

Russ McKendrick, Scott Gallagher

Docker

Programowanie
aplikacji dla
zaawansowanych

Wydanie II

Tytuł oryginału: Mastering Docker - Second Edition

Tłumaczenie: Konrad Matuk

ISBN: 978-83-283-4308-5

Copyright © Packt Publishing 2017. First published in the English language under the title 'Mastering Docker - Second Edition – (9781787280243)'

Polish edition copyright © 2018 by Helion SA
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

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 Wydawnictwo HELION 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 Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/dockaz>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/dockaz.zip>

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

O autorach	9
O recenzentach	11
Wstęp	13
Rozdział 1. Docker — wprowadzenie	17
Czym jest Docker?	17
Programiści	18
Administratorzy	19
Firmy	21
Różnice pomiędzy dedykowanymi hostami, maszynami wirtualnymi i Dockerem	22
Instalacja Dockera	23
Instalacja w systemie Linux (Ubuntu 16.04)	24
Instalacja w systemie macOS	25
Instalacja w systemie Windows 10 Professional	27
Starsze systemy operacyjne	28
Klient Dockera w wierszu poleceń	29
Ekosystem Dockera	32
Podsumowanie	33
Rozdział 2. Tworzenie obrazów kontenerów	35
Plik Dockerfile — wprowadzenie	35
Szczegółowa analiza pliku Dockerfile	37
Dobre praktyki pracy z plikami Dockerfile	41
Budowanie obrazów Dockera	42
Polecenie docker build	42
Korzystanie z utworzonego wcześniej kontenera	44
Budowanie od podstaw	48
Zmienne środowiskowe	50
Umieszczanie zmiennych środowiskowych w pliku Dockerfile	50
Czas wszystko połączyć ze sobą	50
Podsumowanie	56

Rozdział 3. Przechowywanie obrazów i ich dystrybucja	57
Repozytorium Docker Hub	57
Panel główny	58
Przycisk Explore	59
Przycisk Organizations	60
Menu Create	60
Profil i ustawienia	61
Strona Stars	62
Automatyzacja budowy obrazu	63
Ładowanie własnych obrazów	68
Serwis Docker Store	70
Rejestr Docker Registry	71
Docker Registry — informacje ogólne	71
Wdrażanie własnego rejestru	72
Rejestr Docker Trusted Registry	74
Niezależne rejestry	75
Quay	75
Rejestr Amazon EC2 Container Registry	78
Microbadger	79
Podsumowanie	82
Rozdział 4. Zarządzanie kontenerami	83
Polecenia służące do obsługi kontenerów Dockera	84
Podstawy	84
Komunikowanie się z kontenerami	88
Dzienniki i informacje o procesach	91
Ograniczenia zasobów	93
Stany kontenerów i pozostałe polecenia	95
Usuwanie kontenerów	97
Obsługa sieci i wolumenów	100
Sieć	100
Wolumeny Dockera	107
Podsumowanie	112
Rozdział 5. Docker Machine	113
Wprowadzenie do narzędzia Docker Machine	113
Docker Machine i wdrażanie lokalnych hostów Dockera	114
Uruchamianie hostów Dockera w chmurze	119
DigitalOcean	119
Amazon Web Services	121
Więcej o sieciowych możliwościach Dockera	124
Stosowanie innych bazowych systemów operacyjnych	128
System CoreOS w chmurze DigitalOcean	128
System RancherOS w maszynie VirtualBox	130
Podsumowanie	131

Rozdział 6. Docker Compose	133
Wprowadzenie do Docker Compose	133
Uruchamianie aplikacji za pomocą narzędzia Docker Compose	135
Plik YAML narzędzia Docker Compose	137
Plik YAML aplikacji Mobycounter	137
Aplikacja do głosowania	139
Polecenia Docker Compose	146
Up i PS	147
Config	148
Pull, build i create	148
Start, stop, restart, pause i unpause	149
Top, logs i events	149
Exec i run	151
Scale	152
Kill, rm i down	153
Podsumowanie	154
Rozdział 7. Docker Swarm	155
Docker Swarm — instalacja	156
Role Docker Swarm	156
Menedżer Swarm	157
Wykonawca Swarm	157
Korzystanie z trybu Docker Swarm	158
Tworzenie klastra	158
Dołączanie wykonawców	160
Listy węzłów	161
Zarządzanie klastrem	161
Promowanie hosta roboczego	164
Degradacja węzła menedżera	165
Drenaż węzła	166
Usługi i stosy Docker Swarm	168
Usługi	168
Stosy	171
Kasowanie klastra Swarm	173
Równoważenie obciążeń, nakładki i tworzenie harmonogramów	174
Równoważenie obciążeń wejściowych	174
Nakładki sieciowe	175
Tworzenie harmonogramu	176
Podsumowanie	176
Rozdział 8. Portainer	177
Historia prac nad narzędziem Portainer	177
Uruchamianie narzędzia Portainer	178
Korzystanie z narzędzia Portainer	180
Panel główny	181
Szablony aplikacji	181

Kontenery	183
Obrazy	187
Sieci i wolumeny	190
Zdarzenia	190
Docker	191
Portainer i Docker Swarm	191
Tworzenie klastra	192
Usługa Portainer	193
Różnice związane z pracą w klastrze	194
Podsumowanie	199
Rozdział 9. Rancher	201
Instalacja i konfiguracja uwierzytelniania	201
Instalacja	202
Konfiguracja uwierzytelniania	204
Tworzenie stada	207
Uruchamianie stosów	209
Stosy definiowane przez użytkownika	210
Podgląd właściwości kontenerów	216
Katalog	217
Usuwanie stada	217
Inne klastry	218
Podsumowanie	221
Rozdział 10. Usługa Docker Cloud	223
Zakładanie konta	224
Łączenie kont	225
DigitalOcean	226
Amazon Web Services	227
Uruchamianie węzłów	231
Uruchamianie stosu	234
Tryb Swarm	238
Docker dla Amazon Web Services	239
Podsumowanie	244
Rozdział 11. Bezpieczeństwo platformy Docker	245
Bezpieczeństwo kontenerów	245
Zalety	246
Host Dockera	246
Zaufane źródła obrazów	247
Polecenia Dockera	247
Polecenie run	247
Polecenie diff	249
Dobre praktyki	250
Dobre praktyki pracy w Dockerze	250
Zalecenia organizacji Center for Internet Security	251

Aplikacja Docker Bench Security	252
Uruchamianie narzędzia w systemach macOS i Windows	253
Uruchamianie narzędzia w systemie Linux Ubuntu	253
Analiza zwracanych informacji	255
Docker Bench — podsumowanie	260
Skanowanie zabezpieczeń Dockera	260
Niezależne usługi poprawiające bezpieczeństwo	262
Quay	262
Clair	263
Podsumowanie	264
 Rozdział 12. Przepływ zadań w platformie Docker	 265
Docker i prace programistyczne	265
Monitorowanie	277
Rozszerzanie na zewnętrzne platformy	286
Instalator Tectonic	286
Platforma Heroku	289
Usługa Amazon Elastic Container Service	290
Jak wygląda środowisko produkcyjne?	291
Hosty Dockera	291
Obsługa klastrów	292
Rejestry obrazów	293
Podsumowanie	294
 Rozdział 13. Dalsze kroki z Dockerem	 295
Wykrywanie usług	295
Consul	295
Narzędzie etcd	304
Interfejs Docker API	304
Projekt Moby Project	306
Własny wkład w rozwój Dockera	308
Rozwój kodu	308
Pomoc innym	309
Inny wkład w rozwój Dockera	310
Podsumowanie	310
 Skorowidz	 311

Docker

— wprowadzenie

Witaj w drugim wydaniu książki *Docker. Programowanie aplikacji dla zaawansowanych*. W rozdziale 1. znajdziesz podstawowe wiadomości dotyczące Dockera, które powinieneś opanować przed przystąpieniem do lektury kolejnych rozdziałów. Jeżeli nie masz zbyt dużego doświadczenia w pracy z Dockerem, to nie przejmuj się, po przeczytaniu tego rozdziału dowiesz się wszystkiego, co niezbędne, aby pójść dalej. Po przeczytaniu całej książki powinieneś zostać mistrzem Dockera potrafiącym implementować Dockera we własnych środowiskach, będącym w stanie zbudować i uruchomić aplikacje w kontenerach.

W tym rozdziale opiszemy następujące wysokopoziomowe zagadnienia:

- Czym jest Docker?
- Czym się różni Docker od standardowych maszyn wirtualnych?
- Jak wygląda proces instalacji Dockera?
- Jakie polecenia obsługuje Docker?
- Jak wygląda ekosystem Dockera?

Czym jest Docker?

