DOI: https://doi.org/10.36910/6775-2524-0560-2024-57-15
УДК 004.4:004.415:004.678:004.7:004.9
Орлов Микола Вікторович[1], аспірант
https://orcid.org/0000-0002-5361-8854
Дуда Олексій Михайлович[2], к.т.н., доцент
https://orcid.org/0000-0002-5828-067X
Жовнір Юрій Іванович[1], аспірант
https://orcid.org/0009-0006-6186-2861
Грибовський Олег Миколайович [1], аспірант
https://orcid.org/0009-0005-6318-3611
[1]Національний університет «Львівська політехніка», м. Львів, Україна
[2]Тернопільський технічний університет ім. Івана Пулюя, м. Тернопіль, Україна

ІНСТРУМЕНТИ МЕТОДОЛОГІЇ DEVOPS В ІНФОРМАЦІЙНИХ СИСТЕМАХ НА ОСНОВІ ТЕХНОЛОГІЙ ІОТ

Орлов М. В., Дуда О. М., Жовнір Ю. І., Грибовський О.М. Інструменти методології DevOps в інформаційних системах на основі технологій IoT. В статті розглядається основні інструменти методології DevOps та їх застосування в інформаційних системах, що базуються на технологіях Інтернету речей (IoT). Зростаюча кількість IoT-пристроїв, які постійно генерують великі обсяги даних, вимагає ефективного підходу до управління процесами розробки, тестування та розгортання програмного забезпечення. Методологія DevOps забезпечує автоматизацію та інтеграцію цих процесів, дозволяючи зменшити затримки та підвищити якість сервісів. У роботі акцентується увага на інструментах CI/CD (безперервної інтеграції та доставки) і контейнеризації, які є ключовими для досягнення надійності та гнучкості в IoT-екосистемах. Розглядаються конкретні сценарії застосування DevOps в IT-інфраструктурі, зокрема опрацювання даних на периферійних та хмарних платформах, автоматизація управління інфраструктурою та забезпечення кібербезпеки. Крім того, обговорюються особливості моніторингу та обслуговування IoT-рішень у режимі реального масштабу часу з використанням DevOps інструментів для виявлення й усунення помилок. Представлені рекомендації допоможуть спеціалістам краще зрозуміти можливості DevOps для створення масштабованих та безпечних IoT-систем, що відповідають сучасним вимогам бізнесу та кібербезпеки.
Ключові слова: CI/CD (Безперервна інтеграція та доставка); автоматизація; хмарна інфраструктура; моніторинг у реальному часі; кібербезпека; масштабування; управління життєвим циклом програмного забезпечення; інтеграція апаратного та програмного забезпечення.

Orlov M., Duda O., Zhovnir Yu., Hrybovskyi O. DevOps methodology tools in information systems based on IoT technologies. The article provides an in-depth analysis of the primary tools within the DevOps methodology and their implementation in information systems based on Internet of Things (IoT) technologies. As IoT devices multiply at a rapid pace and continually produce large volumes of data, managing the software development, testing, and deployment processes efficiently becomes increasingly critical. DevOps methodology presents a structured approach to automation and integration, promoting collaboration across development, operations, and security teams. This integration reduces latency in system updates and increases service quality by enabling continuous iteration. Key DevOps practices, such as Continuous Integration and Continuous Delivery (CI/CD), alongside containerization, are highlighted as foundational to achieving the scalability, reliability, and adaptability required for IoT ecosystems. The article investigates various use cases of DevOps in IT infrastructures, emphasizing data processing on both edge devices and cloud platforms. Edge computing, in particular, supports real-time processing and data analysis closer to IoT endpoints, while cloud platforms offer scalable resources for extended storage, analytics, and archiving. DevOps tools facilitate seamless synchronization between these layers, allowing the IoT system to meet performance demands effectively. The study further explores the automation of infrastructure management through Infrastructure as Code (IaC) practices, which enable developers and operators to define, deploy, and monitor infrastructure in a highly consistent and repeatable way, crucial for IoT's distributed environments. Cybersecurity is another significant focus, as IoT systems are often vulnerable to network-based attacks due to their interconnected nature. By integrating DevSecOps practices—embedding security into every stage of development and deployment—IoT solutions gain automated security checks that mitigate vulnerabilities without slowing down the release process. These security practices include automated vulnerability scans, code analysis, and compliance checks that are conducted continuously to ensure robust protection. The article also emphasizes the importance of real-time monitoring and logging of IoT components, which DevOps tools support to detect and troubleshoot errors immediately. With centralized monitoring dashboards and automated alerting systems, teams can rapidly respond to incidents, maintaining stable operation across the IoT ecosystem. The recommendations outlined in the article are aimed at industry professionals seeking to leverage DevOps to enhance the efficiency and resilience of IoT infrastructures. The adoption of DevOps practices in IoT not only accelerates innovation but also supports the development of reliable, secure, and scalable IoT systems that align with modern business and cybersecurity standards. By fostering a culture of continuous improvement, DevOps enables organizations to integrate new IoT features and optimize performance without compromising stability or security, positioning it as a valuable methodology for sustainable IoT growth.
Keywords: CI/CD (Continuous Integration and Delivery); automation, cloud infrastructure; real-time monitoring; cybersecurity; scaling; software lifecycle management; hardware-software integration.

