УДК 04.416.6
DOI: https://doi.org/10.36994/2788-5518-2024-01-07-07

АРХІТЕКТУРА ЕФЕКТИВНОЇ CI/CD-СИСТЕМИ ДЛЯ ПРОГРАМНИХ РІШЕНЬ, ЗАСНОВАНИХ НА MSA

Загорулько А. В., *Відкритий міжнародний університет розвитку людини «Україна», Київ, Україна. https://orcid.org/0009-0004-9478-5642, azago85@gmail.com*

Павленко В. І., *к. ф.-м. наук, Відкритий міжнародний університет розвитку людини «Україна», Київ, Україна. https://orcid.org/0000-0002-3958-0415, pavlenko.v@i.ua*

Анотація. *Мікросервісна архітектура (MSA) здобула популярність завдяки своїй гнучкості, масштабованості та ефективності, але перехід на неї є складним та ресурсоємним процесом. Основні виклики включають зростання кількості сервісів та репозиторіїв коду, інфраструктурні витрати, підтримку конфігурацій для різних середовищ, керування залежностями між сервісами та використання різних платформ.*

У дослідженні пропонується архітектура ефективної системи неперервної збірки та розгортання (CI/CD), що вирішує зазначені проблеми і покращує весь цикл розроблення програмного забезпечення (SDLC). Основні принципи запропонованої архітектури включають модульність, використання відкритого набору специфікацій (API) та формування потоку подій. Модульність дає змогу виділити компоненти відповідно до задач SDLC, відкритий API забезпечує гнучкість інтеграції та оновлення, а потік подій забезпечує інформацію для аналізу повного SDLC продукту, створення метрик, системи реагування на надзвичайні ситуації та прозорого моніторингу. Саме потік подій є ключовою особливістю, що виокремлює запропоновану архітектуру ефективної CI/CD-системи серед наявних рішень управління програмними рішеннями, заснованими на MSA.

Кожен модуль CI/CD системи генерує події, які акумулюються для статистичного аналізу та побудови кореляційних моделей із використанням методів машинного навчання. Це дає змогу отримувати точні відповіді на системні питання, такі як вплив якості вимог на строки поставки продукту, відповідність архітектури системи вимогам, пошук причин помилок та вплив додаткових ресурсів на продуктивність системи.

Запропонована архітектура CI/CD-системи підвищує швидкість і якість розроблення, спрощуючи процеси оновлення та міграції на нову інфраструктуру, дає можливість аналізувати вплив змін на будь-якому етапі SDLC-продукту на його продуктивність, забезпечуючи конкурентні переваги на ринку розроблення програмного забезпечення.

Ключові слова: *мікросервіс, CI, CD, розгортання, машинне навчання, ML, SDLC.*

ARCHITECTURE OF AN EFFICIENT CI/CD SYSTEM FOR SOFTWARE SOLUTIONS BASED ON MSA

Anton Zahorulko, *Open International University of Human Development* «Ukraine», Kyiv, Ukraine. *https://orcid.org/0009-0004-9478-5642, azago85@gmail.com*

Volodymyr Pavlenko, *Ph.D., Open International University of Human Development «Ukraine», Kyiv, Ukraine. https://orcid.org/0000-0002-3958-0415, pavlenko.v@i.ua*

Abstract. *Microservices architecture (MSA) has gained popularity due to its flexibility, scalability, and efficiency, but the migration to it is a complex and resource-intensive process. The main challenges include the increase in the number of services and code repositories, infrastructure costs, maintenance of configurations for different environments, management of dependencies between services, and the use of various platforms.*

This study proposes an architecture for an efficient continuous integration and deployment (CI/CD) system that addresses these issues and enhances the entire software development lifecycle (SDLC). The core principles of the proposed architecture include modularity, the use of an open set of specifications (API), and the formation of an event stream. Modularity allows the separation of components according to SDLC tasks, an open API ensures flexible integration and updates, and the event stream provides information for the analysis of the complete SDLC of a product, creation of metrics, emergency response systems, and transparent monitoring. The event stream is the key feature that distinguishes the proposed architecture of an efficient CI/CD system among existing solutions for managing MSA based software.

