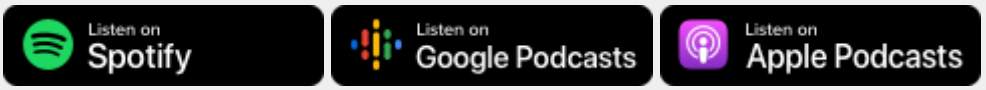
Pierwsze kroki w IT

PODCAST



Co to jest Docker

Gość: Wojciech Lepczyński

Wszystkie polecane materiały i linki znajdziesz na stronie odcinka:

https://devmentor.pl/b/co-to-jest-docker

Dziś moim gościem jest Wojciech Lepczyński. Wojtek opowie nam o konteneryzacji i świetnym narzędziu, jakim jest Docker.

Wojtku, dziękuję, że przyjąłeś moje zaproszenie na rozmowę!

Cześć! Dzięki za kolejne zaproszenie. Super, że możemy sobie znowu pogadać i przy okazji dzielić się wiedzą z innymi.

To tradycyjnie zacznijmy od Twojej osoby. Powiedz nam coś więcej o sobie i co łączy Cię z branżą IT.

Witam wszystkich jeszcze raz bardzo serdecznie, nazywam się Wojciech Lepczyński. Aktualnie pracuję jako senior DevOps w GFT. Pomagam tam tworzyć cyfrowy bank w chmurze dla klienta z Azji – taki bank bez fizycznych biur, budynków, który funkcjonuje w 100% w przestrzeni cyfrowej.

W IT pracuję od ponad 12 lat. Byłem już administratorem, trochę architektem i konsultantem chmurowym, teraz głównie jestem DevOpsem. Najbardziej lubię pracować z chmurą, z którą mam bliższą znajomość od jakichś 7 lat.

Można powiedzieć, że jestem pasjonatem chmury, mam kilka certyfikatów potwierdzających wiedzę w różnych chmurach. Aktualnie najwięcej pracuję z chmurą od AWS. Dzielę się swoją wiedzą dotyczącą chmury, prowadzę swój blog i kanał na YouTubie z różnymi tutorialami i poradami. Chyba dlatego zostałem też AWS Community Builderem. Chcę pomagać ludziom zrozumieć chmurę i dzielić się swoją wiedzą. W Polsce jest aktualnie około 30 osób należących do tego programu, a na całym świecie – coś ponad 2000. Nie chcę tym razem wchodzić w szczegóły, czym jest ten program i co oferuje, bo rozmawialiśmy o tym ostatnio. Teraz lepiej porozmawiajmy o kontenerach.

Tak jest. Jeżeli ktoś ma ochotę, to zapraszamy do poprzedniego nagrania.

A dziś będziemy mówić o Dockerze. Zanim o nim, to powiedz: czy jest on bardzo popularny? Czy zaznajomienie się z nim może zwiększyć nasze szanse na otrzymanie pracy?

Krótko mówiąc: TAK. Jeśli chciałbyś, żebym rozwinął swoją odpowiedź…

Oczywiście, że tak.

To według mnie każda nowa umiejętność zwiększa nasze szanse na znalezienie wymarzonej pracy, no i oczywiście zwiększa naszą wartość na rynku pracy. Wspomniałeś o Dockerze, ja może powiedziałbym, że bardziej chodzi o technologię: o kontenery, konteneryzację. Ludzie często używają tych nazw zamiennie – kontenery i Docker to dla niech to samo, choć tak naprawdę rodzajów kontenerów jest więcej, np. LXC czy Podman, ale o tym później. Może zacznijmy od czegoś prostszego.

No dobrze. Nie wiem, czy to będzie prostsze, ale proponuję, żebyśmy zaczęli do tego, czym jest wirtualizacja.

OK, zacznijmy od podstaw. Żeby łatwiej to zrozumieć, można powiedzieć, że wirtualizacja pozwala uruchomić wiele systemów operacyjnych na tym samym fizycznym sprzęcie. Takie systemy operacyjne są uruchamiane na wirtualnych maszynach, które – można powiedzieć – udają fizyczne komputery. Taka wirtualna maszyna jest niezależna od systemu operacyjnego hosta i jest w pełni od niego odizolowana. Możemy sobie wyobrazić, że jeden zwykły laptop pozwala na uruchomienie kilku systemów w tym samym czasie: linuxy, windowsy, co chcesz.

Wiele firm postanowiło stworzyć rozwiązania służące do wirtualizacji sprzętu – na pewno słyszałeś o czymś takim jak VMware, Hyper-V czy VirtualBox. Rozwiązania takie pozwalają na tworzenie tak zwanych wirtualnych maszyn. Właściwie nie tylko na tworzenie, ale także na zarządzanie nimi. Wirtualne maszyny oferują wirtualne środowiska: każda posiada swój system operacyjny i aplikacje. Ale taka wirtualna maszyna nie może wchodzić bezpośrednio w interakcję z fizycznym sprzętem, potrzebuje tak zwanego hypervisora.

Hypervisor nadzoruje pracę wirtualnych maszyn i pozwala je kontrolować, przydzielać im fizyczne zasoby, takie jak pamięć, dysk, procesor itd. No i oczywiście koordynuje komunikację z fizycznym sprzętem.

