УДК 004.415.2
DOI: https://doi.org/10.53920/ITS-2023-1-5

Валерій Вікторович ЗАВГОРОДНІЙ,
доктор технічних наук, професор,
завідувач кафедри інформаційних технологій,
Державного університету інфраструктури та технологій
ORCID ID: 0000-0002-8347-7183
Ганна Анатоліївна ЗАВГОРОДНЯ,
кандидат технічних наук, доцент,
доцент кафедри інформаційних технологій,
Державний університет інфраструктури та технологій
ORCID ID: 0000-0001-8523-1761
Ігор Андрійович ЯКИМЕНКО,
магістр кафедри інформаційних технологій,
Державний університет інфраструктури та технологій
ORCID ID: 0009-0005-7631-9469
Максим Юрійович САВЧУК,
магістр кафедри інформаційних технологій,
Державний університет інфраструктури та технологій
ORCID ID: 0009-0007-0665-2332

ПРОЄКТУВАННЯ ВІРТУАЛЬНИХ СЕРВЕРІВ НА ОСНОВІ ТЕХНОЛОГІЇ КОНТЕЙНЕРИЗАЦІЇ

Було надано детальний опис архітектури проєкту, включаючи взаємодію компонентів додатка та інші аспекти проєктування хмарної платформи компанії Netcracker.

Роз'яснено поняття віртуалізації та основні типи віртуалізації, які існують на сьогоднішній день, а також обґрунтовано вибір конкретного стеку технологій для проєктування хмарної платформи компанії Netcracker. Незважаючи на те, що хмарна платформа компанії Netcracker вже працює та показує високу продуктивність, зі збільшенням кількості контейнерів зростатиме навантаження на оркестратор та кластер в цілому. Тому в даний час необхідно провести дослідження з метою виявлення слабких місць у системі та вжиття заходів для їх усунення, використовуючи нові підходи до технології контейнеризації, що постійно оновлюються новими інструментами та отримують підтримку від різних виробників рішень, базованих на віртуалізації на рівні операційної системи.

Завдяки використанню каналів передачі даних компанії Netcracker було розроблено рішення, яке характеризується високою продуктивністю, масштабованістю та легкістю внесення змін.

Були наведені ілюстрації та діаграми майбутньої хмарної платформи Netcracker, яка об'єднує різні технології для створення повноцінного веб-додатку. Також ця платформа включає технології, необхідні для створення розширеного середовища, шляхом виділення окремих елементів системи ізольованих сервісів.

Було проведено системний аналіз проєкту, вивчено архітектуру мікросервісів та технологію віртуалізації з метою розробки системи на основі мікросервісної архітектури. В результаті дослідження було встановлено, що для розробки систем, які ґрунтуються на мікросервісах, найбільш підходящою є технологія віртуалізації на рівні операційної системи.

Проаналізовано приклади абстрактних проєктів, які надають короткий опис принципу взаємодії контейнерів між собою. Виявлено, що контейнеризація є ефективним підходом для розробки таких систем, оскільки дозволяє ізолювати та масштабувати окремі компоненти системи, забезпечуючи гнучкість та надійність у взаємодії між контейнерами.

Ключові слова: *архітектура, мікросервіси, технологія віртуалізації, хмарна платформа, Netcracker, контейнеризація.*

Valerii ZAVGORODNII
Doctor of technical sciences, Professor,
Head of the Department of Information Technologies,
State University of Infrastructure and Technologies
Anna ZAVGORODNYA
Candidate of technical sciences, Associate Professor,
Associate Professor of the Department of Information Technologies,
State University of Infrastructure and Technologies
Ihor YAKYMENKO
Master of the Department of Information Technologies,
State University of Infrastructure and Technologies
Maksym SAVCHUK
Master of the Department of Information Technologies,
State University of Infrastructure and Technologies

DESIGN OF VIRTUAL SERVERS BASED ON CONTAINERIZATION TECHNOLOGY

A detailed description of the project architecture was provided, including the interaction of the application components and other aspects of Netcracker's cloud platform design.

