Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
http://nv.nltu.edu.ua
https://doi.org/10.15421/40270920
Article received 22.11.2017 p.
Article accepted 28.11.2017 p.
УДК 004.42

ISSN 1994-7836 (print)
ISSN 2519-2477 (online)

@ Correspondence author
O. A. Andrushko
info@symfoni-esm.com

О. А. Андрушко[1], Ю. О. Борзов[2], І. О. Малець[2], О. В. Придатко[2]

[1] *Symfoni ESM – a Fujitsu Company, м. Осло, Норвегія*
[2] *Львівський державний університет безпеки життєдіяльності, м. Львів, Україна*

АНАЛІЗ ПРОЦЕСІВ ВИКОРИСТАННЯ DOCKER ДЛЯ ПОБУДОВИ МІКРОСЕРВІСІВ

Висвітлено складність командної роботи в процесі розробки програмних продуктів за умови віддаленого доступу. Доповнено знання про технології контейнеризації щодо особливостей використання Docker для побудови мікросервісів. Визначено особливості використання та основні переваги Docker-технологій у процесі побудови мікросервісів. Висвітлено переваги контейнерування над процесом віртуалізації машини як інструменту оптимізації використання обчислювальних ресурсів. Описано шляхи уникнення конфліктів між різними середовищами розробки шляхом використання Docker, за рахунок гарантії того, що мікросервіси додатків працюватимуть у відокремлених від операційної системи середовищах. Описано принцип оптимізації середовищ, із необхідними бібліотеками та мовами, для різних учасників проектної команди на основі Docker Hub. Визначено основні переваги застосування процесу контейнеризації над використанням віртуальних машин у межах одного серверу. Розглянуто архітектуру мікросервісу для кращого уявлення про особливості застосування Docker у процесі розроблення програм. Розглянуто архітектуру Docker та описано основні компоненти. Описано Docker-системи оркестрування контейнерами для динамічного управління додатками.

Ключові слова: контейнеризація; розроблення мікросервісів; віртуалізація; Docker-технологія.

Вступ. Розроблення серверних веб-додатків значно змінилося після дебюту Docker'a. Поява Docker спростила створення масштабованих та керованих додатків, побудованих за допомогою мікросервісів. Для кращого розуміння, що таке мікросервіс і як Docker допомагає їх реалізовувати, розглянемо приклад. Уявімо, що у команді веб-розробників працює три програмісти, які для створення одного й того ж додатку використовують різні операційні системи: macOS, Windows та Debian. Передусім, кожне з перелічених середовищ потребує окремих унікальних налаштувань. Окрім цього, розробники повинні встановити та налаштувати різні бібліотеки для своїх мов програмування. Неминучим є той факт, що бібліотеки та мови програмування конфліктуватимуть між різними середовищами. Додаймо ще сервери для тестування та виконання, після чого стає зрозумілим наскільки складно забезпечити однакові умови для всіх середовищ.

Аналіз останніх досліджень та публікацій. Починаючи з 2013 р., коли розробники Docker Inc. презентували першу версію інструменту для віртуалізації рівня операційної системи, багато з них почали адаптовувати Docker під власні потреби. Разом із цим проведено наукові дослідження щодо визначення особливостей ефективного використання Docker. У науковій праці (Musaev, Gazul & Anantchenko, 2014) розглянуто лабораторії віртуалізації, що дають змогу створювати віртуальні лабораторії на платформі Docker. Також, в оглянутій праці наведено порівняння архітектури віртуальних IT-сервісів на основі Docker та віртуальних машин, що дало підстави до розроблення рекомендацій щодо створення віртуальних лабораторій на основі контейнерів під загальним управління Docker.

У роботі (Belonozhko et al., 2016) проведено аналітичний огляд наявних хмарних сервісів, а також апаратних, програмних і організаційних особливостей їх реалізації. Значну увагу приділено огляду Docker-інструментів, що дає змогу усвідомити його місце в технологіях створення хмарних сервісів.

Стаття (Korolov, Gavrikov & Smirnov, 2017) спрямована на огляд та висвітлення переваг процесу віртуалізації, а також його впливу на використання обчислювальної потужності апаратної частини. Особливу увагу приділено огляду переваг нової технології віртуалізації-контейнеризації.

З проведеного аналізу зрозуміло, що питання особливостей використання технології контейнеризації Docker розкрито досить широко, проте все ж таки актуаль-

Інформація про авторів:

Адрушко Олег Андрійович, магістр, технічний консультант. **Email:** info@symfoni-esm.com

Борзов Юрій Олексійович, канд. техн. наук, доцент кафедри управління проектами, інформаційних технологій та телекомунікацій. **Email:** borzovuo@ukr.net

Малець Ігор Остапович, канд. техн. наук, доцент, професор кафедри управління проектами, інформаційних технологій та телекомунікацій. **Email:** igor.malets@gmail.com

Придатко Олександр Володимирович, канд. техн. наук, заступник начальника кафедри управління проектами, інформаційних технологій та телекомунікацій. **Email:** o_prydatko@ukr.net

Цитування за ДСТУ: Андрушко О. А., Борзов Ю. О., Малець І. О., Придатко О. В. Аналіз процесів використання docker для побудови мікросервісів. Науковий вісник НЛТУ України. 2017. Вип. 27(9). С. 95–98.