Each module of the CI/CD system generates events that are accumulated for statistical analysis and as the source to build correlation models using machine learning methods (ML). Such an approach allows one to obtain an accurate answer to the complex questions, such as the impact of requirements quality on product delivery timelines, the compliance of system architecture with requirements, identification of error causes, the impact of additional resources on system performance, etc.

The proposed CI/CD system architecture increases development speed and quality, simplifies the software product update and migration to new infrastructure stack, and enables the analysis of the impact of changes at any stage of the product's SDLC on its performance, providing competitive advantages in the software development market.

Key words: *microservice, CI, CD, deployment, machine learning, ML, SDLC.*

Вступ

Застосування мікросервісної архітектурі (MSA) є досить поширеною практикою для побудови програмних рішень в останні десятиліття. Такий успіх пов'язаний із тими перевагами, що надає використання MSA [1; 11; 12], основними з них є:

− розширені можливості масштабування;
− вищий рівень ізоляції помилок;
− підвищення ефективності розроблення продукту;
− підвищення гнучкості та швидкості розгортання продукту;
− збільшення економічної ефективності у цілому.

Упровадження MSA складається з двох складників:

− зміни архітектури продукту відповідно до принципів та практик MSA;
− перебудови циклу розроблення програмного забезпечення (SDLC) та CI/CD відповідно до архітектурних змін продукту.

Досить часто рівень складності другого складника є недооціненим або взагалі втрачається з поля зору під час проєктування та оцінки ресурсів за переходу на MSA.

Для оцінки кількісних і якісних змін, необхідних для адаптації наявних SDLC та CI/CD, варто врахувати такі чинники:

− збільшення кількості незалежних компонентів (сервісів) продукту, відповідно, і репозиторіїв вихідного коду;
− ускладнення процесу інтеграції та розгортання продукту;
− необхідність підтримувати конфігурацію кожного сервісу на кожному оточенні розгортання окремо;
− підвищення витрат на управління спільними бібліотеками вихідного коду та спільною інфраструктурою розгортання;
− необхідність поєднання в одному продукті компонентів різних інфраструктурних стеків (AWS-лямбд, Kubernetes-сервісів) [2].

Переважна більшість наявних CI/CD-систем має на меті автоматизувати рутинні операції для підвищення надійності та швидкості впровадження продукту у виробництво [7; 8], але вони лишають поза увагою інші частини SDLC, такі як розроблення вимог, проєктування, планування та моніторинг системи у виробництві. Для прикладу, буває досить важко локалізувати причину некоректної роботи системи, оскільки треба проаналізувати всю послідовність викликів сервісів, беручи до уваги, що помилка в одному сервісі може спричинити цілий каскад помилок у клієнтських сервісах [9]. Із погляду вимог проєктування та планування корисно мати змогу точної оцінки обсягів роботи та необхідних ресурсів, спираючись на вже розроблені сервіси та кількість ресурсів, витрачених на їх підтримку у виробництві.

Виконання досліджень

Розроблення архітектури ефективної CI/CD-системи

Мета цього дослідження – розробити ефективну CI/CD-систему, що дає змогу подолати проблеми, породжені особливостями MSA [3], та надати нові можливості, що якісно вплинуть на весь цикл SDLC.

Серед **функціональних вимог** до ефективної CI/CD системи можна виділити такі:

− інтеграція та розгортання продукту не залежить або має логарифмічну залежність відповідно до кількості його окремих складників;
− повністю автоматизована інтеграція та розгортання продукту на всіх оточеннях;
− моніторинг усіх фаз SDLC-продукту;
− прогнозований календар оновлень продукту;
− незалежність архітектури та технологічного стеку кожного окремого компонента від інших.