Mamy dwa rodzaje hypervisorów: takie, które działają bezpośrednio na fizycznym sprzęcie (jak np. Hyper-V), i takie, które są uruchamiane dopiero na systemie operacyjnym (jak np. VirtualBox), i są zwykłą aplikacją, która przydziela zasoby wewnątrz systemu operacyjnego hosta. Oczywiście lepszą wydajność uzyskujemy, nie używając systemu operacyjnego hosta.

Tutaj muszę wspomnieć, że wirtualne maszyny działają niezależnie i nie zakłócają pracy pozostałych maszyn – przynajmniej nie powinny, jeśli dobrze przydzielimy im zasoby. Nie chciałbym wchodzić w zbyt duże szczegóły, bo mogę o tym opowiadać godzinami, a chyba chciałbyś porozmawiać też jeszcze o czymś innym.

Jasne, to może tylko napomknę, że fajnie byłoby też wspomnieć o wadach i zaletach tego rozwiązania. Już mniej więcej o tym powiedziałeś, ale może potem jeszcze do tego wrócimy.

Do czego służy Docker i dzięki czemu ma przewagę nad wirtualizacją?

O, i tutaj doszliśmy do kolejnego punktu. Docker czy też inne kontenery korzystają z systemu operacyjnego komputera, na którym są uruchomione. Dlatego nie trzeba na nich uruchamiać systemu operacyjnego ani ładować bibliotek. Dzięki temu kontenery są znacznie bardziej wydajne, nie zużywają tak dużo procesora i RAM-u, no i oczywiście zajmują mniej miejsca.

Kontener można sobie wyobrazić jako pudełko zawierające aplikację lub jej część wraz z zestawem plików niezbędnych do uruchomienia kodu. Takie skonteneryzowane aplikacje uruchamiają się w ciągu kilku sekund, a na jednym komputerze można uruchomić dużo więcej kontenerów niż maszyn wirtualnych, ponieważ zużywają mniej zasobów.

Chociaż kontenery są przenośne, są związane z systemem operacyjnym hosta. Tutaj należy dodać jedną bardzo ważną rzecz: chodzi o to, że kontenery nie są tak dobrze odizolowane od siebie jak maszyny wirtualne, bo dzielą jądro systemu hosta z innymi kontenerami.

To może dopytałbym o jeszcze jedną rzecz. Wspominaliśmy, że kontenery są uruchamianie na systemie operacyjnym hosta. To jak to wygląda, jeśli chodzi o postawienie sobie np. serwera Apache na Windowsie, który ma tak naprawdę być systemem linuxowym? Czy jest to do zrobienia i jest normalne, czy pod spodem dzieje się mnóstwo rzeczy, żeby tak było?

Pod spodem dzieje się dość dużo. Może do przykładów użycia przejdziemy później, a na razie wspomnę, że kiedyś było bardzo głośno o takiej podatności kontenerów, która pozwalała uzyskać dostęp do roota na głównym systemie. No i chyba wiesz sam, do czego może takie coś doprowadzić.

Zgadza się, zgadza się. Dobrze, to jeszcze wrócimy do tego tematu.

Powiedz nam, czym jest konteneryzacja?

Konteneryzacja, najprościej mówiąc, jest to upychanie aplikacji do kontenerów. Kiedyś programy tworzyło się jako olbrzymie monolity. Teraz jest trend, by aplikacje tworzyć w częściach jako mikroserwisy. Niektórzy próbują też przerabiać takie monolity pod mikroserwisy i muszę przyznać, że różnie im to wychodzi – jednym lepiej, innym gorzej – ale może o tym porozmawiamy za chwilkę. Według mnie należy zrozumieć, dlaczego się to robi, a nie podążać ślepo za trendami.

Konteneryzacja pozwoliła na szybszą ekspansję mikroserwisów. Architektura mikroserwisowa wprowadza inne podejście do tworzenia oprogramowania. Dzieli nam aplikację na niezależne części. Wprowadzanie zmian staje się prostsze i mniej bolesne, ponieważ zajmujemy się tylko jednym modułem, w którym nastąpiła zmiana, a nie całą aplikacją.

Wracając do zamiany monolitu na mikroserwisy: dzisiaj mało kto nie wie, co to jest Netflix, ale nie każdy wie, że ta firma jako jedna z pierwszych zaczęła używać kontenerów. Co ważne, firma chętnie dzieli się wiedzą na temat swojej architektury w internecie. W 2009 mało osób słyszało o

kontenerach, a oni już ich używali. Aktualnie całość aplikacji składa się z ponad 500 skonteneryzowanych mikroserwisów (wiem, bo sprawdzałem przed nagraniem). Jeśli dobrze pamiętam, zaczęli ich używać z powodu rosnącej liczby użytkowników i problemów z wydajnością monolitycznej aplikacji. Przez lata pracowali nad swoim idealnym środowiskiem i dzięki temu miliony ludzi na świecie nie ma problemu z oglądaniem swojego ulubionego serialu nawet w godzinach szczytu.

Nie wiem, czy wiesz, ale wymyślili coś takiego jak Chaos Monkey – narzędzie, które losowo ubija mikroserwisy na produkcji (co ważne: na produkcji), dzięki czemu są dobrze przygotowani na awarie. Ale się rozgadałem. Może wrócimy do głównego tematu, czyli kontenerów.

Jasne, to może zanim do nich wrócimy, to dopowiem: chyba im się nudzi, skoro sami sobie szukają problemów. Oczywiście trochę się śmieję, ale myślę, że programiści albo administratorzy niekoniecznie są zadowoleni, gdy wiedzą, że wszystko jest OK, a pojawił się problem związany z tym, że sami sobie sprawdzają architekturę.