Definiowanie, czym jest Docker, zacznijmy od cytatu ze strony internetowej tej platformy:

Docker to obecnie najpopularniejsza platforma przeznaczona do konteneryzacji aplikacji. Umożliwia ona rozwiązanie problemów typu „a u mnie działa” powstających podczas tworzenia aplikacji przez różnych programistów. Administratorzy korzystają z Dockera w celu uruchamiania na jednym serwerze wielu aplikacji w oddzielonych od siebie

kontenerach i zarządzania nimi. Docker jest używany w wielu firmach podczas pracy zgodnie z metodologią programowania zwinnego, umożliwiającą szybsze tworzenie nowych funkcji, a także zwiększenie bezpieczeństwa w aplikacjach serwerowych systemów Linux i Windows.

Te słowa nie oddają wszystkiego. Oto liczby opisujące popularność Dockera przedstawione przez prezesa firmy będącej twórcą Dockera, Bena Goluba, podczas otwarcia konferencji DockerCon w 2017 r.:

- 14 milionów hostów Dockera;
- 900 tysięcy aplikacji korzystających z Dockera;
- wzrost o 77 000% liczby miejsc pracy związanych z Dockerem;
- 12 miliardów pobrań obrazów;
- 3300 osób zaangażowanych w rozwój platformy.

Te liczby robią wrażenie, jeżeli weźmiemy pod uwagę, że Docker to technologia obecna na rynku dopiero od trzech lat. Materiał wideo z tym wystąpieniem możesz obejrzeć na stronie <https://www.slideshare.net/Docker/dockercon-2017-general-session-day-1-ben-golub>.

Spróbujmy zrozumieć problemy, które rozwiązuje Docker. Zaczniemy od problemów programistów.

Programiści

Poniższy mem to chyba najlepsza ilustracja problemu „na moim komputerze działa bez problemów”. Obraz ten zaczął kilka lat temu pojawiać się na prezentacjach, forach dyskusyjnych i w kanałach Slacka:

NA DEVIE DZIAŁAŁO



SKĄD WZIAŁ SIĘ TEN PROBLEM?

To dość śmieszna ilustracja częstego problemu, z którym spotykałem się osobiście.

Problem

Pomimo zachowania najlepszych praktyk metodologii DevOps wciąż możliwe jest istnienie różnic pomiędzy środowiskiem programistycznym i produkcyjnym.

Programista korzystający z wersji PHP przystosowanej do pracy w systemie macOS prawdopodobnie będzie pracował w innej wersji niż serwer produkcyjny pracujący pod kontrolą systemu Linux. Nawet jeżeli obie wersje PHP są identyczne, to środowiska te mogą charakteryzować się inną konfiguracją, innymi przywilejami dotyczącymi korzystania z plików itd. Oto jedno potencjalne źródło problemu.

Programista wdrażający kod w środowisku produkcyjnym musi sobie odpowiedzieć na następujące pytanie: Czy lepiej jest skonfigurować środowisko produkcyjne tak, aby jego konfiguracja odpowiadała środowisku deweloperskiemu, czy lepiej skonfigurować środowisko deweloperskie tak, aby było skonfigurowane tak samo jak środowisko produkcyjne?

Idealnie byłoby, aby wszystko było skonfigurowane w identyczny sposób. Dotyczy to laptopa programisty i produkcyjnego serwera, ale to dość utopijna wizja, którą trudno zrealizować. Każdy pracuje w nieco inny sposób. Zgodność platform trudno jest uzyskać nawet wtedy, gdy wszystko obsługuje ta sama osoba, a w przypadku zespołu składającego się z setki programistów jest to praktycznie niemożliwe.

Rozwiązanie problemu za pomocą Dockera

Programista korzystający z platformy Docker przeznaczonej dla systemu Windows lub Mac może obudować swój kod kontenerem, który może być zdefiniowany samodzielnie albo utworzony za pomocą pliku Dockerfile (zagadnienie to opiszemy w rozdziale 2. „Tworzenie obrazów kontenerów”) lub pliku Docker Compose (zagadnienie to opiszemy w rozdziale 6. „Docker Compose”).

Dzięki temu rozwiązaniu możliwe jest korzystanie z wybranego środowiska programistycznego i preferowanych sposobów pracy nad kodem. Podczas lektury kolejnych rozdziałów przekonasz się o tym, że instalacja i obsługa Dockera wcale nie są trudne. Korzystanie z niego jest o wiele łatwiejsze od dbania o zgodność środowisk nawet za pomocą mechanizmów automatyzacji.

Administratorzy

Pracę administratora wykonywałem dłużej, niż powinienem, i z mojego doświadczenia wynika, że opisany niżej problem pojawia się dość regularnie.

Problem

Żałuję, że opiekujesz się pięcioma serwerami: trzy z nich to serwery sieciowe o dość zrównoważonym obciążeniu, a dwa pozostałe to serwery bazodanowe działające w konfiguracji serwer nadrzędny i serwer podrzędny, na których uruchomiona jest aplikacja 1. Stosem oprogramowania

zarządzasz automatycznie za pomocą narzędzia takiego jak Puppet lub Chef. Narzędzie to jest również używane do konfiguracji wszystkich pięciu serwerów.

Wszystko działało poprawnie, dopóki nie dostałeś zadania wdrożenia aplikacji 2 na tych samych serwerach, na których działa aplikacja 1. To na pozór nic problematycznego. Możesz zmodyfikować konfigurację narzędzia Puppet lub Chef w celu dodania nowych użytkowników i wirtualnych hostów oraz zaciągnięcia nowego kodu, ale po chwili orientujesz się, że aplikacja 2 wymaga wyższej wersji oprogramowania od aplikacji 1.

Okazuje się, że aplikacja 1 nie chce współpracować z nowym stosem oprogramowania, a aplikacja 2 nie jest wstecznie kompatybilna.

Istnieje kilka tradycyjnych rozwiązań tego problemu, ale żadne z nich nie jest idealne:

1. Możesz poprosić o dodatkowe serwery. Technicznie rzecz biorąc, jest to najbezpieczniejsze rozwiązanie, ale wiąże się to z poniesieniem wydatków na dodatkową infrastrukturę.
2. Możesz zmienić architekturę aplikacji. Wyjęcie jednego z serwerów sieciowych i bazodanowych będącego kopią lub poprawiającego obciążenia robocze i wdrożenie na nim stosu aplikacji 2 może wydawać się również dobrym rozwiązaniem. Jednakże sprawi ono, że aplikacja 2 będzie podatna na awarie, a stabilność pracy aplikacji 1 może ulec pogorszeniu (prawdopodobnie z jakiegoś powodu działała ona na trzech serwerach sieciowych i dwóch serwerach bazodanowych).
3. Możesz podjąć próbę równoległej instalacji nowego stosu oprogramowania. Oczywiście jest to możliwe i może wydawać się dobrym tymczasowym rozwiązaniem umożliwiającym uruchomienie nowego projektu, ale wdrożenie krytycznej poprawki zabezpieczeń któregoś ze stosów może być wtedy bardzo kłopotliwe.

Rozwiązanie problemu za pomocą Dockera

Zastosowanie Dockera to doskonale rozwiązanie tego problemu. Gdyby aplikacja 1 była uruchomiona na trzech serwerach w kontenerach, to mógłbyś nawet zdublować te kontenery, co pozwoliłoby na gładkie wdrażanie nowych wersji tej aplikacji „w locie”, bez ograniczania jej dostępności.

Wdrożenie aplikacji 2 w tym środowisku sprowadza się po prostu do uruchomienia większej liczby kontenerów w trzech hostach i skonfigurowania systemu równoważenia obciążenia. Wdrażając kontenery, nie musisz się przejmować problemami wynikającymi z wdrażania, konfigurowania i zarządzania dwiema wersjami tego samego stosu oprogramowania na tym samym serwerze.

Dokładnie taki scenariusz opiszemy w rozdziale 6. „Docker Compose”.

Firmy

Firmy cierpią z powodu opisanych wcześniej problemów, ponieważ zatrudniają programistów i administratorów, ale problemy te w skali firmy wydają się znacznie większe i wiąże się z nimi większe ryzyko.

Problem

Przerwa w działaniu aplikacji wiąże się z poniesieniem kosztów finansowych, a także utratą reputacji. Firmy muszą więc testować każde wdrożenie przed jego udostępnieniem. W związku z tym nowe funkcje i poprawki są zawieszane z powodu:

- Tworzenia i konfiguracji środowisk testowych.
- Wdrażania aplikacji w nowych środowiskach.
- Wykonywania zaplanowanych testów i poprawiania aplikacji i konfiguracji aż do poprawnego przejścia testów.
- Zmian i omawiania żądań modyfikacji aplikacji podczas wdrażania jej kodu w środowisku produkcyjnym.