The concept of virtualization and the main types of virtualization that exist today are explained, as well as the choice of a specific stack of technologies for the design of the cloud platform of the Netcracker company is justified. Despite the fact that Netcracker's cloud platform is already working and showing high performance, with the increase in the number of containers, the load on the orchestrator and the cluster as a whole will increase. Therefore, it is currently necessary to conduct research in order to identify weak points in the system and take measures to eliminate them, using new approaches to containerization technology, which are constantly updated with new tools and receive support from various manufacturers of solutions based on virtualization at the level of the operating system.

Thanks to the use of Netcracker's data channels, a solution has been developed that is characterized by high performance, scalability and ease of modification.

Illustrations and diagrams of the upcoming Netcracker cloud platform, which combines various technologies to create a full-fledged web application, were shown. Also, this platform includes the technologies necessary to create an extended environment by separating individual elements of the system of isolated services.

System analysis of the project was carried out, microservices architecture and virtualization technology were studied in order to develop a system based on microservices architecture. As a result of the research, it was established that for the development of systems based on microservices, virtualization technology at the level of the operating system is the most suitable.

Examples of abstract projects that provide a brief description of the principle of interaction between containers have been analyzed. Containerization has been found to be an effective approach for the development of such systems, as it allows isolation and scaling of individual system components, providing flexibility and reliability in the interaction between containers.

Keywords: *architecture, microservices, virtualization technology, cloud platform, Netcracker, containerization.*

Постановка проблеми. У сучасному світі багато компаній, що займаються розробкою програмного забезпечення, використовують віртуалізацію для створення масштабованих, надійних та повнофункціональних систем. Ці системи також є економічно вигідними і конкурентоспроможними у швидкозростаючій бізнес-моделі, заснованій на хмарних сервісах [1, 2].

Актуальність і наукова значимість даного дослідження полягають у тому, що в наш час, особливо враховуючи епідеміологічну ситуацію, зростає попит на онлайн-бізнес. Це призводить до значного збільшення користувачів як на нових, так і на існуючих системах. У даній роботі проводяться дослідження різних технологій віртуалізації та архітектур для розробки програмного забезпечення, що відповідають вимогам високої надійності, продуктивності та гнучкості розгортання.

Головна проблема полягає в пошуку потрібної технології, яка змогла б вирішити низку питань, що виникли через епідеміологічну ситуацію у світі для компаній, що займаються інтернет-бізнесом. Факт полягає в тому, що з наведених причин велика кількість користувачів почала робити покупки або здійснювати дії в цифровому сегменті, що призвело до перевантаження систем. З метою вирішення цієї проблеми різні компанії почали масштабувати своє апаратне забезпечення, що призвело до значних фінансових витрат. Як результат, виникла суперечка щодо проєктування систем у багатьох компаній, і тому виникло питання про більш ефективне використання технологій з метою зниження витрат.

Аналіз останніх досліджень і публікацій. Згідно з [3], віртуалізація відіграє важливу роль у технології хмарних обчислень. У хмарних обчисленнях користувачі зазвичай обмінюються даними, що містяться в додатках, розгорнутих на хмарних платформах. «Хмара», або хмарна обчислювальна система, представляє собою онлайн-сховище, що складається з великої кількості серверів, які об'єднані в одну загальну мережу. Доступ до цього сховища здійснюється за допомогою спеціальних веб-сайтів або додатків. Але насправді, завдяки використанню віртуалізації, дана інфраструктура розподіляється для досягнення гнучкості і стабільності системи.

