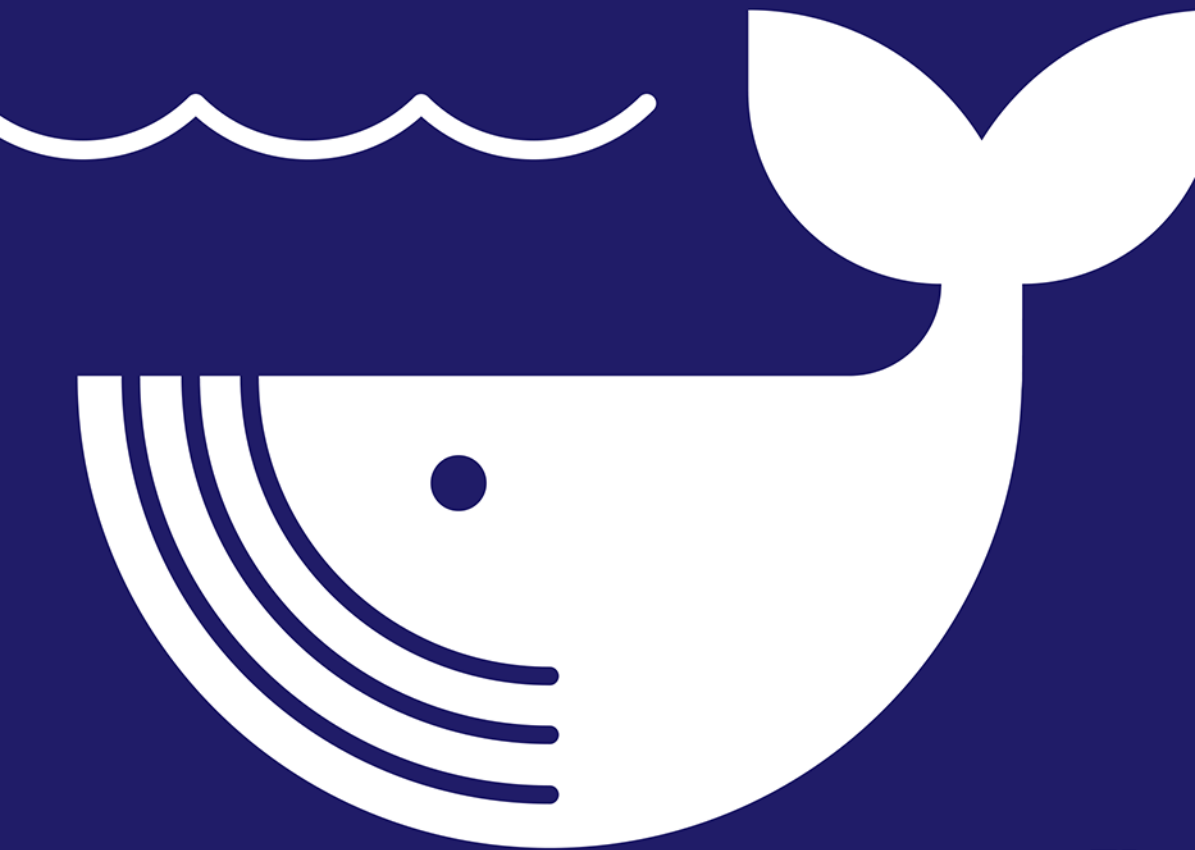
Piotr Chudzik

Konteneryzacja z wykorzystaniem Dockera

PODSTAWY



Helion

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/abckon
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-289-0576-4

Copyright © Helion S.A. 2024

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

ROZDZIAŁ 4.

Pierwszy kontener

Uruchomienie kontenera

Aby uruchomić nasz pierwszy kontener, potrzebujemy obrazu (który omówiliśmy w poprzednim rozdziale) oraz jednego polecenia: `docker run [OPCJE] <obraz>[:<tag>] [POLECENIE]`.