Proces ten może trwać od kilku dni do tygodni, a nawet miesięcy. Zależy to od złożoności aplikacji i ryzyka wprowadzanego przez zmiany. Proces ten jest niezbędny do zapewnienia ciągłości i dostępności na poziomie technologicznym, ale wiąże się z potencjalnymi zagrożeniami natury biznesowej. Co, jeżeli konkurencja udostępni podobne lub, o zgrozo, identyczne funkcje przed Tobą?

Takie zdarzenie może mieć tak samo negatywny wpływ na sprzedaż i reputację firmy jak udostępnienie nieprzetestowanego wdrożenia.

Rozwiązanie problemu za pomocą Dockera

Zacznijmy od stwierdzenia, że Docker nie zwalnia z wykonywania opisanego wcześniej procesu, ale jak już pisaliśmy, ułatwia wykonywanie pracy w sposób ciągły. Dzięki niemu programiści mogą pracować w tak samo skonfigurowanym kontenerze jak kontener wdrożony na produkcji, co z pewnością uprości metodologię jego testowania.

Gdy np. programista ładuje kod, który według niego działa w jego lokalnym środowisku (środowisku, w którym przeprowadzane prace programistyczne), to możliwe jest takie skonfigurowanie narzędzia testującego, aby uruchomiło te same kontenery w celu przeprowadzenia automatycznego testu. Po zakończeniu testowania kontenery mogą zostać usunięte w celu zwolnienia zasobów dla kolejnego zestawu testów. Technika ta sprawia, że proces testowania i związane z nim procedury są o wiele bardziej elastyczne i możliwe staje się wielokrotne korzystanie z tego samego środowiska, co zwalnia z konieczności ponownego wdrażania lub odtwarzania serwerów z obrazów przed przeprowadzeniem kolejnego zestawu testów.

Opisane rozwiązanie usprawnia proces wdrażania nowych aplikacji w środowisku produkcyjnym.

Im szybciej zostanie on przeprowadzony, tym wcześniej będziesz mógł pewnie wprowadzić nowe funkcje lub poprawki, nie dając się wyprzedzić konkurencji.

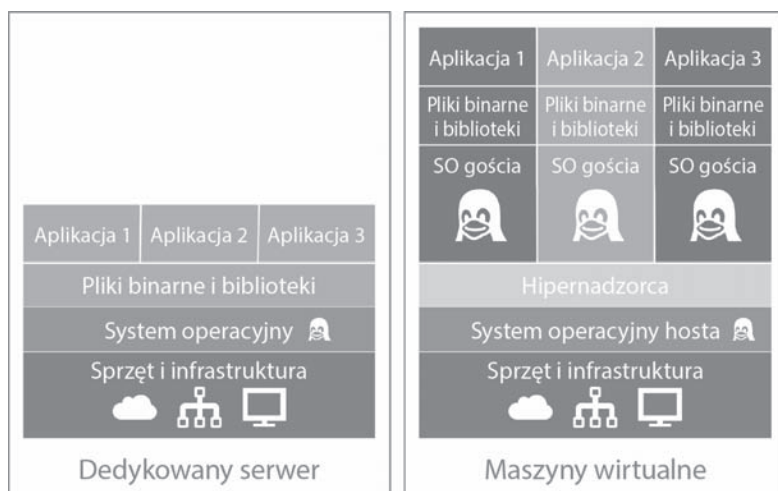
Różnice pomiędzy dedykowanymi hostami, maszynami wirtualnymi i Dockerem

Opisaliśmy problemy, które Docker miał rozwiązać. Teraz zajmiemy się opisem szczegółów mechanizmu działania tej platformy.

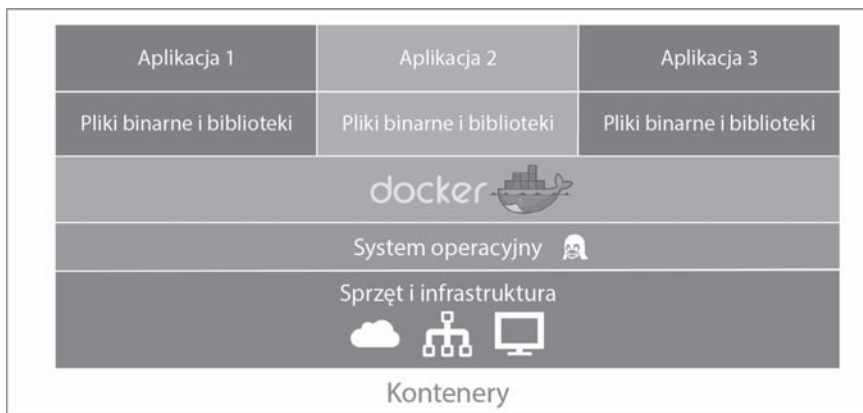
Docker to system zarządzania, który ułatwia obsługę kontenerów **LXC (Linux Containers)**. Rozwiązanie to umożliwia tworzenie obrazów w środowiskach wirtualnych na laptopach programistów i uruchamianie na nich poleceń. Docker umożliwia wykonywanie na kontenerach uruchomionych lokalnie tych samych operacji, które będą przeprowadzane w środowisku produkcyjnym.

W związku z tym nie ma konieczności robienia czegoś w sposób inny w środowisku lokalnym i w środowisku serwerowym. Porównajmy różnice pomiędzy kontenerami Dockera i typowymi środowiskami maszyn wirtualnych.

Poniższy schemat przedstawia różnice pomiędzy dedykowanym, fizycznym serwerem i serwerem uruchomionym w maszynie wirtualnej:



Jak widzisz, w przypadku dedykowanego serwera wszystkie trzy aplikacje współdzielił ten sam stos oprogramowania wyróżniony kolorem pomarańczowym. Maszyny wirtualne umożliwiają uruchomienie trzech aplikacji na dwóch zupełnie innych stosach oprogramowania. Na poniższym schemacie pokazano te same aplikacje (wyróżnione kolorem pomarańczowym i zielonym) uruchomione w kontenerach Dockera:



Schemat ten przedstawia największą zaletę Dockera — brak potrzeby uruchamiania całego nowego systemu operacyjnego podczas dodawania nowych kontenerów, co pozwala na zredukowanie ich rozmiaru. Docker korzysta z jądra systemu Linux hosta. Większość wersji systemu Linux korzysta ze standardowych modeli jądra, a więc Docker może działać w dystrybucjach takich jak Red Hat, CentOS i Ubuntu. W związku z tym systemem hosta może być prawie każda dystrybucja systemu Linux. Aplikacja w kontenerze jest traktowana tak, jakby system operacyjny nie był kompletny, a posiadał tylko niezbędne pliki binarne, takie jak np. menedżer pakietów, serwer Apache-PHP i biblioteki potrzebne do uruchomienia aplikacji.

W przypadku wcześniejszej ilustracji pomarańczowa aplikacja mogła działać w dystrybucji Red Hat, a zielona w dystrybucji Debian, ale żadna z tych dystrybucji nie musiałaby być systemem hosta. Kolejną zaletą Dockera jest rozmiar obrazów. Obrazy nie muszą zawierać dużych elementów, takich jak jądro systemu operacyjnego lub cały system operacyjny. Dzięki temu są one bardzo małe i łatwe do przesyłania.

Instalacja Dockera

Instalator Dockera to pierwszy element tego ekosystemu, który należy uruchomić w swoim lokalnym środowisku, a także na serwerach. Oto lista środowisk, w których można zainstalować Dockera:

- Linux (różne dystrybucje);
- Apple macOS;
- Windows 10 Professional.

Ponadto instalator Dockera może zostać uruchomiony w chmurze takiej jak np. Amazon Web Services, Microsoft Azure i DigitalOcean. Docker integruje się w różny sposób z systemami operacyjnymi, dlatego dla każdego systemu przewidziano inny instalator. Podstawowym środowiskiem pracy Dockera jest Linux, a więc jeżeli korzystasz z tego systemu, jego uruchomienie nie będzie stanowiło żadnego problemu. Docker działa nieco inaczej w systemach macOS i Windows 10. Wynika to z tego, że korzysta on wewnętrznie z rozwiązań systemu Linux.

Przyjrzyjmy się procesowi instalacji Dockera na komputerze stacjonarnym pracującym pod kontrolą systemu Linux — Ubuntu 16.04, a następnie systemów macOS i Windows 10.