W zasadzie masz rację. Skoro awarie się zdarzają, to można też wywołać je samemu. Pozwala testować to niezawodność i odporność na awarie oraz być lepiej przygotowanym.

OK, jak bym był w zespole, to nie do końca podzielałbym to zdanie.

Wróćmy do kontenerów. Powiedz nam, proszę, czym są obrazy.

OK. Chciałem uporządkować trochę to, co powiedzieliśmy. Mamy trzy podstawowe pojęcia dotyczące kontenerów. Mamy kontenery, o których już mówiliśmy, ale są jeszcze obrazy kontenerów i rejestr kontenerów/obrazów, różnie to ludzie tłumaczą.

Zacznijmy od obrazów. Obraz kontenera to taka statyczna reprezentacja aplikacji. Obrazy są niezmienne, a zmiany obrazu wymagają utworzenia nowego obrazu. Mogę opowiedzieć Ci trochę o tym, jak ja próbowałem to zrozumieć. Ja jestem z pokolenia, gdy jeszcze używało się dyskietek, ale płyty CD, DVD itp. też były. Wyobrażałem sobie więc kiedyś taki obraz kontenera jako płytę CD, na której mogę zapisać program, ale nie mogę nic na niej zmienić. No i potem, żeby uruchomić kontener, musiałem użyć tej „płyty", bo tam były wszystkie informacje. Ale wiesz, ja tak to sobie wyobrażałem, dla mnie to zadziałało. Załapałem, o co w tym chodzi, i potem już było z górki.

W zasadzie podzielam Twoją analogię, bo pamiętam, że miałem płytkę CD ze swoimi, powiedzmy, programami, a gdy chciałem mieć nowszą wersję, to nie mogłem jej dograć do tej samej płytki. Musiałem stworzyć sobie nową płytkę z nowymi wersjami. Niektóre się powielały, a niektóre były nowsze, więc myślę, że to fajna analogia.

Ostatnia rzecz: rejestr obrazów/kontenerów to miejsce, gdzie są przechowywane obrazy. Czyli pobieramy sobie obraz z takiego repozytorium – czy to publicznego, czy naszego prywatnego – i na jego podstawie uruchamiamy kontener. Tutaj, jak mówiłem, obraz się nie zmienia. Wszystkie zmiany, które robisz, tworzysz w kontenerze. Gdy się pomylisz, zawsze możesz utworzyć nowy kontener z dokładnie tego

samego obrazu i zacząć od początku. Oczywiście są sposoby, by zachować swoją pracę na kontenerze, ale o tym może później.

A czym są obrazy bazowe?

OK, no dobra, już mówiliśmy o tym, że kontenery tworzy się za pomocą obrazów. Ale jak tworzy się obrazy? No więc obrazy można tworzyć za pomocą pliku tekstowego, tak zwanego Dockerfile'a.

Dockerfile to taki plik zawierający instrukcje o tym, jak stworzyć obraz. I teraz najlepsze: zawiera on także informację o tym, jakiego obrazu bazowego użyć. Obraz bazowy to taki nasz punkt startowy, od którego zaczynamy instalować biblioteki, programy itp. Obrazem bazowym może być np. Ubuntu w wersji Alpine (do którego będziemy musieli czasem sporo dołożyć), ale może być równie dobrze obraz zawierający WordPressa. Obrazem bazowym może być też obraz, który stworzyłeś pięć minut wcześniej.

Obrazy kontenerów mogą być publicznie dostępne, np. w Docker Hub, i zawierają różne oprogramowanie, np. wspomnianego wcześniej WordPressa albo nginex, samego PHP w konkretnej wersji, czy co tam byś chciał – jest tego naprawdę, naprawdę dużo.

Obrazy kontenerów powstają w różnych wersjach. Wersje obrazów nazywają się tagami. Weźmy sobie na przykład taki obraz WordPressa. Gdy użyjesz polecenia *docker pull wordpress*, to pobierzesz najnowszą wersję obrazu kontenera z WordPressem, oznaczoną jako *latest*. Ale możesz też użyć tagu np. *php8.1* i dzięki temu masz pewność, że pobierasz za każdym razem WordPressa z PHP w wersji 8.1.

Oczywiście tagów jest mnóstwo. Bardzo popularne są tagi zawierające w sobie nazwę *alpine*, co oznacza, że są one bardzo lekkie, zawierają minimum. Na przykład zwykły obraz kontenera z WordPressem zawiera ponad 200 MB, a WordPress z tagiem *alpine* poniżej 100 MB (wiem, bo sprawdziłem przed naszą rozmową). Chodzi mi o to, że zawiera on minimum potrzebnych rzeczy i dzięki temu zajmuje mało miejsca oraz szybko się uruchamia. Ludzie zazwyczaj biorą taki obraz, często właśnie z tagiem *alpine*, i instalują na nim rzeczy, których potrzebują, konfigurują go pod siebie – właśnie dlatego, żeby zajmował mniej miejsca i szybciej się uruchamiał.

Ja także staram się używać najlżejszej wersji obrazu. Gdy pobiorę obraz z oficjalnego repozytorium, to tworzę swój obraz bazowy zawierający to, czego zawsze używam, i zapisuję go w prywatnym repozytorium. Potem pobieram obraz ze swojego prywatnego repozytorium i na podstawie tego obrazu tworzę obrazy, których potrzebuję w różnych zadaniach, i one zawierają dodatkowe narzędzia i konfiguracje pod konkretne zadania.

