**УДК 338.47**

# КОНТЕЙНЕРИЗАЦІЯ ТА ОРКЕСТРАЦІЯ: DOCKER ТА KUBERNETES

**Маковєєва В. К.**
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: valeriia.makovieieva@nure.ua

В даній роботі були розглянуті питання використання та переваг технологій Docker та Kubernetes у сучасній розробці програмного забезпечення. В ході проведеного аналізу були розглянуті основні переваги Docker, які полягають в його здатності до контейнеризації, портативності, ефективному використанні ресурсів та прискоренні розробки. Висвітлено випадки використання цієї потужної комбінації технологій у реальних проектах, підкреслюючи їхню важливість у сфері розробки програмного забезпечення.

**Ключові слова:** контейнеризація, портативність веб-розгортання, контейнер, оркестратор, декларативний підхід, мережеві правила, ізоляція додатків, масштабованість.

# CONTAINERIZATION AND ORCHESTRATION: DOCKER AND KUBERNETES

Makovieieva V. K.
Kharkiv Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av, 14
E-mail: valeriia.makovieieva@nure.ua

This paper examines use and benefits of Docker and Kubernetes technologies in modern software development. The analysis highlights key advantages of Docker, including its containerization capability, portability, efficient resource utilization, and accelerated development. Real-world cases of using this powerful technology combination in projects are discussed, emphasizing their significance in field of software development.

**Keywords:** containerization, web, deployment, portability, container, orchestrator

Інформаційні технології залишаються вкрай актуальними в сучасному світі, перетворюючи бізнес-процеси, покращуючи системи охорони здоров'я, модернізуючи освіту, забезпечуючи глобальну зв'язність через соціальні мережі та відіграючи важливу роль у наукових дослідженнях. Важливим аспектом також є забезпечення безпеки даних, що стає дедалі критичнішим в умовах цифрової трансформації та збільшення обсягів інформації [1-6].

Тобто у сучасному світі інформаційних технологій, де швидкість розробки та впровадження нового програмного забезпечення визначає конкурентоспроможність компанії, технології контейнеризації, зокрема Docker та Kubernetes, стали невід'ємною частиною розробки та управління додатками.

Docker – це платформа контейнеризації та середовище виконання, а Kubernetes – це платформа для запуску та керування контейнерами з багатьох середовищ виконання контейнерів. Kubernetes підтримує численні середовища виконання контейнерів, зокрема Docker.

Коли Docker був представлений у 2013 році, він приніс нам сучасну еру контейнерів і започаткував обчислювальну модель, засновану на мікросервісах. Оскільки контейнери не залежать від власної операційної системи, вони полегшують розробку слабко пов'язаних і масштабованих мікросервісів, дозволяючи командам декларативно пакувати додаток, його залежності та конфігурацію разом у вигляді образу контейнера.

Проте зі зростанням складності додатків, які потребували розміщення контейнерів, розподілених між численними серверами, виникли проблеми, зокрема: як координувати та планувати роботу декількох контейнерів, як забезпечити зв'язок між контейнерами, як

масштабувати екземпляри контейнерів та багато іншого. Kubernetes був представлений як спосіб вирішення цих проблем [7].

Коли мова заходить про контейнерну технологію, найчастіше зустрічаються такі назви, як Docker та Kubernetes. Ви можете запитати: яка з них краща? Але часто справа не в тому, яка з них краща, а в тому, як ви можете використовувати обидві з них на свою користь.

Docker – це комерційна платформа контейнеризації та середовище виконання, що допомагає розробникам створювати, розгортати та запускати контейнери. Він використовує клієнт-серверну архітектуру з простими командами та автоматизацією через єдиний API.

Docker також надає інструментарій, який зазвичай використовується для пакування додатків у незмінні образи контейнерів шляхом написання Docker-файлу, а потім запуску відповідних команд для створення образу за допомогою сервера Docker.

Розробники можуть створювати контейнери і без Docker, але платформа Docker робить це простіше. Ці образи контейнерів можна розгорнути і запустити на будь-якій платформі, що підтримує контейнери, наприклад, Kubernetes, Docker Swarm, Mesos або HashiCorp Nomad.

Хоча Docker забезпечує ефективний спосіб пакування та розповсюдження контейнерних додатків, запуск та управління контейнерами в масштабі за допомогою одного лише Docker є складним завданням.

Координація та планування контейнерів на декількох серверах/кластерах, оновлення або розгортання додатків з нульовим часом простою, а також моніторинг стану контейнерів - це лише деякі з аспектів, які необхідно враховувати.

