**УДК 004.432.2:004.738.5:004.41:004.451**
**DOI https://doi.org/10.32689/maup.it.2025.1.23**

***Артем ПОДВИЖЕНКО***
*старший викладач кафедри комп'ютерних наук та програмної інженерії, Приватний вищий навчальний заклад «Європейський університет», artem.podvizhenko@e-u.edu.ua*
***ORCID:** 0000-0003-1826-7566*

***Антон ПЕРЕВЕРЗЄВ***
*старший викладач кафедри комп'ютерних наук та програмної інженерії, Приватний вищий навчальний заклад «Європейський університет», anton.pereverziev@e-u.edu.ua*
***ORCID:** 0000-0003-0778-9203*

***Леонід ЛИТВИНЕНКО***
*кандидат технічних наук, доцент кафедри комп'ютерних наук та програмної інженерії, Приватний вищий навчальний заклад «Європейський університет», leonid.lytvynenko@e-u.edu.ua*
***ORCID:** 0000-0002-0828-383X*
***Scopus Author ID:** 57193444117*

***Олег СКЛЯРЕНКО***
*аспірант, Приватний вищий навчальний заклад «Європейський університет», osklyarenko@e-u.edu.ua*
***ORCID:** 0009-0003-4908-0187*

# РОЛЬ КОНТЕЙНЕРИЗАЦІЇ ТА ВІРТУАЛІЗАЦІЇ НА РІВНІ ОПЕРАЦІЙНОЇ СИСТЕМИ В РОЗВИТКУ ХМАРНО ОРІЄНТОВАНИХ ДОДАТКІВ

***Анотація. Мета** цієї роботи дослідити вплив сучасних технологій віртуалізації та контейнеризації на трансформацію підходів до проєктування, розробки та експлуатації інформаційних систем у хмарних середовищах, зокрема в контексті переходу від монолітної до мікросервісної архітектури.*

***Методологія.** У статті проведено глибокий аналіз фундаментальної ролі технологій віртуалізації та контейнеризації у трансформації хмарних обчислень. Розглянуто історичний розвиток віртуалізації, вплив гіпервізорів, вимоги до гостьових ОС, а також переваги контейнеризації з акцентом на інструменти Docker та Kubernetes. Проаналізовано архітектуру контейнеризованих середовищ, їхню інтеграцію з хмарною інфраструктурою та нові вимоги до базових ОС, включаючи використання незмінних систем.*

***Наукова новизна.** У роботі узагальнено сучасний підхід до проєктування операційних систем для контейнеризованих хмарних середовищ. Особливо підкреслено еволюцію вимог до ОС, розвиток спеціалізованих дистрибутивів, зміну безпекових моделей, а також роль Kubernetes як розподіленої операційної системи для хмари. Окреслено тренди майбутнього розвитку ОС, пов'язані з інтелектуалізацією управління хмарними середовищами.*

***Висновки.** Розуміння ролі та впливу технологій віртуалізації й контейнеризації є критично важливим для ефективного проєктування, розробки та підтримки хмарно-орієнтованих додатків. Подальший розвиток операційних систем йтиме шляхом посилення безпеки, оптимізації взаємодії з апаратним забезпеченням та впровадження інтелектуальних механізмів управління для забезпечення стабільності та масштабованості в умовах високої динаміки хмарних середовищ.*

***Ключові слова:** віртуалізація, контейнеризація, хмарні технології, операційні системи, Docker, Kubernetes, хмарно-орієнтовані додатки, мікросервіси, гіпервізор, оркестрація контейнерів.*

## Artem PODVIZHENKO, Anton PEREVERZIEV, Leonid LYTVYNENKO, Oleh SKLIARENKO. THE ROLE OF CONTAINERIZATION AND OS-LEVEL VIRTUALIZATION IN THE DEVELOPMENT OF CLOUD-NATIVE APPLICATIONS