Постановка наукової проблеми. Зростаюча популярність технологій Інтернету речей (IoT) у різних галузях — від житлових комплексів до інфраструктури міст — породжує нові вимоги до інформаційних систем, зокрема у питаннях масштабованості, надійності, швидкості розгортання та забезпечення кібербезпеки. Через велику кількість підключених пристроїв і величезний обсяг даних, що опрацьовуються в режимі реального масштабу часу, такі системи потребують ефективного управління, швидкої адаптації до змін та мінімізації простоїв. Традиційні методології розроблення не завжди можуть забезпечити необхідний рівень продуктивності та безпеки в умовах швидкого розвитку IoT-рішень.

Методологія DevOps, що об'єднує процеси розроблення і операційного супроводу з акцентом на автоматизацію і безперервну інтеграцію, має потенціал значно покращувати ефективність управління IoT-системами. Проте використання DevOps у контексті IoT супроводжується низкою специфічних викликів, таких як інтеграція програмного та апаратного забезпечення, забезпечення надійної роботи в умовах швидкозмінного навантаження, а також захист від кіберзагроз. Існує потреба у глибокому дослідженні того, як конкретні інструменти DevOps можуть бути оптимально адаптовані для управління інформаційними системами на основі IoT, забезпечуючи їхню стабільність, безперебійність та масштабованість.

Отже, постає проблема визначення найбільш ефективних інструментів та практик DevOps, які можуть підтримувати потреби IoT-систем, а також розробки рекомендацій щодо їхнього впровадження для досягнення високої продуктивності та безпеки у сучасних інформаційних системах.

Аналіз досліджень. Проведений авторами аналіз наукових публікацій дозволив з'ясувати множину інструментів методології DevOps в інформаційних системах на основі технологій IoT. Аналіз проводився за такими основними напрямами.

Автоматизація процесів CI/CD у IoT-середовищах

Кітгенгальмом Б. та Чартером С. [1] запропоновано методику дослідження, і в якій детально подаються процеси побудови системних аналітичних оглядів літератури в галузі програмної інженерії. На думку дослідників, реалізація підходу розпочинається з визначення чітких і конкретних дослідницьких питань, на які слід дати відповідь за результатами аналітичних процедур. Автори статті [2] розглянули особливості автоматизації процесів безперервної інтеграції та доставки (CI/CD) та окреслили її роль у впровадженні нових функцій у IoT-системи. У статті [3] відзначено, що процедури контролю версій, тестування, контейнеризації, моніторингу і забезпечення рівня безпеки сприяють ефективному розгортання систем. В роботі [4] наголошено на вагомій ролі методології DeVops для покращення процесів автоматизації розроблення інформаційних систем.

