УДК 004.042
DOI https://doi.org/10.32782/tnv-tech.2023.6.4

РЕАЛІЗАЦІЯ ТЕХНОЛОГІЙ ДЛЯ РОЗГОРТАННЯ ПРОГРАМ У КОНТЕЙНЕРІ

***Киричек Г. Г.** – кандидат технічних наук,*
доцент кафедри комп'ютерних систем та мереж
Національного університету «Запорізька політехніка»
ORCID ID: 0000-0002-0405-7122

***Тягунова М. Ю.** – кандидат технічних наук,*
доцент кафедри комп'ютерних систем та мереж
Національного університету «Запорізька політехніка»
ORCID ID: 0000-0002-9166-5897

***Смірнов В. В.** – магістр факультету комп'ютерних наук та технологій*
Національного університету «Запорізька політехніка»
ORCID ID: 0009-0006-3314-5072

Застосування віртуалізації та контейнеризації в хмарі, створює рівень абстракції між апаратними ресурсами та програмними компонентами, спрощуючи управління ресурсами та підвищуючи їх ефективність. Усі ці фактори підкреслюють важливість впровадження додатків у хмарі, як фундаментальної стратегії сучасних організацій, що прагнуть зберегти конкурентоспроможність та адаптивність в умовах постійних змін та зростаючих вимог інформаційного ринку. Метою роботи є дослідження методів та реалізація технологій для розгортання програм у контейнері. Об'єктом дослідження є процес реалізації системи розгортання програм у контейнері із використанням Google Kubernetes Engine, Terraform та Tekton. Предметом дослідження є моделі, методи та засоби реалізації системи розгортання програм у контейнері. Виходячи з того, що контейнери є самодостатніми віртуальними середовищами, які містять додатки та їх залежності, маємо можливість ефективно створювати, розгортати і керувати контейнерами для різних хмарних додатків і сервісів, забезпечуючи гнучкість та оптимальне використання ресурсів. Цей підхід сприяє швидкому масштабуванню додатків і забезпечує їх єдність у різних середовищах хмарного обчислення. Само розгортання застосунків, пов'язане з використанням інтерфейсів і стандартів взаємодії між частинами системи, розташованими в хмарі, що потребує підтримки обміну даними та ресурсами через мережу, використовуючи методи балансування та віртуалізації. Тому важливим є створення модульних та незалежних компонентів, які є масштабованими і розгортаються окремо. Також, балансування завантаження є методом, що використовується в інформаційних системах та комп'ютерних мережах з метою розподілу запитів, завдань та трафіку між різними серверами або ресурсами для досягнення найкращої продуктивності, надійності та доступності системи. А віртуалізація є технологією, яка дозволяє створювати віртуальні версії обчислювальних, мережевих, та інших ресурсів, забезпечуючи абстракцію між апаратними ресурсами та програмними засобами або сервісами, які їх використовують.

***Ключові слова:** кластер, контейнер, kubernetes, docker, віртуалізація.*

Kyrychek H. H., Tiahunova M. Yu., Smirnov V. V. Implementation of technologies for deploying programs in a container

Application of virtualization and containerization in the cloud creates a level of abstraction between hardware resources and software components, simplifying resource management and increasing their efficiency. All these factors emphasize the importance of implementing applications in the cloud, as a fundamental strategy of modern organizations that seek to maintain competitiveness and adaptability in the face of constant changes and growing requirements of the information market. The purpose of the work is to research methods and implement technologies for deploying applications in a container. The object of research

is the process of implementing a system of deploying applications in a container using Google Kubernetes Engine, Terraform and Tekton. The subject of research is the models, methods and means of implementation of the application deployment system in the container. Based on the fact that containers are self-sufficient virtual environments that contain applications and their dependencies, we have the ability to efficiently create, deploy and manage containers for various cloud applications and services, ensuring flexibility and optimal use of resources. This approach facilitates the rapid scaling of applications and ensures their unity in different cloud computing environments. Application deployment itself involves the use of interfaces and standards of interaction between parts of the system located in the cloud, which requires support for the exchange of data and resources over the network, using methods of balancing and virtualization. Therefore, it is important to create modular and independent components that are scalable and deployable separately. Also, load balancing is a method used in information systems and computer networks to distribute requests, tasks and traffic between different servers or resources to achieve the best performance, reliability and availability of the system. And virtualization is a technology that allows you to create virtual versions of computing, network, and other resources, providing abstraction between hardware resources and software or services that use them.