Citation APA: Andrushko, O. A., Borzov, Yu. O., Malets, I. O., & Prydatko, O. V. (2017). Analysis of the Use of Docker to Build Microservices. *Scientific Bulletin of UNFU, 27*(9), 95–98. https://doi.org/10.15421/40270920

ними залишаються не розглянуті питання. Мета нашої роботи полягає у доповненні знань про технології контейнеризації в світлі особливостей використання Docker для побудови мікросервісів.

Мета дослідження. Зважаючи на окреслену проблему та беручи до уваги результати проведеного аналізу наукових праць, визначити особливості, переваги та недоліки використання Docker-технологій у процесі побудови мікросервісів.

Виклад основного матеріалу дослідження. Описана вище проблема є актуальною за умови побудови монолітних програм та стає ще нагальнішою, якщо слідувати сучасній тенденції розроблення додатків на базі мікросервісів.

Оскільки мікросервіси є автономними, незалежними прикладними блоками, кожен з яких виконує лише одну конкретну бізнес-функцію, їх можна розглядати як невеликі самостійні програми. Що станеться, коли створити десяток мікросервісів для одного додатку? І що, коли необхідно побудувати кілька мікросервісів з різними стеками технологій? Дуже швидко команда програмістів зіткнеться з багатьма перешкодами, оскільки розробники повинні керувати ще більшою кількістю середовищ, ніж у традиційному монолітному додатку.

Однак, існує рішення – використання мікросервісів та контейнерів для інкапсуляції кожного такого сервісу і Docker допомагає керувати цими контейнерами. Docker – це інструмент контейнерування, побудований на основі контейнерів Linux, для забезпечення простого керування контейнерними додатками. Далі розглянемо переваги Docker для розробки мікросервісів на основі аналізу процесів його використання.

Контейнерування, як альтернатива віртуалізації, завжди могло вдосконалити спосіб створення додатків, а Docker, як інструмент контейнерування, часто порівнюється з віртуальними машинами.

Як відомо, віртуальні машини (VM) створено для оптимізації використання обчислювальних ресурсів. За умови запуску кількох віртуальних машин на одному сервері та розгортання кожного екземпляра програми на окремій віртуальній машині, забезпечуватиметься стабільне середовище для кожного екземпляра. Однак, на жаль, масштабуючи програму, швидко виникне проблема, пов'язана з продуктивністю, оскільки VM споживають багато ресурсів.

Наведена схема (рис. 1) демонструє місце hypervisor у системі керування кількома операційними (гостьовими) системами на одному сервері, ізолюючи їх одна від одної. У найпростіших випадках, hypervisor допомагає зменшити ресурси, необхідні для запуску декількох операційних систем.

Оскільки мікросервіси подібні до невеликих додатків, потрібно розгортати мікроконтролери для власних віртуальних екземплярів (щоб забезпечити дискретне середовище). І варто погодитись з тим фактом, що присвоєння всієї віртуальної машини для розгортання лише невеликої частини додатку – не найефективніший варіант. Однак за допомогою Docker можна зменшити втрати продуктивності та розгортати тисячі мікросервісів на одному сервері, оскільки контейнер Docker потребує набагато менше обчислювальних ресурсів (Bylina, 2016), ніж віртуальні машини.

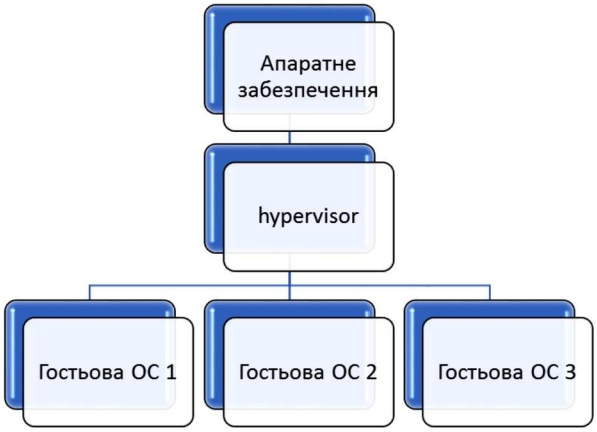

Рис. 1. Модель керування роботою VM з допомогою hypervisor

До цього моменту наш огляд зосереджувався на керуванні середовищами для одного додатка. Проте далі розглянемо випадок розроблення двох різних проектів або тестування двох різних версій того самого додатку. Конфлікти між версіями додатків або бібліотеками у цьому разі неминучі, і підтримка двох різних середовищ на одній VM є складним завданням. Тут доречно задати питання: "Чим Docker кращий за віртуальні машини?". На відміну від роботи віртуальних машин, Docker не потребує постійного створення нових середовищ, намагаючись уникнути конфліктів. Docker гарантує, що мікросервіси додатків працюватимуть у власних середовищах, які повністю відокремлені від операційної системи.

