Nauka DOCKERA W MIESIĄC

ELTON STONEMAN

MANNING

Helion

Tytuł oryginału: Learn Docker in a Month of Lunches

Tłumaczenie: Piotr Rajca

ISBN: 978-83-283-7600-7

Helion S.A.
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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
https://ftp.helion.pl/przyklady/naudoc.zip

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

Printed in Poland.

Spis treści

1 Zanim zaczniesz

Docker jest platformą służącą do uruchamiania aplikacji w lekkich jednostkach nazywanych *kontenerami* (ang. *containers*). Kontenery są obecnie wszechobecne w branży oprogramowania i można je znaleźć wszędzie, zaczynając od funkcji bezserwerowych wykonywanych w chmurze, a kończąc na korporacyjnych narzędziach planowania strategicznego. Znajomość Dockera staje się jedną z kluczowych kompetencji dla operatorów i programistów z całej branży — w ankiecie przeprowadzonej przez serwis Stack Overflow w 2019 r. Docker znalazł się na pierwszym miejscu listy „najbardziej pożądanych" technologii (*https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted*).

Docker jest technologią stosunkowo łatwą do nauki. Możesz sięgnąć po tę książkę jako zupełny nowicjusz, a w rozdziale 2. już będziesz uruchamiał kontenery i w rozdziale 3. pakował aplikacje wykonywane w tych kontenerach. Każdy z rozdziałów niniejszej książki koncentruje się na praktycznych zadaniach i przedstawia przykłady działające na wszystkich platformach, na których można uruchomić Dockera — bez różnicy mogą z nich korzystać użytkownicy systemów Windows, Linux czy macOS.

Podróż, którą odbędziesz na stronach tej książki, została opracowana w trakcie moich wieloletnich doświadczeń związanych z nauczaniem Dockera. Każdy rozdział, z wyjątkiem tego, ma praktyczny charakter. Jednak ważne jest, abyś zanim zaczniesz poznawać Dockera, dowiedział się, w jaki sposób jego kontenery są używane w rzeczywistych zastosowaniach oraz jakiego rodzaju problemy można za ich pomocą rozwiązywać. Właśnie tymi zagadnieniami zajmiemy się w tym rozdziale. Opiszę tu także, w jaki sposób będę uczył o Dockerze, więc czytając go, zorientujesz się, czy ta książka spełni Twoje wymagania.

A teraz dowiedzmy się, do czego ludzie używają kontenerów — przedstawię tu pięć podstawowych scenariuszy, w których organizacje z powodzeniem stosują Dockera. Przekonasz się, że można go używać do rozwiązywania szerokiego zakresu

problemów, z których kilka na pewno będzie odpowiadać scenariuszom znanym Ci z własnej pracy. Pod koniec tego rozdziału będziesz już rozumiał, dlaczego Docker jest technologią, którą powinieneś znać, i będziesz wiedział, w jaki sposób tak książka Ci w tym pomoże.

1.1. Dlaczego kontenery podbiją świat?

Ja zacząłem swoją przygodę z Dockerem w roku 2014, kiedy pracowałem nad projektem związanym z dostarczaniem API na potrzeby urządzeń z Androidem. Zaczęliśmy używać Dockera jako narzędzia wspomagającego proces tworzenia oprogramowania — serwerów zarządzających kodem oraz serwerów budowania. Później nabraliśmy nieco pewności siebie i zaczęliśmy uruchamiać API w kontenerach jako środowiskach testowych. Pod koniec prac nad projektem już każde z używanych przez nas środowisk było obsługiwane przez Dockera, w tym także środowisko produkcyjne, podlegające rygorystycznym wymaganiom dotyczącym dostępności i skalowalności.

Kiedy zakończyłem prace nad projektem, nowy zespół otrzymał w spadku po nas jedynie pojedynczy plik README w repozytorium na GitHubie. Jedynym wymaganiem związanym z budowaniem, wdrażaniem i zarządzaniem aplikacją — w dowolnym środowisku — był Docker. Nowi programiści mogli pobrać kod źródłowy i jednym poleceniem zbudować aplikację oraz uruchomić ją lokalnie. Administratorzy z kolei używali dokładnie tego samego narzędzia do wdrażania kontenerów i zarządzania nimi w klastrze produkcyjnym.

W projektach tej wielkości przekazywanie efektów pracy nowemu zespołowi zajmuje około dwóch tygodni. Nowi programiści muszą zainstalować konkretne wersje kilku bądź kilkunastu narzędzi; to samo dotyczy administratorów, którzy także muszą zainstalować przynajmniej kilka zupełnie różnych programów narzędziowych. Docker centralizuje ten zestaw niezbędnych narzędzi i w ten sposób tak bardzo wszystkim wszystko upraszcza, że pomyślałem wtedy, że kiedyś kontenery będą używane w każdym projekcie.

W roku 2016 dołączyłem do zespołu Dockera i w ciągu ostatnich paru lat obserwowałem, jak moja wizja staje się rzeczywistością. Docker powoli staje się wszechobecny, po części dlatego, że w tak znaczny sposób upraszcza dostarczanie oprogramowania, a po części dlatego, że jest tak bardzo elastyczny — możemy go używać we wszystkich naszych projektach, starych i nowych, w Windowsach i Linuksie. Przekonajmy się zatem, jak można dopasować kontenery do tych różnych projektów.