Gdy chcesz się bawić w konteneryzację i wirtualizację tak na poważnie, to zalecam automatyzację tworzenia takich obrazów kontenerów, np. za pomocą Packera, gdzie za pomocą kodu możemy tworzyć piękne obrazy kontenerów, ale nie tylko. Packer umożliwia także budowanie obrazów maszyn wirtualnych. Jest to w ogóle bardzo uniwersalne i fajne narzędzie. Oczywiście podobnych narzędzi jest więcej, ja podaję tutaj tylko przykład. Coś chciałbyś dodać, polecić?

Myślę, że nie, że na razie wystarczy. Chyba i tak mamy tu sporo informacji jak na początek.

Czy możesz powiedzieć, po co łączy się ze sobą kontenery, i wyjaśnić działanie tego na jakimś front-endowym przykładzie?

OK, spróbuję. Ja mam blog na WordPressie, więc może opowiem o takim prostym przykładzie, jak można uruchomić WordPressa ze zwykłą stroną internetową na Dockerze w domu. Kiedyś nawet sam się tak bawiłem: zrobiłem sobie takiego lokalnego WordPressa, do którego miałem dostęp tylko z sieci domowej, więc to nie będzie czysta teoria, tylko coś z praktyki. Może być?

No jasne, prosimy.

Ja lubię porządek. Tworząc sobie WordPressa na kontenerach, nie musisz mieć zainstalowanego na swoim komputerze ani Apache, ani PHP, ani MySQL, ani nic innego. Kiedy jakiś projekt staje się zbędny, to po prostu usuwasz kontenery, współdzielone pliki i zapominasz, że miałeś coś takiego na komputerze.

Już mówiliśmy o tym, że kontenery mogą zawierać aplikacje i im mniej usług mają uruchomionych, tym lepiej. Dlatego zazwyczaj uruchamia się kilka kontenerów. Takie kontenery, gdy je odpowiednio skonfigurujesz, mogą się bez problemu ze sobą komunikować. W tym przykładzie przyda się kontener z WordPressem i jeszcze jeden z bazą danych, np. MySQL.

Tak więc jeden kontener ma zainstalowanego i skonfigurowanego WordPressa, a drugi ma bazę danych potrzebną do tego WordPressa. Klienci łączą się z pierwszym kontenerem, a on może łączyć się z bazą danych na innym kontenerze. Przy bazach danych bym uważał, bo to

trzeba dobrze skonfigurować, ale oczywiście można tego używać. Ja tylko chciałem podać przykład, który łatwo sobie wyobrazić.

To może dopytam, bo zastanawiam się, jak wygląda temat wydajności takiego rozwiązania. Czy to, że jeden kontener łączy się z drugim, nie powoduje problemów z wydajnością? Czy to jest bardzo widoczne, czy niezauważalne?

W zasadzie zależy to od tego, jak szybki masz komputer i jakie zasoby przydzieliłeś kontenerom. Głównie od tego. Gdy kontenery są na różnych maszynach, to dochodzi jeszcze kwestia szybkości sieci.

Możesz dokładać sobie kolejne komponenty, np. jakąś kolejkę albo serwer mailowy w kolejnym kontenerze itd. Jeśli chciałbyś zajmować się większą liczbą stron internetowych, to możesz postawić więcej kontenerów z WordPressem i podłączyć je do tego samego kontenera z bazą danych. Ma to taki plus, że każdy WordPress może być uruchomiony z innego obrazu i mieć np. inną wersję PHP. Fajne jest to, że awaria kontenera z jednym WordPressem nie ma wpływu na inny, co jest super. Kontener może się zrestartować i za chwilę znowu działać poprawnie.

Ja krótko po tym, jak poznałem Dockera, zacząłem bawić się Kubernetesem – i to dopiero była zabawa. Kubernetes to taki jakby wyższy poziom służący do zarządzania kontenerami.

To może powiedz nam coś więcej na temat tego rozwiązania. Czy bardziej chodzi o to, że jest ono w stanie automatycznie skalować naszą aplikację, czy mówimy tu o zarządzaniu na całkiem innym poziomie?

I tutaj jest zaleta kontenerów, bo mogą one np. automatycznie się skalować, czyli jeśli nasza apka stanie się bardzo popularna i nagle zwiększy się nam ruch, to nie padnie nam strona, bo gdy kontener będzie przeciążony, to automatycznie może utworzyć się następny i następny i kolejne zapytania będą wysyłane do wielu kontenerów. Gdy obciążenie się zmniejszy, to liczba kontenerów się zmniejszy. Tylko to już jest taki kolejny poziom. Według mnie wpierw powinno się zrozumieć, jak działają kontenery, żeby potem móc nimi sprawnie zarządzać, bo to nie jest takie proste.

Do czego może się nam przydać Docker, gdy dopiero uczymy się programowania?

Docker pozwoli Ci zająć się tym, na czym Ci zależy. Gdy chcesz się uczyć programować, to skupiasz się na nauce programowania, nie musisz tracić czasu na instalację bazy danych i innych rzeczy. Nie musisz zajmować się instalacją, konfiguracją itd., bo obrazy Dockera mogą mieć już to wszystko zrobione. Po prostu uruchamiasz i działa. Skupiasz się na nauce programowania czy czego tam chcesz się uczyć. Wybierasz kontener, który ma zainstalowane to, czego potrzebujesz, w wersji, jakiej potrzebujesz – i tyle. Potrzebujesz nowszą albo starszą wersję, to nie bawisz się w instalację, aktualizację, tylko wybierasz inny obraz i gotowe.

