BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 29 No. 2. 2024

UDC 004.7:658.512
DOI: 10.62660/bcstu/2.2024.57

Strategies for implementing or strengthening the DevOps approach in organizations: Analysis and examples

Bohdan Fedoryshyn*
Master
Lviv Polytechnic National University
79000, 12 Stepan Bandera Str., Lviv, Ukraine
https://orcid.org/0009-0005-3779-0186

Abstract. The aim of the study was to analyse the implementation of DevOps in organizations, in particular, to assess the impact of process automation, CI/CD, monitoring and orchestration of microservices on the efficiency of development and management. A methodology was created that allows organizations to effectively implement and enhance the DevOps approach, achieving high results in software development and management. The study looked at cultural change and training strategies, including building a culture of collaboration between teams and developing skills in modern DevOps tools. Process automation, including test automation and integration and deployment, plays an important role in improving code quality and reducing development time. Infrastructure as code allows managing and automating infrastructure configuration, enabling rapid scaling of environments. The work has shown that performance monitoring and feedback are critical to detecting problems early and continuously improving the product. Microservices orchestration, in particular, with Kubernetes, automates the deployment and scaling of containerized applications, which is critical for modern architectures with high performance and availability requirements. Case studies of companies such as Netflix, Spotify, and Airbnb demonstrate the successful application of DevOps practices and technologies to achieve high availability, scalability, and reliability of infrastructures. The study results confirm that the implementation of DevOps leads to a significant increase in development efficiency, software quality, and optimization of infrastructure management costs. In addition, the introduction of cultural changes and increased collaboration between development and operations teams contributes to greater flexibility and speed of response to changing market conditions

Keywords: microservices orchestration; tools; test automation; CI/CD processes; development efficiency

Article's History: Received: 10.12.2023; Revised: 05.04.2024; Accepted: 27.05.2024.

INTRODUCTION

In today's digital environment, organizations are increasingly adopting the DevOps approach to improve the efficiency of software development and operations management. DevOps, which combines development and operations practices, aims to shorten the development cycle, improve software quality and simplify the introduction of new features. This approach allows organizations to respond quickly to market changes and ensure high performance and reliability of their systems. DevOps provides integration and orchestration of microservices, process automation, and continuous monitoring. In particular, the automation of testing and build, integration and deployment (CI/CD) processes allows for the rapid and efficient introduction of new features and updates. Infrastructure as a Code (IaC) simplifies infrastructure management by enabling rapid scaling and repeatable environments. Continuous performance monitoring and well-established feedback processes help to identify and resolve issues early on, as well as continuously improve the product.

This study focuses on identifying the most effective strategies for implementing and strengthening the

Suggested Citation:
Fedoryshyn, B. (2024). Strategies for implementing or strengthening the DevOps approach in organizations: Analysis and examples. *Bulletin of Cherkasy State Technological University*, 29(2), 57-69. doi: 10.62660/bcstu/2.2024.57.

*Corresponding author

DevOps approach in organizations, taking into account numerous challenges such as cultural change, training and skills development, process automation, microservice orchestration, and monitoring and feedback. Implementing DevOps requires significant changes in the organization's culture, a shift to a more open and collaborative approach between development and operations teams, and the adoption of new tools and methodologies such as CI/CD, containerization, test automation, and IaC. Automating processes, such as building, testing, and deploying code, is a resource-intensive and complex process that requires significant investment in new technologies and tools. Managing microservices in complex architectures requires the use of orchestration systems such as Kubernetes, which also adds to the complexity. Monitoring performance and receiving feedback is critical to detecting and fixing problems at an early stage. Analysing examples of DevOps implementations at companies such as Netflix, Spotify, and Airbnb helps to highlight best practices and methodologies that other organizations can apply to achieve success.

The main challenge in this area is the difficulty of integrating DevOps into traditional organizations with established processes and structures, as well as the lack of knowledge and skills among staff. S. Schünemann (2023) investigated the use of automated test pipelines in continuous software development. As a result, he described methods and tools for the effective implementation of automated tests in the continuous development process. S. Kornitskyi (2024) studied the processes of implementing a service quality management system in accordance with ISO/IEC 20000 and described the benefits of standardization and its impact on the quality of services in technology companies. The impact of DevOps on organizational culture, metrics effectiveness, tool integration, staff training, innovation processes, security, and adaptation for different sizes of organizations should be further studied. Another existing challenge is the difficulty of integrating new methodologies into existing processes and structures of the organization, as well as the lack of knowledge and skills among staff to effectively apply DevOps practices.

N. Govil *et al.* (2020) describe methods and approaches for implementing DevOps in web development, including tools and practices that ensure the efficiency of web application development and maintenance. M. Muñoz & M.N. Rodríguez (2021) identified strategic approaches to DevOps implementation and provided examples of successful application of these strategies in organizations. Gaps that require further study include the impact of DevOps process automation on the quality of the final product and user experience. The problem in the field under study is the difficulty of transitioning to new methodologies and cultural changes, as well as the lack of readiness of organizations to adapt DevOps practices. S. Rafi *et al.* (2020) developed a model to assess the readiness of organizations to implement DevOps, which included an assessment of infrastructure, processes, and cultural aspects.

M.S. Khan *et al.* (2022) identified the main challenges organizations face when integrating a DevOps culture, including resistance to change and communication issues between teams. It is necessary to consider how to effectively manage cultural change and overcome resistance when implementing DevOps. A common problem is the difficulty of integrating new approaches such as DevSecOps and assessing the maturity of DevOps processes in organizations. M.A. Akbar *et al.* (2022) described a decision-making framework for the successful implementation of DevSecOps in software development organizations, which included methods for integrating security into DevOps processes. J. Radstaak (2019) developed a DevOps maturity model to assess the level of DevOps adoption in organizations, allowing organizations to measure and improve their DevOps processes. It is important to explore how to successfully integrate security into DevOps processes, including DevSecOps, as well as the management decisions required to do so.

The aim of the study was to analyse the implementation of the DevOps approach in companies, in particular, to study the impact of process automation, continuous integration and deployment (CI/CD), monitoring and orchestration of microservices on the efficiency of development and management. To achieve this goal, the following tasks were identified:

1. Analysing the cultural changes required for successful DevOps implementation, including the importance of building a culture of collaboration between development and operations teams.
2. Studying modern DevOps tools and practices, such as CI/CD, containerization and test automation, with a focus on training and development of employee skills.