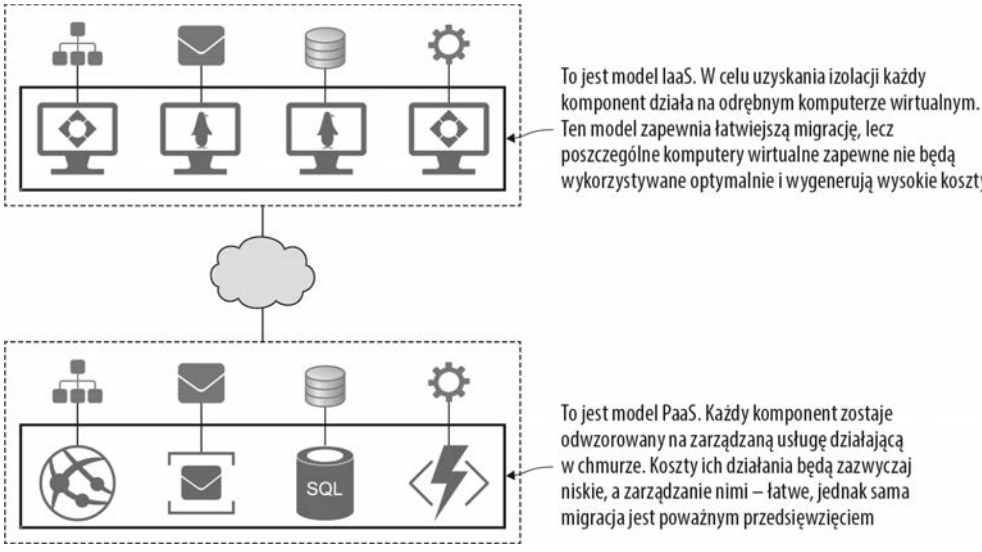
1.1.1. Przenoszenie aplikacji do chmury

Przeniesienie aplikacji do chmury znajduje się na samym początku listy priorytetów wielu organizacji. Jest to bardzo atrakcyjna opcja — niech to Microsoft, Amazon lub Google martwi się serwerami, dyskami, sieciami oraz ich zasilaniem. My zadbamy, by nasza aplikacja działała w globalnych centrach danych i miała praktycznie nieograniczone możliwości skalowalności. Zapewnimy sobie możliwość wdrażania nowych środowisk w ciągu paru minut, a koszty będziemy ponosić tylko za te zasoby, których faktycznie używamy. Ale w jaki sposób przenieść nasze aplikacje do chmury?

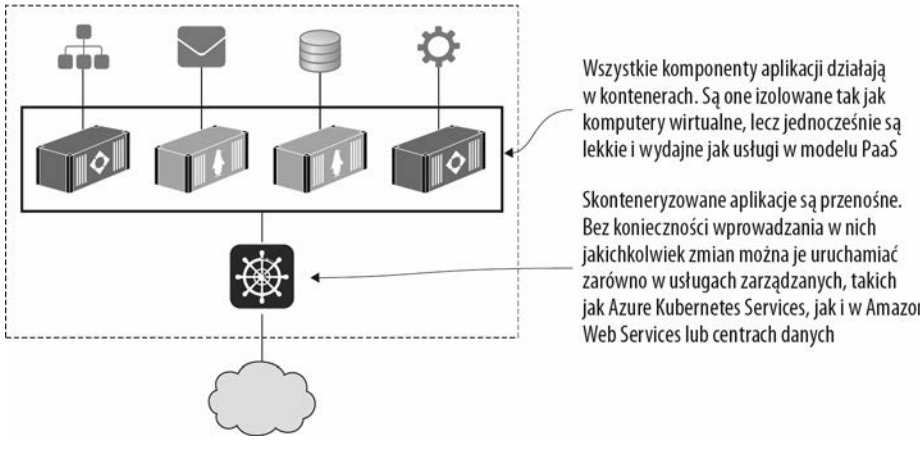
Niegdyś istniały dwa sposoby przeprowadzania migracji aplikacji do chmury: stworzenie rozwiązania typu „infrastruktura jako usługa" (*IaaS* — ang. *Infrastructure as a Service*) lub typu „platforma jako usługa" (*PaaS* — ang. *Platform as a Service*). Żadne z nich nie było rewelacyjne. Wybór między nimi sprowadzał się do znalezienia kompromisu. Wybranie rozwiązania typu PaaS oznaczało konieczność przeniesienia wszystkich elementów aplikacji do odpowiednich zarządzanych usług w chmurze. To trudny projekt, który co więcej zmusza nas do korzystania z jednej chmury, choć jednocześnie pozwala ograniczyć koszty. Alternatywą jest rozwiązanie typu IaaS, które polega na stworzeniu wirtualnej maszyny dla każdego komponentu naszej aplikacji. Zyskujemy w ten sposób możliwość stosowania różnych chmur, jednak koszty działania aplikacji są znacznie wyższe. Rysunek 1.1 pokazuje, jak wygląda typowa aplikacja rozproszona po przeniesieniu jej do chmury w formie rozwiązań typu IaaS oraz PaaS.

Docker stanowi trzecie rozwiązanie, w którym nie są konieczne żadne kompromisy. Każdy z elementów aplikacji jest przenoszony do kontenera, a następnie całą aplikację uruchamiamy w tych kontenerach, używając Azure Kubernetes Service lub Elastic Container Service Amazona, bądź też na własnym klastrze Dockera działającym w jakimś centrum danych. W rozdziale 7. dowiesz się, jak pakować i uruchamiać takie rozproszone aplikacje w kontenerach, natomiast w rozdziałach 13. i 14. — jak można je uruchamiać i skalować w środowiskach produkcyjnych. Na rysunku 1.2 przedstawiłem sposób migracji aplikacji rozproszonej do chmury z użyciem Dockera, który zapewnia nam przygotowanie przenośnej aplikacji, którą będziemy mogli przy niskich kosztach uruchamiać na dowolnej chmurze albo w centrum danych, albo na własnym laptopie.

Oczywiście przeniesienie aplikacji do kontenerów też wymaga pewnych nakładów: trzeba będzie zapisać poszczególne etapy istniejącego procesu instalacji do skryptu, tak zwanego pliku *Dockerfile*, a dokumenty wdrożeniowe do odpowiednich, opisowych manifestów w formatach Docker Composer lub Kubernetes. Nie trzeba wprowadzać żadnych zmian w kodzie, a w końcowym efekcie w każdym środowisku, czy to na laptopie, czy w chmurze, aplikacja będzie działać tak samo i używać tego samego stosu technologicznego.