***Abstract. The aim** of this study is to examine the impact of modern virtualization and containerization technologies on the transformation of approaches to the design, development, and operation of information systems in cloud environments, particularly in the context of the shift from monolithic to microservices architecture.*

***Methodology.** The paper presents an in-depth analysis of the fundamental role of virtualization and containerization technologies in the transformation of cloud computing. It explores the historical development of virtualization, the influence of hypervisors, requirements for guest operating systems, and the benefits of containerization with a focus on tools such as Docker and Kubernetes. The architecture of containerized environments is analyzed, along with their integration with cloud infrastructure and the emerging requirements for base operating systems, including the use of immutable systems.*

***Scientific Novelty.** The study summarizes the modern approach to designing operating systems for containerized cloud environments. It particularly emphasizes the evolution of OS requirements, the development of specialized distributions, changes in security models, and the role of Kubernetes as a distributed cloud operating system. Future trends in OS development are outlined, including the growing role of intelligent management in cloud environments.*

***Conclusions.*** *Understanding the role and impact of virtualization and containerization technologies is critically important for the effective design, development, and maintenance of cloud-oriented applications. The future evolution of operating systems will move toward enhanced security, optimized hardware interaction, and the implementation of intelligent management mechanisms to ensure stability and scalability in highly dynamic cloud environments.*

***Key words:*** *virtualization, containerization, cloud technologies, operating systems, Docker, Kubernetes, cloud-oriented applications, microservices, hypervisor, container orchestration.*

**Постановка проблеми в загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями.** Епоха цифрової трансформації та експоненційне зростання обсягів даних стимулюють постійний розвиток обчислювальних парадигм. Хмарні обчислення стали домінуючою моделлю, пропонуючи безпрецедентну гнучкість, масштабованість та економічну ефективність. Цей перехід супроводжується зміною архітектурних підходів до побудови додатків – від монолітних гігантів до незалежно розгортаних та масштабованих мікросервісів. Така декомпозиція вимагає відповідних змін на інфраструктурному рівні, і саме тут технології віртуалізації та контейнеризації виступають як ключові фактори успіху. Вони забезпечують необхідний рівень абстракції апаратних ресурсів та операційних систем, дозволяючи розробникам та операційним командам зосередитися на логіці додатків, а не на складнощах базової інфраструктури [11, 4].

Історично першою масовою технологією, що забезпечила ефективне спільне використання фізичних ресурсів, стала віртуалізація. Вона дозволила запускати на одному сервері безліч ізольованих віртуальних машин (ВМ), кожна зі своєю повноцінною операційною системою. Це стало основою для появи концепції інфраструктури як послуги (IaaS) у хмарі [14]. Однак з розвитком мікросервісної архітектури та потребою у ще більшій щільності розміщення додатків та швидкості їх розгортання, на передній план вийшла контейнеризація. Ця технологія пропонує легший механізм ізоляції, що використовує спільне ядро операційної системи хоста, забезпечуючи при цьому необхідний рівень безпеки та портативності [11, 4]. Поява Docker значно спростила процес створення та управління контейнерами, зробивши контейнеризацію доступною та популярною технологією [5, 4]. А з поширенням контейнерів виникла гостра потреба в ефективних інструментах їх оркестрації, що призвело до домінування Kubernetes як платформи для автоматизованого управління життєвим циклом контейнеризованих додатків у масштабі [10, 16].

Ці технології не тільки змінили методи розробки та розгортання, але й суттєво вплинули на вимоги до операційних систем, що функціонують у хмарних середовищах. Традиційні ОС, розроблені для роботи на голому залізі, потребують адаптації та еволюції для ефективної підтримки віртуалізованих та контейнеризованих робочих навантажень.

**Аналіз останніх досліджень та публікацій.** Проблема оптимізації розгортання та управління додатками в хмарних середовищах є предметом активних досліджень протягом останнього десятиліття. Поява та стрімкий розвиток технологій контейнеризації та віртуалізації на рівні операційної системи (ОС) суттєво змінили підходи до розробки, тестування та експлуатації хмарно-орієнтованих систем. Аналіз наукових публікацій та технічних звітів дозволяє виділити кілька ключових напрямків досліджень у цій сфері.