Згідно з дослідженням [4], усі облаштовані на облікованій платформі хмарного хостингу ресурси повинні ефективно розподілятися між численними користувачами з метою забезпечення

безпеки додатків, що є важливими згідно вимог замовника. Такий підхід передбачає спільне використання ресурсів, таких як обчислювальна потужність, оперативна пам'ять та дисковий простір, на загальній фізичній інфраструктурі у форматі багатокористувацького середовища.

За допомогою розвитку хмарних технологій, розгортання мікросервісних систем стало більш ефективним, гнучким і економічно вигідним підходом. Проте в [5] відзначається, що мікросервіси є складною темою, яку з особливим інтересом вивчають академічні установи, IT-компанії та промисловість. Вперше термін «мікросервіси» обговорювався на конференції «Архітектура програмного забезпечення» у травні 2011 року з метою узгодження загального архітектурного стилю серед учасників. Через рік, на цій же конференції, термін «мікросервіси» був офіційно підтверджений інженерною групою. Фактично, мікросервісна архітектура була розроблена як реакція на проблеми, що виникають у монолітних додатках, які мають обмежену масштабованість та гнучкість [6, 7].

Монолітна архітектура відноситься до способу побудови програмного додатка, де модулі не можуть працювати незалежно один від одного. Однак розробка, ґрунтована на мікросервісній архітектурі, розглядається як єдиний спосіб виконувати незалежні інструкції, що не взаємодіють між собою [8].

За словами Річардсона, мікросервісна архітектура представляє собою варіант сервісно-орієнтованої архітектури програмного забезпечення, що акцентує на взаємодії невеликих, слабо зв'язаних та легко змінюваних модулів, відомих як мікросервіси. Цей підхід став широко розповсюдженим у середині 2010-х років завдяки розвитку гнучких методик розробки та DevOps [9].

На відміну від традиційних варіантів сервісно-орієнтованої архітектури, де модулі можуть бути складними програмними системами, а взаємодія між ними зазвичай ґрунтується на стандартизованих тяжких протоколах (наприклад, SOAP, XML-RPC), мікросервісна архітектура ґрунтується на компонентах, що виконують прості функції, та використовує ефективні мережеві комунікаційні протоколи (наприклад, REST з використанням JSON, Protocol Buffers, Thrift) для взаємодії між ними [10].

Зважаючи на необхідність ізоляції різних компонентів додатка та підтримки модульної незалежності при використанні техно-

логії мікросервісів, інженерна спільнота стикалася з проблемою вибору відповідної технології для досягнення цієї мети [8, 10]. Таким рішенням стала технологія віртуалізації.

Використання технології контейнерної віртуалізації разом із мікросервісною архітектурою дозволяє розробляти масштабні веб-додатки, що задовольняють вимогам високої продуктивності, надійності та гнучкості системи.

Мета статті – дослідження можливостей технологій контейнерної віртуалізації та мікросервісної архітектури для розробки хмарної платформи, що вирішує проблеми високої продуктивності, гнучкості розгортання та надійності.

Виклад основного матеріалу дослідження. Технологія віртуалізації є основною складовою хмарних обчислень. У традиційному підході до хмарних обчислень використовуються віртуальні машини для розподілу ресурсів та створення ізольованого середовища для користувачів. На одному фізичному сервері в хмарній інфраструктурі можуть бути розгорнуті та запущені кілька віртуальних машин з власною операційною системою та службами. Проте, останнім часом все більш популярною стає легковажна технологія віртуалізації на основі контейнерів. Основна відмінність між віртуальними машинами та контейнерами полягає в тому, що контейнери використовують одну загальну базову операційну систему.

Мікросервісна архітектура з використанням контейнерної віртуалізації широко прийнята як основний інструмент для розробки великих додатків. Ця концепція отримала значну підтримку від спільноти Інтернету, що привело до потреби використання цієї технології при розробці нових хмарних рішень, зокрема в сфері електронної комерції. Запит на такі рішення виник серед замовників разом із розширенням хмарних обчислень. У зв'язку з цим компанія Netcracker прийняла рішення розробити хмарну платформу, яка надасть замовникам гнучку, масштабовану та надійну систему, засновану на мікросервісній архітектурі та супутніх технологіях.