Для вирішення цих та інших проблем з'явилися рішення для оркестрування контейнерів у вигляді Kubernetes, Docker Swarm, Mesos, HashiCorp Nomad та інших. Вони дозволяють організаціям керувати великою кількістю контейнерів та користувачів, ефективно балансувати навантаження, забезпечувати автентифікацію та безпеку, багатоплатформне розгортання тощо [7-10].

Kubernetes (іноді згадується як K8s) – це популярна платформа з відкритим вихідним кодом, яка організовує системи виконання контейнерів на кластері мережевих ресурсів. Kubernetes можна використовувати з Docker або без нього.

Спочатку Kubernetes був розробленений компанією Google, яка потребувала нового способу запуску мільярдів контейнерів на тиждень у великих масштабах. Kubernetes був випущений Google з відкритим вихідним кодом у 2014 році і зараз широко вважається лідером ринку і стандартним інструментом оркестрування для контейнерів і розподіленого розгортання додатків.

Google зазначає, що "основна мета розробки Kubernetes – полегшити розгортання та управління складними розподіленими системами, одночасно отримуючи вигоду від покращеного використання, яке забезпечують контейнери".

Kubernetes об'єднує набір контейнерів у групу, якою можна керувати на одній машині, щоб зменшити накладні витрати на мережу та підвищити ефективність використання ресурсів. Прикладом набору контейнерів є сервер додатків, кеш Redis та база даних sql. Контейнери Docker – це один процес на контейнер.

Kubernetes надає надзвичайну користь командам DevOps, сприяючи ефективному виявленню сервісів, розподілу навантаження в кластері, автоматизованому розгортанню та відкатам, а також забезпечуючи самовідновлення контейнерів, які можуть вийти з ладу. Цей інструмент також займається ефективним управлінням конфігурацією. Для команд, що працюють у сфері розробки та впровадження програм, Kubernetes стає критично важливим фактором. Важливо відзначити, що в контексті побудови надійних конвеєрів DevOps CI/CD, Kubernetes виявляється невід'ємним інструментом, сприяючи забезпеченню стабільності та ефективності процесів розробки та впровадження програмного забезпечення [7, 8].

Однак Kubernetes не є повноцінною платформою як послугою (PaaS), і є багато міркувань, які слід враховувати при створенні та управлінні кластерами Kubernetes.

Складність управління Kubernetes є основним фактором, чому багато клієнтів вирішують використовувати керовані сервіси Kubernetes від хмарних постачальників.

Kubernetes, який часто називають "хмарним Linux", є найпопулярнішою платформою для оркестрування контейнерів не без причини. Ось деякі з них:

- Kubernetes постачається з потужним API та інструментом командного рядка під назвою kubectl, який виконує більшу частину важкої роботи з управління контейнерами, дозволяючи вам автоматизувати ваші операції. Шаблон контролера у Kubernetes гарантує, що програми/контейнери працюватимуть саме так, як вказано;
- Kubernetes керує ресурсами, наданими йому від вашого імені. Це дозволяє розробникам зосередитися на написанні коду додатків, а не на базовій обчислювальній, мережевій інфраструктурі чи інфраструктурі зберігання даних;
- Kubernetes стежить за робочим середовищем і порівнює його з бажаним станом. Він виконує автоматичну перевірку працездатності сервісів і перезапускає контейнери, які вийшли з ладу або зупинилися. Kubernetes робить сервіси доступними лише тоді, коли вони працюють і готові до роботи.

В той час як Docker є середовищем виконання контейнерів, Kubernetes є платформою для запуску та керування контейнерами з багатьох середовищ виконання контейнерів.

Kubernetes підтримує численні контейнерні середовища виконання, включаючи Docker, containerd, CRI-O та будь-яку реалізацію Kubernetes CRI (Container Runtime Interface). Хороша метафора: Kubernetes – це "операційна система", а контейнери Docker – це "програми", які ви встановлюєте на "операційну систему" [9]. Як співпрацюють ці технології зображено на рис. 1.

