Jarosław Krochmalski

Docker

Projektowanie i wdrażanie aplikacji

Helion

Packt›

Tytuł oryginału: Developing with Docker

Tłumaczenie: Piotr Pilch

ISBN: 978-83-283-3534-9

First published in the English language under the title ‘Developing with Docker - (9781786469908)’

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/docpro
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 1

Wprowadzenie do Dockera

Pierwotnie Docker został utworzony jako wewnętrzne narzędzie przez firmę o nazwie **dotCloud**, która oferowała usługę PaaS (ang. *Platform as a Service*). Później, w marcu 2013 r., Docker udostępniono w postaci oprogramowania *open source*. Oprócz zespołu firmy Docker Inc., która jest głównym twórcą, wkład w rozwój tego narzędzia ma kilka innych dużych firm, między innymi Red Hat, IBM, Microsoft, Google i Cisco Systems. Narzędzia do tworzenia oprogramowania muszą obecnie cechować się „zwinnością" i zapewniać możliwość szybkiego reagowania na zmiany. Stosowane są metodologie, takie jak Scrum, nakład pracy szacowany jest za pomocą jednostek względnych (ang. *story points*), a ponadto mają miejsce codzienne spotkania. Jak jednak wygląda kwestia przygotowania oprogramowania do dostarczenia i wdrożenia? Dowiedzmy się, jak Docker sprawdza się w takim scenariuszu, a także w jaki sposób może ułatwić uzyskanie „zwinności".

W rozdziale zostaną omówione następujące zagadnienia:

- podstawowa idea przyświecająca powstaniu Dockera,
- różnica między wirtualizacją i konteneryzacją,
- korzyści wynikające z zastosowania Dockera,
- komponenty dostępne do zainstalowania.

Zaczniemy od podstawowej idei przyświecającej powstaniu tego wspaniałego narzędzia.

Podstawowa idea

Podstawową ideą przyświecającą powstaniu narzędzia Docker jest umieszczenie aplikacji z jej wszystkimi zależnościami (mogą to być dane binarne, biblioteki, pliki konfiguracyjne, skrypty, pliki JAR itd.) w jednostce standaryzowanej pod kątem procesu tworzenia i wdrażania oprogramowania. Kontenery Dockera zawierają nie tylko kod programu, ale posiadają dodatkowo narzędzia i biblioteki systemowe, środowisko uruchomieniowe wraz z gotową konfiguracją, czyli wszystko to, co musiałoby być zainstalowane na serwerze. Gwarantuje to, że oprogramowanie zawsze będzie działać w taki sam sposób, niezależnie od tego, w jakim środowisku zostanie wdrożone. Dzięki Dockerowi możesz utworzyć projekt dla środowiska Node.js lub Java (nie jesteś oczywiście ograniczony wyłącznie do nich) bez konieczności instalowania któregokolwiek z tych środowisk na komputerze hosta. Gdy projekt już nie będzie potrzebny, możesz po prostu usunąć obraz Dockera, tak jakby nic wcześniej nie miało miejsca. Docker nie jest językiem programowania ani platformą programistyczną. Traktuj go raczej jak narzędzie, które ułatwia rozwiązywanie typowych problemów, takich jak instalacja i dystrybucja oprogramowania oraz zarządzanie nim. Docker pozwala programistom i osobom korzystającym z metodyki DevOps tworzyć, dostarczać i uruchamiać kod w dowolnym miejscu.

Możesz uznać, że Docker to mechanizm wirtualizacji, ale jak się wkrótce okaże, narzędzie to dalekie jest od tego.

Porównanie konteneryzacji i wirtualizacji

Aby w pełni zrozumieć, czym naprawdę jest Docker, konieczne jest najpierw uzmysłowienie sobie różnicy między tradycyjną wirtualizacją i konteneryzacją. Porównajmy teraz te dwie technologie.

Tradycyjna wirtualizacja

Tradycyjna maszyna wirtualna, która reprezentuje wirtualizację na poziomie sprzętowym, to zasadniczo kompletny system operacyjny, działający na bazie systemu operacyjnego hosta. Istnieją dwa typy hipernadzorcy wirtualizacji: **typ 1.** i **typ 2**. Hipernadzorcy typu 1. zapewniają wirtualizację serwera na sprzęcie pozbawionym tradycyjnego systemu operacyjnego obsługiwanego przez zwykłych użytkowników. Z kolei hipernadzorcy typu 2. są powszechnie wykorzystywani do wirtualizacji desktopów. W tym wypadku mechanizm wirtualizacji działa na bazie systemu operacyjnego stosowanego przez użytkownika. Istnieje mnóstwo zastosowań, które korzystają z wirtualizacji. Największą zaletą jest możliwość uruchomienia na jednym hoście wielu maszyn wirtualnych z całkowicie odmiennymi systemami operacyjnymi.