Rysunek 1.1. *Początkowo dostępne sposoby migracji do chmury — rozwiązanie typu IaaS wiązało się z uruchamianiem wielu mało wydajnych wirtualnych komputerów i ponoszeniem wysokich kosztów miesięcznych, a rozwiązanie typu PaaS pozwalało ograniczyć koszty, lecz komplikowało i wydłużało proces migracji*



Rysunek 1.2. *Ta sama aplikacja przeniesiona na Dockera przed uruchomieniem w chmurze. Zapewnia ona zalety modelu PaaS, jeśli chodzi o koszty oraz przenośność modelu IaaS, a przy tym cechuje się łatwością użycia, którą zapewnia jedynie Docker*

1.1.2. Modernizacja starych aplikacji

W kontenerze działającym w chmurze można uruchomić praktycznie każdą aplikację, jednak nie sposób będzie w pełni wykorzystać możliwości Dockera oraz platform chmurowych w przypadku aplikacji stworzonych w oparciu o stosowany wcześniej monolityczny projekt. Takie monolity dobrze działają w kontenerach, jednak ograniczają naszą sprawność. Stosując kontenery, możemy wykonać zautomatyzo-

wane, etapowe wdrożenie nowych możliwości w środowisku produkcyjnym w czasie 30 sekund. Jeśli jednak nowa możliwość jest elementem monolitycznego rozwiązania, które w całości liczy dwa miliony wierszy kodu, to przed jego wdrożeniem najprawdopodobniej trzeba będzie poświęcić dwa tygodnie na wykonanie niezbędnych testów regresyjnych.

Przeniesienie aplikacji na Dockera jest doskonałym pierwszym krokiem na drodze do modernizacji jej architektury i zastosowania nowych wzorców bez konieczności przepisywania całego jej kodu. To rozwiązanie jest całkiem proste: zaczynamy od przeniesienia aplikacji do jednego kontenera, używając przy tym składni plików Dockerfile i Docker Compose przedstawionych w dalszej części tej książki. W ten sposób uzyskamy monolityczną aplikację działającą w kontenerze.

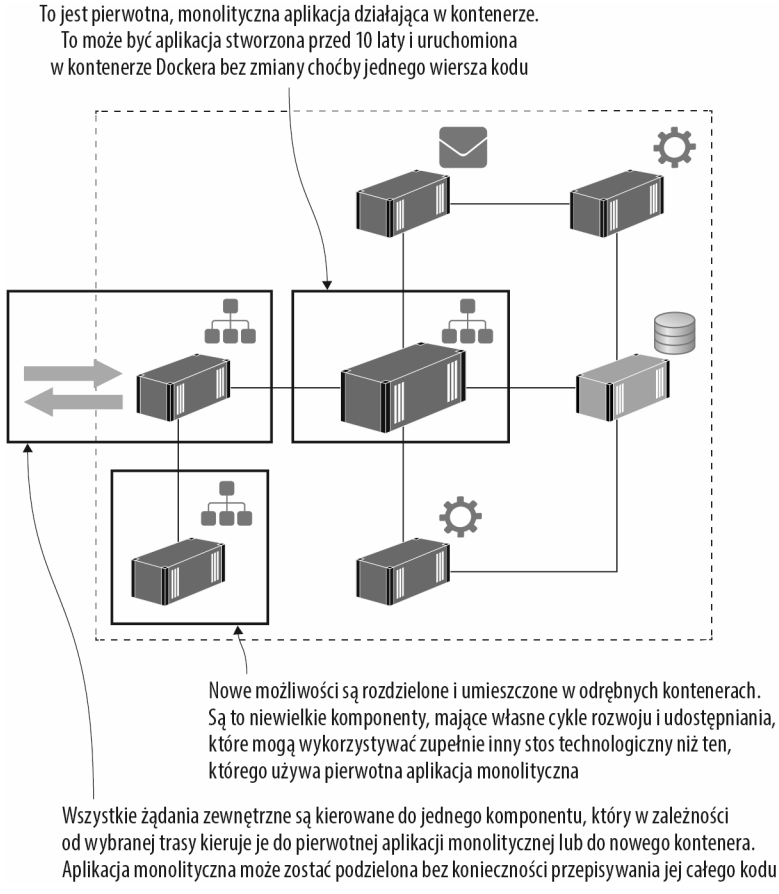
Kontenery mogą tworzyć swoje własne sieci wirtualne, więc mogą komunikować się ze sobą bez udostępniania czegokolwiek światu zewnętrznemu. To z kolei oznacza, że możemy zacząć dzielić naszą aplikację na fragmenty i przenosić jej poszczególne możliwości funkcjonalne do odrębnych kontenerów — w ten sposób początkowe monolityczne rozwiązanie można stopniowo przekształcać w aplikację rozproszoną, której kompletny zestaw możliwości funkcjonalnych będzie zapewniany przez grupę kontenerów. Na rysunku 1.3 pokazałem, jak takie rozwiązanie mogłoby wyglądać w przypadku przykładowej architektury aplikacji.

W taki sposób możemy zyskać wiele korzyści, które zapewnia architektura bazująca na mikrousługach. Kluczowe możliwości funkcjonalne naszej aplikacji zostaną rozdzielona na niewielkie, izolowane jednostki, którymi można zarządzać niezależnie od innych. To z kolei oznacza, że zyskujemy możliwość szybkiego testowania zmian, gdyż nie zmieniamy jednego monolitu, a jedynie kontenery, w których działa konkretna, zmodyfikowana możliwość funkcjonalna aplikacji. Jej poszczególne możliwości można także dowolnie skalować, można także używać innych technologii, które będą zaspokajać nasze wymagania.

Modernizowanie architektur starych aplikacji z wykorzystaniem Dockera jest łatwe — przekonasz się o tym sam, wykonując praktyczne przykłady przedstawione w rozdziałach 20. i 21. Robiąc to, możesz dostarczyć bardziej zwinną, skalowalną i odporną aplikację, a co więcej, możesz to robić etapami, a nie poświęcając 18 miesięcy na przepisanie aplikacji od podstaw.

1.1.3. Tworzenie nowej aplikacji chmurowej

Docker pomaga przenosić istniejące aplikacje do chmury i to niezależnie od tego, czy są to aplikacje rozproszone, czy monolityczne. Docker ułatwia dzielenie ich i nadanie im nowoczesnej architektury niezależnie od tego, czy będą one uruchamiane w chmurze, czy w centrum danych. A jeśli chodzi o nowe projekty, budowane od początku w oparciu o zasady tworzenia rozwiązań chmurowych, to zastosowanie Dockera może w ogromnym stopniu przyspieszyć prace nad nimi.