MATERIALS AND METHODS

The study of the theoretical foundations of DevOps, its components, such as CI/CD, test automation, containerization, and infrastructure as code, allowed to form a knowledge base for further analysis. Successful case studies from real organizations were analysed to gain a practical understanding of DevOps implementation. Examples of companies such as Netflix, Spotify, and Airbnb that have implemented DevOps and achieved significant results in improving development processes and operations are considered. The necessary cultural changes for the successful implementation of DevOps were studied. The importance of creating a culture of collaboration between development and operations teams is assessed, which is critical to ensure effective information sharing and joint work on projects. An overview of modern DevOps tools and practices, such as CI/CD, containerization and test automation, is provided. The importance of employee training in these areas is emphasized.

The study included a detailed examination of the use of automated tests and continuous integration and deployment (CI/CD) systems, which are critical to ensuring code quality and automating development and deployment processes. The methodology involves developing test scenarios that cover various aspects of the application's functionality and performance, integrating tests with CI/CD processes to execute them every time a change is made to the code, and incorporating different types of testing, such as unit tests, integration tests, functional tests, and regression tests. IaC management tools such as Terraform, Ansible, and AWS CloudFormation are explored. This allows exploring how automating configurations and infrastructure management contributes to the rapid scalability and repeatability of environments. The overview includes an analysis of the functionality of each tool, as well as use cases for improving scalability and infrastructure management.

The methods of continuous performance monitoring and feedback are reviewed. This confirms the importance of these processes for identifying and fixing problems at an early stage, as well as for continuous product improvement. The study analyses in detail the role of microservices orchestration in modern DevOps practices, with a particular focus on managing deployed services using Kubernetes. This is an important aspect for ensuring the scalability and reliability of microservice architectures. The work also focuses on evaluating and comparing popular tools and platforms used to implement the DevOps approach, including Jenkins, Kubernetes, Docker, and IaC management systems. As part of the preliminary research, a comprehensive review of scientific publications, technical articles, reports, and documentation related to these tools was conducted (Smart, 2011; Morris, 2020). The review included a study of architectural solutions, functionalities, limitations, and features of Jenkins, Kubernetes, Docker, and IaC in the context of DevOps, which allowed to assess in detail their impact on software development and deployment processes. The results of DevOps implementation, including increased development speed, improved software quality, reduced infrastructure management costs, and the role of Kubernetes in ensuring high scalability and reliability, are evaluated.

RESULTS

Creating a culture of collaboration is one of the main strategies for successful DevOps implementation in organizations. It is critical for ensuring effective interaction between development and operations teams, reducing barriers between these groups and creating a unified working environment. The process of creating such a culture begins with the recognition of the need for constant information sharing and joint work on projects. In traditional approaches to software development, development and operations teams often work in isolation, which can lead to delays in implementing changes and unforeseen problems in the course of operation. DevOps seeks to break down the barriers between development and operations teams by integrating both teams at all stages of software development and deployment. To achieve this goal, DevOps implements several key initiatives:

1. Changing the way of thinking and approaches to work. DevOps focuses on changing the way we think and work, encouraging teams to work as a unit with a common goal of delivering new features and fixing bugs quickly and efficiently. This includes changes in communication, process organization and project management.

2. Regular meetings and information exchange. DevOps ensures that information is shared regularly between teams through joint meetings, huddles, and shared communication platforms. This allows all project participants to be aware of changes, problems, and needs that arise during development and operation.

3. Implementation of common tools. The use of common tools for project management, monitoring and reporting is central to DevOps. This helps to improve process transparency and provides a single source of truth for all project participants.

4. Cross-functional training. DevOps involves upskilling employees through training in new tools and methodologies used in both development and operations. This helps reduce dependence on specialized knowledge and promotes greater flexibility in work.

5. Management support. The leadership of organizations plays an important role in DevOps by actively supporting initiatives and creating the conditions for successful change. This includes financial and moral support for teams working on DevOps implementation.

Fostering a culture of collaboration is critical to the success of DevOps adoption, as it allows organizations to respond to change more effectively, introduce new features faster, and improve software quality (Luz *et al.*, 2018). When teams work together, their ability to innovate and solve problems increases significantly, leading to the achievement of common goals and increased competitiveness of the organization. Training and skills development are critical to the successful implementation of DevOps in organizations. Employees need to acquire in-depth knowledge and skills in modern DevOps tools and practices, such as CI/CD, containerization, and test automation, to effectively implement these practices and ensure their successful operation. The learning and skill development process includes an introduction to CI/CD tools such as Jenkins, GitLab CI/CD, CircleCI, or Travis CI, which automate the processes of building, testing, and deploying software. Mastery of these tools helps to reduce the time of development and release of new versions of products, as well as ensure their high quality. In addition, knowledge of containerization and container management principles is critical for the development and implementation of microservices. Employees need to be able to work with containerization tools such as Docker and understand how to manage containers using orchestration systems such as Kubernetes. Containerization simplifies deployment processes, provides isolation of environments, and makes it easier to scale applications.

Test automation is another crucial aspect that helps ensure software quality at all stages of development. Employees should be able to use automated testing tools such as Selenium, JUnit, or TestNG to create and execute test scripts. This helps to identify errors faster, reduce manual testing, and improve the overall quality of the product. In addition to technical skills, it is relevant to train employees in DevOps practices, which include building effective development and deployment processes, configuration management, and performance monitoring and analysis. This requires knowledge of DevOps best practices and standards, such as IaC and DevOps Culture. DevOps technologies and practices are constantly evolving, so it's important to ensure that employees are constantly trained and certified. This may include participation in trainings, webinars, and courses, as well as obtaining professional certifications that confirm knowledge and skills in the field of DevOps.

Implementing and maintaining training programmes for employees helps organizations ensure a high level of skill and readiness to work with modern DevOps tools and practices. This helps to improve the efficiency of development and deployment processes, increase software quality and reduce time to market. Active investment in employee skill development is critical to achieving success in DevOps implementation and provides a competitive advantage for the organization. Test automation is a critical component in DevOps implementation, as it allows effectively ensuring high-quality software at all stages of development. Implementing automated tests helps reduce bugs, shorten development time, and improve overall software stability and reliability.

The test automation process begins with the creation of test scripts that define which aspects of the software should be tested. These can be unit tests to check individual code components, integration tests to check the interaction between components, or system tests to check the operation of the entire application. After defining the test scenarios, the next step is to use automation tools. Tools such as Selenium for web applications, JUnit or TestNG for Java applications, or NUnit for NET applications help automate test execution and reduce manual work. The integration of automated tests with continuous integration (CI) and continuous deployment (CD) processes is an essential aspect. This allows running tests automatically every time it is necessary to make a change to the code, which helps to detect bugs at an early stage and ensures product stability. CI/CD tools such as Jenkins, GitLab CI/CD, or CircleCI are used to automate this process and ensure continuous testing (Singh, 2022).