Поточне дослідження пропонує взяти за основу **архітектури** ефективної CI/CD-системи три основні складники: модульність, базування на відкритому набору специфікацій (API), формування потоку подій [3].

Модульність має на меті виділити компоненти CI/CD системи відповідно до фази SDLC та конкретної функціональності, що забезпечується модулем.

Рис. 1. Модульна структура ефективної CI/CD-системи

Підтримка відкритого **API** для кожного модуля дає можливість:

− розробляти модулі системи різними виробниками;

− полегшувати впровадження системи за рахунок можливості комбінування реалізації модулів під вимоги конкретної системи;

− незалежно оновлювати окремі компоненти системи.

Потік подій є ключовою особливістю архітектури ефективної CI/CD-системи. Його основне завдання – агрегувати в одному місці всі події, що відбуваються в системі та охоплюють увесь SDLC продукту – від створення вимог до функціонування системи у виробництві [3–6].

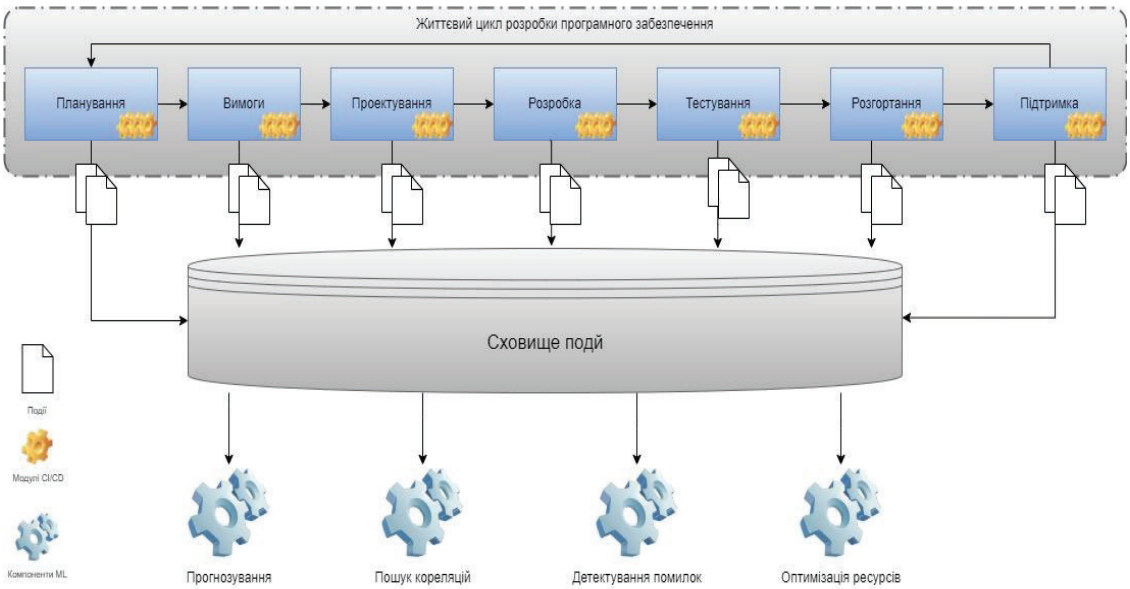

Рис. 2. Архітектура ефективної системи CI/CD

Кожен модуль системи продукує свій набір подій відповідно до API-модуля. Акумулюючи події від усіх модулів CI/CD-системи, пропонується формувати джерело достовірної інформації для статистичного аналізу даних, побудови кореляційних моделей із залученням методів машинного навчання [3].

Модель даних подій кожного компонента системи є фундаментом для подальшого аналізу стану системи та має відповідати таким вимогам:

− містити необхідну і достатню інформацію для аналізу роботи компонента;
− містити посилання на попередні події, що безпосередньо призвели до поточної події, таким чином зв'язуючи події між собою;
− містити унікальний ідентифікатор події;
− містити інформацію для забезпечення дедуплікації подій.