Ponieważ maszyny wirtualne są w pełni izolowane, są bardzo bezpieczne. Nic jednak nie odbywa się bez ponoszenia kosztów. Wyróżnić można wiele mankamentów, które obejmują wszystkie elementy, jakie musi mieć system operacyjny: sterowniki urządzeń, podstawowe biblioteki

systemowe itp. Maszyny wirtualne są ciężkie, zużywają zwykle wiele zasobów, a ponadto nie są łatwe do skonfigurowania i wymagają przeprowadzenia pełnej instalacji systemu. Dodatkowo podczas działania zużywają więcej zasobów na własne potrzeby. Aby pomyślnie uruchomić aplikację na maszynie wirtualnej, hipernadzorca musi ją najpierw zaimportować, a następnie załadować, co wymaga czasu. Co więcej, wydajność maszyn może znacznie spaść. W rezultacie na jednym komputerze efektywnie może zostać uruchomionych tylko kilka maszyn wirtualnych.

Konteneryzacja

Narzędzie Docker działa w izolowanym środowisku nazywanym **kontenerem Dockera**. Nie jest to maszyna wirtualna w ogólnie przyjętym znaczeniu. Kontener Dockera reprezentuje wirtualizację systemu operacyjnego. Każdy obraz maszyny wirtualnej działa w niezależnym systemie operacyjnym gościa, natomiast obrazy Dockera funkcjonują w obrębie jądra tego samego systemu operacyjnego. Kontener ma własny system plików i zmienne środowiskowe. Jest samowystarczalny. Ponieważ kontenery działają wewnątrz jednego jądra, zużywają mniej zasobów systemowych. Podstawowy kontener może być, i zwykle jest, bardzo lekki. Warto wiedzieć, że kontenery Dockera są izolowane nie tylko od bazowego systemu operacyjnego, ale też wzajemnie od siebie samych. Nie jest generowane żadne dodatkowe obciążenie związane z klasycznym hipernadzorcą wirtualizacji i systemem operacyjnym gościa. Pozwala to na uzyskanie wydajności równej niemal wydajności sprzętowej bez narzutu innego systemu operacyjnego. Czas rozruchu aplikacji **korzystającej z Dockera** jest zwykle bardzo krótki, co wynika z niewielkiego obciążenia powodowanego przez kontenery. Możliwe jest również przyspieszenie procesu wprowadzania w ciągu kilku sekund setek kontenerów aplikacji i skrócenie czasu, jaki zajmuje dostarczenie oprogramowania.

Jak widać, Docker w dość znacznym stopniu różni się od tradycyjnych mechanizmów wirtualizacji. Miej świadomość tego, że kontenery nie mogą zastąpić maszyn wirtualnych we wszystkich zastosowaniach. Wnikliwa ocena wymagań w dalszym ciągu niezbędna jest do stwierdzenia, co jest najlepsze z punktu widzenia aplikacji. Oba rozwiązania mają swoje zalety. Z jednej strony dysponujemy w pełni izolowaną i bezpieczną maszyną wirtualną o średniej wydajności, a z drugiej kontenerami pozbawionymi części kluczowych cech (np. całkowita izolacja), ale zapewniającymi dużą wydajność oraz możliwość szybszego wdrożenia w środowisku.

Dowiedzmy się, jakie inne korzyści osiągane są podczas stosowania konteneryzacji Dockera.

Korzyści wynikające z zastosowania Dockera

Porównując kontenery Dockera z tradycyjnymi maszynami wirtualnymi, wspomniano o niektórych ich zaletach. Pora je teraz dokładniej przeanalizować i zaprezentować kilka dodatkowych korzyści.

Szybkość i wielkość

Jak już wcześniej wspomniano, pierwszą widoczną korzyścią wynikającą z użycia Dockera będzie bardzo zadowalająca wydajność i krótki czas dostarczania. Możliwe jest szybkie i łatwe tworzenie lub usuwanie kontenerów. Docker współużytkuje wyłącznie jądro i nic ponadto. Korzysta jednak ponownie z warstw obrazów, na bazie których tworzony jest konkretny obraz. W rezultacie wiele wersji aplikacji uruchomionych w kontenerach będzie bardzo lekkich. Dzięki temu zapewniane są szybsze wdrożenie, łatwiejsza migracja i krótkie czasy rozruchu.