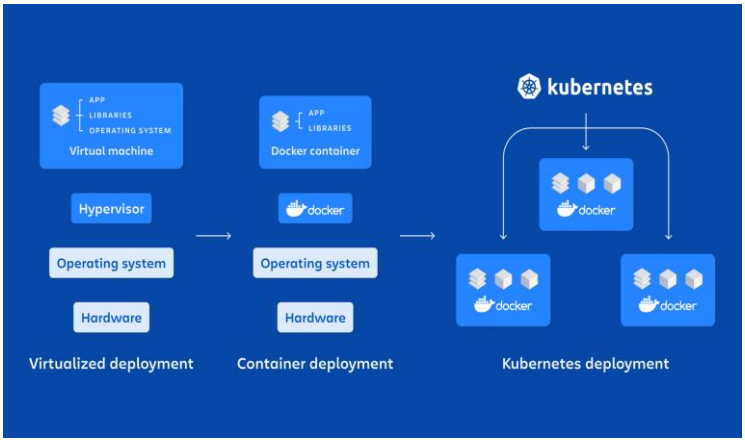


Рисунок 1 – Принцип роботи Docker з Kubernetes

Сам по собі Docker дуже корисний для сучасної розробки додатків. Він вирішує класичну проблему "працює на моїй машині", але більше ніде. Інструмент оркестрування контейнерів Docker Swarm здатний впоратися з розгортанням виробничого контейнера, що складається з декількох контейнерів. Коли система зростає і потрібно додати багато контейнерів, об'єднаних у мережу, автономний Docker може зіткнутися з деякими проблемами, які Kubernetes допомагає вирішити.

Якщо порівнювати ці дві системи, то краще порівнювати Kubernetes з Docker Swarm. Docker Swarm або режим рою Docker – це інструмент оркестрування контейнерів, подібний до Kubernetes, тобто він дозволяє керувати декількома контейнерами, розгорнутими на декількох

хостах, на яких запущено сервер Docker. Режим рою за замовчуванням вимкнений, і його потрібно встановлювати та налаштовувати команді DevOps.

Docker Swarm зазвичай вимагає менше налаштувань та конфігурації, ніж Kubernetes, якщо ви будуєте та запускаєте власну інфраструктуру. Він пропонує ті ж переваги, що і Kubernetes, наприклад, розгортання вашого додатку за допомогою декларативних YAML-файлів, автоматичне масштабування сервісів до бажаного стану, балансування навантаження між контейнерами в кластері, а також безпеку та контроль доступу до ваших сервісів. Якщо у вас небагато робочих навантажень, ви не проти керувати власною інфраструктурою або вам не потрібні специфічні функції, які пропонує Kubernetes, то Docker Swarm може бути чудовим вибором.

Kubernetes складніший у налаштуванні на початку, але пропонує більшу гнучкість та можливості. Він також має широку підтримку від активної спільноти з відкритим кодом. Kubernetes підтримує декілька стратегій розгортання з коробки, може керувати входом в мережу та забезпечує спостережливість з коробки у ваших контейнерах. Всі основні постачальники хмарних сервісів пропонують керовані сервіси Kubernetes, які значно полегшують початок роботи та використання переваг хмарних функцій, таких як автоматичне масштабування. Якщо ви використовуєте багато робочих навантажень і потребуєте хмарної інтероперабельності, а також маєте багато команд у вашій організації, що створює потребу в більшій ізоляції сервісів, то Kubernetes, ймовірно, є тією платформою, яку варто розглянути.

Наприклад, такі компанії, як Google, Microsoft, Amazon, і Facebook, традиційно використовують у роботі контейнери і оркестратори. Деякі інші відомі компанії, які також активно використовують Docker та Kubernetes, включають:

- у Netflix використовують контейнери та Kubernetes для розгортання та керування своїми мікросервісами;
- в Uber використовують Docker для контейнеризації додатків та Kubernetes для оркестрації цих контейнерів;
- Spotify також використовує Docker та Kubernetes для розгортання та управління своїми додатками;
- IBM працює з контейнерами та Kubernetes у своїх хмарних рішеннях, таких як IBM Cloud Kubernetes Service;
- Twitter також використовує контейнери та оркестратори для покращення масштабованості та надійності своїх систем;
- як провідний постачальник відкритого програмного забезпечення, Red Hat активно підтримує Docker та Kubernetes через свої продукти, такі як Red Hat OpenShift.

Це лише кілька прикладів, і багато компаній у світі використовують Docker та Kubernetes або подібні технології для побудови та управління своїми інфраструктурами. Тренд використання контейнерів і оркестраторів залишається дуже активним, і нові компанії приєднуються до користувачів цих технологій.

Таким чином, в цій роботі мова йде про технологій Docker та Kubernetes у сучасній розробці програмного забезпечення. Kubernetes, хоча і складніший у налаштуванні на початковому етапі, пропонує більше гнучкості та можливостей порівняно з Docker Swarm. Він забезпечує широку підтримку від відкритої спільноти, підтримує різні стратегії розгортання, керує ресурсами та забезпечує спостережливість у ваших контейнерах. Зокрема, Kubernetes дозволяє легко взаємодіяти з різними постачальниками хмарних послуг та надає керовані сервіси Kubernetes для спрощення використання хмарних функцій, таких як автоматичне масштабування. Компанії, такі як Google, Microsoft, Amazon, Facebook, Netflix, Uber, Spotify, IBM, Twitter та Red Hat, свідчать про широкий розповсюдження Docker та Kubernetes в сучасному IT-середовищі. Ці технології стають невід'ємною частиною інфраструктури

багатьох великих компаній, що дозволяє їм підтримувати масштабовані та ефективні інфраструктури для розгортання та управління додатками.