Instalacja w systemie Linux (Ubuntu 16.04)

Zgodnie z tym, co wcześniej pisaliśmy, instalacja Dockera w systemie Linux jest najprostsza. W celu zainstalowania Dockera uruchom poniższe polecenia w sesji Terminala:

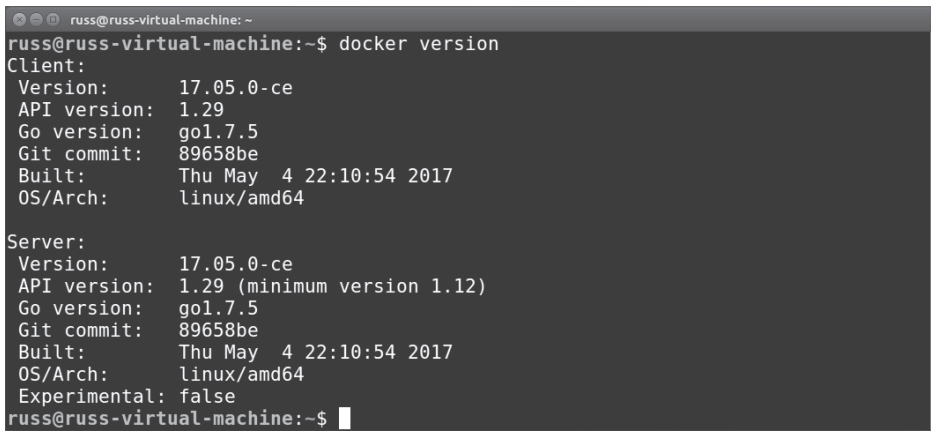
```
$ curl -sSL https://get.docker.com/ | sh
$ sudo systemctl start docker
```

Uruchomienie tych poleceń spowoduje pobranie, zainstalowanie i skonfigurowanie najnowszej wersji Dockera. Program zostanie pobrany bezpośrednio z serwera jego autorów. W momencie pisania tej książki najnowszą wersją Dockera instalowaną przez oficjalny skrypt jest 17.05.

W celu sprawdzenia poprawności instalacji i działania Dockera uruchom polecenie:

```
$ docker version
```

W Terminalu powinien pojawić się komunikat podobny do następującego:



```
russ@russ-virtual-machine: ~
russ@russ-virtual-machine:~$ docker version
Client:
 Version:      17.05.0-ce
 API version:  1.29
 Go version:   gol.7.5
 Git commit:   89658be
 Built:        Thu May  4 22:10:54 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.05.0-ce
 API version:  1.29 (minimum version 1.12)
 Go version:   gol.7.5
 Git commit:   89658be
 Built:        Thu May  4 22:10:54 2017
 OS/Arch:      linux/amd64
 Experimental: false
russ@russ-virtual-machine:~$
```

W dalszej części książki będziemy korzystać z dwóch narzędzi pomocniczych, które są automatycznie instalowane wraz z Dockerem w systemach macOS i Windows. Zainstaluj je teraz, dzięki czemu nie będziesz tracić na to czasu później. Pierwszym z tych narzędzi jest Docker Machine. W celu zainstalowania go uruchom poniższe polecenia:

Aby sprawdzić, czy instalujesz najnowszą wersję narzędzia, wejdź w sekcję z wersjami znajdującą się na stronie projektu w repozytorium GitHub — <https://github.com/docker/machine/releases/>. W celu zainstalowania wersji innej niż v.0.13.0 po prostu umieść inny numer wersji w poniższym poleceniu.

```
$ curl -L https://github.com/docker/machine/releases/download/v0.13.0/
↳ docker-machine-`uname -s`-`uname -m` >/tmp/docker-machine &&
  chmod +x /tmp/docker-machine &&
  sudo cp /tmp/docker-machine /usr/local/bin/docker-machine
```

W celu pobrania i zainstalowania narzędzia Docker Compose uruchom poniższe polecenie. Tu również sprawdź, czy pobiera ono najnowszą wersję — wejdź na stronę <https://github.com/docker/compose/releases/>:

```
$ sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/
↳ docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose &&
  chmod +x /usr/local/bin/docker-compose
```

Po poprawnym zainstalowaniu narzędzia można uruchomić następujące polecenia:

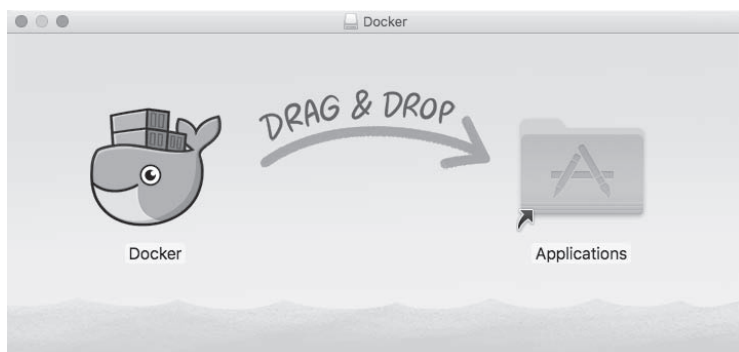
```
$ docker-compose version
$ docker-machine version
```

Instalacja w systemie macOS

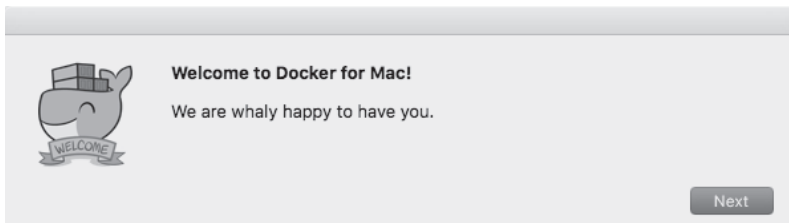
Docker w systemie macOS instalowany jest za pomocą graficznego instalatora.

Przed pobraniem instalatora upewnij się, że pracujesz w systemie Apple macOS Yosemite 10.10.3 lub nowszym. Jeżeli korzystasz ze starszej wersji tego systemu, to możesz również i w niej uruchomić Dockera — zajrzyj do sekcji „Starsze systemy operacyjne”.

Instalator możesz pobrać z serwisu Docker Store (<https://store.docker.com/editions/community/docker-ce-desktop-mac/>). Kliknij przycisk *Get Docker* (pobierz Dockera). Spowoduje to pobranie pliku DMG. Kliknij go dwukrotnie w celu zamontowania obrazu i uruchomienia go. Na ekranie Twojego komputera powinno pojawić się następujące okno:



Po przeciągnięciu ikony Dockera do folderu aplikacji kliknij ją dwukrotnie. Zostaniesz zapytany, czy chcesz otworzyć pobraną aplikację. Kliknij *Tak*, co uruchomi instalator:



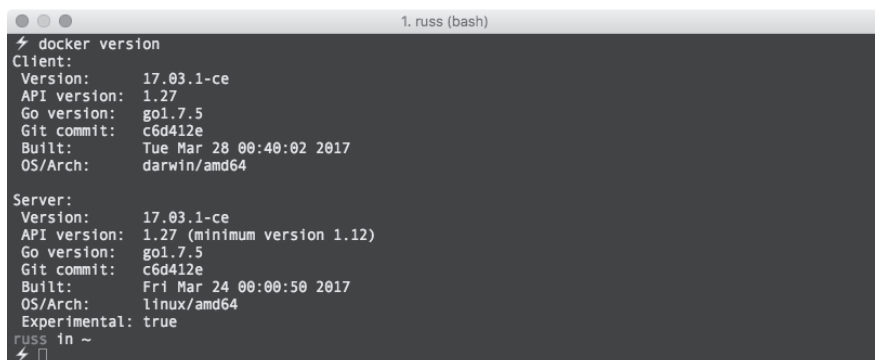
Kliknij przycisk *Next* (dalej) i wykonuj instrukcje wyświetlane na ekranie. Po zainstalowaniu i uruchomieniu Dockera jego ikona powinna pojawić się na belce widocznej w prawym górnym rogu ekranu. Kliknięcie tej ikony i wybranie opcji *About Docker* (o Dockerze) spowoduje wyświetlenie okna podobnego do poniższego:



Możesz również otworzyć okno Terminala i uruchomić w nim to samo polecenie, z którego korzystaliśmy w systemie Linux:

```
$ docker version
```

Powinno ono spowodować wyświetlenie informacji o Dockerze:



Oto polecenia, dzięki którym sprawdzisz wersje narzędzi Docker Compose i Docker Machine zainstalowanych wraz z silnikiem Docker Engine:

```
$ docker-compose version
$ docker-machine version
```

Instalacja w systemie Windows 10 Professional

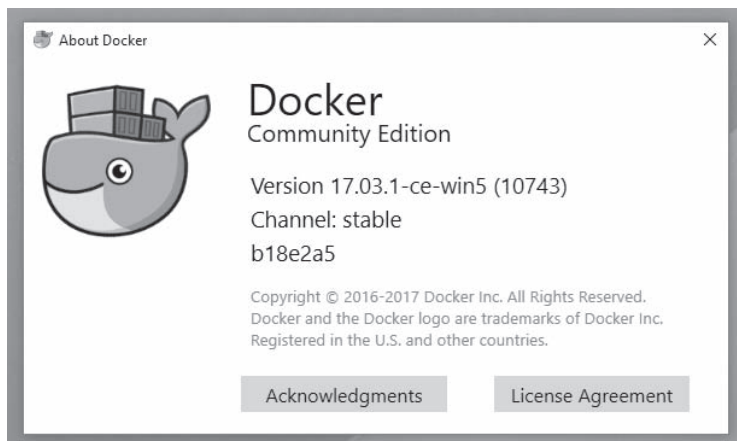
W systemie Windows, podobnie jak w systemie macOS, Docker jest instalowany również za pomocą graficznego instalatora.