Запропонована архітектура дає змогу отримувати досить точні відповіді на питання, у яких можна поєднувати фази SDLC, команди розробки, ресурси оточення та роботу множини сервісів продукту у виробництві. Такий аналіз є недосяжним для поточних архітектур систем CI/CD. Можна навести приклади завдань, що вирішуються запропонованою архітектурою:

− вплив якості вимог на строки поставки системи у виробництво, якість роботи системи у виробництві та обсяг додаткових зусиль у фазі підтримки;
− відповідність архітектури системи поставленим вимогам, як то виявлення взаємозалежних сервісів, сервісів із великою кількістю помилок, вузьких місць системи, що впливають на її продуктивність;
− вплив термінів поставки системи у виробництво на якість її роботи;
− пошук функціоналу, що викликає більшість помилок у виробництві, та причини їх виникнення, ураховуючи всі фази SDLC та задіяні ресурси;
− аналіз впливу додаткового виділення обчислювальних ресурсів на продуктивність системи.

Подальший розвиток наведеної у даному дослідженні архітектури вбачається в додаванні зворотного зв'язку між модулями машинного навчання та системою, таким чином можна автоматично вирішувати знайдені проблеми без залучення оператора, наприклад:

− розгортати або відкочувати нову версію конкретного сервісу;
− оновлювати версії бібліотек та платформ, що використовуються у продукті;
− керувати строками поставки у виробництво, ураховуючи залежності між сервісами, командами розробки, змінами вимог тощо;
− керувати розгортанням або згортанням апаратних ресурсів виробництва відповідно до навантаження та вимог до продуктивності системи.

Висновки

Запропонована архітектура ефективної системи CI/CD для програмних продуктів, заснованих на архітектурі MSA, спрямована на вирішення не лише поточних проблем упровадження та функціонування MSA-систем, а й на суттєве розширення можливостей контролю та прогнозування процесів розроблення та підтримки таких систем у виробництві.

Створення масиву даних за допомогою потоку подій забезпечує основу для аналізу минулих, поточних та майбутніх станів продукту, використовуючи методи статистичного аналізу, прогнозування та машинного навчання.

Ці вдосконалення мають значно розширити та переосмислити завдання систем CI/CD, інтегруючи технології машинного навчання, а в майбутньому й інші технології штучного інтелекту для підвищення ефективності SDLC та загальної якості програмних рішень.