After running automated tests, it is critical to collect and analyse the test results. This allows identifying problems, getting information about application performance, and defining areas for improvement. Metrics and reporting tools such as SonarQube or Allure help in visualizing test results and tracking trends. Regularly updating test scenarios is essential as software is constantly evolving and changing. This ensures that tests remain relevant and effective by adapting them to new features and requirements. Test automation also reduces the need for manual testing, which can be time-consuming and error-prone. Automated tests are faster and less likely to make human errors, which increases the accuracy and efficiency of testing. Implementing automated tests helps to identify defects faster, reduce testing time, and improve overall product quality, which is key to successful DevOps implementation and competitive advantage.

Implementation of continuous integration and deployment (CI/CD) systems is a key element of successful DevOps implementation, as these systems automate the processes of building, testing, and deploying code, which significantly increases development efficiency and ensures high-quality software (Singh & Mansotra, 2021). Continuous Integration (CI) involves automatically building and testing code whenever changes are made to the repository. This allows detecting defects in the early stages of development and reducing the likelihood of errors in the finished product. CI systems, such as Jenkins, GitLab CI/CD, CircleCI, or Travis CI, automate these processes by ensuring that tests and builds are performed regularly and automatically at each commit. This helps to identify and fix bugs faster, facilitates the integration of new features, and reduces development delays.

Continuous Deployment (CD) extends the CI concept by automating the process of deploying code to a test or production environment. This ensures fast and reliable deployment of new software releases, reducing the time from development to deployment. CD systems automatically check the code, deploy it to test environments, and, if everything goes well, to the production environment. This allows quickly releasing new versions of the product and maintaining a high level of stability and quality (Fig. 1).

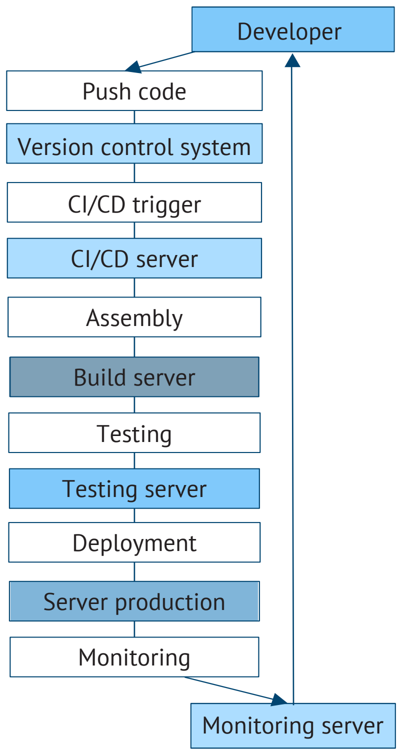

Figure 1. Diagram of interaction between CI/CD components
Source: compiled by the author based on O. Kravchuk (2023)

IaC is an important component in the implementation of CI/CD. IaC automates infrastructure management by using configuration files to describe and manage environment components such as servers, networks, and databases. This provides greater flexibility and scalability, and ensures consistent environments for development, testing, and performance. By using IaC tools such as Terraform, Ansible, or AWS CloudFormation, organizations can automatically build and manage infrastructure, which significantly reduces the likelihood of human error and increases the speed and accuracy of deployment. IaC also makes it easy to track changes to infrastructure, manage versions, and recover systems after a disaster. The integration of CI/CD and IaC provides a continuous development and deployment process that automates key stages of the software lifecycle, reduces time to market, improves product quality, and provides flexibility in scaling and infrastructure management. This allows organizations to respond quickly to changes, reduce risks and achieve high performance in a competitive environment.

Infrastructure as code is a relevant practice in modern DevOps strategies that allows managing and automating the configuration and management of infrastructure through code (Ljunggren, 2023). This significantly increases the efficiency and flexibility of infrastructure deployment processes and provides a number of benefits. Infrastructure as code uses configuration files or scripts written in specialized languages or formats, such as YAML Ain't Markup Language™ (YAML), JavaScript Object Notation (JSON), and HashiCorp Configuration Language (HCL), to describe and manage infrastructure resources. These files contain all the instructions needed to create, configure, and manage server instances, networks, storage, and other infrastructure components.

One of the main benefits of IaC is the ability to scale infrastructure quickly. With IaC, organizations can quickly and efficiently scale resources up or down to meet business needs without having to manually configure each component of the infrastructure. This is especially important in the face of dynamic and changing workloads, where the speed of scaling can affect the performance and availability of services. Infrastructure as code also ensures repeatability and reproducibility of environments. The code that describes the infrastructure can be saved in version control systems such as Git. This makes it easy to reproduce and reuse infrastructure configurations in different environments, such as development, testing, and production. All changes to the infrastructure are documented and verifiable, making it easier to manage configurations and reduce the likelihood of errors.

Automating infrastructure management is also a key benefit of IaC. Instead of manually configuring and administering resources, organizations can automate the creation, configuration and updating of infrastructure. This helps to reduce administration costs, reduce the likelihood of human error, and ensure consistency across all environments. IaC supports the use of infrastructure automation and management tools such as Terraform, Ansible, Puppet, or Chef. These tools allow describing one's infrastructure as code and automating its deployment and management, which simplifies processes and provides high efficiency and control over one's infrastructure. Performance monitoring is an essential component of successful DevOps implementation (Azad, 2022). It allows constantly monitoring the state of the system, identifying and fixing problems at an early stage. Ongoing monitoring ensures a high level of software availability and reliability, which is critical to providing a positive user experience and supporting business processes. The main purpose of performance monitoring is to collect and analyse metrics, logs, and bug tracking. Metrics allow tracking key system performance indicators, such as response time, CPU utilization, memory usage, network speed, and other important parameters. Collecting and analysing this data allows detecting anomalies, overloads, or other issues that may affect system performance in a timely manner.

Metrics collection tools provide automated collection and storage of system health data. They can provide real-time monitoring and the ability to view graphs and charts illustrating system performance. Tools such as Prometheus, Grafana, Datadog, or New Relic allow collecting data from a variety of sources, including services, applications, and infrastructure. Logs (log files) are another major part of monitoring, providing detailed information about events occurring in the system. Logs can contain records of errors, warnings, informational messages, and details about the execution of requests. Analysing logs helps to identify problems, track their causes, and understand how the system responds to different conditions. Tools for working with logs, such as ELK Stack (Elasticsearch, Logstash, Kibana) or Splunk, allow centralizing, analysing and visualizing logs to identify and resolve problems faster.