Przed pobraniem instalatora upewnij się, że pracujesz w 64-bitowej wersji systemu Microsoft Windows 10 Professional lub Enterprise. Jeżeli korzystasz ze starszej wersji tego systemu lub nieobsługiwanej wersji Windows 10, zajrzyj do sekcji „Starsze systemy operacyjne”.

Docker wymaga określonej wersji systemu Windows, ponieważ korzysta z Hyper-V — hipernadzorcy wbudowanego w system Windows, umożliwiającego uruchamianie aplikacji o architekturze x86-64. Mogą to być aplikacje systemu Windows 10 Professional lub Windows Server. Wspomniany hipernadzorca wchodzi nawet w skład systemu operacyjnego konsoli Xbox One.

Instalator Dockera dla systemu Windows możesz pobrać z serwisu Docker Store <https://store.docker.com/editions/community/docker-ce-desktop-windows/>. Kliknij przycisk *Get Docker* (pobierz Dockera) w celu pobrania instalatora. Po uruchomieniu instalatora Docker zostanie automatycznie rozpakowany i zainstalowany. Opcja uruchomienia hipernadzorcy pojawi się przy pierwszym uruchomieniu Dockera.

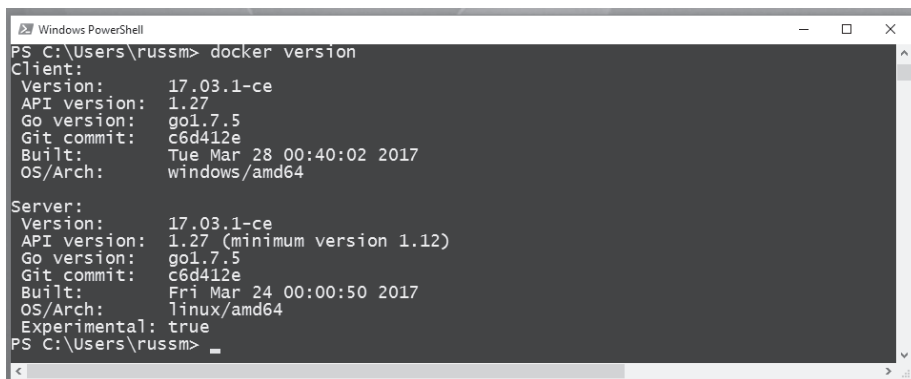
Po zakończeniu procesu instalacji ikona Dockera pojawi się w zasobniku systemowym wyświetlanym w prawym dolnym rogu ekranu. Kliknij ją i wybierz opcję *About Docker*, co spowoduje wyświetlenie okna z informacjami o zainstalowanej wersji Dockera:



Otwórz okno *PowerShell* i uruchom w nim polecenie:

```
$ docker version
```

W oknie pojawiają się informacje podobne do tych, które były wyświetlane w wersjach przeznaczonych dla systemów Mac i Linux:



```
PS C:\Users\russs> docker version
Client:
Version:      17.03.1-ce
API version:  1.27
Go version:   go1.7.5
Git commit:   c6d412e
Built:        Tue Mar 28 00:40:02 2017
OS/Arch:      windows/amd64

Server:
Version:      17.03.1-ce
API version:  1.27 (minimum version 1.12)
Go version:   go1.7.5
Git commit:   c6d412e
Built:        Fri Mar 24 00:00:50 2017
OS/Arch:      linux/amd64
Experimental: true
PS C:\Users\russs>
```

W systemie również możesz uruchomić następujące polecenia:

```
$ docker-compose version
$ docker-machine version
```

Starsze systemy operacyjne

Jeżeli korzystasz ze starszej wersji systemu macOS lub Windows, to musisz skorzystać z narzędzia Docker Toolbox. Być może zauważyłeś, że po uruchomieniu polecenia:

```
$ docker version
```

w przypadku zaprezentowanych trzech instalacji wyświetlone zostały informacje o dwóch wersjach: klienta i serwera. W przypadku instalacji w systemie Linux komunikat zawierał informacje o tym, że architektura klienta i serwera to Linux; w przypadku wersji zainstalowanej w systemie macOS komunikat informował o tym, że klient działa na architekturze Darwin, czyli uniksowym jądrze firmy Apple, a w systemie Windows klient działa na architekturze systemu Windows, ale wszystkie serwery działają według komunikatu na architekturze systemu Linux. Jak to możliwe?

Wynika to z tego, że wersje Dockera pracujące w systemach Windows i macOS pobierają i uruchamiają w tle maszynę wirtualną, w której uruchomiony jest lekki system operacyjny Alpine Linux. Maszyna ta korzysta z własnych bibliotek Dockera, które łączą się z wbudowanym hipernadzorcą wybranego środowiska. W przypadku systemu macOS jest to wbudowany moduł Hypervisor Framework (<https://developer.apple.com/documentation/hypervisor>), a w systemie Windows jest to Hyper-V (<https://www.microsoft.com/en-gb/cloud-platform/server-virtualization>).

Istnieją wersje Dockera przeznaczone dla starszych wersji systemów macOS i Windows. Nie korzystają one z wbudowanych hipernadzorców. Zamiast nich używają programu VirtualBox w charakterze hipernadzorcy — lokalny klient łączy się z uruchomionym w tym programie serwerem Linux.

VirtualBox to otwarte narzędzie wirtualizacyjne przeznaczone dla architektur x86 i AMD64/Intel64 opracowane przez firmę Oracle. Działa ono w hostach Windows, Linux, Macintosh i Solaris. Obsługuje wiele wirtualizowanych systemów operacyjnych z rodzin Linux, Unix i Windows. Więcej informacji na temat programu VirtualBox znajdziesz na stronie <https://www.virtualbox.org>.

Więcej informacji na temat projektu Docker Toolbox znajdziesz na stronie <https://www.docker.com/get-docker>, z której możesz również pobrać programy instalacyjne systemów macOS i Windows.

W tej książce zakładamy, że zainstalowałeś najnowszą wersję Dockera w systemie Linux lub korzystasz z Dockera w wersji dla systemu macOS lub Windows. Co prawda instalacje wykonane przy użyciu narzędzia Docker Toolbox powinny obsługiwać polecenia opisane w tej książce, ale pracując w takiej instalacji, możesz spotkać się z problemami wynikającymi z braku uprawnień do plików podczas montowania w kontenerze danych z lokalnej maszyny.

Klient Dockera w wierszu poleceń

Po zainstalowaniu Dockera czas przypomnieć sobie jego najważniejsze polecenia. Zaczniemy od tych, które są używane najczęściej, przejdziemy do poleceń, które są używane podczas zarządzania obrazami, a na koniec zajmiemy się poleceniami dotyczącymi kontenerów.

Docker niedawno zmienił strukturę klienta obsługiwanego z poziomu wiersza poleceń. Polecenia zostały pogrupowane w sposób bardziej logiczny. Zabieg ten wynikał z szybkiego rozwijania się platformy i krzyżowania poleceń. W tej książce będziemy korzystać z nowej struktury. Więcej informacji na temat zmian wprowadzonych w kliencie obsługiwanym za pomocą wiersza poleceń znajdziesz w artykule <https://blog.docker.com/2017/01/whats-new-in-docker-1-13/>.