To może dodam jedną rzecz: przy naude front endu pewnie nie potrzebujecie nic, bo wystarczy przeglądarka, wystarczy edytor i w zasadzie już możecie pisać HTML-a, CSS-a, JavaScript i Wam działa. Ale jeżeli macie język back-endowy np. PHP, to musicie

zainstalować przynajmniej PHP, Apache, jakąś bazę, żeby w ogóle móc zacząć działać. Przy front endzie jest trochę łatwiej, a z back endem trzeba się trochę pobawić, żeby móc wystartować. I tyle, możesz kontynuować. Wybacz, że Ci przerwałem.

Nie, spoko, OK.

Fajne jest to, że pomaga zachować porządek na swoim komputerze – nie musisz instalować tony rzeczy i co chwila czegoś dodawać, usuwać, zmieniać wersji itd., bo masz to w kontenerze. Nie będziesz miał śmieci na komputerze i nie będziesz musiał się zastanawiać, jaką to właściwie wersję masz aktualnie zainstalowaną.

Oczywiście czasem lepszym wyborem będą maszyny wirtualne – w zależności od tego, czego potrzebujesz. Tylko pamiętaj, że przygotowanie i konfiguracja takiej maszyny wirtualnej wymaga nieco więcej pracy. Gdy popełnisz błąd i będziesz chciał zacząć od początku, to zajmie to więcej czasu. A co, jeśli będziesz musiał uruchomić kilka wirtualnych maszyn w tym samym czasie? No to Twój sprzęt raczej nie będzie zadowolony. Kontenery mają tę przewagę, że są lżejsze i jak wspomnieliśmy wcześniej, nie zużywają tylu zasobów.

Super jest jeszcze to, że gdy bawisz się na takim kontenerze i coś zepsujesz, to się tym nie przejmujesz. Po prostu go restartujesz i zaczynasz od początku. Często spotykałem ludzi, którzy instalowali u siebie na lapku soft i próbowali się czegoś nauczyć, tylko często kończyło się to tak, że po tych swoich próbach musieli reinstalować system, bo coś poszło nie tak. Potem wpadli na pomysł że zrobią sobie backup, zanim zaczną pracować, a gdy coś pójdzie nie tak, to do niego wrócą – ale to zazwyczaj trwa sporo czasu. Dużo lepszym rozwiązaniem

są kontenery. Gdy coś nie wyjdzie, restartujesz kontener i zaczynasz od początku, a gdy wszystko działa, to możesz stworzyć nowy obraz i zaczynasz z miejsca, w którym wszystko działa.

To może znów się wtrącę. To już były zaszłe czasy, ale miałem wersję PHP 4-coś, 5.0, 5.6 i potem sam już się gubiłem, co gdzie jest, czy działa, czy nie działa, dlaczego nie działa, tu porty jakieś poblokowane. Aj… Na pewno dużo by to ułatwiło. Tylko tyle, kontynuuj.

Tylko pamiętaj jeszcze, żeby dobrze nazwać te obrazy, żebyś potem wiedział, co w którym jest.

Tak jest, tak jest.

Tutaj jeszcze chciałbym dodać, że kontenery mogą także mieć dostęp do plików z naszego komputera. Dzięki temu po restarcie kontenera nasza praca nie zniknie. Możemy współdzielić pliki, które będą kodem aplikacji internetowej. Aplikacje możemy uruchomić za pomocą np. serwera HTTP, oczywiście w kontenerze, i normalnie jej używać. To samo dotyczy bazy danych: nie musimy upychać jej do kontenera. Pliki bazy danych mogą być na naszym dysku i możemy je współdzielić w kontenerze za pomocą specjalnych woluminów jak w poprzednim przykładzie.

Czy rozpoczęcie pracy z Dockerem jest trudne? Czego potrzebujemy?

Tak naprawdę, jak chcemy się uczyć, to Dockera możemy pobrać za darmo. Nie ma on dużych wymagań sprzętowych. Ale według mnie trzeba to zrozumieć, nie iść ślepo w nieznane bez zrozumienia podstawowych pojęć. Jest tutaj pewien próg wejścia, ale jest on, powiedziałbym, niski. Trzeba zrozumieć, czym jest ta technologia, czym jest kontener, czym jest obraz itd. Gdy wiesz, na czym to polega, jak to działa, to już potem z górki. Warto poznać tę technologię, zwłaszcza gdy jesteś na początku drogi, bo możesz zaoszczędzić dużo czasu. Chciałbyś tutaj coś dodać?

Myślę, że chciałbym się Ciebie dopytać, co trzeba doczytać po naszej rozmowie, aby móc zacząć z Dockerem.

Najlepszą odpowiedzią jest chyba „to zależy”. To jest bardzo indywidualna kwestia. My tutaj sporo rzeczy wyjaśniamy, ale zależy, jaki masz backgroud, co już umiesz i czy rozumiesz, jak to działa. Jedno to znać teorię, a drugie to znać mechanizm działania.