Значна увага приділяється порівняльному аналізу продуктивності та ефективності використання ресурсів між традиційною віртуалізацією (гіпервізори типу 1 і 2) та віртуалізацією на рівні ОС (контейнерами). Роботи [2] демонструють, що контейнеризація, завдяки спільному використанню ядра ОС хоста, зазвичай забезпечує менші накладні витрати на процесорний час та пам'ять порівняно з віртуальними машинами, що є критичним для мікросервісних архітектур та високо навантажених додатків [9]. Дослідження також фокусуються на вимірюванні затримок мережевого обміну та продуктивності дискових операцій в контейнеризованих середовищах.

Важливим аспектом є питання безпеки контейнерів. Оскільки контейнери на одному хості ділять спільне ядро ОС, це створює потенційні вектори атак. Дослідники аналізують механізми ізоляції (namespaces, cgroups), вразливості образів контейнерів, безпеку мережевої взаємодії та можливості підвищення рівня безпеки за допомогою спеціалізованих інструментів сканування та моніторингу [4, 5]. Порівняння моделей безпеки контейнерів та віртуальних машин є постійною темою дискусій [14].

Активно досліджується роль систем оркестрації контейнерів, таких як Kubernetes та Docker Swarm, у забезпеченні масштабованості, відмовостійкості та автоматизації управління життєвим циклом хмарно-орієнтованих додатків. Аналізуються алгоритми планування контейнерів, механізми авто масштабування, сервіс-дискавері, управління конфігураціями та секретами. Дослідження також стосуються складності впровадження та підтримки таких систем [7].

Вивчається вплив контейнеризації на процеси розробки та DevOps. Технології, такі як Docker, спрощують створення відтворюваних середовищ розробки, тестування та розгортання, сприяючи

впровадженню практик неперервної інтеграції та неперервної доставки (CI/CD) [1]. Досліджується ефективність інтеграції контейнерів у конвеєри CI/CD та їхній вплив на швидкість виведення продуктів на ринок.

Незважаючи на значний обсяг досліджень, залишаються актуальними питання оптимізації продуктивності для специфічних навантажень (наприклад, stateful-додатків), розробки уніфікованих підходів до забезпечення безпеки в гетерогенних контейнерних середовищах, а також оцінки довгострокових економічних ефектів від переходу на контейнеризовані хмарно-орієнтовані архітектури. Дане дослідження спрямоване на систематизацію переваг та викликів використання контейнеризації та віртуалізації на рівні ОС, а також на аналіз їхньої синергетичної ролі у становленні сучасних хмарних додатків.

**Мета статті** полягає в детальному аналізі фундаментальної ролі технологій віртуалізації та контейнеризації у трансформації сучасних хмарних середовищ. Основна увага зосереджена на дослідженні глибокого впливу цих технологій на дизайн, функціональність, еволюцію та нові вимоги до операційних систем, що лежать в основі розробки та експлуатації хмарно-орієнтованих додатків.

**Виклад основного матеріалу.** Віртуалізація є наріжним каменем хмарних обчислень. Вона дозволяє хмарним провайдерам абстрагувати базове фізичне обладнання та надавати користувачам віртуальні ресурси за запитом. Цю абстракцію забезпечують гіпервізори (Virtual Machine Monitor – VMM), які створюють та управляють віртуальними машинами на одному фізичному сервері [15]. Існують два основні типи гіпервізорів: Type 1 (або «bare-metal»), які працюють безпосередньо на апаратному забезпеченні (наприклад, VMware ESXi, KVM, Xen), та Type 2 (або «hosted»), які працюють як звичайні програми на існуючій операційній системі (наприклад, VirtualBox, VMware Workstation) [15]. У хмарних центрах обробки даних переважно використовуються гіпервізори Type 1 через їхню продуктивність та безпеку.