Zaczniemy od najważniejszego polecenia — tego wyświetlającego pomoc. W celu uruchomienia pomocy uruchom polecenie:

```
$ docker help
```

Na ekranie pojawi się lista wszystkich dostępnych poleceń Dockera wraz z krótkimi opisami ich funkcji. W celu uzyskania bardziej szczegółowych informacji dotyczących wybranego polecenia wpisz:

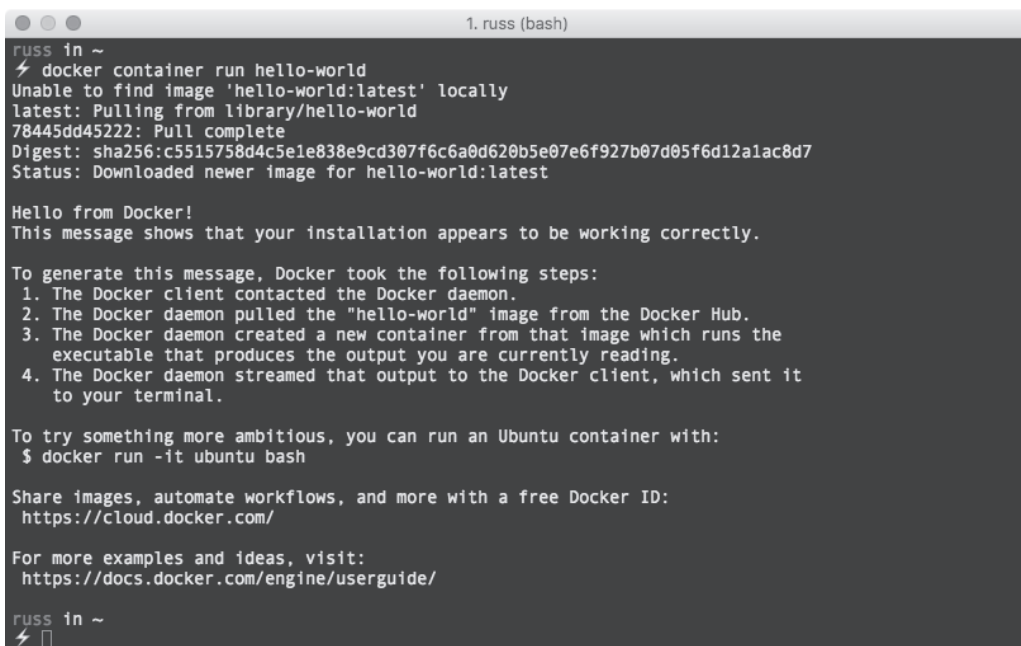
```
$ docker <POLECENIE> --help
```

Teraz możemy przystąpić do uruchomienia kontenera `hello-world`. Zrobimy to za pomocą polecenia (w zależności od konfiguracji systemu Linux może istnieć konieczność poprzedzenia poleceń instrukcją `sudo`):

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

Nieważne, na jakim hoście uruchomiono Dockera. W systemach Linux, macOS i Windows zostaną wykonane te same operacje — Docker pobierze obraz kontenera `hello-world`, uruchomi go, a następnie zatrzyma po wykonaniu znajdującego się w nim kodu.

W oknie Terminala powinny zostać wyświetlone następujące komunikaty:



```
1. russ (bash)
russ in ~
$ docker container run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
78445dd45222: Pull complete
Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7
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.
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://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/

russ in ~
$
```

Zrobmy coś ciekawszego. Spróbujmy pobrać i uruchomić kontener NGINX. W tym celu należy uruchomić następujące polecenia:

```
$ docker image pull nginx
$ docker container run -d --name nginx-test -p 8080:80 nginx
```

Pierwsze polecenie pobiera obraz kontenera NGINX, a drugie uruchamia go w tle — tworzy kontener `nginx-test` na bazie pobranego obrazu `nginx`. Drugie polecenie mapuje również port 8080 hosta do portu 80 kontenera, dzięki czemu można uzyskać do niego dostęp lokalnie za pomocą przeglądarki internetowej (pod adresem `http://localhost:8080/`).

Polecenia te zadziałają tak samo we wszystkich trzech systemach operacyjnych (o czym przekonasz się, patrząc na trzy kolejne rysunki). Oto zrzut z systemu Linux:

```

russ@russ-virtual-machine: ~
russ@russ-virtual-machine:~$ docker image pull nginx
Using default tag: latest
^[[latest: Pulling from library/nginx
ff3d52d8f55f: Pull complete
226f4ec56ba3: Pull complete
53d7dd52b97d: Pull complete
Digest: sha256:41ad9967ea448d7c2b203c699b429abeled5af331cd92533900c6d77490e0268
Status: Downloaded newer image for nginx:latest
russ@russ-virtual-machine:~$ docker container run -d --name nginx-test -p 8080:80 nginx
05a46b559fa8f57b895d88243cb0f539bd3a135553c1bc2a675d00cfd8b45866
russ@russ-virtual-machine:~$

```

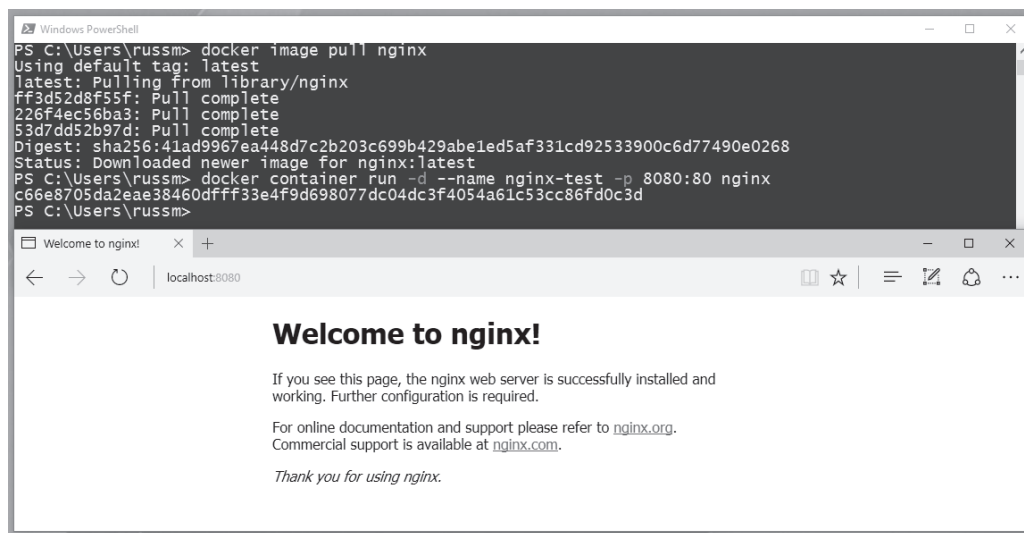
Oto zrzut z systemu macOS:

```

1. russ (bash)
russ in ~
$ docker image pull nginx
Using default tag: latest
latest: Pulling from library/nginx
ff3d52d8f55f: Pull complete
226f4ec56ba3: Pull complete
53d7dd52b97d: Pull complete
Digest: sha256:41ad9967ea448d7c2b203c699b429abeled5af331cd92533900c6d77490e0268
Status: Downloaded newer image for nginx:latest
russ in ~
$ docker container run -d --name nginx-test -p 8080:80 nginx
09e24e5d36561147f4a625a964308f2aeef008b20c5427a724699994dd50d10
russ in ~
$

```

A to zrzut z systemu Windows:



W kolejnych trzech rozdziałach poszerzysz swoją wiedzę dotyczącą klienta obsługiwanego za pomocą wiersza poleceń. Zatrzymaj i usuń kontener `nginx-test` za pomocą poleceń:

```
$ docker container stop nginx-test
$ docker container rm nginx-test
```

Ekosystem Dockera

Docker poza dołączonymi do niego narzędziami obsługuje również wiele dodatkowych narzędzi. Część z nich opisujemy w kolejnych rozdziałach. Na koniec pierwszego rozdziału chcieliśmy stworzyć zarys narzędzi, z których będziemy korzystać. Najważniejszym z nich jest **Docker Engine**.

To jądro Dockera. Korzystają z niego wszystkie pozostałe opisane przez nas narzędzia. Korzystaliśmy z niego w sekcjach związanych z instalacją Dockera. Obecnie utrzymywane są dwie wersje silnika Docker Engine: **Docker Enterprise Edition (Docker EE)** i **Docker Community Edition (CE)**. W tej książce korzystamy z Dockera CE. Niedawno cykl wydawania nowych wersji silnika Docker Engine został zmieniony, dzięki czemu jest bardziej przewidywalny.