Розвиток хмарних технологій та технологій віртуалізації активізує роботу компаній, що спеціалізуються на наданні програмних рішень на основі хмарних обчислень. Однією з таких компаній є Netcracker.

Netcracker Technology є підрозділом корпорації NEC, спеціалізуються на розробці, впровадженні та підтримці систем опера-

тивної підтримки (OSS), систем підтримки бізнесу (BSS) і рішень SDN/NFV для операторів зв'язку, великих підприємств і державних установ.

Протягом багатьох років рішення Netcracker BSS і OSS не були масштабованими і стали занадто складними для подальшого використання, що обмежувало їх конкурентоспроможність. У зв'язку з великим ризиком масштабних трансформацій системи, постачальники послуг зараз частіше надають перевагу тому, щоб почати з невеликої системи та поетапно розширювати її під свої потреби.

Використання хмарних технологій стає необхідним кроком для забезпечення бізнесу та операцій наступного покоління, оскільки це сприяє збільшенню гнучкості, безпеки, надійності та масштабованості. Netcracker пропонує власну хмарну платформу, відому як Netcracker Cloud Platform, яка дозволяє клієнтам ефективно керувати своїм бізнесом та операціями, використовуючи переваги хмарних технологій.

Розділення додатків на мікросервіси

З метою визначення необхідних мікросервісів для хмарної платформи компанії Netcracker було проведено аналіз у співпраці з замовником. Основною метою аналізу було зібрати критерії та узгодити потреби замовника з можливостями платформи. Були виокремлені такі завдання:

1. Розробка платформи, яка дозволить користувачам купувати хмарні сервіси з можливістю підписки. Для досягнення цієї мети необхідно створити портал для цифрових сервісів нового покоління, хмарних додатків і віртуалізованих сервісів. Основна функція цього порталу полягатиме в забезпеченні взаємодії з кінцевими користувачами. Для нього пропонується назва «Shopping Portal» або «інтернет-магазин».

2. Розробка порталу для адміністрування інтернет-магазину, який буде відповідальним за редагування, видалення та додавання нових продуктів, а також за всі взаємодії з користувачами та налаштування інтернет-магазину. Цей портал отримав назву «Admin Portal» або «портал для адміністрування інтернет-магазину».

У цьому випадку, використання віртуальних серверів дозволить забезпечити швидку та безпечну роботу з порталами. Шляхом розбиття додатку на окремі сервіси, які незалежно виконують

свої функції, можна досягти більшої ефективності. Це дозволить легко та швидко редагувати та оновлювати окремі частини додатку без зайвих труднощів.

Візьмемо, наприклад, модель взаємодії між користувачем інтернет-магазину та динамічним оновленням списку продуктів після внесення змін через портал адміністрування. Обидва ці портали вже є окремими мікросервісами, які будуть мати власні контейнери та віртуальні сервери. Поділ додатка на такі окремі частини спростить взаємодію між порталами та дозволить виділити загальну логіку в окремий контейнер (рис. 1).