Література

1. Newman S., Building Microservices, 2nd Edition / Sam Newman. – O'Reilly Media, Inc., 2021. 617 p.
2. Davis A., Bootstrapping Microservices with Docker, Kubernetes, and Terraform / Ashley Davis. New York, USA: Manning Publications, 2021. 440 p.
3. Eramo R., Muttilo V., Berarrdinelli L., Bruneliere H., AIDOaRt: AI-augmented Automation for DevOps, a Model-based Framework for Continuous Development in Cyber-Physical Systems. [Electronic resource] / Conference: 24th Euromicro Conference on Digital System Design (DSD 2021), Palermo, Itally – Mode of access: https://www.researchgate.net/publication/353268043_AIDOaRt_AI-augmented_Automation_for_DevOps_a_Model-based_Framework_for_Continuous_Development_in_Cyber-Physical_Systems
4. Sabharwal N., Bhardwaj G., Hands-on AIOps Best Practices Guide to Implementing AIOps / Navin Sabharwal, Gaurav Bhardwaj. New York, USA: Apress, 2022. 259 p.
5. Josu Diaz de Arcaya, Ana-Isabel Torre-Bastida, Gorka Zarate, Raúl Miñón, Aitor Almeida, A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey [Electronic resource] / October 2023ACM Computing Surveys 56(4):1–30 Mode of access: https://www.researchgate.net/publication/374911984_A_Joint_Study_of_the_Challenges_Opportunities_and_Roadmap_of_MLOps_and_AIOps_A_Systematic_Survey
6. Pranjal Gupta, Harshit Kumar, Debanjana Kar, Karan Bhukar, Pooja Aggarwal, Prateeti Mohapatra, Learning Representations on Logs for AIOps [Electronic resource] / August 2023. Mode of access: https://www.researchgate.net/publication/373298002_Learning_Representations_on_Logs_for_AIOps
7. Roozbeh Aghili, Heng Li, Foutse Khomh, Studying the characteristics of AIOps projects on GitHub [Electronic resource] / October 2023Empirical Software Engineering 28(6). Mode of access: https://www.researchgate.net/publication/374810736_Studying_the_characteristics_of_AIOps_projects_on_GitHub
8. Saima Rafi, Muhammad Azeem Akbar, Sajjad Mahmood, Ahmed Alsanad, Abdulrahman Alothaim, Selection of DevOps best test practices: A hybrid approach using ISM and fuzzy TOPSIS analysis [Electronic resource] / March 2022 Journal of Software: Evolution and Process. Mode of access: https://www.researchgate.net/publication/359164520_Selection_of_DevOps_best_test_practices_A_hybrid_approach_using_ISM_and_fuzzy_TOPSIS_analysis

9. Mingxu Jin, Aoran Lv, Yuanpeng Zhu, Zijiang Wen, Yubin Zhong, Zexin Zhao, Jiang Wu, Hejie Li, Hanheng He, Fengyi Chen, An Anomaly Detection Algorithm for Microservice Architecture Based on Robust Principal Component Analysis [Electronic resource] / December 2020IEEE Access PP(99):1–1. Mode of access: https://www.researchgate.net/publication/347630664_An_Anomaly_Detection_Algorithm_for_Microservice_Architecture_Based_on_Robust_Principal_Component_Analysis

10. Jonas Sorgalla, Philip Wizenty, Florian Rademacher, Sabine Sachweh, Albert Zündorf, Applying Model-Driven Engineering to Stimulate the Adoption of DevOps Processes in Small and Medium-Sized Development Organizations [Electronic resource] / July 2021. Mode of access: https://www.researchgate.net/publication/353510651_Applying_Model-Driven_Engineering_to_Stimulate_the_Adoption_of_DevOps_Processes_in_Small_and_Medium-Sized_Development_Organizations

11. Ahmad Alnafessah, Alim Ul Gias, Runan Wang, Lulai Zhu, Giuliano Casale, Antonio Filieri, Quality-Aware DevOps Research: Where Do We Stand? [Electronic resource] / March 2021 IEEE Access 9:44476–44489. Mode of access: https://www.researchgate.net/publication/349934738_Quality-Aware_DevOps_Research_Where_Do_We_Stand

12. Tom Černý, Michael J. Donahoo, Michal Trnka, Contextual understanding of microservice architecture: current and future directions [Electronic resource] / January 2018 ACM SIGAPP Applied Computing Review 17(4):29–45. Mode of access: https://www.researchgate.net/publication/322842819_Contextual_understanding_of_microservice_architecture_current_and_future_directions

References

1. Newman S., Building Microservices, 2nd Edition / Sam Newman. – O'Reilly Media, Inc., 2021. 617 p.

2. Davis A., Bootstrapping Microservices with Docker, Kubernetes, and Terraform / Ashley Davis. New York, USA: Manning Publications, 2021. 440 p.