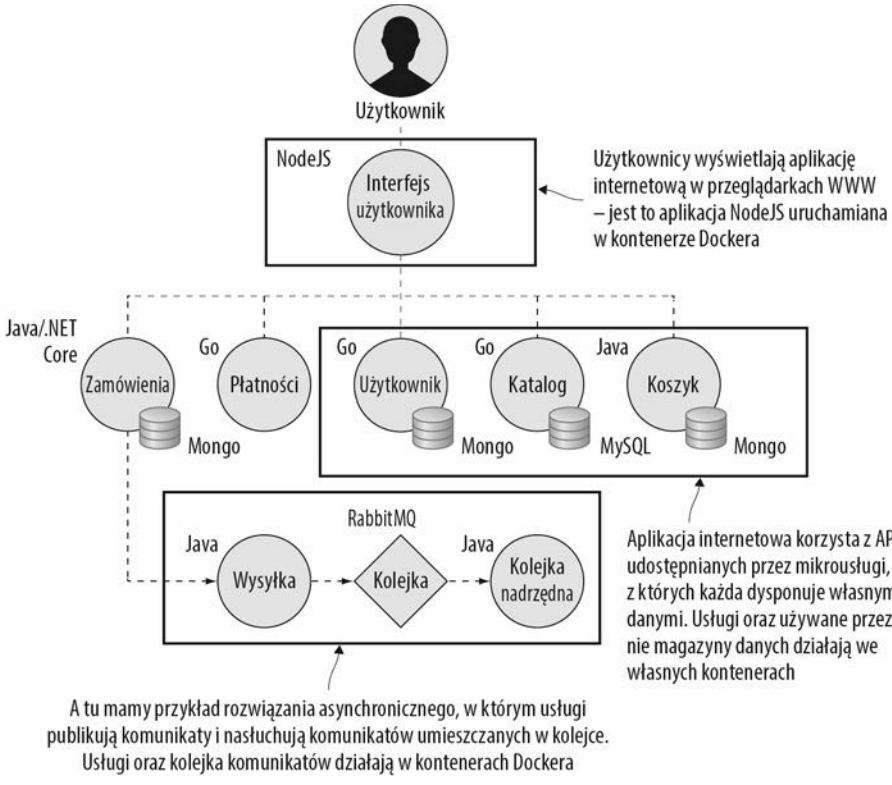


Rysunek 1.3. *Dzielenie monolitu do postaci aplikacji rozproszonej bez przepisywania całego projektu od podstaw. Wszystkie jej komponenty działają w kontenerach Dockera, a komponent kierujący ruchem sieciowym decyduje, czy żądania będą obsługiwane przez monolit, czy przez nowe mikrousługi*

Organizacja Cloud Native Computing Foundation (CNCF) opisuje te nowe architektury jako używające „stosu oprogramowania typu *open source* do wdrażania aplikacji jako mikrousług, umieszczonych w odrębnych kontenerach i dynamicznie zarządzających tymi kontenerami w celu optymalizacji wykorzystania zasobów".

Typową architekturę nowych aplikacji stworzonych jako mikrousługi przedstawiłem na rysunku 1.4; jest to demonstracyjna aplikacja stworzona przez społeczność, dostępna w serwisie GitHub na stronie *https://github.com/microservices-demo*.

To fantastyczna przykładowa aplikacja, jeśli ktoś chce przekonać się, jak w praktyce można implementować mikrousługi. Każdy komponent dysponuje własnymi danymi i udostępnia je za pośrednictwem API. Interfejsem użytkownika jest aplikacja internetowa używająca tych API. Ta przykładowa aplikacja korzysta z różnych języków programowania i różnych technologii baz danych, jednak każdy jej kom-



Rysunek 1.4. *Aplikacje chmurowe są tworzone z wykorzystaniem architektur bazujących na mikrousługach, których poszczególne komponenty działają w odrębnych kontenerach*

ponent posiada własny plik Dockerfile pozwalający na umieszczenie go w kontenerze, a cała aplikacja jest zdefiniowana w pliku Docker Compose.

W rozdziale 4. dowiesz się, jak używać Dockera do kompilowania kodu w ramach procesu pakowania aplikacji. Oznacza to, że do budowania i uruchamiania aplikacji w taki sposób nie będziemy potrzebować żadnych narzędzi programistycznych. Wystarczy, że programiści zainstalują Dockera, sklonują kod źródłowy, a następnie zbudują i uruchomią aplikację przy użyciu jednego polecenia.

Docker ułatwia także dodawanie do aplikacji oprogramowania innych, niezależnych twórców i wzbogacania w ten sposób jej możliwości bez pisania żadnego kodu. Docker Hub jest publicznym serwisem, na którym twórcy mogą udostępniać swoje oprogramowanie działające w kontenerach. Fundacja CNCF publikuje listę projektów typu *open source* obejmujących rozwiązania, zaczynając od monitorowania, a kończąc na kolejkach, których można używać i które są dostępne za darmo w serwisie Docker Hub.