Error tracking involves monitoring and logging errors that occur in a system. Error tracking tools such as Sentry, Rollbar, or Raygun help to automatically detect and report errors, including their nature, frequency, and impact on the system. This allows developers to respond quickly to issues by providing detailed reports and context for resolution. By continuously monitoring performance, organizations can detect problems early, reduce time to resolution, and improve overall system stability and efficiency. Monitoring also provides crucial feedback on the state of the system, enabling informed decisions to be made about further development and optimization of the software and infrastructure.

Feedback is a critical element for continuous product improvement and ensuring that a product meets the needs of users. Establishing user feedback processes and automated systems allows organizations to make improvements based on real-world data and user experience. The first aspect is collecting feedback from users. It is relevant to establish effective channels for collecting feedback, such as online surveys, feedback forms, user forums, or functionality to send feedback directly from the application or website interface. Providing a convenient and easy way for users to provide feedback allows getting a variety of perspectives and identify potential problems or areas for improvement. The second aspect is the use of data analytics. Analysing feedback data helps to identify trends, common problems, and user preferences. Data analytics tools, such as Google Analytics, Mixpanel, or Amplitude, allow collecting and processing data on user interaction with the product (Mykhalchenko & Tytarenko, 2023). This includes information about the frequency of use, session duration, failure points, and other metrics that help to understand how the product is used and where there may be problems. The third aspect is integration with automated systems. Automated systems can provide an additional layer of feedback through performance, log and error monitoring. Performance monitoring tools such as Datadog or New Relic can provide information about anomalies or issues that users may not be able to detect directly. Integrating these systems can automatically generate reports and alerts about potential problems, making it easier to identify and fix errors. The fourth aspect is continuous product improvement. Feedback from users and automated systems should be used to regularly update and improve the product. This may include releasing new versions, fixing identified defects, optimizing functionality, and improving the interface. A continuous improvement cycle allows adapting the product to the changing needs and preferences of users, ensuring its competitiveness in the market. The last but not the least aspect is interaction with development teams. Establishing a feedback process also includes effective communication between development and support teams. Feedback should be systematically passed on to developers for analysis and implementation of changes. Regular meetings where feedback is discussed, and next steps are planned foster better interaction between teams and ensure focus on improving the product. Successfully establishing feedback processes allows organizations to adapt to user and market needs, ensuring continuous improvement in product quality and user satisfaction. It also helps build trust and loyalty among users, which is critical to the long-term success of a product.

Microservices orchestration is an important part of the modern DevOps approach, as it allows for efficient management of deployed services in a complex architecture (Waseem *et al.*, 2020). The growing popularity of microservice architecture, which breaks down large monolithic applications into small, independent services, has led to the need for tools that help coordinate and manage these services. Microservice orchestration automates the deployment, scaling, and management of containers, which increases the flexibility and efficiency of software development and operation. One of the key tools for orchestrating microservices is Kubernetes, which provides powerful capabilities for the automated management of containerized applications. Kubernetes allows automatically deploying, scaling, and managing containers, providing load balancing, disaster recovery, and simplified configuration management. Thanks to its powerful features, Kubernetes is becoming the de facto standard for microservice orchestration in many organizations. Microservice orchestration includes several key aspects. Firstly, it is the automation of deployment. With the help of orchestration tools, it is possible to automatically deploy new versions of services in different environments, from test to production, which ensures fast and secure implementation of changes. This reduces human error and speeds up the deployment process, which is critical in today's dynamic environments. The second crucial aspect is scaling. Microservice orchestrators allow automatically scaling services depending on the load. For example, Kubernetes can automatically add or remove container instances based on current resource requirements, which allows for efficiently using available resources and ensuring high system performance even during peak loads. Configuration management is another critical aspect of microservice orchestration.

Orchestration tools allow centrally managing the configuration of services, which simplifies the upgrade process and reduces the likelihood of configuration errors. It also allows quickly implementing changes and adapting services to new requirements.

In addition, microservice orchestration provides disaster recovery. Orchestration tools such as Kubernetes can automatically restart failed containers and move services to other nodes in the event of hardware failures, ensuring high reliability and availability of the system. Monitoring and logging are also important aspects of microservice orchestration. Orchestration tools are often integrated with monitoring and logging systems such as Prometheus and ELK Stack, which allows for centralized collection and analysis of data on the status and performance of services. This helps to identify problems in time and make informed decisions to resolve them. Kubernetes is a powerful container orchestration system designed to automate the deployment, scaling, and management of containerized applications (Zhou *et al.*, 2021). It provides centralized management of large container clusters, making it an indispensable tool for modern IT infrastructures. Kubernetes has a wide range of functionalities that provide high availability, automatic disaster recovery, load balancing, and automatic container scaling. High availability is achieved by automatically distributing the load between different nodes, which ensures the uninterrupted operation of applications even in the event of a failure of individual components. Automatic disaster recovery includes automatic restart of containers in the event of a malfunction, as well as the ability to move containers to other nodes in the event of hardware problems. Load balancing ensures an even distribution of requests between containers, which increases system performance and stability. Automatic scaling allows dynamically adding or removing containers depending on the current resource requirements, which ensures efficient use of resources and high system performance.

Kubernetes integrates with CI/CD processes to automate the deployment of new versions of applications, providing version and environment control. Integration with CI/CD tools such as Jenkins, GitLab CI/CD, CircleCI, or Travis CI allows automatically deploying code updates to different environments, which speeds up the development and deployment process. Kubernetes also supports the IaC concept, which allows describing infrastructure configurations as code and automatically applying them across different environments. This ensures high repeatability and consistency of configurations, reduces the likelihood of errors and simplifies infrastructure management. Kubernetes also supports high availability configurations, which allows applications to remain available even in the event of a node failure. In addition, Kubernetes fits well with the microservices architectural style. With Kubernetes, each microservice can be deployed as a separate container in a module, which enables independent scaling, management, and rapid development of microservice-based applications (Fig. 2).