Key words: *cluster, container, kubernetes, docker, virtualization.*

Постановка проблеми. На даний час підприємства та організації пристосовуються до змін та забезпечують доступність своїх сервісів, залишаючись попереду в технологічних інноваціях [1]. Одним із ключових підходів для досягнення цих цілей є розгортання застосунків у хмарі, що є частиною процесу управління хмарними ресурсами хмарних обчислень [2]. Під системою розгортання застосунків у хмарі, розуміємо набір програмних компонентів і інструментів, які дозволяють організаціям розгортати застосунки, які упаковані разом з усіма залежностями в один автономний об'єкт – контейнер. Контейнери запускаються на будь-якій хмарній інфраструктурі, що підтримує контейнерні технології [3]. Така система складається з декількох формуючих її компонентів та, зазвичай, включає: контейнерний менеджер, що відповідає за запуск, зупинку, перезапуск та масштабування контейнерів; контейнерний реєстр, що дозволяє зберігати та керувати контейнерами та скрипти розгортання, які автоматизують процес розгортання застосунків у хмарі.

Аналіз останніх досліджень та публікацій. Найбільш популярною технологією керування контейнерами є Kubernetes, так як більшість інструментів розгортання контейнеризованих застосунків його підтримують. Хоча альтернативою є інші технології, такі як Docker Swarm або Mesos [3,4]. Також найбільш популярними хмарами є Amazon Web Services (AWS), Microsoft Azure і Google Cloud Platform (GCP), але інструменти розгортання контейнеризованих застосунків підтримують більшість хмарних платформ. При чому Google Kubernetes Engine (GKE) [5] є хмарним сервісом, який дозволяє розгортати та експлуатувати контейнеризовані застосунки на Google Cloud Platform та використовує Kubernetes для управління контейнеризованими застосунками. Маючи велику кількість CI/CD (Continuous Integration/Continuous Delivery) інструментів, в першу чергу орієнтуємось на ті, які є універсальними та кросплатформеними для використання на різних хмарних платформах, тому аналізуємо наявні інструменти, такі як [6, 7]: Tekton CI/CD; Jenkins; Bitbucket Pipelines та GitHub Actions [8]. Починаючи з інструментів автоматизації інфраструктури, що дозволяє створювати, змінювати та знищувати інфраструктуру в середовищах, обираємо для аналізу Terraform [6], Docker [3], Habitat [5] та Ansible [4]. Усі інструменти забезпечують автоматизацію процесу розгортання контейнеризованих застосунків у хмарі, а також підтримують масштабованість, моніторинг і логування, технологію контейнеризації та мікросервісну архітектуру [9]. Та за результатами порівняння робимо висновки,

що всі ці інструменти мають майже однаковий функціонал, а відрізняються лише у способах реалізації та інтерфейсах. Всі вони є потужними інструментами, які можуть використовуватися для створення, розгортання та управління контейнеризованими застосунками в мікросервісній архітектурі [4].

Постановка завдання. Метою роботи є дослідження методів та реалізація технологій для розгортання програм у контейнері. Об'єктом дослідження є процес реалізації системи розгортання програм у контейнері із використанням Google Kubernetes Engine, Terraform та Tekton. Предметом дослідження є моделі, методи та засоби реалізації системи розгортання програм у контейнері. Виходячи з того, що контейнери є самодостатніми віртуальними середовищами, які містять додатки та їх залежності, маємо можливість ефективно створювати, розгортати і керувати контейнерами для різних хмарних додатків і сервісів, забезпечуючи гнучкість та оптимальне використання ресурсів [4]. Цей підхід сприяє швидкому масштабуванню додатків і забезпечує їх єдність у різних середовищах хмарного обчислення.