Завдяки Docker, перед командою розробників не виникає необхідності примусово працювати в ідентичних середовищах. Натомість, один розробник може створити стабільне середовище з усіма необхідними бібліотеками та мовами, після чого зберегти це налаштування в Docker Hub. Іншим розробникам залишається лише завантажити налаштування. Завдяки цій особливості Docker може заощадити багато часу.

Зважаючи на описані особливості використання Docker, можна узагальнити основні переваги використання цієї технології:

- зменшення часу запуску (контейнер Docker запускається за лічені секунди, оскільки контейнер – це лише процес операційної системи. Запуск віртуальної машини з повною ОС може тривати кілька хвилин);
- швидке розгортання (не має необхідності створювати нове середовище. Членам команди веб-розробки потрібно лише завантажити образ Docker, щоб запустити його на іншому сервері);
- зручне керування та масштабування контейнерів (знищення та запуск контейнерів швидше ніж знищення і запуск віртуальної машини);
- використання обчислювальних ресурсів (можливість запуску більшої кількості контейнерів, ніж віртуальних машин на одному сервері);
- підтримка різних операційних систем (Docker працює під Windows, Mac, Debian та інші ОС).

Для кращого уявлення про особливості застосування Docker у процесі розроблення програм, що базуються на мікросервісах, варто розглянути їх архітектуру. Почнемо з огляду простого мікросервісу.

Рис. 2. Докеризований додаток (мікросервіс)

Програма (мікросервіс), зображена на рис. 2, складається лише із трьох служб і дає змогу реалізувати блог для веб-сайту. Кожна зі служб, Nginx (веб-сервер), MySQL (база даних) і Wordpress (рушій блогу), інкапсулюється в контейнер. Архітектура Docker містить три основні компоненти – образ, контейнери та реєстри. Розглянемо кожен компонент окремо.

Образи (Docker Images) слугують шаблонами для створення Docker-контейнерів. Контейнери, своєю чергою, є екземплярами образів та вважаються другим основним компонентом в архітектурі Docker.

Реєстри образів (репозиторії). Отже, образи та контейнери – два основних компоненти архітектури Docker. Але де вони розташовані? Усі контейнери в наведеному прикладі побудовано зі стандартних образів, що збережено в Docker Hub (реєстрі). Реєстри наділені можливістю додавання образів та їх завантаження з реєстру.

Далі, для повноти аналізу процесів використання Docker, варто розглянути особливості управління сотнями або тисячами контейнерів Docker на кількох серверах.

Докер дає змогу розгортати мікросервіси один за одним на одному хості (сервері). Невеликий додаток, у прикладі вище, не потребує складного управління. Але що робити у випадку, коли додаток збільшується? Як розгорнути кілька контейнерів у кожному з декількох запущених серверів? Як масштабувати ці сервери "догори" та "донизу"? Для вирішення цих окреслених питань, Docker включає системи оркестрування контейнерами.

Контейнерна оркестрова система (container orchestration system) – це додатковий інструмент, який необхідно використовувати з Docker. До середини 2016 р. Docker фактично не надав жодного конкретного способу керування додатками, побудованими з тисяч мікросервісів. Але тепер є Docker Swarm – вбудована платформа для оркестрування контейнерів тощо.

Docker Swarm дає змогу використовувати Docker CLI для запуску команд, тому можна легко ініціалізувати групи контейнерів, додавати та видаляти контейнери з цих груп.

Як хмарні рішення, які можуть допомогти запустити докеровані додатки та оркеструвати контейнери, потрібно розглядати такі сервіси:

- *Google Cloud Platform* з підтримкою Kubernetes;
- *Amazon ECS*. Web-сервери Amazon дають змогу використовувати службу Elastic Compute Cloud (EC2) для запуску та оброблення контейнерів Docker;
- *Azure Container Service* є хостинговим рішенням, подібним до Amazon ECS, і підтримує різні способи для оркестрування докерованих додатків, включаючи Kubernetes, DC / OS з Mesos та Docker Swarm.

Отже, використання мікросервісів та контейнерів вважають ефективним сучасним способом створення масштабованих і керованих веб-програм. У разі не контейнеризації мікросервісу виникатимуть труднощі під час їх розгортання та керування. Саме тому, для уникнення будь-яких проблем під час їх розгортання, рекомендовано використання Docker. Окрім цього, контейнерна система оркестрування дає змогу ефективно керувати комплексними докеризованими програмами.