Odtwarzalne i przenośne kompilacje

Użycie Dockera pozwala na wdrożenie oprogramowania gotowego do uruchomienia, które jest przenośne i szczególnie łatwo może być rozprowadzane (proces tworzenia obrazu zostanie omówiony w rozdziale 6., „Tworzenie obrazów"). Aplikacja objęta konteneryzacją po prostu działa w obrębie własnego kontenera: nie trzeba przeprowadzać tradycyjnej instalacji. Kluczową korzyścią związaną z obrazem Dockera jest to, że zawiera on wszystkie zależności, jakie aplikacja musi posiadać, aby poprawnie się uruchomić. Brak instalacji zależności na serwerze stanowi ogromną zaletę. Eliminuje to takie problemy, jak konflikty oprogramowania i bibliotek, a nawet problemy ze zgodnością sterowników. Dzięki środowisku odtwarzalnych kompilacji narzędzia Docker jest ono szczególnie dobrze przystosowane do testowania, zwłaszcza w przypadku przepływu integracji ciągłej (ang. *Continuous Integration*). W celu przeprowadzenia testów możesz szybko dokonać rozruchu identycznych środowisk. Ponieważ obrazy kontenera są każdorazowo takie same, możliwa jest dystrybucja obciążenia i bezproblemowe uruchamianie równoległych testów. Programiści mogą na swoich komputerach uruchamiać ten sam obraz, który później zostanie użyty na komputerze produkcyjnym. I tym razem zapewnia to ogromną korzyść związaną z testowaniem. Użycie kontenerów Dockera przyspiesza integrację ciągłą. Nie występuje już nieskończona liczba cykli kompilowanie-testowanie-wdrażanie. Kontenery Dockera zapewniają, że aplikacje działają identycznie w środowiskach projektowych, testowych i produkcyjnych.

Jedna z najwspanialszych cech Dockera to przenośność. Kontenery Dockera są przenośne, więc mogą być uruchomione gdziekolwiek: na komputerze lokalnym, na pobliskim lub odległym serwerze albo w prywatnej lub publicznej chmurze. Skoro mowa o chmurze — wszyscy najważniejsi dostawcy usług obliczeniowych opartych na chmurze, takich jak Amazon Web Services i Google Compute Platform, zdali sobie sprawę z dostępności narzędzia Docker, które jest obecnie przez nich obsługiwane. Kontenery Dockera mogą być uruchamiane wewnątrz instancji usługi Amazon EC2 lub instancji usługi Google Compute Engine, pod warunkiem że Docker obsługiwany jest przez system operacyjny hosta. Kontener działający w instancji usługi Amazon EC2 z łatwością może zostać przeniesiony do jakiegoś innego środowiska, osiągając identyczny poziom spójności i funkcjonalności. Docker bardzo dobrze współpracuje z różnymi innymi dostawcami modelu **IaaS** (ang. *Infrastructure-as-a-Service*), takimi jak Microsoft Azure, IBM SoftLayer lub OpenStack. Taki dodatkowy poziom abstrakcji względem warstwy infrastruktury jest czymś niezastąpionym. Oprogramowanie może być po prostu tworzone bez martwienia się tym, na jakiej później platformie zostanie uruchomione. Jest to rzeczywiście rozwiązanie typu *utwórz raz i uruchamiaj gdziekolwiek*.

Trwała i „zwinna" infrastruktura

Utrzymywanie prawdziwie niezmiennej bazy kodu zarządzania konfiguracją może być złożonym i czasochłonnym procesem. Z upływem czasu kod zwiększa swoją wielkość i staje się coraz bardziej kłopotliwy. Z tego właśnie powodu idea trwałej infrastruktury zyskuje obecnie coraz większą popularność. Na ratunek przychodzi konteneryzacja. Użycie kontenerów podczas procesu programowania i wdrażania aplikacji pozwala uprościć proces. Zastosowanie lekkiego serwera Dockera, który nie wymaga prawie żadnego zarządzania konfiguracją, powoduje, że zarządzanie aplikacjami sprowadza się do wielokrotnego wdrażania kontenerów na serwerze. I tym razem lekkie kontenery sprawiają, że ich użycie wymaga poświęcenia jedynie kilku sekund.

Na początek możesz pobrać wstępnie utworzony obraz Dockera z serwisu Docker Hub, który przypomina repozytorium gotowych do użycia obrazów. W serwisie dostępnych jest wiele opcji wyboru serwerów WWW, platform środowiska uruchomieniowego, baz danych, serwerów przesyłania komunikatów itp. Serwis jest jak prawdziwa kopalnia złota w postaci darmowego oprogramowania, które może być fundamentem Twojego własnego projektu. Serwis Docker Hub oraz proces wyszukiwania w nim obrazów zostanie omówiony w rozdziale 5., „Znajdowanie obrazów".