Silniki Docker CE i EE mają stabilne wersje, które są aktualizowane co kwartał. Jak zapewne zauważyłeś, w systemie macOS i Windows zainstalowaliśmy Dockera w wersji CE v17.03.x. Istnieje jeszcze wersja Edge, w której nowe funkcje są wprowadzane co miesiąc, ale lepiej nie korzystać z niej w środowiskach produkcyjnych. W systemie Linux zainstalowaliśmy Dockera w wersji Edge CE 17.05.x. Oto inne elementy ekosystemu Dockera:

- **Docker Compose** to narzędzie pozwalające na tworzenie i udostępnianie zespołów kontenerów. Więcej informacji na ten temat znajdziesz w rozdziale 6. „Docker Compose”.
- **Docker Machine** to narzędzie przeznaczone do uruchamiania hostów Dockera na różnych platformach. Więcej informacji na ten temat znajdziesz w rozdziale 5. „Docker Machine”.
- **Docker Hub** to repozytorium przeznaczone do przechowywania obrazów Dockera. Zagadnienia z nim związane zostaną poruszone w trzech kolejnych rozdziałach.
- **Docker Store** to serwis przeznaczony do pobierania oficjalnych obrazów i wtyczek Dockera, a także innych licencjonowanych produktów. Zagadnienia związane z tym tematem również zostaną poruszone w trzech kolejnych rozdziałach.
- **Docker Swarm** to narzędzie przeznaczone do orkiestracji mogące pracować z wieloma hostami. Więcej informacji na ten temat znajdziesz w rozdziale 7. „Docker Swarm”.
- **Docker dla systemu macOS**. Zagadnienie to opisaliśmy w tym rozdziale.
- **Docker Cloud** — platforma CaaS (**kontener jako usługa**) opisana jest szczegółowo w rozdziale 10. „Usługa Docker Cloud”.
- **Docker dla systemu Windows**. Zagadnienie to opisaliśmy w tym rozdziale.
- **Docker dla usług Amazon Web Services**. Najlepsze praktyki instalacji narzędzia Docker Swarm w kontekście obsługi usług AWS opiszemy w rozdziale 10. „Usługa Docker Cloud”.
- **Docker dla Azure**. Najlepsze praktyki instalacji narzędzia Docker Swarm w kontekście obsługi usługi Azure opiszemy w rozdziale 10. „Usługa Docker Cloud”.

W kolejnych rozdziałach przyjrzymy się również wybranym usługom niezależnych twórców.

Podsumowanie

W tym rozdziale przedstawiliśmy podstawowe zagadnienia, które są niezbędne podczas lektury kolejnych rozdziałów. Dowiedziałeś się, czym jest i jak działa Docker. Opisaliśmy przebieg instalacji Dockera w różnych systemach operacyjnych i obsługę podstawowych funkcji Dockera z poziomu wiersza poleceń. Pamiętaj o konieczności sprawdzenia wymagań przed uruchomieniem instalatora — korzystaj z instalatora właściwego dla swojego systemu operacyjnego.

W dalszej części tego rozdziału opisaliśmy podstawowe polecenia umożliwiające rozpoczęcie pracy z Dockerem. W kolejnych rozdziałach opiszemy wszystkie polecenia służące do zarządzania kontenerami. Przedstawimy działanie poszczególnych poleceń, a także ich zastosowania. Pod koniec tego rozdziału wymieniliśmy narzędzia wchodzące w skład ekosystemu Dockera.

W kolejnych rozdziałach przyjrzymy się procesowi tworzenia bazowych kontenerów, opiszemy zagadnienia związane z plikami Dockerfile, przechowywaniem obrazów, a także korzystaniem ze zmiennych środowiskowych i wolumenów Dockera.

Skorowidz

A

- ADD, 38
- administratorzy, 19
- Amazon
 - EC2 Container Registry, 78
 - Elastic Block Storage, 290
 - Elastic Compute Cloud, 290
 - Elastic Container Service, 290
 - Web Services, 121, 227
- analiza pliku Dockerfile, 37
- API, 304
- aplikacja
 - do głosowania, 139
 - Docker Bench Security, 252
 - Mobycounter, 133, 137
 - NGINX, 50
- architektury referencyjne, 293
- attach, 88
- Auto Scaling, 291
- automatyzacja budowy obrazu, 63
- AWS, Amazon Web Services, 121, 227

B

- bezpieczeństwo, 262
 - kontenerów, 245
 - platformy Docker, 245
- budowane
 - pliki kontenerów, 252
 - obrazów, 42, 48
 - automatyzacja, 63
 - Dockera, 42
 - własnych, 43
- build, 148

C

- chmura
 - DigitalOcean, 119, 128
 - Docker Cloud, 223
 - Google Cloud Storage, 74
 - Microsoft Azure, 74
 - VPC, 122
- chmury
 - prywatne, 122
 - uruchamianie hostów, 119
- Clair, 263
- CMD, 40
- config, 148
- Consul, 295
 - dostępność, 304
 - szablony narzędzia, 302
 - uruchamianie narzędzia, 296
 - usługa Registrar, 296
 - zapytania, 297
- COPY, 38
- CoreOS, 128
- create, 148

D

- degradacja węzła menedżera, 165
- demon Dockera, 257
- diff, 249
- DigitalOcean, 119, 128, 226
- dobre praktyki, 250
- Docker, 17
 - Bench Security, 252
 - analiza informacji, 255
 - konfiguracja demona, 256
 - konfiguracja hosta, 255

Docker

- obrazy kontenerów, 257
- operacje bezpieczeństwa, 260
- pliki konfiguracyjne demona, 257
- środowisko robocze kontenerów, 258
- uruchamianie narzędzia, 253
- Cloud, 33, 223
 - Amazon Web Services, 239
 - łączenie kont, 225
 - tryb Swarm, 238
 - uruchamianie stosu, 234
 - uruchamianie węzłów, 231
 - zakładanie konta, 224
- Community Edition, 32
- Compose, 33, 133
 - polecenia, 146
 - uruchamianie aplikacji, 135
- dla Azure, 33
- dla systemu macOS, 33
- dla systemu Windows, 33
- dla usług Amazon Web Services, 33
- Engine, 32, 246
- Enterprise Edition, 32
- Hub, 33, 57
 - konfiguracja serwisu, 64
 - menu Create, 60
 - panel główny, 58
 - profil, 61
 - przycisk Explore, 59
 - przycisk Organizations, 60
 - strona Stars, 62
 - ustawienia, 61
- Machine, 33, 113
 - uruchamianie hostów, 119
 - wdrażanie lokalnych hostów, 114
- Registry, 71
 - wdrażanie rejestru, 72
- Store, 33, 70
- Swarm, 33, 155
 - degradacja węzła menedżera, 165
 - dołączanie wykonawców, 160
 - drenaż węzła, 166
 - instalacja, 156
 - kasowanie klastra, 173
 - listy węzłów, 161
 - menedżer Swarm, 157
 - nakładki sieciowe, 175
 - promowanie hosta, 164
 - role, 156

- równoważenie obciążeń, 174
- stosy, 168, 171
- tworzenie harmonogramu, 176
- tworzenie klastra, 158
- usługi, 168
- wykonawca Swarm, 157
- zarządzanie klastrem, 161
- Trusted Registry, 74, 293
- dołączanie wykonawców, 160
- down, 153
- drenaż węzła, 166
- dystribucja obrazów, 57
- dzienniki, 91, 186

E

- EBS, 290
- EC2, 290
- ekosystem, 32
- ekran główny, 58
- Elastic Load Balancing, 290
- ELB, 290
- ENTRYPOINT, 40
- ENV, 41
- etcd, 304
- events, 149
- exec, 89, 151
- EXPOSE, 40

F

- FROM, 37

G

- GitHub, 76
- Grafana, 280, 282

H

- harmonogram, 176
- Heroku, 289
- hosty
 - dedykowane, 22
 - Dockera, 246, 291
 - lokalne, 114
 - robocze, 164
 - uruchamianie, 119

I

informacje
 o obciążeniu procesora, 186
 o pamięci, 186
 o połączeniach sieciowych, 186
 o procesach, 91
 instalacja, 23
 Docker Swarm, 156
 Rancher, 202
 uwierzytelniania, 201
 w systemie Linux, 24
 w systemie macOS, 25
 w systemie Windows 10, 27
 instalator Tectonic, 286
 interfejs Docker API, 304

K

kasowanie klastra Swarm, 173
 kill, 96, 153
 klaster, 161, 218, 292
 Swarm, 173, 195
 tworzenie, 192
 klient Dockera, 29
 komunikacja klastra, 293
 konfiguracja
 demona Dockera, 251, 256
 hosta, 251, 255
 serwisu Docker Hub, 64
 uwierzytelniania, 201, 204
 kontenery, 35, 44, 183
 bezpieczeństwo, 245
 komunikacja, 88
 podgląd właściwości, 216
 polecenia, 84, 95
 stany, 95
 usuwanie, 97
 zarządzanie, 83

L

LABEL, 37
 Linux Containers, 22
 listy węzłów, 161
 logs, 91, 149

Ł

ładowanie własnych obrazów, 68

M

maszyna wirtualna, 22
 VirtualBox, 130
 menedżer Swarm, 157
 menu Create, 60
 Microbadger, 79
 mieszanie procesów, 291
 Moby Project, 306
 Mobycounter, 137
 monitorowanie, 277
 narzędzia, 285