3. Eramo R., Muttilo V., Berarrdinelli L., Bruneliere H., AIDOaRt: AI-augmented Automation for DevOps, a Model-based Framework for Continuous Development in Cyber-Physical Systems. [Electronic resource] / Conference: 24th Euromicro Conference on Digital System Design (DSD 2021), Palermo, Itally – Mode of access: https://www.researchgate.net/publication/353268043_AIDOaRt_AI-augmented_Automation_for_DevOps_a_Model-based_Framework_for_Continuous_Development_in_Cyber-Physical_Systems

4. Sabharwal N., Bhardwaj G., Hands-on AIOps Best Practices Guide to Implementing AIOps / Navin Sabharwal, Gaurav Bhardwaj. New York, USA: Apress, 2022. 259 p.

5. Josu Diaz de Arcaya, Ana-Isabel Torre-Bastida, Gorka Zarate, Raúl Miñón, Aitor Almeida, A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey [Electronic resource] / October 2023ACM Computing Surveys 56(4):1–30 Mode of access: https://www.researchgate.net/publication/374911984_A_Joint_Study_of_the_Challenges_Opportunities_and_Roadmap_of_MLOps_and_AIOps_A_Systematic_Survey

6. Pranjal Gupta, Harshit Kumar, Debanjana Kar, Karan Bhukar, Pooja Aggarwal, Prateeti Mohapatra, Learning Representations on Logs for AIOps [Electronic resource] / August 2023. Mode of access: https://www.researchgate.net/publication/373298002_Learning_Representations_on_Logs_for_AIOps

7. Roozbeh Aghili, Heng Li, Foutse Khomh, Studying the characteristics of AIOps projects on GitHub [Electronic resource] / October 2023Empirical Software Engineering 28(6). Mode of access: https://www.researchgate.net/publication/374810736_Studying_the_characteristics_of_AIOps_projects_on_GitHub

8. Saima Rafi, Muhammad Azeem Akbar, Sajjad Mahmood, Ahmed Alsanad, Abdulrahman Alothaim, Selection of DevOps best test practices: A hybrid approach using ISM and fuzzy TOPSIS analysis [Electronic resource] / March 2022 Journal of Software: Evolution and Process. Mode of access: https://www.researchgate.net/publication/359164520_Selection_of_DevOps_best_test_practices_A_hybrid_approach_using_ISM_and_fuzzy_TOPSIS_analysis

9. Mingxu Jin, Aoran Lv, Yuanpeng Zhu, Zijiang Wen, Yubin Zhong, Zexin Zhao, Jiang Wu, Hejie Li, Hanheng He, Fengyi Chen, An Anomaly Detection Algorithm for Microservice Architecture Based on Robust Principal Component Analysis [Electronic resource] / December 2020IEEE Access PP(99):1–1. Mode of access: https://www.researchgate.net/publication/347630664_An_Anomaly_Detection_Algorithm_for_Microservice_Architecture_Based_on_Robust_Principal_Component_Analysis

10. Jonas Sorgalla, Philip Wizenty, Florian Rademacher, Sabine Sachweh, Albert Zündorf, Applying Model-Driven Engineering to Stimulate the Adoption of DevOps Processes in Small and Medium-Sized Development Organizations [Electronic resource] / July 2021. Mode of access: https://www.researchgate.net/publication/353510651_Applying_Model-Driven_Engineering_to_Stimulate_the_Adoption_of_DevOps_Processes_in_Small_and_Medium-Sized_Development_Organizations

11. Ahmad Alnafessah, Alim Ul Gias, Runan Wang, Lulai Zhu, Giuliano Casale, Antonio Filieri, Quality-Aware DevOps Research: Where Do We Stand? [Electronic resource] / March 2021 IEEE Access 9:44476–44489. Mode of access: https://www.researchgate.net/publication/349934738_Quality-Aware_DevOps_Research_Where_Do_We_Stand

12. Tom Černý, Michael J. Donahoo, Michal Trnka, Contextual understanding of microservice architecture: current and future directions [Electronic resource] / January 2018 ACM SIGAPP Applied Computing Review 17(4):29–45. Mode of access: https://www.researchgate.net/publication/322842819_Contextual_understanding_of_microservice_architecture_current_and_future_directions