Хмарна інфраструктура забезпечує фізичні елементи для розгортання контейнерів, включаючи сервери, мережі та сховища. Розробники та адміністратори використовують будь-яку хмарну інфраструктуру, яка підтримує контейнери, надаючи їм можливість легко масштабувати ресурси, забезпечувати гнучкість та швидкість реагування на зміни в робочих навантаженнях [10]. При цьому контейнери, як стандартизовані одиниці, працюють на будь-якому хмарному або фізичному сервері, який їх підтримує.

Виклад основного матеріалу. Реєстр контейнерів функціонує як централізований сервіс зберігання та керування контейнерами, дозволяючи використовувати вже створені контейнери та є або загальнодоступним, як Docker Hub, чи приватним, як Amazon Elastic Container Registry [11]. Контролер розгортання відповідає за керування процесом розгортання контейнерів, отримує деталі програм від розробників, планує та виконує розгортання контейнерів на хмарних ресурсах (рис. 1). Структурований та модульний підхід до розгортання контейнеризованих систем у хмарі забезпечує спрямований робочий процес, надає можливості адаптації до потреб та ефективній інтеграції з іншими компонентами системи, сприяючи підвищенню її ефективності та продуктивності, створюючи комплексне та добре взаємодіюче інфраструктурне середовище.

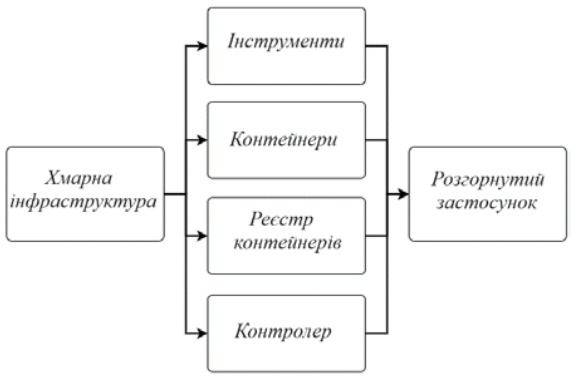

Рис. 1. Модель системи

На основі наведеної моделі побудуємо компоненти модулів, отримавши в кінцевому результаті цілісну систему. Архітектура самої системи є ефективним рішенням для розгортання та управління контейнеризованими застосунками, маючи структурований та модульний підхід (рис. 2).

Рис. 2. Модулі розгортання

Хмарною платформою є Google Cloud Platform, з необхідною вбудованою інфраструктурою, системою Google Kubernetes Engine та можливістю інтеграції з Terraform та Tekton CI/CD [12]. Тому першим етапом є реалізація скриптів для автоматизованого розгортання контейнерів та ресурсів у середовищі Google Kubernetes Engine, яке надає інфраструктуру для розгортання контейнерів Kubernetes у хмарі Google Cloud Platform.

В процесі реалізації системи маємо завдання з розгортання всього програмного забезпечення. Тому, для неперервної інтеграції та постійної доставки (CI/CD), застосовуємо фреймворк Tekton, що надає можливості автоматизувати етапи реалізації, розгортання і управління програмним забезпеченням. Тобто, за допомогою цього фреймворку розв’язуємо наступні завдання: створення контейнерних зображень; запуск тестових сценаріїв для перевірки якості програм; розгортання

програмного забезпечення у інфраструктурі; управління моніторингом та масштабуванням системи [4]. При цьому Tekton дозволяє автоматизувати виконання усіх цих завдань.

Розглянемо архітектуру проекту, який починається з репозитарію на GitHub, де зберігається весь вихідний код, конфігураційні файли та інші ресурси. Цей репозитарій є центральним елементом для спільної роботи команди та автоматизації процесу розгортання. У вигляді системи для контейнеру, як сутності, використовуємо Kubernetes у середовищі Google Kubernetes Engine, а як допоміжні інструменти застосовуємо Terraform та Tekton [4]. Текстові файли конфігурації використовують Terraform для опису інфраструктури Google Cloud Platform, такої як мережі, класи машин, а також інших ресурсів. Ці файли конфігурації зберігаємо в окремій теки у репозитарії. Automated Deployment Pipeline: Tekton використовується для створення автоматизованого конвеєра розгортання. Пайплайн Tekton включає кроки з побудови інфраструктури за допомогою Terraform, розгортання Kubernetes-кластера та завантаження контейнерів з додатком WordPress у Google Kubernetes Engine. Kubernetes Cluster використовує Kubernetes у середовищі Google Kubernetes Engine (GKE). Він відповідає за керування контейнерами, оркестрацію служб, масштабування та управління ресурсами, де керування кластером здійснюється за допомогою kubectl. WordPress Docker Image застосовує образ WordPress разом з конфігураційними файлами та іншими ресурсами, розміщений у репозитарії на GitHub та використовується під час розгортання. Kubernetes дозволяє масштабувати додаток та проводити оновлення без перерв у роботі сервісу, а Tekton допомагає автоматизувати виконання процесів. На рисунку 3 наведена архітектура реалізованої системи.