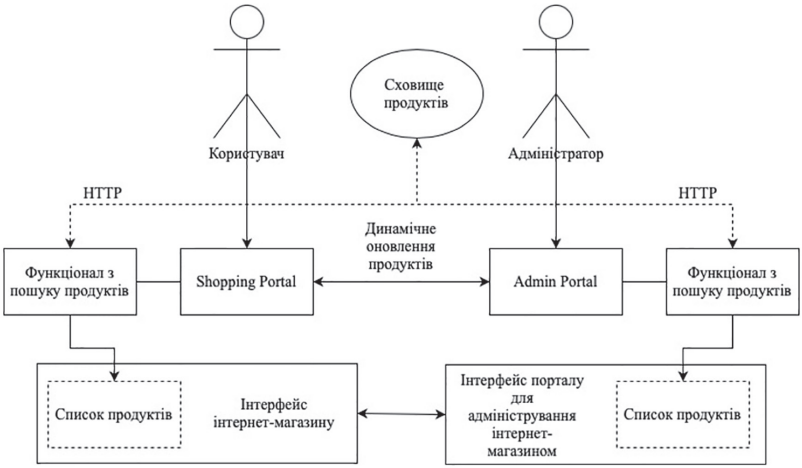

Рис. 1. Клієнтська модель взаємодії користувача з інтернет-магазином та динамічне оновлення списку продуктів

Як бачимо на рисунку 1, інтернет-магазин та портал адміністрування мають обидва відображати список продуктів. Це вимагає реалізації функції отримання продуктів з бази даних на одному з порталів, що створює певні труднощі, оскільки цей функціонал повинен бути доступним для використання на інших порталах. Щоб вирішити цю проблему, ми можемо скористатися можливостями мікросервісної архітектури та контейнеризації.

Для цього ми виділимо всю необхідну логіку в окремий мікросервіс, який буде працювати у власному контейнері. Мікросервіс

матиме свої власні змінні середовища та може мати власну базу даних або використовувати базу даних інших мікросервісів. Таке розподілення функціоналу дозволить нам забезпечити незалежність цієї функціональності від конкретного порталу і спростить її використання на інших платформах.

Таким способом досягається відокремлення логіки роботи з продуктами від загального сервісу. На рисунку 2 показано демонстрацію цього сервісу.

Новий мікросервіс отримає назву «Product Storage» і включатиме наступні функціональні завдання:

1. Зберігання продуктів у базі даних та можливість змінювати, додавати та видаляти продукти зі списку.

2. Надання відкритих посилань для взаємодії із продуктами через протокол HTTP з іншими мікросервісами.

3. Універсальність функціоналу, що підходитиме для різних порталів.

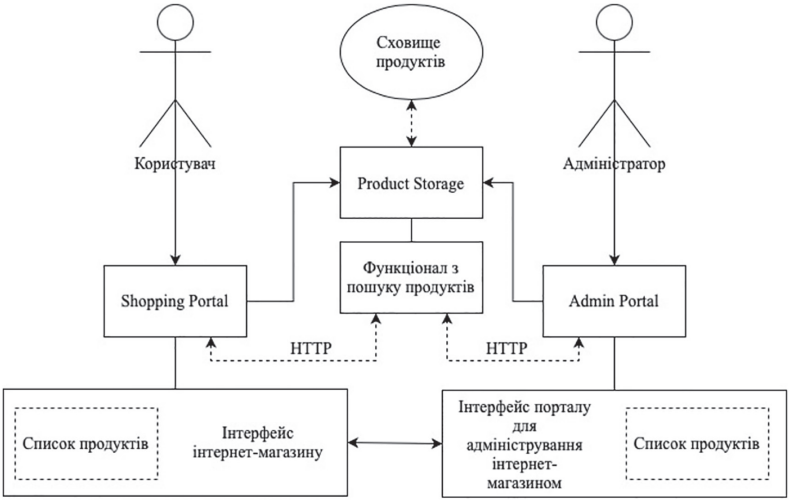

Рис. 2. Виділення функціоналу для отримання списку продуктів на окремий сервіс

У даному випадку, технологія контейнеризації грає пряму роль, оскільки для перевірки функціоналу необхідно оновлення версій контейнерних образів.

На рисунку 3 відображений реальний досвід роботи з контейнерами та представлено початкову схему розробки хмарної платформи компанії Netcracker.