Тренд використання контейнерів та оркестраторів, зокрема Docker та Kubernetes, продовжує залишатися високоактивним, а нові компанії активно приєднуються до користувачів цих технологій. Це свідчить про значущий внесок цих інструментів у сучасний IT-ландшафт та їхню важливу роль у розвитку інфраструктури та розгортання програмного забезпечення. Використання Docker та Kubernetes дозволяє підвищити ефективність розробки та впровадження програмного забезпечення. Можливість упаковування додатків у контейнери надає їм портативність та незалежність від середовища виконання, що полегшує розгортання на різних платформах. Крім того, використання оркестратора контейнерів, такого як Kubernetes, сприяє автоматизації багатьох аспектів управління додатками, включаючи масштабування, моніторинг, та відновлення від збоїв. Це дозволяє розробникам та DevOps-командам зосередитися на розробці функціоналу, забезпечуючи при цьому надійність та стабільність роботи додатків у виробничому середовищі. У сфері безпеки, Docker та Kubernetes також надають інструменти для контролю доступу, ізоляції контейнерів, аудиту та моніторингу подій, що важливо для забезпечення високого рівня захисту даних та дотримання вимог щодо безпеки інформації. Інтеграція з іншими інструментами та технологіями, такими як системи контролю версій, CI/CD пайплайни, та інші, також забезпечує повністю зрошену екосистему для розробки та управління додатками. Це робить Docker та Kubernetes потужними інструментами для будь-якого розміру організації, що прагне до ефективності та інновацій у своїй розробці.

ЛІТЕРАТУРА

1. Deineko Z. Confidentiality of Information when Using QR-Coding // International Journal of Academic Information Systems Research (IJAISR) / Z. Deineko, S. Sotnik, V. Lyashenko, 2022. – Vol. 6, Issue 9. – pp. 10-15.

2. Al-Sherrawi M. H. Information model of plastic products formation process duration by injection molding method // International Journal of Mechanical Engineering and Technology (IJMET), 2018. – Volume 9, Issue 3. – pp. 357 -366.

3. Sotnik S. V. Gamification in science: game platforms for learning // Матеріали конференції «Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації - 2023» / S. V. Sotnik, A. S. Andreiev, 2023. – pp. 87- 89.

4. Deineko Z. Multimedia Systems in Education // International Journal of Academic Information Systems Research (IJAISR) / Z. Deineko, S. Sotnik, V. Lyashenko, 2022. – vol. 6, issue 7. – pp. 23-28.

5. Borysenko I. A. Chat gpt features in data search // The 9 th International scientific and practical conference “Scientific progress: innovations, achievements and prospects” (May 29-31, 2023) MDPC Publishing, Munich, Germany / I. A. Borysenko, S. V. Sotnik, 2023. – pp. 139 – 143.

6. Sotnik S. РАЗРАБОТКА ИНФОРМАЦИОННО-ПОИСКОВОЙ СИСТЕМЫ ВЫБОРА ФОРМУЮЩИХ ЭЛЕМЕНТОВ ЛИТЬЕВЫХ ФОРМ //INNOVATIVE TECHNOLOGIES AND SCIENTIFIC SOLUTIONS FOR INDUSTRIES / S. Sotnik, V. Nevliudova, I. Malaya, 2017. – №. 2 (2). – С. 86-92.

7. Morabito, R. (2017) Virtualization on Internet of Things edge devices with Container technologies: a performance evaluation documentation [Електроний ресурс]. – Режим доступу до ресурсу: https://doi.org/10.1109/ACCESS.2017.2704444.

8. The Kubernetes authors (2020). Kubernetes documentation [Електроний ресурс]. – Режим доступу до ресурсу: https://kubernetes.io/docs/home/.

9. Binani, H . (2018). Kubernetes vs Docker Swarm – A comprehensive comparison . [Електроний ресурс]. – Режим доступу до ресурсу: https://hackernoon.com/kubernetes-vs-docker-swarm-a-comprehensive-comparison-73058543771e/.

10. Docker Inc (2020). Docker Overview [Електроний ресурс]. – Режим доступу до ресурсу: https://docs.docker.com/get-started/overview/.