Для операційних систем, що працюють як гостьові ОС у віртуалізованому середовищі, віртуалізація ставить низку специфічних вимог та викликів. Насамперед, це стосується взаємодії з гіпервізором. Історично, повна віртуалізація вимагала бінарної трансляції привілейованих інструкцій, що могло негативно впливати на продуктивність. Альтернативний підхід, пара-віртуалізація, передбачає модифікацію гостьової ОС для включення спеціальних «гіпервикликів» (hypercalls), які дозволяють їй безпосередньо взаємодіяти з гіпервізором, підвищуючи ефективність [14]. Хоча сучасні процесори мають апаратну підтримку віртуалізації, що значно зменшує накладні витрати, оптимізація драйверів та компонентів гостьової ОС для роботи з віртуалізованим обладнанням (мережевими адаптерами, контролерами сховища) все ще залишається важливою задачею.

Іншим важливим аспектом є динамічне управління ресурсами. У хмарних середовищах ресурси віртуальних машин, такі як CPU та RAM, можуть змінюватися динамічно. Гостьова ОС повинна вміти ефективно реагувати на ці зміни, адекватно перерозподіляти ресурси всередині себе та забезпечувати стабільну роботу додатків [14]. Це вимагає оптимізації механізмів планування процесів та управління пам'яттю в ОС спеціально для віртуалізованого середовища.

Крім того, швидкість операцій вводу/виводу нерідко виявляється обмежуючим фактором у віртуалізованих системах. Для ефективної роботи гостьова ОС потребує оптимізованих віртуальних драйверів для взаємодії з віртуалізованими пристроями, які, своєю чергою, взаємодіють з гіпервізором та фізичним обладнанням. Розвиток технологій, зокрема SR-IOV (Single Root I/O Virtualization), дозволяє ВМ отримувати прямий доступ до апаратних пристроїв, минаючи гіпервізор, що суттєво підвищує продуктивність вводу/виводу, однак вимагає відповідної підтримки з боку ОС.

Зрештою, гостьова ОС має коректно обробляти міграцію ВМ. Можливість «живої» міграції, тобто переміщення працюючої ВМ між фізичними хостами без переривання роботи, є ключовою функцією хмарних платформ для балансування навантаження та обслуговування. Гостьова ОС повинна бути спроможна впоратися з процесом міграції, зберігаючи стан запущених додатків та активні мережеві з'єднання. Таким чином, віртуалізація трансформувала операційні системи, зробивши їх більш «обізнаними» про віртуалізоване середовище, в якому вони працюють, та оптимізованими для ефективної роботи під управлінням гіпервізора.

Контейнеризація пропонує інший рівень абстракції – на рівні операційної системи, а не апаратного забезпечення. Контейнер – це стандартизований, виконуваний пакет програмного забезпечення, який включає код додатка, бібліотеки, системні інструменти, час виконання та все необхідне для його запуску. Ключова відмінність від ВМ полягає в тому, що контейнери використовують спільне ядро операційної системи хоста, тоді як кожна ВМ має власну повноцінну гостьову ОС [4, 11]. Це робить контейнери значно легшими, швидше запускаються та споживають менше ресурсів порівняно з ВМ [11].

Docker став синонімом контейнеризації, надавши простий та зручний інструментарій для створення, пакування та розповсюдження контейнерів за допомогою образів. Образ Docker – це шаблон тільки для читання, що містить інструкції для створення контейнера. Контейнер – це екземпляр образу, що

виконується [4, 5]. Простота використання Docker сприяла вибуховому зростанню популярності контейнеризації серед розробників.