Роль DevOps у забезпеченні масштабованості та гнучкості IoT-рішень

Ряд авторів у своїх дослідженнях наголошують на перевагах методології DevOps у швидкій адаптації IoT-систем до змінних вимог та зростаючого навантаження. Методологія DevOps розглядається як високотехнологічний процес розроблення та поширення програмного забезпечення. Він поєднує в собі культуру співпраці та консолідації підходів до управління, технологій кодування та системних технік інтеграції. Методологія DevOps подається як нова інноваційна парадигма та концепція, яка впроваджується в галузі IT і стосується програмного забезпечення та вбудованих систем в таких аплікаціях як робототехніка та інтелектуальні агенти. Екосистема IoT подається як набір фізичних пристроїв, таких як давачі та виконавчі пристрої, зінтегрованих в розлогих мережевих структурах, що містять набори серверів та шлюзів, які забезпечують надійне їх підключення та взаємодію. Зазначені пристрої можуть інсталюватись на трьох рівнях, що реалізують три обчислювальні концепції, а саме пограничних обчислень, туманних обчислень та хмарних обчислень [5].

У ряді робіт аналізуються питання, як саме розподілені хмарні середовища та інфраструктура як код (IaC) полегшують масштабування великих IoT-мереж. Дослідники вважають безперервну інтеграцію [6] процесом розроблення, у якому розробники та інтегратори імплементують код у спільні інструменти. Зазначена практика дозволяє автоматизувати збірку та тестування нових версій програмного забезпечення. Це базова практика, яка підтримує безперервність роботи програмного забезпечення розумних агентів, які є основою систем, на базі технологій Інтернету речей. Базуючись на конкретних характеристиках, були розроблені та розгорнуті інформаційні системи з використанням ряду фреймворків для систем на базі IoT. Автори запропонували метамодель

інформаційних технологій Інтернету речей, що спрямовано на стандартизацію існуючих архітектур систем на базі IoT, а також тих, які будуть запропоновані в майбутньому. У [7] роботі відзначається, що сучасні системи на основі технологій IoT є розподіленими розлогими мережевими системами, які об'єднують підходи, засновані на хмарних, пограничних і туманних обчисленнях в залежності від методики та способу розподілення обчислювальних можливостей, опрацювання та захисту даних. Такий розподіл і неоднорідність обчислювальних середовищ роблять конвеєри розроблення та розгортання доволі складними та фрагментованими з кількома кінцевими точками доставки. В свою чергу це перешкоджає швидкому розвитку систем та перетворює роботу та моніторинг такого класу систем складним і виснажливим завданням.

Забезпечення безпеки в системах на основі технологій IoT за допомогою методології DevSecOps

У ряді публікацій дослідники зосереджуються на імплементації підходів методології DevSecOps для забезпечення кібербезпеки у складних IoT-екосистемах. В роботах подаються описи того, як автоматизовані перевірки рівнів безпеки та інтеграція систем моніторингу сприяють зменшенню ризиків атак і підвищенню надійності IoT-рішень. Автори [8] вдало визначили та формалізували діяльність, яка підтримує «швидкий і безперервний зворотний зв'язок від операціоністів до розробників», формулюючи гнучку інфраструктуру моніторингу, для того щоб команди могли налаштувати свої служби моніторингу та оповіщення відповідно до певних критеріїв з метою отримання швидкого і безперервного зворотнього зв'язку для кращого передбачення проблем під час розгортання виробництва.

Моніторинг у реальному часі як частина DevOps-процесів у системах, базованих на технологіях IoT

Дослідники обшрунтовують доцільність використання моніторингових інструментів у IoT-системах, що сприяє оперативному реагуванню на зміни у роботі системи та виявленню відхилень[5]. Заслуговує уваги подана метамодель, до складу якої входить 5 основних компонентів: IoT, мікросервіси, Ansible, Docker та Kubernetes [6].

Результати аналізу наукових публікацій дозволяють відзначити, що методологія DevOps є ключовою методологією підвищення ефективності та надійності IoT-екосистем, а її інструменти сприяють ефективному вирішенню ряду задач.

Метою даної роботи є аналіз та обґрунтування ефективності використання інструментів методології DevOps у розробленні та підтримці інформаційних систем на основі технологій Інтернету речей (IoT). Дослідження має на меті визначити, як DevOps сприяє оптимізації процесів розроблення, тестування, розгортання і обслуговування IoT-систем, забезпечуючи підвищення швидкості, надійності, та кібербезпеки у виконанні цих процесів.