Висновки. За результатами проведеного аналізу процесів використання Docker означено низку переваг, які полягають в оптимізації використання обчислювальних ресурсів, працездатності у відокремлених від ОС середовищах та можливості оркестрування комплексними програмами, що унеможливлює виникнення конфліктів між різними середовищами розробки під час побудови мікросервісів.

Перелік використаних джерел

Amazon Web Services. (n.d.). Amazon Web Services – cloud computing services. Retrieved from: http://aws.amazon.com/ru/

Belonozhko, P., Belous, V., Kutsevych, N., & Khramov, D. (2016). Free cloud hardware and software platforms. *Analytical review. Science: Internet Magazine, 8*(6). https://doi.org/10.15862/61TVN616

Bylina, A. (2016). Automate the deployment of infrastructure for business. *Problems and prospects of modern science, 4*, 87–91. [in Russian].

Docker. (n.d.). Docker – Build, Ship, and Run Any App, Anywhere. Retrieved from: https://www.docker.com/

Korolov, O., Gavrikov, I., & Smirnov, A. (2017). The economic role of virtualization in information systems. *International scientific review, 5*(36), 69–70. [in Russian].

Musaev, A., Gazul, S., & Anantchenko, I. (2014). The information infrastructure design of an educational organization using virtualization technologies. *Bulletin of the Saint Petersburg State Institute of Technology (Technical University), 27*(53), 71–76. [in Russian].

О. А. Андрушко[1], Ю. А. Борзов[2], И. О. Малець[2], А. В. Придатко[2]

[1] Symfoni ESM – a Fujitsu Company, г. Осло, Норвегия

[2] Львовский государственный университет безопасности жизнедеятельности, г. Львов, Украина

АНАЛИЗ ПРОЦЕССОВ ИСПОЛЬЗОВАНИЯ DOCKER ДЛЯ ПОСТРОЕНИЯ МИКРОСЕРВИСОВ

Описана проблема сложности командной работы в процессе разработки программных продуктов при удаленном доступе. Дополнен существующий багаж знаний о технологиях контейнеризации в свете особенностей использования Docker для построения микросервисов. Определены особенности использования и основные преимущества Docker-технологий в процессе построения микросервисов. Освещены преимущества контейнерной технологии над процессом виртуализации машины как инструмента оптимизации использования вычислительных ресурсов. Описаны пути избегания конфликтов между различными средами разработки путем использования Docker, за счет гарантии того, что микросервисы приложений будут работать в полностью отделенных от операционной системы средах. Описан принцип оптимизации сред, с необходимыми библиотеками и языками, для различных участников проектной команды на основе Docker Hub. Определен ряд основных преимуществ применения процесса контейнеризации над использованием виртуальных машин в пределах одного сервера с целью построения микросервисов. Рассмотрена архитектура микросервиса для лучшего понимания особенностей применения Docker в процессе разработки программ. Рассмотрена архитектура Docker и описаны основные компоненты. Описаны Docker-системы для динамического управления контейнерами.

Ключевые слова: контейнеризация; разработка микросервисов; виртуализация; Docker-технология.

O. A. Andrushko[1], Yu. O. Borzov[2], I. O. Malets[2], O. V. Prydatko[2]
[1] Symfoni ESM – a Fujitsu Company, Oslo, Norway
[2] Lviv State University of Life Safety, Lviv, Ukraine

ANALYSIS OF THE USE OF DOCKER TO BUILD MICROSERVICES

The problem of the complexity of teamwork in the process of developing software products with remote access is described. The existing knowledge of containerization technologies is supplemented in the light of the peculiarities of using Docker for creating microservices. The features of use and the main advantages of Docker-technologies in the process of building micro-services are determined. The advantages of container technology over the process of machine virtualization as a tool for optimizing the use of computing resources are highlighted. Described ways of avoiding conflicts between different development environments by using Docker, due to the guarantee that application micro services will operate in completely separate environments from the operating system. The optimization principle of environments, with necessary libraries and languages, for various participants of a design team based on Docker Hub is described. A number of main advantages of using the containerization process over the virtual machines within a single server for the purpose of creating micro-services are determined. Microservice architecture is considered for better understanding of Docker application features in the process of program development. Docker architecture is considered and the main components are described. Description of Docker-system for containers dynamic management is presented. According to the analysis of the processes of using Docker results, there is a number of advantages that include optimizing the computing resources use, efficiency in separated from the OS environments and the possibility of orchestration by complex programs, which allows avoiding conflicts between different development environments in the process of development of microservices.

Keywords: containerization; development of microservice; virtualization; Docker-technology.