli „181”

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp
$ docker container start -i 181
Say hello from method:
Hello World from Docker container!
```

# Co dalej?

A co jeśli chcemy umieścić obraz naszej super aplikacji w repozytorium Dockera?

1. Jeśli posiadamy już konto w repozytorium Dockera to musimy się zalogować lokalnie na swoje konto poprzez wykonanie polecenia „docker login”

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp
$ winpty docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username (tadamczyk): tadamczyk
Password:
Login Succeeded
```

2. Następnie ponownie budujemy aplikację, tylko tym razem dodajemy przed nazwą aplikacji nazwę użytkownika z serwisu repozytorium Dockera oraz opcjonalnie tag po nazwie obrazu

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp
$ docker build -t tadamczyk/helloworld-java-app:latest .
Sending build context to Docker daemon 3.072kB
Step 1/5 : FROM java:8
--> d23bdf5b1b1b
Step 2/5 : COPY . /var/www/java
--> Using cache
--> 9b7da7e9cb1d
Step 3/5 : WORKDIR /var/www/java
--> Using cache
--> bb579e2d16e8
Step 4/5 : RUN javac HelloWorld.java
--> Using cache
--> aa1fd8573db0
Step 5/5 : CMD ["java", "HelloWorld"]
--> Using cache
--> e218e0794f06
Successfully built e218e0794f06
Successfully tagged tadamczyk/helloworld-java-app:latest
```

### 3. Następnie „wypychamy” naszą aplikację do repozytorium za pomocą poniższego polecenia

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp
$ docker push tadamczyk/helloworld-java-app:latest
The push refers to repository [docker.io/tadamczyk/helloworld-java-app]
313653a2356e: Preparing
05795dba90ef: Preparing
35c20f26d188: Preparing
c3fe59dd9556: Preparing
6ed1a81ba5b6: Preparing
a3483ce177ce: Preparing
ce6c8756685b: Preparing
30339f20ced0: Preparing
0eb22bfb707d: Preparing
a2ae92ffcd29: Preparing
a3483ce177ce: Waiting
ce6c8756685b: Waiting
30339f20ced0: Waiting
0eb22bfb707d: Waiting
a2ae92ffcd29: Waiting
```

# Obraz naszej aplikacji jest już dostępny w repozytorium Dockera

The screenshot shows the Docker Hub profile page for user 'tadamczyk'. The page has a dark blue header with the Docker logo, a search bar, and navigation links: Dashboard, Explore, Organizations, Create, and the user's profile. Below the header, there's a sub-header with the user's name 'tadamczyk', a dropdown menu, and tabs for Repositories, Stars, and Contributed. A 'Private Repositories: Using 0 of 1' status is also shown. The main section is titled 'Repositories' and includes a 'Create Repository +' button. A search bar prompts the user to 'Type to filter repositories by name'. Below this, two repository entries are listed. The first entry is 'tadamczyk/dockerized-akteam' with 0 stars and 14 pulls. The second entry, 'tadamczyk/helloworld-java-app', is highlighted with a red border and shows 0 stars and 1 pull. Each entry includes a user profile picture, the repository name, its visibility (public), and a 'DETAILS' link.

Repository Name	Stars	Pulls	Details
tadamczyk/dockerized-akteam public	0 STARS	14 PULLS	> DETAILS
tadamczyk/helloworld-java-app public	0 STARS	1 PULLS	> DETAILS

Jeśli chcemy „ściągnąć” obraz naszej aplikacji należy wykonać poniższe polecenie:

```
Tomasz@Tomasz-WINDOWS MINGW64 ~/Desktop/JavaApp
$ docker pull tadamczyk/helloworld-java-app
Using default tag: latest
latest: Pulling from tadamczyk/helloworld-java-app
Digest: sha256:a68e0932cde9039260a7e3f6c49463863e6ca61ea4de9abb777331e517a86c73
Status: Image is up to date for tadamczyk/helloworld-java-app:latest
```



# Podsumowanie

Przedstawiony przykład jest oczywiście bardzo trywialny, lecz w przejrzysty sposób obrazuje, jak można utworzyć obraz swojej aplikacji, jak „stawiać” i używać kontenery korzystające z obrazu aplikacji oraz jak wysłać i ściągnąć obraz aplikacji wykorzystując repozytorium Dockera.

Poniżej znajduje się zbiór najprzydatniejszych poleceń linii komend dla Dockera.

komenda	opis
<b>docker version</b>	wyświetla zainstalowaną lokalną wersję Dockera
<b>docker login</b>	umożliwia zalogowanie się lokalnie na repozytorium
<b>docker images</b>	wyświetla wszystkie lokalne obrazy Dockera
<b>docker ps -a</b>	wyświetla wszystkie lokalne kontenery Dockera
<b>docker rm &lt;container-id&gt;</b>	usuwa lokalny kontener o podanym ID
<b>docker rmi &lt;image&gt;</b>	usuwa lokalny obraz o podanej nazwie lub ID
<b>docker build -t &lt;username&gt;/&lt;image-name&gt;: &lt;tag-name&gt; .</b>	buduje obraz aplikacji znajdującej się w lokalnym roboczym katalogu
<b>docker run &lt;image-name&gt;</b>	tworzy i uruchamia nowy kontener aplikacji o podanej nazwie obrazu
<b>docker container start -i &lt;container-id&gt;</b>	uruchamia kontener o podanym ID
<b>docker commit &lt;container-id&gt; &lt;username&gt;/&lt;image-name&gt;:&lt;tag-name&gt;</b>	tworzy nowy, lokalny obraz aplikacji bazującej na kontenerze o podanym ID
<b>docker push &lt;username&gt;/&lt;repo-name&gt;: &lt;tag-name&gt;</b>	„wypycha” obraz lokalnej aplikacji do repozytorium Dockera
<b>docker pull &lt;username&gt;/&lt;repo-name&gt;: &lt;tag-name&gt;</b>	„ściąga” obraz aplikacji z repozytorium Dockera i zapisuje go lokalnie