Efekt niezmienności obrazów Dockera wynika ze sposobu, w jaki są one tworzone. Docker korzysta ze specjalnego pliku o nazwie *Dockerfile*. Plik zawiera wszystkie instrukcje instalacyjne dotyczące tego, jak ma zostać utworzony obraz. Instrukcje odwołują się do niezbędnych komponentów, bibliotek, udostępnionych katalogów współużytkowanych, konfiguracji sieciowej itp. Obraz może być bardzo prosty i nie zawierać niczego oprócz podstawowych składników systemu operacyjnego. Obraz może też, co jest częściej spotykane, zawierać cały wstępnie przygotowany stos technologii gotowych do uruchomienia. Obrazy mogą być tworzone ręcznie, ale też może to być zautomatyzowany proces.

Docker tworzy obrazy w sposób warstwowy: każdy dołączany składnik będzie dodawany jako kolejna warstwa podstawowego obrazu. W ten sposób uzyskuje się poważny przyrost szybkości w porównaniu z tradycyjnymi technikami wirtualizacji.

Szczegółami związanymi z tworzeniem obrazów zajmiemy się w rozdziale 6., „Tworzenie obrazów".

Narzędzia i interfejsy API

Docker to oczywiście nie tylko procesor pliku *Dockerfile* i mechanizm środowiska uruchomieniowego. Jest to kompletny pakiet oferujący bogatą gamę narzędzi i interfejsów API, które są pomocne w czasie codziennej pracy programistów i osób korzystających z metodyki DevOps. Przede wszystkim dostępne jest narzędzie Docker Toolbox, które pełni funkcję instalatora pozwalającego szybko i łatwo zainstalować i skonfigurować środowisko Dockera na własnym komputerze. Kitematic to klienckie środowisko graficzne przeznaczone do zarządzania kontenerami oraz obrazami Dockera w systemach Windows i Mac OS X. Dystrybucja Dockera zawiera również całą grupę narzędzi wiersza poleceń, które będą omawiane w książce. Przyjrzyjmy się im teraz.

Przegląd narzędzi

Zależnie od używanej wersji systemu Windows dostępne są dwie opcje. W przypadku systemu Windows 10 lub nowszego możesz skorzystać z narzędzia Docker for Windows. Jeśli używasz jednego z wcześniejszych wersji systemu Windows, dostępne jest narzędzie Docker Toolbox. Najnowsza oferta to narzędzie Docker for Mac, które działa jako natywna aplikacja systemu Mac OS, a ponadto używa narzędzia xhyve do wirtualizacji środowiska Docker Engine i jądra systemu Linux. W przypadku wcześniejszej wersji systemu Mac OS, która nie spełnia wymagań narzędzia Docker for Mac (zostaną one wyszczególnione w rozdziale 2., „Instalowanie Dockera"), należy wybrać narzędzie Docker Toolbox for Mac. Idea przyświecająca narzędziu Docker Toolbox i natywnym aplikacjom Dockera jest taka sama, czyli wirtualizacja jądra systemu Linux i środowiska Docker Engine na bazie używanego systemu operacyjnego. Na potrzeby książki będziemy korzystać z narzędzia Docker Toolbox, ponieważ jest uniwersalne. Będzie ono działać we wszystkich wersjach systemów Windows i Mac OS. Pakiet instalacyjny dla tych systemów zawarty jest w pliku wykonywalnym o nazwie Docker Toolbox. W pakiecie znajdują się wszystkie narzędzia niezbędne do rozpoczęcia pracy z Dockerem. Istnieje oczywiście całe mnóstwo zgodnych z Dockerem dodatkowych narzędzi zewnętrznych dostawców, spośród których część jest bardzo przydatna. Niektóre z nich w skrócie zostaną zaprezentowane w rozdziale 9., „Użycie Dockera do projektowania". Skoncentrujmy się jednak na razie na domyślnym zestawie narzędzi. Przed rozpoczęciem instalacji przyjrzyjmy się narzędziom oferowanym przez pakiet instalatora, aby lepiej zrozumieć, jakie zmiany zostaną wprowadzone w systemie.

Środowisko Docker Engine i jego klient

Docker to aplikacja klient-serwer. Uwzględnia ona demona realizującego ważne zadania, a mianowicie tworzenie i pobieranie obrazów, uruchamianie i zatrzymywanie kontenerów itp. Docker udostępnia interfejs API REST, który określa interfejsy prowadzące interakcję z demonem i używane na potrzeby zdalnego zarządzania. Środowisko Docker Engine akceptuje polecenia Dockera wprowadzane w wierszu poleceń, na przykład polecenie `docker run` *`nazwa-obrazu`* uruchamiające obraz, polecenie `docker ps` wyświetlające listę działających kontenerów oraz polecenie `docker images`, które wyszczególnia obrazy.