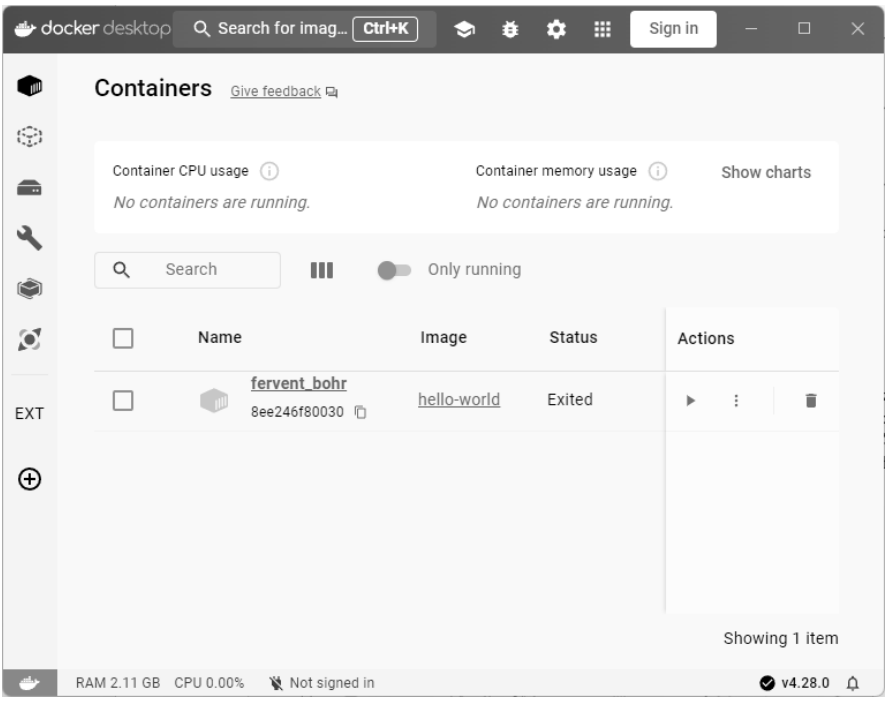
Samo polecenie `run` to potężne narzędzie. Pozwala ono nie tylko uruchomić kontener na podstawie wskazanego przez nas obrazu, ale również wprowadzić odpowiednie zmiany w jego strukturze, czy nawet ustawić odpowiednie limity. W tej książce zapoznam Cię z najważniejszymi opcjami dla polecenia `docker run`. Na początku skupmy się na uruchomieniu naszego pierwszego polecenia. Uruchomimy kontener na podstawie obrazu *hello-world* (w jego przypadku możemy pozwolić sobie na tag *latest*). Będzie to również ostateczny sprawdzian, czy nasze środowisko działa w 100% prawidłowo. W konsoli wykonuję zatem polecenie `docker run hello-world`.

Wykorzystałem tutaj obraz *hello-world*, którego nie pobrałem wcześniej. Możemy dzięki temu zauważyć, że polecenie `run` automatycznie będzie szukać w dostępnych dla Dockera rejestrach wskazanego obrazu, po znalezieniu pobierze go, a następnie uruchomi kontener.

```
cmd: docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:53641cd209a4fecfc68e21a99871ce8c6920b2e7502df0a20671c6fccc73a7c6
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.
    (amd64)
 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/
```

Kup książkę Poleć książkę

Poniżej polecenia możemy zobaczyć efekt działania kontenera. W przypadku *hello-world* jest to jedynie prosta informacja na temat poleceń wykonanych przez Dockera. Jeżeli widzisz taki sam komunikat u siebie, oznacza to, że Twoje środowisko na pewno jest prawidłowo skonfigurowane. Cały powyższy komunikat to efekt działania naszego kontenera! Jeżeli korzystasz z Docker for Desktop, to w zakładce *Containers* zobaczysz nowy wpis:



Wyświetlenie i status kontenerów

Status każdego kontenera możemy w prosty sposób zweryfikować za pomocą polecenia `docker ps [-a | --all]` (w przypadku Docker for Desktop możemy zweryfikować kontenery w zakładce *Containers*).

Polecenie zwróci nam tabelę aktualnie działających kontenerów oraz kilka dodatkowych informacji:

- *CONTAINER ID* — ID kontenera;
- *IMAGE* — wykorzystany obraz;
- *COMMAND* — polecenie, z którym uruchomiono kontener;
- *CREATED* — kiedy został utworzony;
- *STATUS* — stan kontenera;
- *PORTS* — komunikacja sieciowa, czyli przekierowane i wystawione porty;
- *NAMES* — nazwa kontenera.