N

nakładki sieciowe, 175
 narzędzia do monitorowania, 285
 narzędzie
 cadvisor, 279
 Consul, 296
 Docker Bench Security, 252
 Docker Machine, 113
 etcd, 304
 Grafana, 280, 282
 Microbadger, 79
 node-exporter, 279
 Portainer, 177
 Rancher, 201
 WP-CLI, 274

O

obciążenia wejściowe, 174
 obrazy, 187, 252
 Dockera, 42
 dystrybucja, 57
 kontenerów, 35, 257
 przechowywanie, 57
 rejstry, 293
 zaufane źródła, 247
 obsługa
 klastrow, 292
 kontenerów, 84
 kontenerów LXC, 22
 sieci, 100
 wolumenów, 100

ograniczenia zasobów, 93
 ONBUILD, 41
 operacje bezpieczeństwa, 252, 260
 organizacja Center for Internet Security, 251

P

panel główny, 58
 pause, 96, 149
 platforma Heroku, 289
 plik
 .dockerignore, 43
 Dockerfile, 35
 analiza, 37
 budowa obrazu, 42
 dobre praktyki, 41
 polecenia, 37, 40
 zmienné środowiskowe, 50
 YAML, 137
 pliki konfiguracyjne demona Dockera, 251, 257
 podgląd właściwości, 216
 polecenia
 Docker Compose, 146
 Dockera, 247
 obsługa kontenerów, 84
 pliku Dockerfile, 40
 polecenie
 ADD, 38
 attach, 88
 build, 148
 CMD, 40
 config, 148
 COPY, 38
 create, 148
 diff, 249
 docker build, 42
 down, 153
 ENTRYPOINT, 40
 ENV, 41
 events, 149
 exec, 89, 151
 EXPOSE, 40
 FROM, 37
 kill, 96, 153
 LABEL, 37
 logs, 91, 149
 ONBUILD, 41
 pause, 96, 149
 prune, 97
 ps, 147
 pull, 148
 restar, 96
 restart, 149
 rm, 98, 153
 run, 38, 151, 247
 scale, 152
 service, 168
 start, 96, 149
 stats, 93
 stop, 96, 149
 top, 92, 149
 unpause, 96, 149
 up, 147
 USER, 41
 WORKDIR, 41
 Portainer, 177
 Docker, 191
 dzienniki, 186
 konsola, 187
 kontenery, 183
 obrazy, 187
 panel główny, 181
 sieci, 190
 statystyki, 186
 stosowanie, 180
 szablony aplikacji, 181
 uruchamianie, 178
 usługi, 193, 196
 węzły końcowe, 198
 wolumeny, 190
 zdarzenia, 190
 prace programistyczne, 265
 profil, 61
 projekt Moby Project, 306
 przechowywanie obrazów, 57
 przepływ zadań, 265
 przycisk
 Explore, 59
 Organizations, 60
 PS, 147
 pull, 148

Q

Quay, 75, 262

R

Rancher, 201
 instalacja, 202
 katalog, 217
 konfiguracja uwierzytelniania, 204
 tworzenie stada, 207
 uruchamianie stosów, 209
 usuwanie stada, 217
 RancherOS, 130
 Registrator, 296
 rejestr
 Amazon EC2 Container Registry, 78
 Docker Registry, 71
 Docker Trusted Registry, 74
 rejestry
 niezależne, 75
 obrazów, 293
 repozytorium, 63
 Docker Hub, 57
 GitHub, 76
 restart, 96
 restart, 149
 rm, 153
 role Docker Swarm, 156
 rozszerzenie Weave Net, 124
 run, 38, 151, 247

S

scale, 152
 serwis
 Docker, 58
 Docker Hub, 57
 Docker Store, 70
 sieci, 100, 190
 silnik Consul, 295
 skanowanie zabezpieczeń Dockera, 260
 stany kontenerów, 95
 start, 96, 149
 stats, 93
 stop, 96, 149
 stos, 171, 209, 210, 234
 stosy Docker Swarm, 168
 strona Stars, 62
 Swift, 74
 system
 CoreOS, 128
 Kubernetes, 286

plików, 74
 RancherOS, 130
 szablony
 aplikacji, 181
 narzędzia Consul, 302

Ś

środowisko
 produkcyjne, 291
 robocze kontenerów, 252, 258
 wykonawcze, 252

T

Tectonic, 286
 top, 92, 149
 tryb Swarm, 238
 tworzenie
 harmonogramu, 176
 klastra, 158, 192
 obrazów kontenerów, 35
 stada, 207

U

unpause, 96, 149
 up, 147
 uruchamianie
 aplikacji, 135
 hostów, 119
 narzędzia Consul, 296
 stosów, 209, 234
 usługi Registrator, 296
 węzłów, 231
 USER, 41
 usługa, 196
 Amazon Elastic Container Service, 290
 Amazon S3, 74
 Docker Cloud, 223
 Docker Swarm, 168
 Portainer, 193
 Quay, 75
 Registrator, 296
 ustawienia, 61
 usuwanie
 kontenerów, 97
 stada, 217
 uwierzytelnianie, 201, 204

V

VPC, Virtual Private Cloud, 122, 291

W

wdrażanie

 lokalnych hostów, 114

 rejestru, 72

węzły

 końcowe, 198

 menedżera, 165

 uruchamianie, 231

wiersz poleceń, 29

WirtualBox, 130

właściwości kontenerów, 216

wolumeny, 107, 190

WORKDIR, 41

wykonawca Swarm, 157

wykrywanie usług, 295

Z

zabezpieczenia Dockera, 260

zarządzanie

 klastrem, 161

 kontenerami, 83

zasoby, 93

zaufane źródła obrazów, 247

zdarzenia, 190

zewnętrzne platformy, 286

zmienne środowiskowe, 50

Ż

źródła obrazów, 247

Ż

żądania do kontenerów, 292

żądanie SIGTERM, 96

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ 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 WYDAWNICZA



Helion SA

Docker: od teraz aplikacja działa zawsze i wszędzie!

Od czasu premiery w 2013 roku Docker cieszy się rosnącym zainteresowaniem. Liczba deweloperów korzystających z tej platformy rośnie lawinowo. Docker zmienił sposób projektowania i wdrażania aplikacji, włączając w to aplikacje sieciowe. Ma duże możliwości i łączy w sobie prostotę wdrażania aplikacji z prostotą administrowania. Pozwala na rozwiązywanie problemów, z którymi borykają się zespoły programistów i administratorów wdrażających nowe systemy. Jednym słowem, jest to niezwykle użyteczne narzędzie i warto maksymalnie wykorzystać jego potencjał!

Niniejsza książka jest praktycznym podręcznikiem, dzięki któremu szybko zaczniesz efektywnie korzystać z Dockera. Prędko też zauważysz zupełnie nowe możliwości pracy nad oprogramowaniem. Zapoznasz się z podstawowymi koncepcjami związanymi z Dockerem i z takimi zagadnieniami jak budowanie, zarządzanie i przechowywanie obrazów. Dowiesz się, kiedy i w jaki sposób warto rozszerzyć Dockera oraz jak zintegrować go z różnymi platformami i narzędziami. Nauczysz się pracować z kontenerami za pomocą narzędzi Docker Machine, Docker Swarm i Docker Compose. Zapoznasz się również z problematyką bezpieczeństwa tworzonych systemów.

W tej książce między innymi:

- zwięzłe podstawy Dockera oraz zasady pracy z obrazami i kontenerami Dockera
- przechowywanie i dystrybucja obrazów
- praca z narzędziami Portainer i Rancher oraz z usługą Docker Cloud
- zabezpieczanie platformy i zarządzanie przepływem zadań
- ulepszanie aplikacji działającej w kontenerze Dockera

Russ McKendrick — jest architektem oprogramowania. Z branżą IT jest związany od ponad ćwierćwiecza. Korzysta niemal wyłącznie z rozwiązań open source, a Docker jest jego ulubionym narzędziem. Pasjonuje się muzyką na płytach winylowych.

Scott Gallagher — od dzieciństwa fascynuje się programowaniem. Zajmował się takimi technologiami jak serwerowe rozwiązania firm Novell, Microsoft i Red Hat. Obecnie interesuje się środowiskami systemu Linux, a w pracy skupia się na Dockerze i technologiach chmury.

Helion helion.pl 0 801 339900 0 601 339900 INFORMATYKA W NAJLEPSZYM WYDANIU	<i>Sprawdź nasze szkolenia!</i> SZKOLENIA  AKADEMIA IT & BUSINESS WWW.SZKOLENIA.HELION.PL	KOD KORZYŚCI Śięgnij po więcej!  ISBN 978-83-283-4308-5  9 788328 343085 Cena: 59,00 zł
--	---	--

Packt