Klient Dockera to program wiersza poleceń służący do zarządzania hostami Dockera z uruchomionymi kontenerami systemu Linux. Komendy klienta są zamieniane na odpowiednie zapytania interfejsu REST API, a następnie wysyłane do serwera Dockera.

Środowisko Docker Engine działa tylko w systemie Linux. Aby uruchomić Dockera w systemie Windows lub Mac OS albo by zapewnić wiele hostów Dockera w sieci lub chmurze, niezbędne będzie narzędzie Docker Machine.

Docker Machine

Docker Machine to dość nowe narzędzie wiersza poleceń, stworzone przez zespół rozwijający Dockera z myślą o zarządzaniu serwerami Dockera, na których mogą być wdrażane kontenery. Narzędzie to zastąpiło dotychczasową metodę instalowania Dockera za pomocą **Boot2Docker**. Docker Machine eliminuje konieczność ręcznego tworzenia maszyn wirtualnych i instalowania Dockera przed uruchomieniem na nich kontenerów. Docker Machine automatycznie obsługuje proces dostarczania i instalowania Dockera. Inaczej mówiąc, jest to szybki sposób dostarczania nowej maszyny wirtualnej i przygotowania jej do uruchomienia kontenerów Dockera. Narzędzie Docker Machine okazuje się niezastąpione podczas rozwoju architektury modelu **PaaS** (ang. *Platform as a Service*). Nie tylko tworzy nową maszynę wirtualną z zainstalowanym na niej środowiskiem Docker Engine, ale też konfiguruje pliki certyfikatów na potrzeby uwierzytelniania, a następnie konfiguruje klienta Dockera, tak aby mógł komunikować się ze środowiskiem. Z myślą o elastyczności w Docker Machine wprowadzono pojęcie sterowników. Dzięki nim Docker ma możliwość komunikowania się z różnym oprogramowaniem wirtualizacji i dostawcami usługi chmury. Tak naprawdę podczas instalacji Dockera dla systemu Windows lub Mac OS używany będzie domyślny sterownik VirtualBox. W tle zostanie wykonane następujące polecenie:

```
docker-machine create --driver=virtualbox default
```

Kolejny dostępny sterownik to amazonec2, przeznaczony dla usługi Amazon Web Services. Sterownik może posłużyć do instalacji Dockera w chmurze firmy Amazon, co zostanie omówione w dalszej części rozdziału. Istnieje mnóstwo sterowników gotowych do użycia, a ponadto nieustannie pojawiają się kolejne. Lista istniejących oficjalnych sterowników wraz z dokumentacją jest dostępna w witrynie internetowej **Docker Drivers** pod adresem *https://docs.docker.com/machine/drivers*.

Lista obejmuje aktualnie sterowniki dla następujących usług:

- Amazon Web Services,
- Microsoft Azure,
- Digital Ocean,
- Exoscale,
- Google Compute Engine,
- Generic,
- Microsoft Hyper-V,
- OpenStack,
- Rackspace,
- IBM Softlayer,
- Oracle VirtualBox,

- VMware vCloud Air,
- VMware Fusion,
- VMware vSphere.

Oprócz tego istnieje wiele zewnętrznych dodatkowych sterowników dostępnych bezpłatnie w witrynach internetowych, takich jak GitHub. Dodatkowe sterowniki możesz znaleźć dla różnych dostawców chmury i platform wirtualizacji, takich na przykład jak OVH Cloud lub Parallels for Mac OS. Nie ma ograniczenia tylko do usług Amazon AWS lub Oracle VirtualBox. Jak widać, do wyboru jest wiele różnych sterowników.

> Jeśli nie możesz znaleźć konkretnego sterownika dla używanego dostawcy chmury, spróbuj poszukać go w witrynie GitHub.

W przypadku instalowania narzędzia Docker Toolbox w systemie Windows lub Mac OS narzędzie Docker Machine zostanie wybrane domyślnie. Jest ono niezbędne i obecnie stanowi jedyną metodę uruchomienia Dockera w tych systemach operacyjnych. Instalowanie Docker Machine nie jest obligatoryjne w przypadku systemu Linux. Nie ma tutaj potrzeby wirtualizacji jądra systemu Linux. Aby jednak skorzystać z dostawców chmury lub po prostu dysponować wspólnym środowiskiem uruchomieniowym, które może być przenoszone między systemami Mac OS, Windows i Linux, możesz też zainstalować Docker Machine for Linux. Proces ten zostanie opisany w dalszej części rozdziału. Narzędzie Docker Machine będzie również stosowane w tle podczas korzystania z narzędzia graficznego Kitematic, które zostanie zaprezentowane wkrótce.