Однак, управління великою кількістю контейнерів, особливо в розподілених хмарних середовищах, є складним завданням. Саме тут на сцену виходить Kubernetes – відкрита платформа, призначена для автоматизації розгортання, масштабування та управління контейнеризованими додатками. Kubernetes надає механізми для оркестрації контейнерів, що включає автоматизацію процесів розгортання, оновлення, видалення та моніторингу груп контейнерів (відомих як Pod'и) на кластері фізичних або віртуальних вузлів [10, 16]. Платформа також забезпечує автоматичне горизонтальне масштабування Pod'ів, реагуючи на завантаження процесора або інші метрики [16]. Крім того, Kubernetes полегшує управління життєвим циклом додатків, підтримуючи такі стратегії оновлення без простоїв, як канарійські випуски, синьо-зелені розгортання та відкати [10]. Він пропонує інструменти для виявлення сервісів та балансування навантаження, дозволяючи мікросервісам у різних Pod'ах знаходити один одного, взаємодіяти та автоматично розподіляти трафік. Kubernetes також керує зберіганням даних, динамічно надаючи постійні сховища для контейнерів, і забезпечує безпечне зберігання та надання конфігураційної інформації та чутливих даних додаткам [11].

Kubernetes, по суті, стає операційною системою для розподілених хмарно-орієнтованих додатків, працюючи на рівні кластера та абстрагуючи нижчі рівні інфраструктури, включаючи базові ОС вузлів.

Зростання популярності контейнеризації та оркестрації, зокрема через платформи як Kubernetes, значно вплинуло на вимоги до операційних систем, які слугують вузлами кластерів. Хоча традиційні серверні ОС можуть запускати контейнери, вони не є ідеальними для цієї ролі, що призвело до появи спеціалізованих «хмарних» ОС.

Ці нові операційні системи часто є мінімалістичними та незмінними (immutable). Для вузлів Kubernetes оптимальними є легковагі ОС з мінімальним набором компонентів, що включають лише ядро Linux, середовище виконання контейнерів (наприклад, containerd або CRI-O) та агент Kubernetes (kubelet). Прикладами слугують Container-Optimized OS від Google, Fedora CoreOS, RancherOS та Photon OS. Їхня «незмінність» означає, що коренева файлова система доступна лише для читання, а оновлення відбуваються шляхом заміни всього образу ОС, а не окремих пакетів. Такий підхід підвищує безпеку та спрощує управління, усуваючи проблеми з несумісністю пакетів чи помилками конфігурації після оновлень.

Оскільки контейнери використовують спільне ядро ОС хоста, посилена безпека на рівні ядра стає надзвичайно важливою. Це передбачає використання технологій ізоляції, таких як SELinux, AppArmor та seccomp, для обмеження можливостей процесів у контейнерах, а також регулярне оновлення ядра для виправлення вразливостей. Дослідження також ведуться у напрямку використання спеціалізованих мікроядер або унікернелів для подальшого підвищення безпеки та продуктивності контейнерів.

Крім того, необхідна оптимізація роботи з ресурсами для контейнерів. Хоча Kubernetes управляє розподілом ресурсів між Pod'ами, базова ОС відповідає за низькорівневе управління CPU та пам'яттю для процесів у контейнерах. Вона повинна ефективно використовувати cgroups та інші механізми для ізоляції ресурсів та пріоритезації навантажень згідно з політиками Kubernetes.

Важливою є також інтеграція з мережевими плагінами через інтерфейс Container Network Interface (CNI), який Kubernetes використовує для конфігурації мережі Pod'ів. Базова ОС має коректно взаємодіяти з реалізаціями CNI (як-от Calico, Flannel, Cilium) для забезпечення мережевої зв'язності та застосування мережевих політик. У хмарних середовищах часто застосовуються розподілені файлові системи для зберігання даних, доступних з різних вузлів. Тому хмарні ОС повинні мати вбудовану підтримку або можливість легкої інтеграції з такими системами для забезпечення постійного зберігання даних для контейнеризованих додатків.