1.1.4. Innowacje techniczne: rozwiązania bezserwerowe i inne

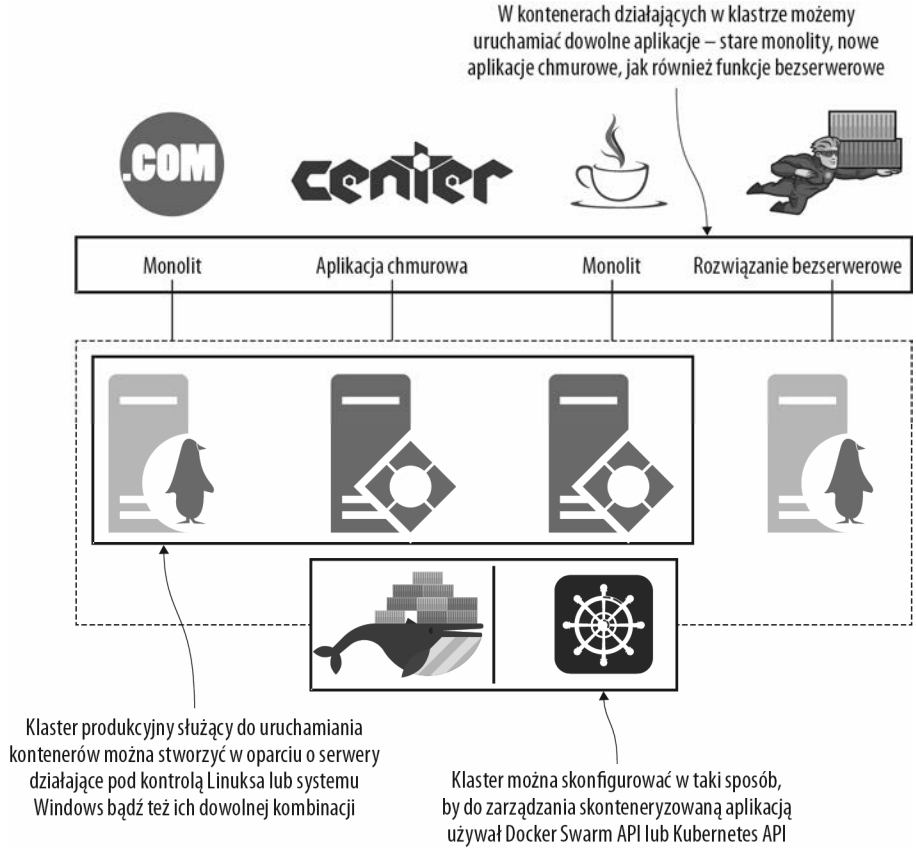
Jednym z kluczowych czynników rozwoju nowoczesnej branży informatycznej jest spójność: zespoły dążą do stosowania tych samych narzędzi, procesów i środowisk uruchomieniowych we wszystkich tworzonych projektach. Można to osiągnąć, stosując Dockera i jego kontenery do wszystkiego, zaczynając od starych, monolitycznych aplikacji .NET przeznaczonych dla systemów Windows, a kończąc na nowych aplikacjach pisanych w języku Go i działających w Linuksie. Można tworzyć klastry Dockera umożliwiające uruchamianie wszystkich tych aplikacji, a zatem możemy budować i wdrażać wszystko, co jest związane z naszą aplikacją, i zarządzać tym, a wszystko w taki sam sposób.

Innowacje techniczne nie powinny być oddzielone od normalnie działających aplikacji. Docker stanowi podstawę dla wielu spośród największych innowacji, możemy więc ciągle używać tych samych narzędzi i technik podczas eksplorowania nowych obszarów. Jedną z najbardziej fascynujących innowacji (oczywiście nie licząc kontenerów) są funkcje bezserwerowe (nazywane także funkcjami *server less* — ang. *serverless functions*). Na rysunku 1.5 pokazałem, w jaki sposób można wykonywać całe aplikacje — stare monolity, nowe aplikacje chmurowe oraz funkcje bezserwerowe — na jednym klastrze Dockera, który może działać zarówno w chmurze, jak i w centrum danych.

Bezserwerowość (ang. *serverless*) jest ściśle powiązana z kontenerami. Celem tworzenia rozwiązań bezserwerowych jest zapewnienie programistom możliwości pisania kodu funkcji, przekazywania ich do usługi, która następnie zbuduje i spakuje ten kod. Kiedy klient będzie chciał użyć takiej funkcji, usługa uruchomi instancję funkcji w celu przetworzenia żądania. W tym przypadku nie ma żadnych serwerów budowania, potoków ani serwerów produkcyjnych, którymi trzeba by zarządzać — wszystkimi tymi zagadnieniami zajmuje się platforma.

W niezauważalny sposób wszystkie chmurowe rozwiązania bezserwerowe używają Dockera do pakowania kodu i tworzenia kontenerów używanych do wykonywania funkcji. Jednak funkcje umieszczone w chmurze nie są przenośne — nie możemy wziąć naszych funkcji działających w usłudze AWS Lambda i uruchomić ich na platformie Azure, gdyż nie istnieje żaden otwarty standard rozwiązań bezserwerowych. Jeśli chcemy tworzyć rozwiązania bezserwerowe, a jednocześnie nie być związanym z konkretną usługą chmurową, bądź też jeśli nasze rozwiązanie ma działać w centrum danych, to możemy uruchomić własną platformę na Dockerze, używając przy tym takiego oprogramowania jak Nuclio, OpenFaaS lub Fn Project — wszystkie stanowią popularne platformy bezserwerowe i są udostępniane jako oprogramowanie typu *open source*.

Inne najważniejsze innowacje, takie jak uczenie maszynowe, blockchain oraz internet rzeczy (ang. *Internet of Things*, w skrócie *IoT*) także mogą skorzystać z jednolitego modelu pakowania i wdrażania zapewnianego przez Dockera. Przekonasz



Rysunek 1.5. *Pojedynczy klaster serwerów z Dockerem może obsługiwać aplikacje dowolnego typu, dając nam jednocześnie możliwość budowania, wdrażania i zarządzania nimi wszystkimi w taki sam sposób niezależnie od używanej przez nie architektury oraz stosu technologicznego*

się, że wszystkie główne projekty można znaleźć w serwisie Docker Hub, czego doskonałymi przykładami mogą być TensorFlow oraz Hperldger. Jest to szczególnie interesujące w kontekście internetu rzeczy, gdyż Docker współpracował z Arm nad zastosowaniem kontenerów jako domyślnego środowiska uruchomieniowego dla urządzeń brzegowych i IoT.

1.1.5. Cyfrowe przetwarzanie przy użyciu rozwoju i operacji

Wszystkie te scenariusze są związane z wykorzystaniem technologii, jednak największy problem, z jakim muszą się zmierzyć organizacje, ma charakter operacyjny — dotyczy to przede wszystkim większych i starszych przedsiębiorstw. Zespoły są w nich tradycyjnie dzielone na „programistów” i „operatorów”, a każda z tych grup jest odpowiedzialna za zupełnie inne fragmenty cyklu rozwoju projektu.