Po przeprowadzeniu procesu instalacji narzędzie Docker Machine będzie dostępne z poziomu wiersza poleceń po wpisaniu polecenia `docker-machine`.

Kitematic

Kitematic to narzędzie umożliwiające uruchamianie kontenerów za pośrednictwem prostego, a przy tym dającego duże możliwości graficznego interfejsu użytkownika (GUI). W 2015 r. firma Docker przejęła firmę tworzącą Kitematica, oczekując, że rozwiązanie dotyczące konteneryzacji zyska uznanie większej liczby programistów, a co za tym idzie, większą popularność.

Kitematic jest obecnie domyślnie dołączany w trakcie instalowania narzędzia Docker Toolbox w systemach Mac OS i Windows. Kitematic pozwala w wygodny sposób wyszukiwać i pobierać niezbędne obrazy z serwisu Docker Hub. Narzędzie to może również być używane do uruchamiania własnych kontenerów. Pozwala edytować zmienne środowiskowe, mapować porty, konfigurować wolumeny, analizować dzienniki i uzyskiwać dostęp do kontenerów z poziomu wiersza poleceń. Warto wspomnieć, że możliwe jest płynne przełączenie się między tym interfejsem i interfejsem wiersza poleceń, aby uruchamiać kontenery aplikacji oraz zarządzać nimi. Kitematic jest bardzo wygodny w użyciu. Jeśli jednak wymagasz większego poziomu kontroli podczas

pracy z kontenerami albo po prostu chcesz uruchomić skrypt, wiersz poleceń będzie lepszym rozwiązaniem. Okazuje się, że Kitematic umożliwia przełączanie się między Docker CLI i interfejsem graficznym. Wszelkie zmiany dokonane w interfejsie wiersza poleceń zostaną bezpośrednio uwzględnione w Kitematicu. Jak się okaże na końcu rozdziału, gdzie zostanie przetestowana instalacja na komputerze z systemem Mac OS lub Windows, narzędzie to jest proste w użyciu. W pozostałej części książki do pracy z Dockerem będzie stosowany interfejs wiersza poleceń.

Narzędzie Compose Dockera

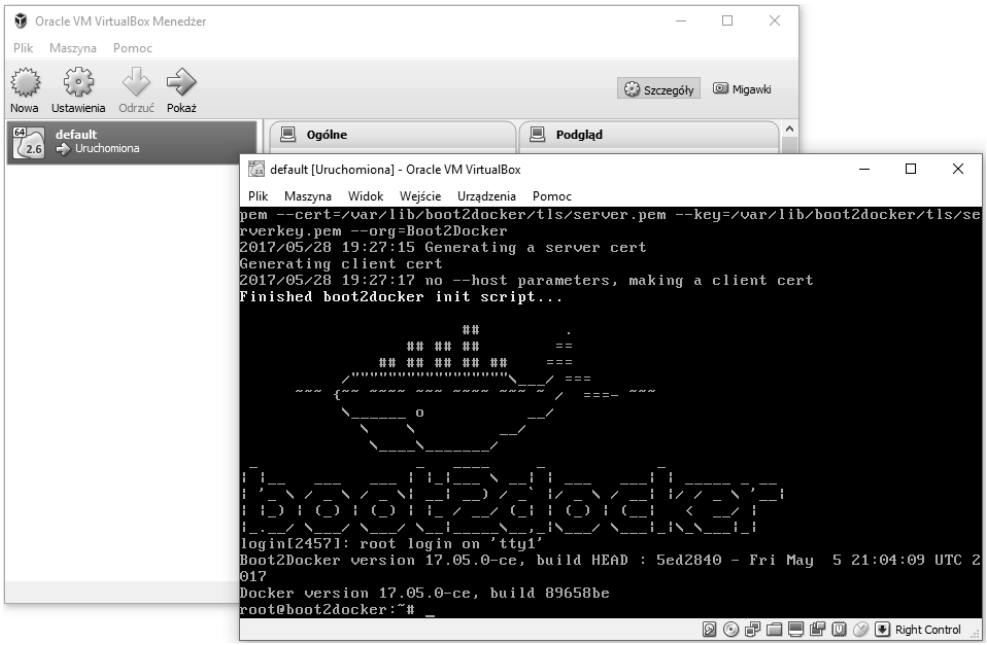
Compose to narzędzie wykonywane z poziomu wiersza poleceń jako polecenie `docker-compose`. Zastępuje ono stare narzędzie *fig*. Compose służy do definiowania i uruchamiania aplikacji Dockera z wieloma kontenerami. Choć bardzo łatwo można sobie wyobrazić taką aplikację (np. serwer WWW w jednym kontenerze, a bazę danych w drugim kontenerze), taki podział nie jest wymagany. A zatem, jeśli zdecydujesz, że aplikacja będzie zawarta w jednym kontenerze Dockera, nie będzie konieczne użycie polecenia `docker-compose`. Jednak w rzeczywistości z bardzo dużym prawdopodobieństwem aplikacja będzie rozmieszczona w wielu kontenerach. Polecenie `docker-compose` pozwala użyć pliku kompozycji do skonfigurowania usług aplikacji tak, aby mogły być uruchamiane razem w wyizolowanym środowisku. Korzystając następnie z jednego polecenia, tworzysz i uruchamiasz wszystkie usługi na bazie utworzonej konfiguracji. W przypadku aplikacji z wieloma kontenerami polecenie `docker-compose` sprawdza się znakomicie podczas programowania i testowania, a także w odniesieniu do przepływów integracji ciągłej.