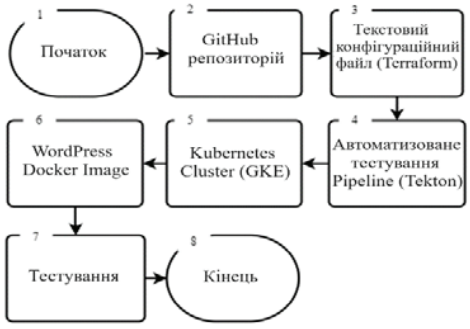

Рис. 3. Архітектура системи

При тестуванні застосовуємо тести, які перевіряють конфігурацію Kubernetes, переконуючись, що ресурси правильно налаштовані та готові до роботи. Це допомагає уникнути проблем, пов'язаних з неправильною конфігурацією під час розгортання, а інтеграція тестування з GitHub дозволяє автоматично запускати тести при кожному внесенні змін до репозитарію, що забезпечує швидке виявлення та вирішення проблем.

На першому етапі створюємо та налаштовуємо репозитарій на GitHub. Він є центральним місцем зберігання вихідного коду, конфігурацій та Docker-образів додатку, що дозволяє спільно працювати над додатком та автоматизувати процеси розгортання:

```
curl -u 'юзернейм' https://api.github.com/user/repos -d '{"name":"назва_репозитарію"}'
git clone https://github.com/юзернейм/назва_репозитарію.git
cd назва_репозитарію
```

Далі проводимо його ініціалізацію та додаємо файли README.md, .gitignore та LICENSE:

```
echo "# Project Title" >> README.md
git add README.md
git commit -m "Додано README"
git add .gitignore
git commit -m "Додано .gitignore"
git add LICENSE
git commit -m "Додано ліцензію"
```

Наступним етапом є налаштування Terraform для опису інфраструктури Google Cloud Platform, створення та керування інфраструктурними компонентами, такими як мережі, класи машин, тощо, що забезпечує автоматизоване розгортання та управління інфраструктурою додатка. Третім етапом є налаштування конвеєра розгортання за допомогою Tekton, який автоматизує розгортання додатків, включаючи етапи побудови інфраструктури, розгортання Kubernetes-кластера та завантаження додатку, забезпечуючи швидкі та стабільні цикли розробки. На четвертому етапі створюємо та конфігуруємо Kubernetes-кластер в середовищі Google Kubernetes Engine (GKE). Він відповідає за управління контейнерами, забезпечуючи їхню оркестрацію, масштабування та ефективне розподілення ресурсів. Наступним етапом є підготовка Docker-образу для додатку, який містить весь необхідний код, конфігурації та ресурси для запуску додатку та зберігається в репозитарії на GitHub. Виконуючи шостий етап маємо можливість динамічно змінювати кількість ресурсів та автоматично виконувати оновлення контейнерів, застосовуючи Kubernetes для легкого масштабування та оновлення додатка без перерв у роботі. І на останньому етапі включаємо тести, які перевіряють правильність конфігурування Kubernetes та готовність додатку до роботи. Тестування дозволяє швидко виявляти та вирішувати можливі проблеми. Завершивши всі етапи, отримаємо готову до використання систему, яка автоматизує реалізацію, розгортання та моніторинг додатку на Kubernetes.