Problemy z przekroczeniem czasu zakończenia projektu stają się cyklem wzajemnych oskarżeń i wdrażane są kontrole jakości, które mają zapobiec przyszłym porażkom. W końcu stosowanych jest tak wiele kontroli jakości, że w ciągu roku udaje się przeprowadzić nie więcej niż dwa lub trzy wdrożenia, a i te są ryzykowne i bardzo pracochłonne.

Rozwiązania określane terminem „rozwój i operacje" (ang. *DevOps — development and operations*) mają na celu zapewnienie zwinności procesowi wytwarzania i utrzymania oprogramowania poprzez wymaganie, by to jeden zespół odpowiadał za cały cykl życia aplikacji — łącząc ze sobą „programowanie" i „operacje". Rozwój i operacje to przede wszystkim zmiana kulturowa, która może sprawić, że firma zamiast wielkich kwartalnych wdrożeń zacznie stosować wdrożenia bardzo małe, lecz przeprowadzane codziennie. Jednak wprowadzenie takiej zmiany bez modyfikacji technologii używanych przez zespół jest bardzo trudne.

Operatorzy mogą dysponować znajomością takich narzędzi jak Bash, Nagios, PowerShell czy też System Center. Programiści z kolei mogą używać takich programów jak Make, Maven, NuGet czy też MSBuild. Bardzo trudno jest stworzyć jeden zespół, kiedy jego poszczególni członkowie nie posługują się żadnymi wspólnymi technologiami. I właśnie w tym może pomóc Docker. Naszą transformację technik rozwoju i operacji możemy oprzeć na zastosowaniu kontenerów, a kiedy to zrobimy, okaże się, że cały zespół używa plików Dockerfile i Docker Compose, mówi tym samym językiem i używa tych samych narzędzi.

Ale to nie wszystko. Istnieje bardzo potężny framework służący do implementacji rozwoju i technologii, nosi on nazwę CALMS, co jest skrótem od angielskich słów: Culture, Automation, Lean, Metrics oraz Sharing — kultura, automatyzacja, prostota, metryki i udostępnianie. Docker doskonale spełnia wszystkie te wytyczne: automatyzacja jest kluczowym aspektem uruchamiania kontenerów, aplikacje rozproszone są tworzone w oparciu o zasadę prostoty, metryki z procesu tworzenia aplikacji oraz jej wdrażania można z łatwością publikować, a serwis Docker Hub został stworzony w celu udostępniania, a nie powielania pracy.

1.2. Dla kogo jest przeznaczona ta książka?

Pięć scenariuszy, które przedstawiłem w poprzednim podrozdziale, obejmuje praktycznie wszystkie działania, jakie obecnie wykonuje się w branży programistycznej, i mam nadzieję, że już jest dla Ciebie jasne, że Docker stanowi ich punkt kluczowy. Ta książka jest dla Ciebie, jeśli zależy Ci na wykorzystaniu Dockera do rozwiązywania tego typu praktycznych problemów. Przeprowadzi Cię ona od przysłowiowego „zera" do umiejętności uruchamiania aplikacji w klastrze o produkcyjnej jakości.

Ta książka ma Cię nauczyć używać Dockera, dlatego też nie opisuję w niej zbyt szczegółowo samych wewnętrznych sposobów jego działania. Nie będę gruntownie opisywał, czym jest `containerd`, pisał o linuksowych rozwiązaniach niskiego poziomu, takich jak `cgroups` i `namespaces`, ani o stosowanej w systemie Windows usłudze Windows Host Capture. Gdybyś jednak chciał poznać te szczegóły, to znajdziesz je w drugim wydaniu książki Jeffa Nickoloffa i Stephena Kuenzliego pt. *Docker in Action*, wydanej przez wydawnictwo Manning.

Przykłady prezentowane w tej książce będą działać na wszystkich platformach systemowych, można więc je wykonywać w systemie Windows, macOS i Linux, a nawet na urządzeniach z procesorami Arm — możesz zatem użyć nawet swojego Raspberry Pi. W przykładach używałem kilku różnych języków programowania, jednak tylko takich, które działają na wielu platformach, dlatego też oprócz innych technologii używam .NET Core zamiast .NET Framework (która działa tylko w systemie Windows). Gdybyś chciał dokładniej poznać zagadnienia związane ze stosowaniem kontenerów w systemie Windows, to doskonałym źródłem informacji będzie mój blog (*https://blog.sixeyed.com*).

I w końcu ta książka jest o Dockerze, dlatego też do wdrożeń produkcyjnych będę używał Docker Swarm — technologii tworzenia klastrów wbudowanej w Dockera. W rozdziale 12. przedstawiłem Kubernetes oraz wyjaśniłem, jak wybrać jedno z tych dwóch konkurencyjnych rozwiązań — Docker Swarm i Kubernetes — jednak nie będę opisywał Kubernetes szczegółowo. Poznanie Kubernetes samo w sobie wymaga co najmniej miesiąca, jednak w gruncie rzeczy jest to po prostu inny sposób uruchamiania kontenerów Dockera, jeśli więc używasz tej technologii, to i tak przyda Ci się wszystko, o czym napisałem w tej książce.

1.3. Tworzenie środowiska roboczego

A teraz bierzmy się do roboty. Wszystkim, czego potrzebujesz, by uczyć się Dockera z tej książki, jest sam Docker oraz kody źródłowe przykładów.

1.3.1. Instalacja Dockera

Darmowa wersja Dockera — Docker Community Edition — w zupełności wystarczy do zastosowań związanych z wytwarzaniem oprogramowania, a nawet do zastosowań produkcyjnych. Jeśli używasz nowej wersji systemu Windows 10 lub macOS, to najlepszym rozwiązaniem będzie zainstalowanie programu Docker Desktop; starsze wersje tych systemów wymagają użycia programu Docker Toolbox. Docker udostępnia także pakiety instalacyjne dla wszystkich głównych dystrybucji Linuksa. Zacznij od zainstalowania Dockera, używając opcji, która będzie Ci najbardziej

odpowiadać; do pobierania plików będziesz musiał utworzyć konto w serwisie Docker Hub, jednak konto jest darmowe i pozwala udostępniać innym przygotowane aplikacje korzystające z Dockera.