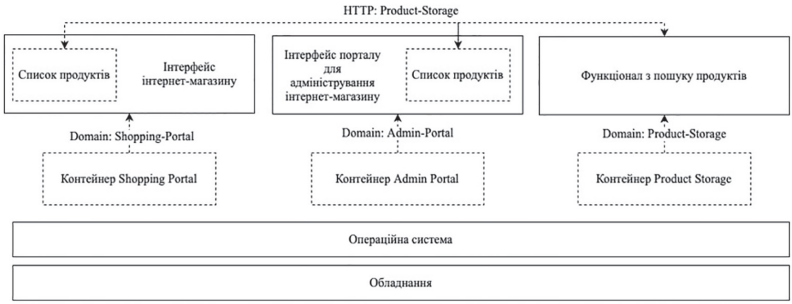

Рис. 3. Логічна схема контейнерів та встановлених на них додатків

Ключові моменти даної схеми:

1. Для кожного додатку виділено свій контейнер, на якому запущений додаток. Додаток може мати кілька сутностей для забезпечення надійності, при цьому кожен контейнер незалежний один від одного та керується контейнерним рушієм.

2. Кожен контейнер має свій домен, що відповідає концепції доменного управління для проєктування мікросервісів. Обмежений контекст явно визначає межі моделі.

3. Взаємодія між додатками та сервісами відбувається за протоколом HTTP.

Перевага такої узгодженості в різних додатках полягає у наявності стандарту при виконанні різних дій.

Сервіси авторизації та механізм виявлення служб

Для створення хмарної платформи Netcracker необхідно побудувати таку архітектуру проєкту, щоб мікросервіси взаємодіяли між собою та утворювали компактні компоненти додатка, які разом створюють стійку екосистему. Розглянемо певну частину сервісів, що складають основну частину хмарної платформи Netcracker, а також їх взаємодію, починаючи з сервісу авторизації.

Сервіс авторизації використовує токен для аутентифікації користувача і не зберігає статус користувача. Клієнт повинен надсилати токен на сервер для аутентифікації кожного разу при запиті.

Основний процес аутентифікації користувача з використанням токена зображений на рисунку 4.

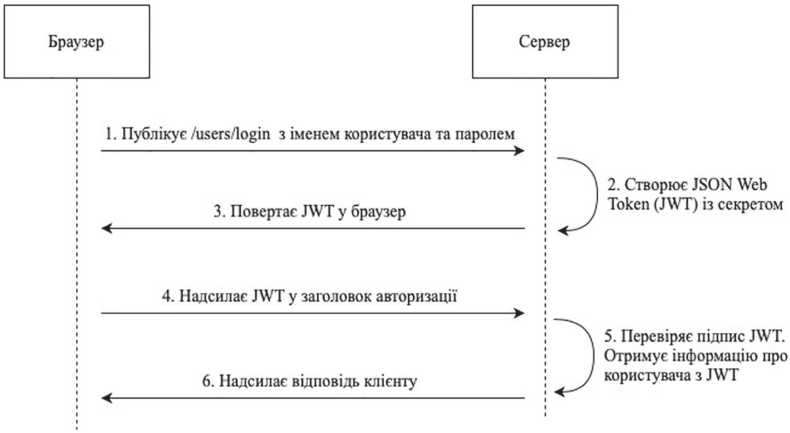

Рис. 4. Процес аутентифікації користувача в режимі токена

Процес аутентифікації користувача аналогічний базовому процесу аутентифікації з токеном, з додаванням шлюзу API як вхідної точки зовнішнього запиту. Цей сценарій передбачає, що всі запити проходять через шлюз API, що ефективно приховує мікросервіси.

З точки зору продуктивності системи порівняно з монолітною системою, такий підхід є вигідним, оскільки сервіси авторизації працюють через механізм виявлення сервісів шлюз API, що є правильним та безпечним з архітектурної точки зору. Щодо шлюзу API, за допомогою технології контейнеризації можна створити кілька екземплярів цього сервісу для збільшення пропускної здатності системи, оскільки більшість запитів будуть направлені саме до цього сервісу.