Застосуємо тестові сценарії, які перевіряють всі етапи цього процесу, враховуючи, що автоматизація гарантує регулярне виконання тестів у стабільному середовищі. Етапи тестування при розгортанні застосунку, передбачають створення Chef InSpec тестів, які перевіряють коректне встановлення та налаштування усіх компонентів системи. При відновленні застосунку після збоїв, тестування включає створення тестів Chef InSpec для перевірки автоматичного відновлення системи. На завершальному етапі інтегруємо тести у CI/CD процес, що дозволяє автоматизувати виконання тестів при внесенні змін, забезпечуючи стабільність та надійність системи протягом її життєвого циклу (рис. 4).

Для створення файлів Chef InSpec та реалізації тестування системи, проводимо аналіз вимог до коректної роботи системи та визначаємо ключові функції, які потрібно перевіряти: конфігурація мережі, аутентифікація користувачів, налаштування безпеки, тощо. Каталог для файлів Chef InSpec створюємо у кореневому каталозі проекту. Він містить файл spec.rb з основними перевірками тестування системи:

```
describe docker_container('my-app') do
 its('image') { should eq 'my-app:latest' }
 its('ports') { should include(80) }
end
```

Далі створюємо збірку Google Cloud Build, що запускає файли Chef InSpec. Вона є автоматизованим процесом, який запускається за розкладом або подією. Тригер тестування є механізмом, який дозволяє запускати збірку Google Cloud Build за певною подією. Також на цьому етапі вказуємо репозитарій, у якому зберігаються файли Terraform. Результати тестування відображаються в консолі Google Cloud Build, у вигляді таблиці. Якщо результати тестування негативні, необхідно провести додаткове дослідження для виявлення причин несправності.

Рис. 4. Алгоритм тестування

Висновки. У роботі проведено дослідження методів та технологій для розгортання програм у контейнері. При виконанні основних задач проведені дослідження принципів та технологій для розгортання застосунків у хмарних платформах за методом контейнеризації. При аналізі враховані особливості конкретних хмарних платформ, таких як Amazon Web Services, Microsoft Azure, Google Cloud Platform, із використанням допоміжних інструментів, таких як Kubernetes, Terraform by HashiCorp, Tekton CI/CD Pipelines, Docker Image. Отримані рішення дозволяють скоротити час на проведення розгортання методом віртуалізації на віддаленому сервері та обслуговування системи. Авторами реалізована система розгортання програм із використанням Google Kubernetes Engine, Terraform та Tekton. Також в процесі реалізації системи були перевірені результати тестування, які відображаються у консолі Google Cloud Build: чи відповідає застосунок заданим вимогам, чи використовується правильний образ Docker, чи працює він на правильному порті. В цілому, такий підхід дозволяє забезпечити якість застосунку та скоротити час реалізації та впровадження.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. Petrosyan D., Astsatryan H. Serverless High-Performance Computing over Cloud. *Cybernetics and Information Technologies.* 2022. Vol. 20(3). P. 82–92.

2. Rashid A., Chaturvedi A. Cloud computing characteristics and services: brief review. *International Journal of Computer Sciences and Engineering. 2019.* Vol. 7(2). P. 421–426.

3. Киричек Г. Г., Смірнов В. В., Тягунова М. Ю. Дослідження застосування контейнерних технологій для розгортання програм на суперкомп'ютерах. *Вісник КрНУ ім. М. Остроградського*. 2023. Вип. 3 (140). С. 29–35.

4. Киричек Г.Г., Щетінін М.О. Конфігурація серверів з використанням Ansible. *Publishing House "Baltija Publishing"*. 2021. P. 15–17.

5. Randal A. Ideal versus the real: Revisiting history of virtual machines and containers. *ACM Computing Surveys (CSUR).* 2020. Vol 53 (1). P. 1–31.

6. Wankhede P., Talati M., Chinchamalatpure R. Comparative study cloud platforms-microsoft azure, google cloud platform and amazon EC2. *J. Res. Eng. Appl. Sci.* 2020. Vol 5(02). P. 60–64.

7. Dubey P., Tiwari A. K., Raja R. Amazon Web Services: the Definitive Guide for Beginners and Advanced Users. *Bentham Science Publishers*. 2023.