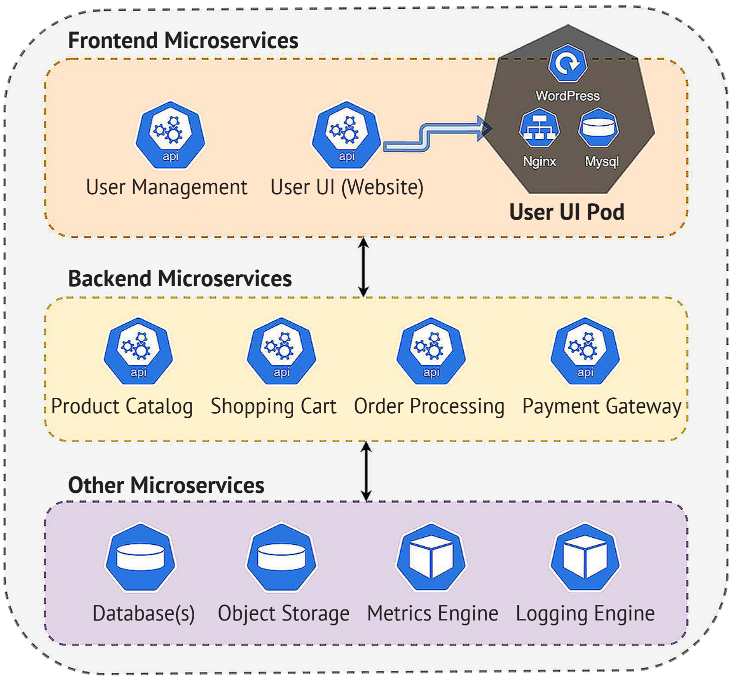

Figure 2. Microservice architecture in Kubernetes

Source: Yu. Vatsyk (2024)

Implementing DevOps is a complex process that requires changes in an organization's culture, the introduction of new tools and practices, and continuous monitoring and improvement of processes. However, the efforts invested in this transformation can significantly increase an organization's efficiency and competitiveness in the market. Netflix was one of the first large companies to implement Kubernetes to manage its microservices. Orchestrating microservices with Kubernetes allows Netflix to automate the deployment, scaling, and management of applications running in containers. This is critical for a company that serves more than 200 million users worldwide. Thanks to Kubernetes, Netflix can dynamically scale its services depending on the load, for example, when releasing new episodes of popular shows when the number of viewers increases dramatically. Kubernetes also allows for automated deployment of new versions of applications, minimizing the risks associated with human error and ensuring high service reliability.

Netflix has implemented continuous integration (CI) to ensure that changes to the code created by multiple development teams are integrated quickly and reliably. CI systems automatically test new changes, allowing potential problems to be identified early in the development process. This approach helps to minimize the time between writing code and deploying it to the production environment. Continuous deployment (CD) allows Netflix to automate the process of releasing new versions of applications, which significantly reduces the time to market for new features and updates. It also provides the ability to respond quickly to user feedback and quickly fix bugs or add new features. Netflix pays considerable attention to monitoring the performance of its services and collecting feedback from users. The company uses its own Atlas platform to collect performance metrics, which allows it to accurately assess the status of each service in real time. This includes monitoring indicators such as response time, number of errors, and other critical parameters that affect the user experience. Netflix has also developed Eureka, a service management system that allows managing the availability and scaling of microservices. Monitoring tools allow quickly identifying and fixing problems, ensuring uninterrupted service operation even during peak periods.

One of the key aspects of successful DevOps implementation at Netflix is cultural and organizational change. The company is creating cross-functional teams that bring together developers and operations specialists. This approach helps to improve communication and collaboration between different teams, which is critical for quick and effective implementation of changes. In addition, Netflix gives its teams a lot of autonomy in decision-making, which allows them to quickly adapt to changes and innovate without bureaucratic delays. This approach allows the company to remain a leader in the streaming services market, ensuring a high speed of development of new features and stability of work. In this way, Netflix demonstrates the effective use of DevOps tools and practices to manage complex infrastructure, which allows it to maintain a high quality of user experience and remain competitive in the market.

Spotify uses Kubernetes as its primary microservices orchestration tool, enabling it to efficiently manage the thousands of containers that power the many features on the platform. Spotify's microservice architecture separates the various functional components of the platform into separate services that can be independently deployed, scaled and maintained. This gives Spotify the ability to quickly adapt to changes in load, for example, during the release of new albums or significant events in the music world, when the number of active users can increase dramatically. Using Kubernetes also allows Spotify to automate the management of its microservices, ensuring that services are reliable and highly available even in the event of system failures.

Spotify actively integrates continuous integration (CI) and continuous deployment (CD) practices, which are key elements of DevOps at the company. CI processes automate the testing and integration of new code changes, which ensures that potential problems are quickly identified and resolved at the earliest stages of development. This helps to minimize the risks associated with releasing new versions and ensures platform stability. Continuous Deployment (CD) automates the process of releasing new features and updates, enabling Spotify to deliver new features to users quickly. For example, new features or updates can be released multiple times a day without interrupting the user experience. Performance monitoring and feedback gathering are an integral part of DevOps at Spotify. The company uses powerful monitoring tools such as Prometheus and Grafana to collect detailed metrics about system performance in real time. Prometheus collects and stores metrics that are used to identify potential issues and analyse performance, while Grafana provides visualization of this data, enabling teams to make quick decisions about optimizing and scaling resources. This is especially important for Spotify, which serves millions of users worldwide, and every delay or performance hit can have a significant impact on the user experience.

One of the key cultural changes that Spotify has implemented to support DevOps is the creation of cross-functional teams known as "Squads". These teams work autonomously and are responsible for specific aspects of the product. Each "Squad" is made up of developers, testers, DevOps engineers, and other specialists who work together to achieve common goals. This structure allows Spotify to quickly adapt to changes and respond quickly to market needs. In addition, Spotify has implemented a "Tribe" structure, which brings together several "Squads" working on broader areas of product development. This organizational structure promotes effective collaboration between different teams, ensuring consistency of action and high speed of project implementation. By implementing these practices and technologies, Spotify has been able to achieve a high level of performance and reliability of its platform, while maintaining flexibility in the development and implementation of new features. This has allowed the company to become one of the leaders in the audio streaming market and maintain a high quality of service for its users.

Airbnb actively uses Kubernetes to manage its microservices' infrastructure. Kubernetes allows Airbnb to automate the deployment, scaling, and management of containers, which ensures high availability and stability of services even during peak loads. For example, during high seasons, such as holidays or major events, when the number of users increases dramatically, Kubernetes automatically scales resources to handle the increased load. This allows Airbnb to keep the service running smoothly, minimizing downtime and disruptions. Using Kubernetes also simplifies infrastructure management, as it allows for rapid deployment and upgrades of individual components without impacting other parts of the system.