Na pewno zalecałbym poznanie podstawowych poleceń. Dobrym pomysłem będzie odwiedzenie strony Docker Hub, a właściwie hub.docker.com z publicznymi repozytoriami. Znajdziesz tam dużo obrazów dockerowych i instrukcje, jak ich używać. Dużo obrazów ma właśnie bardzo fajny opis tego, co zawierają i instrukcję mówiącą, co zrobić krok po kroku, żeby uruchomić konkretny obraz.

Na jakich systemach operacyjnych możemy wykorzystywać Dockera?

Tak naprawdę do nauki możesz mieć dowolny komputer, nie ma znaczenia, czy jest to Linux, Mac czy Windows. Docker nie jest zbyt wymagający. Oczywiście zalecane parametry trzeba spełniać, ale nie są one jakieś wygórowane. Możesz na zwykłym Windowsie pracować z Dockerem, utworzyć obraz, otworzyć go na innym komputerze i pracować dalej. W sieci jest pełno tutoriali, które wystarczą, by zacząć.

OK, to wracając do pytania z samego początku: jeżeli mam Windowsa i chcę na nim uruchomić serwer, na którym uruchomię PHP, to po prostu jest to PHP, ale na Windowsie. A jeżeli chciałbym uruchomić PHP na Linuksie, to bardziej wchodzi w grę wirtualizacja, a nie konteneryzacja, tak?

W sumie tak. To znaczy w Dockerze tam pod spodem jest cały czas Linux.

Aha, OK. No dobrze.

Nie wiem, na ile jest to informacja na teraz, a na ile jest to informacja zaszła, ale czy potwierdzasz, że na systemie MacOS Docker działa zauważalnie gorzej niż na innych systemach? Można sobie z tym jakoś poradzić, jeśli tak faktycznie jest?

Docker jest dostępny, jak już wspominałem, na Linuksie, Windowsie i macOS, ale na Linuksie działa najlepiej. Windows całkiem dobrze sobie radzi z WSL 2, który według mnie sam w sobie świetnie działa i dzięki niemu mogłem zrezygnować tylko z Linuxa. Wolałem mieć Windowsa razem z Linuxem. Wiem, że nie każdy się ze mną zgodzi, ale każdy może pracować na tym, co lubi.

Kiedyś pracowałem tylko na Windowsie, potem tylko na Linuksie, teraz pracuję głównie na maczku, ale mam też inny komputer, na którym mam Windowsa z WSL 2. Według mnie jest to genialne rozwiązanie, które bardzo ułatwia mi pracę, chociaż aktualizacje Windowsa potrafią być strasznie uciążliwe.

To może powiedz nam coś więcej na temat WSL 2.

Tak, racja – chodziło mi dokładniej o Windows Subsystem for Linux w wersji 2. Najprościej mówiąc, to taki Linux w Windowsie, który jest bardzo dobrze z nim zintegrowany. Pozwala na uruchamianie narzędzi i poleceń Linuxa na komputerze z systemem Windows.

Brzmi dobrze. I mówisz, że tak samo dobrze działa.

Według mnie tak. Wiesz, Linux jest super, tylko gdy zarządzałem serwerami głównie windowsowymi z komputera z Linuxem, to nie było zbyt fajnie. Do zarządzania windowsowymi serwerami bardziej się sprawdzał u mnie Windows. Mając Windowsa z WSL, mogłem używać Linuxa, kiedy chciałem, i bez problemu miałem dostęp do windowsowych aplikacji, jak RSAT (Remote Server Administration Tools) czy SQL Server Management Studio. Oczywiście są zamienniki pod inne systemy, ale lepiej mi się pracowało na tych windowsowych aplikacjach.

Podsumowując: jeśli chcesz się uczyć, to nie ma jakiegoś znaczenia, jakiego sprzętu używasz. Według mnie powinno się używać tego, co jest dostępne. Jeśli chcesz, żeby działało najlepiej, to używaj Linuxa, a tak serio, to jakoś się nad tym nie zastanawiałem. Gdy miałem problem, to szukałem odpowiedzi w Google, rozwiązywałem go i tyle.

Nie używam Dockera na macOS, więc nie mam żadnych problemów i niestety też żadnych dobrych rad. Jeśli będę musiał używać, to pewnie sam będę ich szukał. Ale faktycznie słyszałem, jak kiedyś znajomi narzekali głównie ze względu na opóźnienia w synchronizacji plików. Dochodził jeszcze problem cross-kompilacji x86 na ARM.

Możesz potwierdzić zaprzeczyć albo coś doradzić ? Masz jakieś doświadczenie w tym zakresie?

No ja niestety też nie mam, ale może to była jakaś zaszłość, więc po prostu spróbujcie, zobaczcie, czy działa. Może wszystko będzie OK, więc nie ma co się martwić na zapas. Może już ten problem rozwiązano. Sam też więcej nie jestem w stanie powiedzieć.

Przed jakimi innymi błędami i problemami mógłbyś przestrzec osoby, które zaczynają przygodę z Dockerem?

O, jest tego całkiem sporo. Na przykład przed używaniem tagu *latest*, o którym już wcześniej mówiliśmy. Tagi określają wersję danego obrazu. Gdy tworzymy swój obraz, to powinniśmy używać konkretnego tagu obrazu bazowego, a nie *latest*, który pobiera nam najnowszy dostępny obraz. Nie zrozum mnie źle. Używanie najnowszych obrazów nie jest niczym złym, tylko chodzi o kontrolę tego. W przeciwnym razie kiedyś nagle może nam coś przestać działać. Tak więc powinniśmy aktualizować nasze obrazy, ale to my powinniśmy wybierać konkretną wersję i czas.