Важливо зазначити що, ефективний моніторинг та діагностика є ключовими у складних розподілених середовищах. ОС для хмари мають надавати розширені метрики та інструменти для збору логів, які інтегруються з системами моніторингу та оркестрації на рівні Kubernetes, такими як Prometheus, Grafana чи ELK stack. Таким чином, контейнеризація та оркестрація вимагають від операційних систем значно більшої спеціалізації, мінімалізму, посиленої безпеки на низькому рівні та тісної інтеграції з платформами управління хмарною інфраструктурою.

**Висновки з даного дослідження і перспективи подальших розвідок у даному напрямку.** Технології віртуалізації та контейнеризації відіграли і продовжують відігравати вирішальну роль у формуванні сучасного ландшафту хмарних обчислень та розвитку хмарно-орієнтованих додатків.

Віртуалізація заклала фундаментальну основу, дозволивши абстрагувати апаратне забезпечення. Це забезпечило можливість ефективного консолідування та спільного використання фізичних ресурсів (ЦП, пам'ять, сховище, мережа), ізоляції робочих навантажень та стало ключовим елементом для

надання інфраструктури як послуги (IaaS), пропонуючи гнучкість у керуванні ресурсами, яка була неможлива раніше.

Контейнеризація, що набула масового поширення завдяки Docker та системам оркестрації, таким як Kubernetes, вивела ефективність та гнучкість на новий рівень. Вона запропонувала легші, швидші у запуску та менш ресурсоємні (порівняно з віртуальними машинами) одиниці розгортання. Це забезпечило незмінність середовищ між розробкою, тестуванням та продуктивним використанням, значно спростило портативність додатків між різними хмарними провайдерами та локальними інфраструктурами, і стало критично важливим каталізатором для реалізації мікросервісної архітектури та впровадження практик DevOps, прискорюючи цикли розробки та розгортання.

Ці технологічні зрушення неминуче спричинили суттєві зміни у вимогах до операційних систем, що використовуються в хмарних середовищах. Традиційні, універсальні серверні ОС, часто обтяжені зайвими компонентами, великою поверхнею атаки та повільнішим часом завантаження, виявилися неоптимальними для високодинамічних, масштабованих та ефемерних навантажень, характерних для контейнерів.

Як наслідок, відбувається чіткий перехід до спеціалізованих хмарно-орієнтованих операційних систем. Ці дистрибутиви характеризуються мінімалізмом (містять лише необхідні для запуску контейнерів компоненти, зменшуючи розмір образу та потенційні вразливості), підвищеною безпекою (часто реалізуючи принципи незмінної інфраструктури (immutable infrastructure), де ОС не модифікується після розгортання, а оновлюється шляхом заміни цілого образу), та можливістю програмного конфігурування через API для автоматизації управління. Вони оптимізовані для максимальної щільності розміщення контейнерів, швидкого завантаження та глибокої інтеграції з платформами оркестрації, такими як Kubernetes, для ефективного управління життєвим циклом контейнерів.

Майбутнє операційних систем у хмарі, ймовірно, буде пов'язане з подальшою глибокою спеціалізацією під конкретні типи навантажень (наприклад, AI/ML, високопродуктивні обчислення, обробка даних, edge computing). Очікується посилення безпеки на низькому рівні з використанням апаратних можливостей (наприклад, конфіденційні обчислення confidential computing) та впровадженням принципів нульової довіри (zero trust) на рівні ОС. Глибша інтеграція з апаратним забезпеченням, таким як SmartNICs, DPUs та GPU, стане стандартом для розвантаження мережевих функцій, функцій безпеки та прискорення обчислень безпосередньо в інфраструктурі. Ключовим напрямком стане розвиток самооптимізуючих та самовідновлюваних механізмів на основі штучного інтелекту та машинного навчання (AIOps). Це дозволить ОС динамічно керувати розподіленими ресурсами, передбачати та запобігати збоям, автоматично масштабуватись та оптимізувати продуктивність в умовах надзвичайно динамічних та складних хмарних середовищ.