Airbnb actively integrates continuous integration (CI) and continuous deployment (CD) practices as an integral part of their DevOps approach. CI/CD automates the processes of building, testing, and deploying code, which can significantly reduce the time from development to deployment of new features or fixes. For example, new features or bug fixes can be released several times a day, allowing a company to respond quickly to changing user needs or vulnerability patches. This also reduces the risk of human error and improves the quality of the final product. To ensure high performance and stability of its services, Airbnb actively uses tools to monitor and manage feedback. Prometheus collects and stores metrics about system performance, allowing DevOps teams to identify potential problems at an early stage. Using Grafana to visualize this data helps to make quick and accurate decisions about optimizing and scaling resources. For example, if the system detects an increase in latency in responding to requests, teams can respond quickly by adjusting configurations or adding resources to address the issue. This allows Airbnb to maintain a high quality of user experience, which is critical for a company with millions of users worldwide.

The implementation of DevOps at Airbnb also includes important cultural and organizational changes. The company is creating cross-functional teams that bring together developers, DevOps engineers, testers, and other specialists responsible for different aspects of the product. These teams work autonomously, which allows them to quickly adapt to changes and implement innovations. In addition, this structure promotes closer cooperation between development and operations teams, which improves communication and coherence. For example, the team responsible for a particular microservice can quickly make changes or introduce new features without waiting for other teams to make decisions.

Airbnb actively uses the IaC approach to automate infrastructure management and configuration. Thanks to IaC, Airbnb can easily create and manage infrastructure resources such as servers, network settings, and databases using software code. This significantly reduces the risks associated with human error and ensures repeatable and standardized environments. In addition, automating these processes allows the company to quickly scale its resources in response to growing needs, which is especially relevant during peak times. By adopting these practices and technologies, Airbnb has been able to significantly improve the efficiency and reliability of its services, ensuring uninterrupted operation even during high loads. These innovations have allowed the company not only to maintain a high quality of user experience, but also to quickly adapt to changes in the market, remaining one of the leaders in the rental and hospitality industry.

Thus, Kubernetes is a key tool for orchestrating microservices in the context of DevOps. Its functionalities provide automation, high availability, scalability, and integration with CI/CD processes, which allows organizations to effectively manage their containerized applications and ensure high performance and reliability of their services. Analysing the above examples, several important conclusions can be drawn. Firstly, the implementation of Kubernetes for microservices orchestration is a key component of a successful DevOps implementation. It allows automating many aspects of infrastructure management, ensuring high availability, reliability, and scalability of services. Secondly, integration with CI/CD processes is critical to ensure the continuous release of new features and updates, which allows companies to respond quickly to changes in the market and user needs. Third, real-life examples of DevOps implementation at companies such as Netflix, Spotify, and Airbnb demonstrate that the successful use of this methodology can significantly improve development efficiency, and software quality, and optimize infrastructure management costs. They confirm that the correct implementation of DevOps tools and practices can lead to significant improvements in software development and management. These examples also highlight the importance of cultural change in organizations, including building a culture of collaboration between development and operations teams. Investing in employee training in new DevOps tools and practices is critical to success. Continuously monitoring system performance and receiving feedback allows quickly identifying and fixing problems, which contributes to continuous product improvement.

DISCUSSION

This study has examined in detail the implementation and strengthening of the DevOps approach in organizations, which is a complex and multifaceted process that requires a comprehensive approach and a deep understanding of both technical and cultural aspects. One of the most important components of a successful DevOps implementation is a cultural change in the organization. Fostering a culture of collaboration between development and operations teams is critical to ensuring effective information sharing and project collaboration. This includes creating an environment where each team member feels responsible for the end result, not just their own narrow area of work. This environment encourages more open discussion of problems and faster resolution. Employees should be trained in modern DevOps tools and practices, such as CI/CD, containerization, and test automation. For example, Continuous Integration and Continuous Deployment (CI/CD) are fundamental DevOps practices that ensure that new versions of software are constantly updated and released. This reduces the time to release new features and improvements, increasing the speed and flexibility of the organization to respond to market needs.

Similar conclusions were reached by J. Díaz *et al.* (2021), who explored the reasons why many companies are implementing a DevOps culture in their organizations. They noted that DevOps allows organizations to become more agile in their development processes. This means that companies can quickly respond to market changes and user requirements by adapting their products and services to new conditions. DevOps helps to accelerate development and deployment cycles, which allows new features and updates to be brought to market faster. Process automation is one of the key components of DevOps, as it allows achieving high efficiency and quality of software development. One of the main advantages of automation is the ability to detect errors at the early stages of development. This is achieved through the use of automated tests that can quickly identify defects in the code. This allows development teams to fix bugs quickly, reducing overall fix costs and minimizing the risk of serious problems later in the development process. Continuous integration and deployment (CI/CD) systems are an integral part of DevOps process automation. They automate the compilation, testing, and deployment of code, which significantly reduces the time between writing code and deploying it to a production environment. Thanks to CI/CD automation, teams can release new product versions more frequently, allowing them to respond more quickly to changing market and user requirements. The continuity of the development and deployment processes also ensures more stable and reliable software performance, as each change is put through standardized and automated testing procedures.

Studies in the same direction were conducted by I. Karamitsos *et al.* (2020), who investigated the application of DevOps practices in the context of continuous automation of machine learning processes. The authors found that the integration of DevOps practices into machine learning (ML) can significantly improve the efficiency of developing and deploying machine learning models. S.M. Mohammad (2018) studied the optimization of DevOps automation in the context of cloud applications, identifying several crucial aspects that affect the effectiveness of DevOps implementation in cloud environments. He focused on the use of automated tools for managing cloud resources, such as automatic scaling, configuration management, and application deployment.

Continuously monitoring system performance and receiving feedback is an integral part of DevOps and has a significant impact on software quality and stability. Using metrics, logs, and bug tracking tools allows quickly determining and fixing problems at an early stage, which helps maintain a high level of system performance. Monitoring provides continuous control over aspects such as response time, resource utilization, and overall system health, enabling responding quickly to potential issues. Analysing event logs and tracking errors using tools such as Sentry or Rollbar provides detailed information about the causes of faults and anomalies, which is important for their prompt resolution. Establishing processes for receiving feedback from users is also critical for continuous product improvement. Automated systems for collecting and analysing feedback help to identify systemic issues and trends, which ensures faster response to user needs and continuous product improvement, which is the basis of successful DevOps practice.