Często ludzie pytają się mnie, jak często powinni aktualizować swoje kontenery. No i ja najczęściej odpowiadam, że tak często, jak się da,

ponieważ wciąż są odkrywane nowe podatności i na bieżąco są łatane w nowych wersjach kontenerów. Dlatego, jeśli chcemy dbać o bezpieczeństwo, to powinniśmy zwracać uwagę na nowsze wersje obrazów kontenerów.

Wiele firm automatyzuje aktualizacje na środowiskach developerskich albo dodaje task w każdym sprincie, by ktoś z zespołu zajął się aktualizacją obrazu kontenera czy maszyny wirtualnej. Tutaj także trzeba pamiętać, aby aktualizować rzeczy, które dodajemy do naszego obrazu. Często prywatne repozytoria obrazów umożliwiają analizę obrazów kontenerów i automatycznie powiadamiają administratorów, gdy wystąpią jakieś podatności.

Kolejna porada to: warto szukać jak najmniejszego obrazu, by zajmował on jak najmniej miejsca i dzięki temu szybciej się uruchamiał. Na pewno znasz Gita i wiesz, do czego służy plik *.gitignore*. W Dockerze jest plik, który nazywa się *.dockerignore* i pozwala na to samo, czyli pomijanie plików, których nie chcemy dodać do obrazu.

Jeśli już jesteśmy przy pomijaniu plików, to oczywiście nie powinno się dodawać w konfiguracji żadnych kluczy – inaczej każdy, kto będzie miał dostęp do obrazu, będzie mógł je odczytać.

To może od razu powiedz nam, jak sobie radzić z zarządzaniem kluczami w inny sposób.

Można na przykład przez HashiCorp Vault albo w chmurze AWS masz coś takiego jak KMS (Key Management Service), zresztą w innych chmurach też jest coś takiego. Jak już jesteśmy przy chmurze: możemy nadawać różne role, które mają uprawnienia do innych zasobów i

zrezygnować z kluczy. Jak już musisz, to możesz dodać klucze po utworzeniu obrazu. Jeśli ktoś ukradnie Ci obraz albo uzyska do niego nieuprawniony dostęp, to kluczy tam nie znajdzie.

Działanie kontenerów także powinno się monitorować, tak jak to robimy ze zwykłymi serwerami, bo jednak chcielibyśmy wiedzieć, czy nasza aplikacja działa prawidłowo, czy nie. No przynajmniej ja bym chciał.

Mówiłem już dzisiaj o tym, że repozytoria kontenerów mają często możliwość skanowania obrazów, ale sam Docker też ma taką możliwość. Możemy użyć w tym celu polecenia *docker scan*, by znaleźć luki.

Powinniśmy pamiętać o tym, by otwierać tylko potrzebne porty w kontenerach. Jest to zwykła dobra praktyka poprawiająca bezpieczeństwo, która jest znana już z pracy z serwerami.

Powinniśmy pamiętać o tworzeniu Healthchecków. Dzięki temu, w przypadku gdy Docker otrzyma informacje o złym stanie kontenera, powinien utworzyć nowy z prawidłowym statusem.

Lepiej nie przechowywać danych w kontenerach. Powinno się to robić na podpinanych volumach. Kontenery mogą być w każdej chwili zrestartowane, zatrzymane albo zniszczone i nowsza wersja aplikacji powinna mieć możliwość uruchomienia się bez jakiejkolwiek utraty danych.

Według mnie najlepiej obrazy kontenerów tworzyć z Dockerfile'a, a nie za pomocą *docker commit* – czyli z działającego kontenera. Tworzenie obrazu z Dockerfile'a umożliwia proste śledzenie zmian i gwarantuje powtarzalność, czego nie zagwarantuje nam *docker commit*.

W jednym kontenerze powinien być uruchomiony jeden proces. Gdy uruchamiasz ich więcej, to trudniej się nimi zarządza i możesz mieć przez to więcej problemów – na przykład jeśli jeden proces przestanie działać, to kontener nie będzie już działał prawidłowo, ale tego nie wykryje, bo główny proces wciąż działa prawidłowo i kontener nie wie, że powinien się zrestartować.

No i ostatnie, ale bardzo ważne: pamiętaj, żeby uruchamiać swoje kontenery jako użytkownik bez uprawnień administratora, czyli nie jako root.

Wow. Tyle tych dobrych rad, że trudno zapamiętać, a co dopiero wdrożyć, więc na pewno słuchacze będą mieli co robić, nad czym się zastanawiać.

Na szczęście nagrywamy i można przesłuchać jeszcze raz. Na pewno rad byłoby jeszcze więcej, tylko musielibyśmy mieć dużo więcej czasu.

To jeżeli słuchacze będą chętni, to nagramy jakieś uzupełnienie. Jeśli ktoś ma ochotę, to dawajcie znać, będziemy o tym myśleć.

W jakich sytuacjach nie zalecałbyś wykorzystania Dockera?

Przede wszystkim wtedy, gdy pracujesz nad aplikacją desktopową, a nie serwerową. Nie zalecałbym także wtedy, gdy Twoja aplikacja jest mała i prosta – wtedy używanie Dockera czy Kubernetesa do hostowania aplikacji to trochę jak strzelanie z armaty do muchy. Nie wiem, co o tym myślisz.