Kup książkę Poleć książkę

W tym miejscu warto zwrócić uwagę na ID oraz nazwę kontenera. Są one unikalne w naszym środowisku, więc dokładnie jeden kontener może posiadać wybrane ID oraz nazwę. Zapewne się zastanawiasz, gdzie znajduje się nasz kontener uruchomiony wcześniej? Otóż zakończył swoje działanie i aktualnie jest wyłączony. Możemy go wyświetlić, wykonując poprzednie polecenie z opcją `-a` lub `--all`.

```
cmd: docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
eee45f11e607   hello-world   "/hello"   15 minutes ago   Exited (0) 15 minutes ago             vibrant_meitner
```

Po dodaniu opcji możemy zauważyć, że na liście widnieje nasz kontener ze statusem *Exited (0)* oraz informacją, kiedy ten status wystąpił. Należy wiedzieć, że kontenery działają, dopóki ich polecenie (*command*) wykonuje swoje zadanie. Możemy powiedzieć, że to polecenie jest warunkiem działania kontenera. Dopóki wykonuje swoje zadanie, kontener jest w statusie *Up*. Jeżeli polecenie zakończy swoje działanie (jak w przypadku *hello-world*, gdy skrypt */hello* zakończył swoją pracę), to kontener przejdzie w status *Exited*, a wartość 0 informuje nas, że samo działanie kontenera zakończyło się prawidłowo i zgodnie ze skryptem. Jeżeli kontener zakończyłby swoją pracę z powodu błędu lub niekontrolowanego przerwania działania, to zapewne zobaczysz to w kolumnie *STATUS*, gdzie przy *Exited* będzie liczba inna niż zero.

Nazwa i własne polecenie kontenera

W poprzednim rozdziale uruchomiliśmy nasz pierwszy kontener na podstawie *hello-world*. Obraz ten posiada informację, że ma uruchomić kontener poprzez wykonanie skryptu */hello*. Każdy obraz posiada domyślne polecenie, z którym uruchamiany jest kontener. Najczęściej tego polecenia nie można nadpisać (np. w przypadku obrazu narzędzia nginx, MySQL), ale dla obrazów „bazowych" ta opcja jak najbardziej jest dostępna. Aby uruchomić kontener z własnym poleceniem, musimy je podać po nazwie obrazu, tak samo jakbyśmy pisali je w konsoli. Wykorzystamy tutaj obraz *ubuntu:22.04*. Wykonuję zatem polecenia:

```
docker run ubuntu:22.04
docker run ubuntu:22.04 echo 'Witaj, Helion!'
docker ps -a
```

```
cmd: docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                      PORTS
    NAMES
9b83c4bf3209   ubuntu:22.04   "echo 'Witaj, Helion…"   3 seconds ago    Exited (0) 3 seconds ago
    flamboyant_meninsky
bb3c4901ecc4   ubuntu:22.04   "/bin/bash"              15 seconds ago   Exited (0) 14 seconds ago
    cool_dewdney
```

Wykonałem tutaj dwukrotnie polecenie `docker run`. W pierwszym przypadku uruchomiłem po prostu obraz *ubuntu:22.04*, natomiast w drugim dodałem na końcu polecenie `echo „Witaj, Helion!"`, co spowodowało wyświetlenie tekstu *Witaj, Helion!* na konsoli. Następnie wyświetliłem status naszych kontenerów. Oba zakończyły swoje działanie prawidłowo. Pierwszy z nich ma zarejestrowane polecenie `„/bin/bash"` — jest to domyślne polecenie obrazu *ubuntu*, natomiast drugi kontener już wykonał moje polecenie związane z wyświetleniem tekstu na konsoli, dlatego w zakładce *COMMAND* widnieje polecenie `echo 'Witaj, Helion!'`.

Kup książkę Poleć książkę

Wspominałem również, że kontenery mają unikatowe ID oraz nazwę, która jest nadawana przez Dockera. Nie jesteśmy w stanie zmienić ID kontenera, natomiast możemy ustawić własną nazwę dla niego podczas uruchomienia. Służy do tego opcja `--name`, po której podajemy naszą nazwę — oczywiście musi ona być unikatowa w naszym środowisku kontenerów!