Polecenie `docker-compose` zostanie zastosowane do utworzenia aplikacji z wieloma kontenerami w dalszej części książki, w rozdziale 6., „Tworzenie obrazów”.

Oracle VirtualBox

Sterownik Oracle VM VirtualBox to bezpłatny hipernadzorca *open source* firmy Oracle przeznaczony dla komputerów o architekturze x86. Zostanie domyślnie zainstalowany w trakcie instalacji narzędzia Docker Toolbox. Hipernadzorca obsługuje tworzenie maszyn wirtualnych działających w systemach Windows, Linux, BSD, OS/2, Solaris itp., a także zarządzanie nimi. W omawianym przypadku narzędzie Docker Machine korzystające ze sterownika hipernadzorcy VirtualBox użyje go do utworzenia i rozruchu niewielkiej dystrybucji systemu Linux, która umożliwia uruchomienie polecenia `docker-engine`. Warto wspomnieć, że możliwe jest też uruchomienie niewielkiej instalacji systemu Linux poddanej wirtualizacji bezpośrednio z poziomu samego hipernadzorcy VirtualBox.

Każda maszyna Dockera utworzona za pomocą polecenia `docker-machine` lub `Kitematic` będzie widoczna i dostępna do rozruchu w oknie hipernadzorcy VirtualBox uruchomionego bezpośrednio. Prezentuje to poniższy zrzut ekranu.



Możliwe jest uruchamianie, zatrzymywanie, resetowanie, zmienianie ustawień i wczytywanie dzienników w taki sam sposób, jak w przypadku innych systemów operacyjnych poddanych wirtualizacji.

> Hipernadzorca VirtualBox może być używany w systemie Windows lub MacOS do innych celów niż narzędzie Docker.

Git

Git to rozproszony system kontroli wersji, który jest powszechnie wykorzystywany na potrzeby tworzenia oprogramowania oraz innych zadań związanych z kontrolą wersji. System kładzie nacisk na szybkość, integralność danych oraz wsparcie rozproszonych, nieliniowych przepływów informacji. Narzędzie Docker Machine i klient Dockera bazują na modelu pobierania/wypychania systemu Git, aby uzyskiwać z sieci niezbędne zależności. Jeśli na przykład zdecydujesz się na uruchomienie obrazu Dockera, którego nie ma na komputerze lokalnym, Docker uzyska ten obraz z serwisu Docker Hub. Docker nie używa wewnętrznie systemu Git w odniesieniu do jakiegokolwiek rodzaju numeracji wersji zasobów. Bazuje jednak na haszowaniu, aby w unikalny sposób identyfikować warstwy systemu plików, co pod względem działania bardzo przypomina system Git. Początkowo Docker czerpał też inspirację z takich działań, jak zatwierdzanie, wypychanie i pobieranie. Git również wchodzi w skład pakietu instalacyjnego narzędzia Docker Toolbox.

Z perspektywy programisty dostępne są narzędzia, które okazują się szczególnie przydatne w jego codziennej pracy. Może to być wtyczka IntelliJ IDEA Docker Integration przeznaczona dla miłośników języka Java lub oprogramowanie Visual Studio 2015 Tools for Docker dla tych osób, które preferują język C#. Narzędzia te umożliwiają pobieranie i budowanie obrazów Dockera, tworzenie i uruchamianie kontenerów oraz realizowanie innych powiązanych zadań bezpośrednio z poziomu zintegrowanego środowiska programistycznego. Narzędzia te zostaną omówione bardziej szczegółowo w następnych rozdziałach.