Myślę, że się zgodzę. Jestem minimalistą, więc raczej staram się robić jak najmniej, ale żeby było to jakościowo dobre. Nie lubię też dodawać sobie niepotrzebnej pracy, więc dopiero jak muszę, to wtedy robię coś więcej.

Słusznie, słusznie.

Jeszcze chciałbym dodać chyba najważniejszą rzecz. Nie zalecałbym używania produkcyjnego kontenerów osobom, które ich nie znają albo nie mają dużej wiedzy. Wejście do świata kontenerów nie wymaga aż tak dużej wiedzy, ale używanie kontenerów na produkcji wymaga według mnie bardzo, bardzo dużej wiedzy – głównie dotyczącej bezpieczeństwa i tego, jak kontenery mogą się zachowywać.

Kiedy kod Twojej aplikacji jest monolitem i działa dobrze, to czy na pewno musisz usilnie pchać się w kontenery? Nie każdy potrzebuje kontenerów. Według mnie warto upraszczać rzeczy (tak jak też właśnie wspomniałeś). Widziałem firmy, które przerabiały swoje aplikacje, żeby można było ich używać w kontenerach. Straciły dużo czasu i dużo pieniędzy, by stwierdzić, że to nie jest technologia dla nich, i wróciły do tego, co miały wcześniej i co się sprawdzało.

To znaczy ja nie zabraniam nikomu używać kontenerów, tylko po prostu trzeba to robić z głową. Gdy szukasz prostej, łatwo zarządzanej technologii, to Docker nie jest dobrym rozwiązaniem – jak mówiłem, potrzebna jest duża wiedza i doświadczenie, by robić to dobrze. Takie jest moje zdanie. Jakie jest Twoje?

No ja tylko mogę podpisać się pod tym wszystkim, co powiedziałeś. Ty tu jesteś specjalistą, więc w stu procentach się z Tobą zgodzę.

Jaką książkę polecisz osobie, która chce zgłębić działanie konteneryzacji?

Ja uczyłem się z darmowych tutoriali znalezionych w necie. Teraz pewnie bym się zastanowił nad czymś płatnym, gdzie wiedza jest uporządkowana i pozwoli łatwiej zacząć i szybciej zrozumieć temat, bo nie ukrywając, jest tutaj pewien próg wejścia. Żeby używać kontenerów, powinno się rozumieć, jak działają.

Dobrym pomysłem jest udział w kursach i szkoleniach, gdzie masz możliwość zadawania pytań instruktorom. Fajnie się sprawdza, gdy masz na początku pokazane krok po kroku, co zrobić, i co ważne: dlaczego. Potem na podstawie tego masz zrobić coś podobnego, ale już bez podpowiedzi. A później to praktyka praktyka i praktyka.

No dobrze, to może w takim razie Ty stworzysz jakiś kurs, na którym można by się dowiedzieć tego wszystkiego krok po kroku. Nie wiem, czy myślisz o czymś, czy na razie nie masz czasu na takie rzeczy.

Może, może kiedyś coś takiego powstanie. Kiedyś myślałem o zrobieniu jakiegoś kursu, ale na razie są to tylko może jakieś dalekosiężne plany.

Czyli rozumiem, że brak czasu.

No, też.

To może tak: zapytam się Ciebie teraz, gdzie można Cię znaleźć w sieci, a wszyscy ci, którzy chcieliby mieć taki kurs, napiszą do Ciebie i Cię zmotywują, żebyś to zrobił.

W sieci łatwo mnie znaleźć. Znajdziecie mnie w social mediach, na Linkedinie – po wpisaniu „Wojciech Lepczyński" powinienem być na jednej z pierwszych pozycji.

Mogę zaprosić na mój kanał na YouTubie – tam nazwa kanału to, uwaga: Wojciech Lepczynski, bez polskich znaków. Jeśli kogoś interesuje chmura, to zapraszam. Wrzucam tam porady i tutoriale dotyczące chmury. Aktualnie skupiam się bardziej na AWS, ale o Azure też coś kiedyś dodawałem.

No i oczywiście zachęcam do odwiedzania mojego bloga głównie o chmurach, ale kilka porad o Kubernetesie i Terraformie też powinniście tam znaleźć. I tu Was zaskoczę: wystarczy wpisać tylko lepczynski.it. Blog, jak coś, jest po polsku i angielsku.

To ponownie: zachęcamy. Jeżeli chcecie kurs, to piszcie do Wojtka, żeby go zmotywować.

Może kiedyś powstanie.

Bardzo dziękuję Ci, Wojtku, za naszą dzisiejszą rozmowę. Kolejna dawka mnóstwa ciekawych informacji, więc super, że postanowiłeś się nimi podzielić.

Dzięki, dzięki za zaproszenie, za rozmowę. Znowu fajnie mi się z Tobą gadało, a czas szybko zleciał. Co prawda nie sądziłem, że tak szybko ponownie mnie zaprosisz, więc chyba nie zrobiłem najgorszego

pierwszego wrażenia i nie tylko mi dobrze się gadało w poprzednim podcaście.

Jeszcze raz dzięki za zaproszenie, pozdrawiam wszystkich słuchaczy i zachęcam by zainteresować się kontenerami i chmurami – to może pozytywnie zmienić Wasze życie. Dzięki, trzymajcie się, cześć.

Cześć.