Teraz uruchomię własny kontener z nazwą *my_sleep* za pomocą polecenia `docker run --name my_sleep ubuntu:22.04 sleep 15`.

```
cmd: docker run --name my_sleep \
> ubuntu:22.04 sleep 15
cmd: docker ps -a
CONTAINER ID   IMAGE          COMMAND      CREATED          STATUS                      PORTS     NAMES
372470d74d31   ubuntu:22.04   "sleep 15"   35 seconds ago   Exited (0) 19 seconds ago             my_sleep
```

Widzimy, że w działaniu kontenera nic się nie zmieniło, a w kolumnie *NAMES* widnieje ustawiona przeze mnie nazwa *my_sleep*. Jeżeli postaram się uruchomić kolejny kontener, wykorzystując tę nazwę, to zobaczę na konsoli błąd:

```
cmd: docker run --name my_sleep ubuntu:22.04 sleep 15
docker: Error response from daemon: Conflict. The container name "/my_sleep" is already in use by containe
r "372470d74d317d41d3633a32ce836367d9f7ce792778707061476c476d4310e6". You have to remove (or rename) that
container to be able to reuse that name.
See 'docker run --help'.
```

Docker pilnuje, aby nazwy kontenerów były unikatowe. Za każdym razem, kiedy wykonujemy polecenie `run`, Docker sprawdza, czy podana przez nas nazwa widnieje na liście kontenerów. Jeżeli tak, to otrzymujemy powyższy błąd, zalecający nam zmianę nazwy lub usunięcie kontenera o wykorzystywanej nazwie.

Interaktywny kontener oraz tryb detach

Do tej pory uruchamialiśmy kontenery bezpośrednio w konsoli, powodując, że konsola (okno) była zablokowana podczas działania kontenera, a wszelkie przerwanie działania powodowałoby zakończenie działania kontenera. Oczywiście w środowisku produkcyjnym zależy nam na uruchomieniu kontenera w taki sposób, aby nie był on zależny od takich elementów jak aktywna sesja użytkownika. W tej sytuacji bardzo pomocna jest opcja `-d`, oznaczająca *detach*. Dzięki temu nasz kontener zostanie uruchomiony w tle. Wykorzystajmy tę opcję podczas uruchamiania kontenera aplikacji nginx: `docker run -d nginx:1.25`.

```
cmd: docker run -d nginx:1.25
1dfa9740f72805ce3b1a7666cb6194e787d353d9e26d78626d8995d38d94c44b
cmd: docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS        PORTS     NAMES
1dfa9740f728   nginx:1.25   "/docker-entrypoint.…"   2 seconds ago   Up 1 second   80/tcp    elastic_sinou
ssi
```

Jak możemy zauważyć, w odpowiedzi otrzymałem jedynie informację w postaci ID kontenera (jeżeli użyłbym opcji `--name`, to pojawiłaby się nazwa). Natomiast po weryfikacji listy kontenerów za pomocą polecenia `docker ps` widzimy nasz kontener w statusie *Up*. Czyli nasz kontener działa i wykonuje powierzone mu zadanie, a my może wprowadzać kolejne polecenia do terminala.

Kup książkę Poleć książkę

Podczas uruchamiania pierwszych kontenerów zapewne przyszło Ci na myśl, czy można w jakikolwiek sposób wejść w interakcję z uruchomionym kontenerem, skoro według informacji zawartych w rozdziale 1., „Wirtualizacja a konteneryzacja", jest to izolowane środowisko (pudełko). W tym pomogą nam dwie kolejne opcje: `-i`, która pozwala uruchomić terminal w wersji interaktywnej (możemy wysyłać informacje do kontenera), oraz `-t`, która emuluje terminal (pozwala na uruchomienie konsoli kontenera). Najczęściej obie te opcje idą w parze, dlatego często możesz zobaczyć w internecie polecenia z opcją `-it`, będącą ich połączeniem.