L. Giamattei *et al.* (2024) reached a similar conclusion in their paper on monitoring tools for DevOps and microservices. They emphasized that effective monitoring is critical to ensure the stability and performance of systems in a DevOps environment. The authors highlighted the importance of using tools that allow not only tracking system performance, but also analysing data to detect anomalies and potential problems in a timely manner.

Microservices orchestration is critical for successful DevOps implementation, and Kubernetes is one of the key tools in this process. Kubernetes automates the deployment, scaling, and management of containerized applications, ensuring high availability and reliability of the system. Thanks to automatic scaling, the system can adapt to changing workloads, ensuring efficient use of resources and maintaining performance. Load balancing between containers avoids overloading of individual components, which ensures stable operation of applications. Kubernetes also provides automatic disaster recovery, which is critical for maintaining business continuity. Thanks to these features, Kubernetes facilitates the efficient management of large container clusters, which ensures the scalability and reliability of microservice architectures.

Similar conclusions were reached by S. Baškarada *et al.* (2018), who analysed the practical possibilities and challenges of implementing this architecture in their work on microservices' architecture in the context of modern IT systems. They emphasized that although the microservice architecture provides flexibility and scalability, its implementation will require careful orchestration and management planning to ensure the efficient functioning of the system. C. Bühler (2021), studying microservices in the context of DevOps, pointed out the importance of integrating orchestration tools such as

Kubernetes to automate management tasks and ensure continuous application deployment. He noted that Kubernetes makes it easy to manage deployed microservices, which significantly increases the reliability and scalability of applications within the DevOps approach.

Real-life examples of DevOps implementation in companies such as Netflix and Amazon clearly demonstrate the effectiveness of this methodology. Netflix actively uses DevOps to achieve high availability and scalability of its services. The company uses Kubernetes to manage containers and automate the deployment of microservices. This allows it to provide continuous access to services even in the event of heavy loads, downloads, or failures. Amazon is also a prime example of successful DevOps implementation. The company uses test and deployment automation to increase the speed and quality of development. In addition, Amazon uses microservice orchestration to increase the reliability of its infrastructure, which allows it to manage huge amounts of data and complex system architectures without interruption. Both of these examples confirm that DevOps can significantly improve productivity and efficiency in large organizations. V. Sharma *et al.* (2024) focused on improving software automation through DevOps adoption. They noted that DevOps integration provides effective automation of processes, including testing, deployment, and monitoring, which is critical to reducing development time and improving product quality. Overall, the findings show that successfully implementing or strengthening a DevOps approach in organizations requires a comprehensive approach that encompasses both technical and cultural aspects.

CONCLUSIONS

The study found that successful implementation of DevOps requires profound cultural changes in an organization, including the formation of a culture of collaboration between development and operations teams. This ensures effective information sharing and joint work on projects, which contributes to overall productivity. Investing in employee training in new DevOps tools and practices, such as CI/CD, containerization, and test automation, is critical to success. They allow increasing the efficiency and speed of development, as well as adapting to technological changes.

The work showed that test automation and the implementation of continuous integration and deployment (CI/CD) systems are key to ensuring high code quality and reducing the time to release new product versions. This allows quickly identifying and eliminating errors, reducing the cost of fixing them and speeding up the development process. Continuous monitoring of system performance and receiving feedback from users contribute to the continuing improvement of the product. The use of tools for collecting metrics and tracking bugs is essential to maintain the high quality and reliability of the product. Orchestrating microservices using tools such as Kubernetes ensures scalability and reliability. Kubernetes automates the deployment, scaling, and management of containerized applications, which contributes to the high availability and reliability of microservice architectures. Practical examples from companies such as Netflix, Spotify, and Airbnb confirm that DevOps implementation significantly increases the efficiency of software development and management. These examples demonstrate that the successful use of DevOps tools and practices can significantly improve software quality, optimize infrastructure management costs, and ensure the rapid and continuous release of new features.

Prospects for further research include analysing the impact of new technologies, such as artificial intelligence and machine learning, on DevOps processes, including their use for automation, forecasting and improving code quality. The study of the integration of DevOps practices with cloud platforms and services, including the impact of cloud environments on the efficiency and scalability of DevOps processes. The study's limitations stem from the insufficient technical characteristics and capabilities of the DevOps tools and practices under study, which may affect the overall assessment of their effectiveness, as certain aspects of technologies or tools may not be fully represented in the analysis.

ACKNOWLEDGEMENTS

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Akbar, M.A., Smolander, K., Mahmood, S., & Alsanad, A. (2022). Toward successful DevSecOps in software development organizations: A decision-making framework. *Information and Software Technology*, 147, article number 106894. doi: 10.1016/j.infsof.2022.106894.

[2] Azad, N. (2022). Understanding DevOps critical success factors and organizational practices. In *IWSiB '22: Proceedings of the 5th international workshop on software-intensive business: Towards sustainable software business* (pp. 83-90). New York: Association for Computing Machinery. doi: 10.1145/3524614.3528627.

[3] Baškarada, S., Nguyen, V., & Koronios, A. (2018). Architecting microservices: Practical opportunities and challenges. *Journal of Computer Information Systems*, 60(5), 428-436. doi: 10.1080/08874417.2018.1520056.

[4] Bühler, C. (2021). *Microservices in a DevOps context*. (Thesis, OST Eastern Switzerland University of Applied Sciences, Rapperswil-Jona, Switzerland).

[5] Díaz, J., López-Fernández, D., Pérez, J., & González-Prieto, Á. (2021). Why are many businesses instilling a DevOps culture into their organization? *Empirical Software Engineering*, 26(2), article number 25. doi: 10.1007/s10664-020-09919-3.

[6] Giamattei, L., *et al*. (2024). Monitoring tools for DevOps and microservices: A systematic grey literature review. *Journal of Systems and Software*, 208, article number 111906. doi: 10.1016/j.jss.2023.111906.

[7] Govil, N., Saurakhia, M., Agnihotri, P., Shukla, S., & Agarwal, S. (2020). Analyzing the behaviour of applying agile methodologies & DevOps culture in e-commerce web application. In *2020 4th international conference on trends in electronics and informatics (ICOEI) (48184)* (pp. 899-902). Tirunelveli: Institute of Electrical and Electronics Engineers. doi: 10.1109/ICOEI48184.2020.9142895.

[8] Karamitsos, I., Albarhami, S., & Apostolopoulos, C. (2020). Applying DevOps practices of continuous automation for machine learning. *Information*, 11(7), article number 363. doi: 10.3390/info11070363.