Розуміння цієї еволюції – від віртуалізації через контейнеризацію до спеціалізованих, інтелектуальних ОС – є ключовим для архітекторів рішень, розробників та DevOps-інженерів для ефективного проектування, розробки, розгортання та експлуатації надійних, безпечних та продуктивних додатків у сучасній та майбутній хмарі. Це дозволяє приймати обґрунтовані рішення щодо вибору технологічного стеку та стратегій управління інфраструктурою.

**Список використаних джерел:**

1. Кім Дж., Гамбл Дж., Вілліс Дж., Дебуа П. DevOps Handbook: Як налагодити ефективну взаємодію розробників та IT-служби / пер. з англ. Київ : Фабула, 2020. 512 с.

2. Фельдман А. Р., Новицький О. В. Порівняльний аналіз продуктивності технологій віртуалізації та контейнеризації в хмарних обчисленнях. *Системи обробки інформації,* 2021. № 2(165). С. 115–121. DOI: 10.30748/SOI.2021.165.15.

3. Яровий Р., Улічев О., Скляренко О., Пашорін В. Моделювання мультиагентних систем захисту інформаційних ресурсів. *Herald of Khmelnytskyi National University. Technical Sciences,* 2024. 337(3(2)). С. 278–284. https://doi.org/10.31891/2307-5732-2024-337-3-42

4. Cloud Container Technologies: A State-of-the-Art Review. *IEEE Translations on Cloud Computing. Scribd*. URL: https://www.scribd.com/document/ 692088192/Container-review (дата звернення 20.04.2025)

5. Containerization in cloud computing: comparing Docker and Kubernetes for scalable web applications. *International Journal of Science and Research Archive,* 2024, 13(01), 3302–3315. https://doi.org/10.30574/ijsra.2024.13.1.2035

6. Containerization in Multi-Cloud Environment: Roles, Strategies, Challenges, and Solutions for Effective Implementation. *Arxiv*. URL: https://arxiv.org/html/2403.12980v2 (дата звернення 20.04.2025)

7. Hightower K., Burns B., Beda J. *Kubernetes: Up and Running: Dive into the Future of Infrastructure* (2nd ed.). Sebastopol: O'Reilly Media, 2019. 334 p.

8. Native Database Systems and Unikernels: Reimagining OS Abstractions for Modern Hardware. *Technische Universität München*. URL: https://www.cs.cit.tum.de/fileadmin/w00cfj/dis/papers/cumulus.pdf (дата звернення 22.04.2025)

9. Newman S. *Building Microservices: Designing Fine-Grained Systems.* Sebastopol: O'Reilly Media, 2015. 280 p.

10. On the Optimization of Kubernetes toward the Enhancement of Cloud Computing. *Mathematics* 2024. 12(16), 2476 https://doi.org/10.3390/math12162476
11. Managing Persistent Storage in Containers. *CloudOptimo.* URL: https://www.cloudoptimo.com/blog/managing-persistent-storage-in-containers (дата звернення 20.04.2025)
12. Resource Optimization: Top Strategies. *EPAM SolutionsHub*. URL: https://solutionshub.epam.com/blog/post/cloud-resource-optimization (дата звернення 22.04.2025)
13. Security Threats: Detection and Challenges. *Palo Alto Networks*. URL: https://www.paloaltonetworks.com.au/cyberpedia/cloud-security-threats-detection-and-challenges (дата звернення 19.04.2025)
14. Virtualization in Cloud Computing and Types. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/virtualization-cloud-computing-types/ (дата звернення 22.04.2025)
15. What is a Hypervisor? *Amazon*. URL: https://aws.amazon.com/what-is/hypervisor/ (дата звернення 21.04.2025)
16. What is container orchestration? *RedHat*. URL: https://www.redhat.com/en/topics/containers/what-is-container-orchestration (дата звернення 21.04.2025)
17. What Is Cloud Native? *Palo Alto Networks*. URL: https://www.paloaltonetworks.com/cyberpedia/what-is-cloud-native (дата звернення 22.04.2025)