Модель взаємодії мікросервісів у хмарній платформі, що використовує канали передачі даних компанії Netcracker

Монолітний додаток взаємодіє між своїми компонентами шляхом виклику методів або функцій на мовному рівні. Натомість, мікросервісна архітектура полягає у розподіленій системі, що працює на кількох віртуальних машинах. Зазвичай, кожен мікро-

сервіс є окремим процесом. Тому для взаємодії між сервісами необхідно використовувати механізми міжпроцесової взаємодії (IPC).

Кожен сервіс може використовувати різні методи взаємодії. Деяким сервісам достатньо одного механізму IPC, тоді як інші можуть використовувати комбінацію різних механізмів. На прикладі інтернет-магазину (рис. 5), сервіси взаємодіють за допомогою сповіщень, запитів/відповідей та публікацій/підписок, коли користувач додає товар до кошика і отримує сповіщення.

Рис. 5. Взаємодія сервісів між собою при додаванні товару в кошик та виведення повідомлення

Сервіси використовують комбінацію сповіщень, запитів/відповідей та публікацій/підписок. Наприклад, коли користувач взаємодіє з інтернет-магазином і надсилає запит до сервісу кошика, сервіс «Портал покупок» передає запит через «Шлюз API» до сервісу, який відповідає за кошик. Після отримання запиту, сервіс «Кошик покупок» звертається до «Сховища цінових пропозицій», щоб зберегти товар у кошику та перевірити його валідність. У той же час «Кошик покупок» очікує відповіді від сервісу «Сховище цінових пропозицій» і, коли вона надходить, виводить сповіщення за допомогою сервісу «Сервіс сповіщень».

Під час використання обміну повідомленнями, процеси взаємодіють асинхронно шляхом передачі повідомлень. Клієнт надсилає запит до сервісу, відправляючи йому повідомлення. Якщо передбачається отримання відповіді від сервісу, він відправляє

окреме повідомлення з відповіддю клієнту. Оскільки комунікація є асинхронною, клієнт не блокується під час очікування відповіді. Замість цього клієнт очікує відповіді, припускаючи, що вона не буде надіслана негайно.

Висновки та пропозиції. Науковий характер даного дослідження полягає у тому, що хмарна платформа компанії Netcracker має здатність не лише надавати клієнтам рішення щодо продажу цифрових продуктів, але й гнучку можливість створення веб-додатків, які можуть бути швидко адаптовані до потреб певної галузі бізнесу і оперативно введені в експлуатацію.

Використання можливостей віртуалізації дозволяє створювати й підтримувати великі та складні системи, які характеризуються високою гнучкістю, ефективністю, зручністю у супроводженні та високою продуктивністю. В даному контексті віртуалізація означає запуск декількох операційних систем на одному комп'ютері, з використанням спільних апаратних ресурсів.

Даний підхід дозволяє покращити не лише кодову базу, а й сприяє підвищенню надійності та масштабованості системи. Досягнення цих поліпшень забезпечується принципом ізольованості відмов: мікросервіси повинні бути здатні до самостійного розгортання та роботи. Без цього принципу, хмарна платформа Netcracker втратить свої основні критерії – масштабованість та стійкість до відмов.

Практична розробка хмарної платформи на основі контейнерної віртуалізації та мікросервісної архітектури має значущість для комерційних організацій, що потребують розробки стійкої та гнучкої системи з можливістю масштабування.

ЛІТЕРАТУРА

1. Priyanka Tyagi. E Commerce for Entrepreneurs. BPB Publications, 2020. P. 168. ISBN 9789389898408.

2. Dan Croxen-John, Johann van Tonder. E-Commerce Website Optimization. Kogan Page, 2020. P. 272. ISBN 9781789664430.

3. Plauth, M., Feinbube, L., Polze, A. A Performance Survey of Lightweight Virtualization Techniques. In: De Paoli, F., Schulte, S., Broch Johnsen, E. (eds) Service-Oriented and Cloud Computing. ESOCC 2017. Lecture Notes in Com-

puter Science(), vol 10465. Springer, Cham. https://doi.org/10.1007/978-3-319-67262-5_3.