Завдання дослідження**:**

1. Проаналізувати основні інструменти методології DevOps, такі як CI/CD, контейнеризація та автоматизація інфраструктури, та оцінити їхнє застосування в IoT-інфраструктурі.
2. Дослідити специфіку використання DevOps у розподілених IoT-системах, зокрема для опрацювання даних на рівні периферії (edge) та у хмарних середовищах.
3. Оцінити роль DevSecOps у підвищенні рівня безпеки IoT-систем через впровадження автоматизованих перевірок безпеки на кожному етапі розроблення та розгортання.
4. Визначити рівень ефективності автоматизованого моніторингу та обслуговування IoT-рішень за допомогою DevOps для швидкого виявлення та усунення помилок.
5. Сформулювати рекомендації для розробників та інженерів з використання DevOps в IoT-системах для забезпечення масштабованості, надійності та відповідності сучасним бізнес- та кібербезпековим вимогам.

Об'єктом дослідження є процеси розроблення, тестування, розгортання та підтримки інформаційних систем на основі технологій Інтернету речей (IoT), що використовують методологію DevOps для підвищення їх ефективності, надійності та безпеки.

Предметом дослідження є інструменти методології DevOps, зокрема автоматизація CI/CD (безперервної інтеграції та доставки), контейнеризація, інструменти для автоматизації інфраструктури та засоби моніторингу, і їхнє застосування в розподілених IoT-екосистемах. У межах дослідження використовувались сучасні наукові підходи для аналізу процесів розроблення, інтеграції та підтримки IoT-рішень із застосуванням принципів DevOps.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження. У цьому дослідженні автори розглядають інструменти методології DevOps в контексті покращення розроблення інформаційних систем, що базуються на технологіях IoT. Запровадження методології DevOps загалом сприяє тіснішій співпраці команд розробників, тестувальників та операційних фахівців. Це в свою чергу забезпечує більш ефективну комунікацію та швидке вирішення проблем, які виникають у процесі розроблення та розгортання масштабних інформаційних систем, які реалізуються у вигляді відповідних IoT-застосунків. Основна увага цього дослідження зосереджена на аналізі зв'язків між методологією DevOps та програмними комплексами, що реалізують функціонал інформаційних систем, втілених у вигляді розлогих мереж на основі великої кількості IoT пристроїв. Проведене авторами статті аналітичне дослідження дає підстави виокремити шість груп інструментів, що використовуються як компоненти методології DevOps.

Інструменти CI/CD:

1. **AWS CodePipeline** (https://console.aws.amazon.com/codesuite/ codepipeline/home) – CI/CD платформа від Amazon Web Services, яка автоматизує процеси збирання, тестування та доставки застосунків. Для IoT пристроїв та систем підтримує повну інтеграцію з екосистемою AWS, включаючи AWS IoT Core для керування IoT-пристроями. Це дає можливість створювати конвеєри для безперервної доставки IoT-застосунків в хмару та до відповідних пристрої.
2. **Azure DevOps** (https://dev.azure.com/) – комплексний набір інструментів від Microsoft, який включає системи CI/CD, управління вихідним кодом, тестування та моніторинг. Azure DevOps добре інтегрується з Azure IoT Hub, що робить його вдалим вибором для проєктів, в яких використовуються хмарні рішення Azure. Забезпечує масштабоване розгортання на різні пристрої та хмарні сервіси.
3. **CircleCI** (https://circleci.com/) – автоматизована CI/CD платформа, що дає можливість ефективно запускати тести, збірку і розгортання. Підтримує Docker і кросплатформне розгортання, що важливо для тестування різних прошивок і операційних систем IoT-пристроїв.
4. **GitLab CI/CD** (https://about.gitlab.com/) – CI/CD платформа від GitLab, яка інтегрована з системою контролю версій Git, що для IoT пристроїв та систем дає змогу швидко тестувати та розгортати IoT-застосунки на великій кількості пристроїв, завдяки легкій інтеграції з контейнеризацією (наприклад, Docker) та хмарними платформами.
5. **Jenkins** (https://www.jenkins.io/) – один із найпопулярніших CI/CD інструментів з відкритим кодом. Забезпечує автоматизацію збірки, тестування та розгортання ПЗ на IoT-пристроях, підтримує різні плагіни, інтегрується з багатьма інструментами для опрацювання IoT-даних, масштабованих хмарних платформ, а також з пристроями, що працюють в різних операційних середовищах.
6. **Spinnaker** (https://spinnaker.io/) – інструмент для багатоплатформного управління релізами, який інтегрується з Kubernetes, AWS та іншими. Це для IoT пристроїв та систем спрощує процеси розгортання і масштабування застосунків, даючи змогу керувати складними IoT-інфраструктурами.
7. **TeamCity** (jetbrains.com/teamcity/) – CI/CD сервер від JetBrains з розширеними можливостями для управління збірками. Підтримує роботу з різними мовами програмування, що корисно при розробленні багатоплатформних IoT-рішень.
8. **Travis CI** (https://www.travis-ci.com/) – CI платформа для автоматизації збірки і тестування ПЗ. Для IoT пристроїв та систем підтримує декілька мов програмування та операційних систем, що робить його корисним для різних IoT-екосистем.