INSTALACJA PROGRAMU DOCKER DESKTOP W SYSTEMIE WINDOWS 10

Aby zainstalować Docker Desktop, będziesz potrzebował systemu Windows 10 w wersji Professional lub Enterprise. Poza tym będziesz musiał upewnić się, że masz zainstalowane wszystkie aktualizacje systemu — musisz dysponować co najmniej wersją o numerze 1809 (możesz ją sprawdzić, wykonując z poziomu wiersza poleceń komendę `winwer`). Wyświetl w przeglądarce stronę *https://www.docker.com/products/docker-desktop* i kliknij przycisk, aby pobrać wersję programu dla systemu Windows. Po pobraniu programu uruchom go i zaakceptuj wszystkie domyślne ustawienia. Kiedy Docker Desktop będzie działać, w obszarze powiadomień paska zadań, tuż obok zegara, będzie widoczna charakterystyczna ikona Dockera przedstawiająca wieloryba.

INSTALACJA PROGRAMU DOCKER DESKTOP W SYSTEMIE MACOS

Aby używać programu Docker Desktop na komputerze Mac, będziesz potrzebował systemu macOS Sierra 10.12 lub nowszego — aby wyświetlić numer wersji aktualnie używanego systemu, kliknij ikonę Apple w lewym górnym rogu paska menu i wybierz opcję *Ten Mac*. Aby pobrać Docker Desktop, przejdź na stronę *https://www.docker.com/products/docker-desktop* i wybierz wersję programu dla komputerów Mac. Po zakończeniu pobierania uruchom program i zaakceptuj wszystkie ustawienia domyślne. Kiedy Docker Desktop będzie działać, na pasku zadań będzie widoczna obok zegara charakterystyczna ikona Dockera przedstawiająca wieloryba.

INSTALACJA PROGRAMU DOCKER TOOLBOX

Jeśli używasz starszej wersji systemu Windows lub OS X, to możesz zainstalować program Docker Toolbox. Twoje końcowe doświadczenia będą identyczne, jednak rozwiązanie to będzie mieć kilka różnic w stosunku do programu Docker Desktop. Przejdź na stronę *https://docs.docker.com/docker-for-windows/docker-toolbox/* i wykonaj zamieszczone na niej instrukcje — będziesz potrzebował zainstalowanego oprogramowania do tworzenia i uruchamiania maszyn wirtualnych, takiego jak VirtualBox (Docker Desktop jest lepszym rozwiązaniem, oczywiście jeśli możesz go użyć, gdyż nie zmusza do stosowania dodatkowego oprogramowania do wirtualizacji).

INSTALACJA DOCKER COMMUNITY EDITION I DOCKER COMPOSE

Jeśli używasz systemu Linux, to stosowana dystrybucja będzie zapewne udostępniać jakąś wersję Dockera, którą będziesz mógł zainstalować — jednak to nie jest rozwiązanie, z którego będziesz chciał skorzystać. Takie wersje Dockera są zapewne bardzo stare, gdyż zespół twórców Dockera obecnie udostępnia swoje własne pakiety instalacyjne. W środowiskach innych niż produkcyjne można skorzystać ze skryptu udostępnianego przez twórców Dockera i aktualizowanego po wprowadzeniu każdej jego nowej wersji — w tym celu wystarczy wyświetlić stronę *https://get.docker.com* i postępować zgodnie z instrukcją, by pobrać i zainstalować Dockera. Następnie należy wyświetlić stronę *https://docs.docker.com/compose/install/* i wykonać zamieszczone na niej instrukcje, by zainstalować Docker Compose.

INSTALACJA DOCKERA W SYSTEMIE WINDOWS SERVER LUB W SERWEROWYCH DYSTRYBUCJACH LINUKSA

Produkcyjne wdrożenia Dockera mogą używać jego wersji Community Edition, jeśli jednak zależy Ci na wsparciu dla środowiska wykonującego kontenery, to zapewne będziesz chciał skorzystać z komercyjnej wersji Dockera o nazwie Docker Enterprise. Ta wersja bazuje na Docker Community Edition, zatem wszystko, czego się nauczysz w tej książce, będzie obowiązywać także w razie stosowania Docker Enterprise. Docker Enterprise jest dostępny w wersjach na wszystkie najważniejsze dystrybucje Linuksa oraz w wersjach dla systemów Windows Server 2016 oraz Windows Server 2019. Wszystkie wersje Docker Enterprise wraz z instrukcjami dotyczącymi sposobów ich instalacji można znaleźć w serwisie Docker Hub, na stronie *https://hub.docker.com/search/?q=&type=edition&offering=enterprise*.

1.3.2. Weryfikacja działania Dockera

Platforma Dockera składa się z kilku komponentów, jednak na potrzeby tej książki musimy jedynie sprawdzić, czy Docker działa i czy został zainstalowany program Docker Compose.

Zaczniemy od sprawdzenia samego Dockera, co można zrobić przy użyciu polecenia `docker version`:

```
PS> docker version
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea
 Built:             Wed Nov 13 07:22:37 2019
 OS/Arch:           windows/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.5
```

```
API version:      1.40 (minimum version 1.24)
Go version:       go1.12.12
Git commit:       633a0ea
Built:            Wed Nov 13 07:36:50 2019
OS/Arch:          windows/amd64
Experimental:     false
```

Wyniki, które zobaczysz, będą inne, gdyż na pewno zmieni się wersja oprogramowania, a być może będziesz także używał innego systemu operacyjnego; jeśli jednak w wynikach zobaczysz numer wersji klienta (Client) i serwera (Server), będzie to znaczyć, że Docker działa prawidłowo. Nie przejmuj się na razie tym klientem i serwerem — o architekturze Dockera dowiesz się w następnym rozdziale.