Oprócz narzędzi dołączonych do pakietu dystrybucji Dockera (w przypadku starszych wersji systemu Windows będzie to Docker Toolbox albo Docker for Windows i Docker for Mac) istnieją setki niezależnych narzędzi, takich jak Kubernetes i Helios (służą do organizacji Dockera), Prometheus (monitorowanie statystyk) albo Swarm i Shipyard, które umożliwiają zarządzanie klastrami. Wraz z zyskiwaniem coraz większej popularności przez Dockera niemal każdego tygodnia pojawiają się nowe powiązane z nim narzędzia. W rozdziale 9., „Użycie Dockera w programowaniu", spróbujemy w skrócie omówić najbardziej interesujące spośród tych narzędzi, jak również dodatkowe zasoby.

Nie są to jednak jedyne dostępne narzędzia. Dodatkowo Docker zapewnia zestaw interfejsów API, które mogą być bardzo pomocne. Jednym z nich jest interfejs Remote API służący do zarządzania obrazami i kontenerami. Korzystając z niego, będziesz w stanie kierować obrazy do środowiska uruchomieniowego Dockera. Kontener może zostać przeniesiony na inny komputer z działającym Dockerem i uruchomiony na nim bez obaw o problemy ze zgodnością. Może to być szczególnie przydatne podczas tworzenia architektur modelu PaaS (ang. *Platform-as-a-Service*). Dostępny jest też interfejs Stats API, który zapewni bieżące informacje o wykorzystaniu zasobów (np. procesor, pamięć, sieciowe oraz blokowe wejście-wyjście) na potrzeby kontenerów. Ten punkt końcowy API może zostać użyty do tworzenia narzędzi informujących o działaniu kontenerów (na przykład w systemie produkcyjnym).

Podsumowanie

Na tym etapie odróżniasz wirtualizację od konteneryzacji. Mam nadzieję, że zauważyłeś korzyści wynikające z użycia drugiego rozwiązania. Wiesz również, jakie komponenty są dostępne do zainstalowania i wykorzystania. Rozpocznijmy podróż po świecie kontenerów i przejdźmy od razu do dzieła, przeprowadzając instalację oprogramowania.

Skorowidz

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z

Docker: już dziś korzystaj z narzędzi jutra!

Piętnastego marca 2013 roku na jednej z konferencji dla projektantów aplikacji zaprezentowano światu Dockera. Choć wystąpienie to trwało zaledwie kilka minut, wzbudziło ogromne zainteresowanie. Nic dziwnego — nowe narzędzie znacząco uprościło tworzenie oprogramowania i wdrażanie go na dużą skalę w dowolnym środowisku oraz usprawniło przepływ pracy. Docker ułatwia decyzje dotyczące architektury, co przekłada się na tworzenie narzędzi pomocniczych i ich wykorzystanie w różnych aplikacjach. Jednak aby w pełni skorzystać z tych licznych zalet, trzeba zrozumieć specyficzne podejście Dockera do budowy aplikacji.

Dzięki tej książce zrozumiesz, skąd się wziął lawinowy wzrost popularności Dockera. Przyjrzysz mu się z punktu widzenia projektanta i dowiesz się, dlaczego dzięki temu narzędziu projektowanie, testowanie i wdrażanie aplikacji jest szybsze i prostsze. Najpierw zapoznasz się z zagadnieniami podstawowymi, takimi jak kontenery, środowisko wykonawcze i narzędzia systemowe, aby potem skupić się na tworzeniu, uruchamianiu i publikowaniu obrazów Dockera. Znajdziesz tu również informacje o dotyczących go — i przydatnych — zasobach internetowych, a także o wartościowych narzędziach zewnętrznych, znakomicie poprawiających komfort pracy z Dockerem.

Najważniejsze zagadnienia:

- wprowadzenie do Dockera i przygotowanie środowiska pracy
- architektura Dockera: obrazy, woluminy, kontenery
- proces dystrybucji oprogramowania
- najlepsze praktyki tworzenia plików Dockera
- przykłady tworzenia rzeczywistych aplikacji w Dockerze

Jarosław Krochmalski — od kilkunastu lat tworzy oprogramowanie, specjalizuje się w aplikacjach dla branży finansowej. Jest projektantem z pasją, entuzjastą przejrzystego kodu i kunsztu w pisaniu oprogramowania. Otrzymał certyfikat Scrum Master. Szczególnie interesuje się nowymi technologiami związanymi z projektowaniem aplikacji internetowych, wzorcami projektowymi, architekturą dla przedsiębiorstw oraz wzorcami integracji. Brał udział w wielu projektach o dużej skali, takich jak międzynarodowe przekazy pieniężne, płatności ekspresowe i systemy gromadzenia danych. Obecnie pracuje jako konsultant w duńskiej firmie 7N.

Helion

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

Informatyka w najlepszym wydaniu

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

sięgnij po WIĘCEJ

KOD KORZYŚCI

ISBN 978-83-283-3534-9

cena: 54,90 zł

Packt