4. J. Bogner, J. Fritzsch, S. Wagner and A. Zimmermann. Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality, 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), Hamburg, Germany, 2019, pp. 187-195, doi: 10.1109/ICSA-C.2019.00041.

5. Xia, C., Zhang, Y., Wang, L., Coleman, S., & Liu, Y. Microservice-based cloud robotics system for intelligent space. Robotics and Autonomous Systems, 2018, 110, 139-150. https://doi.org/10.1016/j.robot.2018.10.001.

6. Chris Richardson. Inter-Process Communication in a Microservices Architecture. URL: https://dzone.com/articles/building-microservices-inter-process-communication-2 (дата звернення: 30.05.2023).

7. Kalske, M., Mäkitalo, N., Mikkonen, T. (2018). Challenges When Moving from Monolith to Microservice Architecture. In: Garrigós, I., Wimmer, M. (eds) Current Trends in Web Engineering. ICWE 2017. Lecture Notes in Computer Science(), vol 10544. Springer, Cham. https://doi.org/10.1007/978-3-319-74433-9_3.

8. Chris Richardson. Microservices Patterns. Manning Publications, 2018. P.520. ISBN 9781617294549.

9. Julien Vehent. Securing DevOps. Manning Publications, 2018. P.520. ISBN 9781617294136.

10. Shayank Jain. Designing Microservices using Django. BPB Publications, 2020. P.332. ISBN 9789389328790.

REFERENCES

1. Tyagi, P. (2020), «E Commerce for Entrepreneurs». BPB Publications. P. 168. ISBN 9789389898408.

2. Dan Croxen-John, Johann van Tonder (2020), «E-Commerce Website Optimization». Kogan Page. P. 272. ISBN 9781789664430.

3. Plauth, M., Feinbube, L., Polze, A. (2017), A Performance Survey of Lightweight Virtualization Techniques. In: De Paoli, F., Schulte, S., Broch Johnsen, E. (eds) Service-Oriented and Cloud Computing. ESOCC 2017. Lecture Notes in Computer Science(), vol 10465. Springer, Cham. https://doi.org/10.1007/978-3-319-67262-5_3.

4. J. Bogner, J. Fritzsch, S. Wagner and A. Zimmermann (2019), «Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality», 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), Hamburg, Germany, P. 187-195, doi: 10.1109/ICSA-C.2019.00041.

5. Xia, C., Zhang, Y., Wang, L., Coleman, S., & Liu, Y. (2018). Microservice-based cloud robotics system for intelligent space. Robotics and Autonomous Systems, 110, 139 – 150. https://doi.org/10.1016/j.robot.2018.10.001.

6. Richardson, C. (2016), «Inter-Process Communication in a Microservices Architecture», available at: https://dzone.com/articles/building-microservices-inter-process-communication-2 (Accessed 30 May 2023).

7. Kalske, M., Mäkitalo, N., Mikkonen, T. (2018). Challenges When Moving from Monolith to Microservice Architecture. In: Garrigós, I., Wimmer, M. (eds) Current Trends in Web Engineering. ICWE 2017. Lecture Notes in Computer Science, vol 10544. Springer, Cham. https://doi.org/10.1007/978-3-319-74433-9_3.

8. Richardson, C. (2018), «Microservices Patterns», Manning Publications. P.520. ISBN 9781617294549.

9. Vehent, J. (2018), «Securing DevOps», Manning Publications. P. 520. ISBN 9781617294136.

10. Jain, S. (2020), «Designing Microservices using Django», BPB Publications. P. 332. ISBN 9789389328790.

СТАТТЯ НАДІЙШЛА ДО РЕДАКЦІЇ 29.05.2023