8. Qarkaxhija J. Using Cloud Computing як Infrastructure Case Study-Microsoft Azure. *Technium: Romanian Journal Of Applied Sciences And Technology.* 2020. Vol 2(3). P. 93–100.

9. Рудьковський О.Р., Киричек Г.Г. Програмний комплекс з підтримки розподіленої взаємодії мережевих пристроїв та додатків. *Вчені записки ТНУ ім. В.І. Вернадського. Серія «Технічні науки».* 2021. Вип.32(71). №2. С. 229–234.

10. Киричек Г.Г., Гаркуша В.Ю. Віртуалізація хостів на основі Proxmox VE в умовах надлишкового використання ресурсів. *Вчені записки ТНУ імені В.І. Вернадського. Серія «Технічні науки».* 2021. Вип. 32 (71). № 1. С. 78–84.

11. McKendrick R. Learn Ansible: automate cloud, security, and network infrastructure using ansible 2. x. *Packt Publishing Ltd*. 2018.

12. Janani K. та ін. Analysis of CI/CD Application in Architecture of Kubernetes. *Mathematical Statistician and Engineering Applications*. 2022. Vol 71(4). P. 11091–11097.

REFERENCES:

1. Petrosyan, D., Astsatryan, H. (2022). Serverless High-Performance Computing over Cloud. Cybernetics and Information Technologies, 22(3), 82-92.

2. Rashid, A., Chaturvedi, A. (2019). Cloud computing characteristics and services: a brief review. International Journal of Computer Sciences and Engineering, 7 (2), 421-426.

3. Kyrychek, H. H., Smirnov, V. V., Tiahunova, M. Yu. (2023). Doslidzhennia zastosuvannia konteinernykh tekhnolohii dlia rozghortannia prohram na superkomp'iuterakh [Research on the application of container technologies for program development on supercomputers]. Herald of KrNU named after M. Ostrogradskyi, 3 (140), 29-35 [in Ukrainian].

4. Kyrychek H.H., Shchetinin M.O. (2021). Configuring servers using Ansible. Publishing House "Baltija Publishing", 15-17.

5. Randal, A. (2020). Ideal versus the real: Revisiting history of virtual machines and containers. ACM Computing Surveys (CSUR), 53 (1), 1-31.

6. Wankhede, P., Talati, M., Chinchamalatpure, R. (2020). Comparative study of cloud platforms-microsoft azure, google cloud platform and amazon EC2. J. Res. Eng. Appl. Sci, 5 (02), 60-64.

7. Dubey, P., Tiwari, A. K., Raja, R. (2023). Amazon Web Services: the Definitive Guide for Beginners and Advanced Users. Bentham Science Publishers.

8. Qarkaxhija, J. (2020). Using Cloud Computing as an Infrastructure Case Study-Microsoft Azure. Technium: Romanian Journal Of Applied Sciences And Technology, 2(3), 93-100.

9. Rudkovskyi O.R., Kyrychek H.H. Prohramnyi kompleks z pidtrymky rozpodilenoi vzaiemodii merezhevykh prystroiv ta dodatkiv [A software complex supporting the distributed interaction of network devices and applications.]. *Scientific notes of TNU named after V.I. Vernadskyi. Series "Technical Sciences",* 2021, 32(71), 2, 229-234 [in Ukrainian].

10. Kyrychek H.H., Harkusha V.Yu. Virtualizatsiia khostiv na osnovi Proxmox VE v umovakh nadlyshkovoho vykorystannia resursiv [Virtualization of hosts based on Prox-

mox VE in conditions of excessive use of resources]. *Scientific notes of TNU named after V.I. Vernadskyi. Series "Technical Sciences"*. 2021, 32 (71), 1, 78-84 [in Ukrainian].

11. McKendrick, R. (2018). Learn Ansible: Automate Cloud, Security, and Network Infrastructure Using Ansible 2. x. Packt Publishing, Limited.

12. Janani, K., Anuhya, K., Manaswini, VL, Likitha, V., Suneetha, B., Vignesh, T. (2022). Analysis of CI/CD Application in Kubernetes Architecture. Mathematical Statistician and Engineering Applications, 71 (4), 11091-11097.