Pora na przetestowanie opcji i poznanie kontenera „od środka". Wykonujemy w konsoli polecenie `docker run -it --name test_console debian:bookworm`.

```
cmd: docker run -it --name test_console debian:bookworm
root@c9a965164203:/#
```

Widzimy, że okno terminala się zmieniło. Teraz jestem zalogowany na użytkownika *root*, a nazwa maszyny (hosta) to zestaw znaków i cyfr. Jako że skorzystałem z obrazu Debiana, to mam dostępną większość poleceń dla tego systemu operacyjnego, a struktura plików jest praktycznie identyczna.

```
root@c9a965164203:/# ls --color=auto
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@c9a965164203:/# exit
exit
cmd: docker ps -a
CONTAINER ID   IMAGE             COMMAND   CREATED          STATUS                      PORTS     NAMES
c9a965164203   debian:bookworm   "bash"    27 seconds ago   Exited (0) 2 seconds ago              test_
console
```

Widzimy, że mamy dobrze nam znane katalogi, takie jak np. mnt, bin czy usr, oraz możemy zweryfikować użytkownika poleceniem `id`. Oczywiście inne polecenia również są dostępne. Należy pamiętać, że pomimo uruchomienia w trybie interaktywnym kontenera polecenie `exit` zatrzymuje główny proces (polecenie `bash`), zatem nasz kontener został ustawiony w statusie *Exited (0)*.

Podsumowanie

W tym rozdziale poznaliśmy zasady działania kontenera. Wiemy już, że kontener jest uruchamiany na podstawie wskazanego przez nas obrazu i funkcjonuje, dopóki działa jego główny proces. W przypadku niektórych obrazów (głównie tych „bazowych") możemy ustawić własną instrukcję, na podstawie której działa kontener, podając ją w poleceniu `run` po obrazie. Dodatkowo wiemy, jak weryfikować status naszego kontenera, nadać mu własną nazwę i uruchomić go w formie interaktywnej oraz „detach". W następnym rozdziale przedstawię podstawowe polecenia, które pozwolą Ci sprawnie zarządzać istniejącymi już kontenerami.

Kup książkę Poleć książkę

Kup książkę Poleć książkę


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

Twórz, testuj i wdrażaj: konteneryzacja na Twoich zasadach

Pierwsza była wirtualizacja: oprogramowanie symuluje istnienie zasobów logicznych korzystających z zasobów fizycznych. Po niej przyszła konteneryzacja, polegająca na tworzeniu obrazów — kopii danych — zawierających wszystkie pliki potrzebne do uruchomienia danej aplikacji. Środowiska produkcyjne z obu korzystają równolegle, ale to konteneryzacja stała się swojego rodzaju rewolucją w sektorze IT. Pozwoliła bowiem na sprawniejsze wdrażanie mikroserwisów, a także na optymalizację kosztów działania wielu aplikacji.

Jeśli dotąd nie nadarzyła się okazja, by zgłębić temat, zrób to jak najszybciej, ponieważ umiejętność konteneryzacji jest obecnie wymagana na bardzo wielu stanowiskach, od programistów i inżynierów danych po specjalistów DevOps i administratorów. Korzystając z tej książki, zapoznasz się z Dockerem — najpopularniejszym narzędziem do konteneryzacji.

Podczas pracy z naszym poradnikiem między innymi:

- **Zbudujesz swój pierwszy kontener**
- **Nauczysz się zarządzać jego zawartością**
- **Stworzysz sieć kontenerów**
- **Dowiesz się, czym jest Dockerfile**
- **Opanujesz podstawy Docker Compose**

Helion

helion.pl

HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶

ISBN 978-83-289-0576-4

9 788328 905764

Cena: 39,90 zł

Piotr Chudzik

Absolwent Politechniki Łódzkiej, zawodowo specjalizuje się w technologiach *big data* i administrowaniu nowoczesnymi środowiskami IT (korzystającymi między innymi z *cloud computing*, konteneryzacji czy IaC — *infrastructure as code*). Pracuje jako wykładowca na Uniwersytecie Łódzkim, gdzie przekazuje studentom wiedzę na temat Linuksa, baz danych i programowania w języku Python. Jest również trenerem wielu technologii z zakresu DevOps i *data engineering* dla firm. Zawsze otwarty na nowe doświadczenia i wiedzę, którą w przyszłości mógłby się podzielić z innymi. Interesuje się grami komputerowymi, światem nowych technologii i fantastyką. Jest fanem serii o wiedźminie i uniwersum *Warhammera*.