[9] Khan, M.S., Khan, A.W., Khan, F., Khan, M.A., & Whangbo, T.K. (2022). Critical challenges to adopt DevOps culture in software organizations: A systematic review. *IEEE Access*, 10, 14339-143349. doi: 10.1109/ACCESS.2022.3145970.

[10] Kornitskyi, S. (2024). *Implementation of a service quality management system in technology companies based on international standards ISO/IEC 20000*. (Master's thesis, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine).

[11] Kravchuk, O. (2023). CI/CD implementation model for optimizing IT project management. *Measuring and Computing Devices in Technological Processes*, 3, 73-82. doi: 10.31891/2219-9365-2023-75-8.

[12] Ljunggren, D. (2023). *DevOps: Assessing the factors influencing the adoption of infrastructure as code, and the selection of infrastructure as code tools: A case study with Atlas Copco*. (Master's thesis, KTH Royal Institute of Technology, Stockholm, Sweden).

[13] Luz, W.P., Pinto, G., & Bonifácio, R. (2018). Building a collaborative culture: A grounded theory of well succeeded devops adoption in practice. In *ESEM '18: Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement* (article number 6). New York: Association for Computing Machinery. doi: 10.1145/3239235.3240299.

[14] Mohammad, S.M. (2018). Streamlining DevOps automation for Cloud applications. *International Journal of Creative Research Thoughts (IJCRT)*, 6(4), 955-959.

[15] Morris, K. (2020). *Infrastructure as code: Dynamic systems for the cloud age*. Sebastopol: O'Reilly Media, Inc.

[16] Muñoz, M., & Rodríguez, M.N. (2021). A guidance to implement or reinforce a DevOps approach in organizations: A case study. *Journal of Software Evolution and Process*, 36(3), article number e2342. doi: 10.1002/smr.2342.

[17] Mykhalchenko, H., & Tytarenko, M. (2023). Data analytics and personalized marketing strategies in e-commerce platforms. *Futurity Economics & Law*, 3(3), 115-139. doi: 10.57125/FEL.2023.09.25.07.

[18] Radstaak, J. (2019). *Developing a DevOps maturity model: A validated model to evaluate the maturity of DevOps in organizations*. (Master's essay, University of Twente, Enschede, the Netherlands).

[19] Rafi, S., Yu, W., Akbar, M.A., Mahmood, S., Alsanad, A., & Gumaei, A. (2020). Readiness model for DevOps implementation in software organizations. *Journal of Software Evolution and Process*, 33(4), article number e2323. doi: 10.1002/smr.2323.

[20] Schünemann, C. (2023). *Automating the build and test process of a regulated software project using continuous delivery pipelines*. (Bachelor's thesis, Technical University Ingolstadt of Applied Sciences, Ingolstadt, Germany).

[21] Sharma, V., Shrivastava, V., Pandey, A., & Gupta, P. (2024). A basic introduction to DevOps. *International Journal of Research Publication and Reviews*, 5(3), 725-731.

[22] Singh, A., & Mansotra, V. (2021). A comparison on continuous integration and continuous deployment (CI/CD) on cloud based on various deployment and testing strategies. *International Journal for Research in Applied Science and Engineering Technology*, 9(6), 4968-4977. doi: 10.22214/ijraset.2021.36038.

[23] Singh, V. (2022). *Developing a CI/CD pipeline with GitLab*. (Bachelor's thesis, Turku University of Applied Sciences, Turku, Finland).

[24] Smart, J.F. (2011). *Jenkins: The definitive guide*. Sebastopol: O'Reilly Media, Inc.

[25] Vatsyk, Yu. (2024). *Research and analysis of the relevance of Kubernetes in the modern IT market*. (Master's thesis, King Danylo University, Ivano-Frankivsk, Ukraine).

[26] Waseem, M., Liang, P., & Shahin, M. (2020). A systematic mapping study on microservices architecture in DevOps. *Journal of Systems and Software*, 170, article number 110798. doi: 10.1016/j.jss.2020.110798.

[27] Zhou, N., Georgiou, Y., Pospieszny, M., Zhong, L., Zhou, H., Niethammer, C., Pejak, B., Marko, O., & Hoppe, D. (2021). Container orchestration on HPC systems through Kubernetes. *Journal of Cloud Computing*, 10(1), article number 16. doi: 10.1186/s13677-021-00231-z.

Стратегії впровадження або посилення підходу DevOps в організаціях: аналіз та приклади

Богдан Федоришин

Магістр
Національний університет «Львівська політехніка»
79000, вул. Степана Бандери, 12, м. Львів, Україна
https://orcid.org/0009-0005-3779-0186

Анотація. Мета роботи полягала в аналізі впровадження DevOps в організаціях, зокрема в оцінці впливу автоматизації процесів, CI/CD, моніторингу та оркестрації мікросервісів на ефективність розробки та управління. Було створено методологію, яка дозволяє організаціям ефективно впроваджувати та посилювати підхід DevOps, досягаючи високих результатів у розвитку і управлінні програмним забезпеченням. У дослідженні було розглянуто стратегії культурної зміни та навчання, що включають формування культури співпраці між командами та розвиток навичок у сучасних інструментах DevOps. Автоматизація процесів, зокрема автоматизація тестування та інтеграція та розгортання, відіграє важливу роль у підвищенні якості коду і зменшенні часу розробки. Інфраструктура як код дозволяє управляти та автоматизувати конфігурацію інфраструктури, забезпечуючи швидке масштабування середовищ. Робота виявила, що моніторинг продуктивності та зворотний зв'язок є критичними для виявлення проблем на ранніх етапах і постійного вдосконалення продукту. Оркестрація мікросервісів, зокрема за допомогою Kubernetes, забезпечує автоматизацію розгортання і масштабування контейнеризованих додатків, що критично важливо для сучасних архітектур з високими вимогами до продуктивності та доступності. Приклади з практики таких компаній, як Netflix, Spotify і Airbnb, демонструють успішне застосування DevOps практик і технологій для досягнення високої доступності, масштабованості та надійності інфраструктур. Результати дослідження підтверджують, що впровадження DevOps веде до значного підвищення ефективності розробки, покращення якості програмного забезпечення та оптимізації витрат на управління інфраструктурою. Крім того, впровадження культурних змін і підвищення рівня співпраці між командами розробки та операцій сприяє більшій гнучкості та швидкості реагування на зміни ринкових умов

Ключові слова: оркестрація мікросервісів; інструменти; автоматизація тестування; CI/CD процеси; ефективність розробки