Kolejnym krokiem będzie przetestowanie Docker Compose — to kolejny program uruchamiany z poziomu wiersza poleceń, który współdziała z Dockerem. Aby go sprawdzić, wykonaj polecenie `docker-compose version`:

```
PS> docker-compose version
docker-compose version 1.25.4, build 8d51620a
docker-py version: 4.1.0
CPython version: 3.7.4
OpenSSL version: OpenSSL 1.1.1c  28 May 2019
```

Także w tym przypadku wyświetlone informacje mogą być inne od tych przedstawionych w tekście, jeśli jednak zobaczysz listę wersji, a nie komunikaty o błędach, to wszystko będzie w porządku.

1.3.3. Pobieranie kodów źródłowych do książki

Spolonizowane kody źródłowe przykładów do książki są dostępne na stronie wydawnictwa Helion, pod adresem *https://ftp.helion.pl/przyklady/naudoc.zip*. Wystarczy je pobrać i rozpakować do dowolnie wybranego katalogu.

1.3.4. Pamiętanie o poleceniach czyszczących

Docker nie usuwa kontenerów ani pakietów aplikacji automatycznie. Kiedy wyjdziesz z programu Docker Desktop (lub zatrzymasz usługę Dockera), wszystkie kontenery zostaną zatrzymane i nie będą zużywać ani procesora, ani pamięci; jeśli jednak chcesz przeprowadzić dodatkowe porządki, to po zakończeniu każdego z rozdziałów możesz wykonać następujące polecenie:

```
docker container rm -f $(docker container ls -aq)
```

Jeśli chciałbyś odzyskać miejsce na dysku zajmowane przez obrazy Dockera utworzone podczas ćwiczeń, to wykonaj następujące polecenie:

```
docker image rm -f $(docker image ls -f reference='diamol/*' -q)
```

Docker jest na tyle inteligentny, że pobiera wszystko, co jest mu potrzebne do działania, dlatego też nic nie stoi na przeszkodzie, by powyższe polecenia wykonywał w dowolnym momencie. Kiedy następnym razem uruchomisz kontenery i Docker nie znajdzie wszystkiego, czego potrzebuje, na lokalnym komputerze, to pobierze niezbędne elementy.

1.4. Jak być efektywnym od zaraz?

„Bycie efektywnym od zaraz" to kolejna zasada serii książek „w miesiąc". Wszystkie kolejne rozdziały książki koncentrują się na pokazywaniu i nauczaniu praktycznych umiejętności.

Każdy rozdział zaczyna się od krótkiego wprowadzenia do opisywanego zagadnienia, po którym przedstawiane są ćwiczenia pokazujące, jak w praktyce zastosować opisywane rozwiązania, używając do tego Dockera. Następnie zamieszczane jest podsumowanie uzupełnione o dodatkowe szczegóły mogące stanowić odpowiedzi na ewentualne pytania, które nasuną Ci się podczas lektury i wykonywania ćwiczeń. I wreszcie na samym końcu rozdziałów znajduje się praktyczne laboratorium, które wprowadzi Cię w zagadnienia opisywane w następnym rozdziale.

Wszystkie zagadnienia opisywane w tej książce koncentrują się na zadaniach, które są bardzo użyteczne w rzeczywistym świecie. Pozwolą Ci one błyskawicznie, bo już podczas lektury tekstu, nabyć praktyczne umiejętności, a kończąc poszczególne rozdziały, będziesz rozumiał, jak stosować te nowe umiejętności. A zatem spróbujmy uruchomić jakieś kontenery!

DOCKER. NALEŻY ZNAĆ. TRZEBA UŻYWAĆ!

U podstaw sukcesu Dockera leżał bardzo prosty pomysł: spakowanie aplikacji w lekkim, łatwym do zainstalowania kontenerze. Nagle się okazało, że można zarządzać aplikacjami bez budowania złożonej infrastruktury. Kontenery są niezależne od platformy i wszechstronne. Upraszczają opracowanie, testowanie, wdrażanie i skalowanie aplikacji, a także ułatwiają automatyzację przepływu pracy i ciągły rozwój aplikacji. Docker jest przy tym rozwiązaniem bezpłatnym, o otwartych źródłach. Coraz większa liczba użytkowników, programistów i administratorów przekonuje się do tej technologii, co sprawia, że lawinowo rośnie liczba wdrożeń. Dziś Dockera po prostu trzeba znać!

Ta książka składa się z ponad dwudziestu krótkich, praktycznych lekcji, w ramach których przedstawiono najważniejsze koncepcje związane ze stosowaniem Dockera. Dzięki niej szybko i bez problemów zaczniesz wdrażać aplikacje kontenerowe w środowisku produkcyjnym. Skupisz się na rzeczywistych zadaniach i stopniowo będziesz zdobywać doświadczenie związane z Dockerem, aplikacjami rozproszonymi, orkiestracją i ekosystemem kontenerów. Publikacja zawiera mnóstwo wskazówek, przykładów, ćwiczeń i rozbudowanych projektów, które ułatwią nabywanie wiedzy i przydatnych umiejętności. Każdy rozdział powinien Ci zająć nie więcej niż godzinę: w tym czasie zapoznasz się z niewielką porcją teorii, wykonasz ćwiczenia i przeanalizujesz praktyczne przykłady. W rezultacie po lekturze będziesz pewnie tworzyć i uruchamiać aplikacje w Dockerze.

W książce:

- gruntowne wprowadzenie do koncepcji Dockera
- pakowanie aplikacji w kontenerach
- uruchamianie kontenerów w środowisku produkcyjnym
- tworzenie zoptymalizowanych obrazów Dockera
- uruchamianie i skalowanie skonteneryzowanych aplikacji

Elton Stoneman najpierw specjalizował się w zagadnieniach związanych z platformą .NET i projektował duże systemy korporacyjne. Później pracował jako architekt w firmie Docker i do dziś, jako niezależny konsultant, pomaga różnym organizacjom w stosowaniu kontenerów. Od lat posiada przyznany przez Microsoft tytuł MVP. Jest autorem internetowych kursów szkoleniowych.

ISBN 978-83-283-7600-7

Cena: 119,00 zł

MANNING