Інструменти для управління конфігураціями:

1. **Ansible** (https://www.ansible.com/) – інструмент для автоматизації конфігурацій та управління інфраструктурою. Завдяки простоті використання та можливості працювати без агентів (agentless). Ansible підходить для налаштування IoT-пристроїв, забезпечуючи автоматизацію процесів розгортання, оновлення і управління конфігураціями через сценарії playbooks. Може легко масштабуватися для тисяч IoT-пристроїв.
2. **AWS IoT Device Management** (https://aws.amazon.com/iot-device-management/) – хмарний сервіс від Amazon для управління IoT-пристроями. AWS IoT Device Management надає можливість масштабовано управляти конфігураціями тисяч пристроїв у режимі реального часу, централізовано відстежувати стан конфігурацій, проводити віддалені оновлення і спостереження їх стану.

3. **Azure IoT Hub Device Management** (https://azure.microsoft.com/en-us/products/iot-hub/) – хмарне програмно-алгоритмічне рішення від Microsoft для управління IoT-пристроями через Azure IoT Hub. Azure IoT Hub надає можливості для централізованого управління конфігураціями, оновлення програмного забезпечення і керування IoT-пристроями. Це полегшує процеси інтеграції з іншими Azure-сервісами і дає можливість автоматизувати управління великими IoT-інфраструктурами.
4. **Balena** (https://www.balena.io/) – програмно-алгоритмічна платформа для розгортання і управління IoT-пристроями на базі Linux. Balena надає можливість управляти конфігураціями та розгортати оновлення на віддалені IoT-пристрої за допомогою контейнерів. Вона добре підходить для керування флотами пристроїв і забезпечує безперервну доставку програмного забезпечення через хмару.
5. **Chef** (https://www.chef.io/) – потужний інструмент для управління конфігураціями на основі коду. Chef дає можливість автоматизувати процеси налаштування середовищ для IoT-пристроїв, а також відслідковувати стан конфігурацій. Його підхід "інфраструктура як код" дає змогу масштабувати процеси управління конфігураціями для тисяч пристроїв.
6. **Consul (HashiCorp)** (https://www.consul.io/) – система для управління сервісами та конфігураціями. Consul забезпечує централізоване управління конфігураціями для розподілених IoT-пристроїв, даючи змогу автоматично відстежувати зміни і налаштовувати пристрої відповідно до нових вимог.
7. **Mender** (https://mender.io/) – інструмент для управління оновленнями прошивки та конфігураціями IoT-пристроїв. Mender надає можливості централізованого управління прошивками і конфігураціями на віддалених IoT-пристроях. Це забезпечує надійну доставку оновлень програмного забезпечення, що критично важливо для підтримки працездатності IoT-мереж.
8. **Puppet** (https://www.puppet.com/) – популярний інструмент для управління процесами конфігурування та автоматизації інфраструктури. Використовується для управління інфраструктурою як кодом (IaC). Puppet забезпечує централізоване управління конфігураціями для IoT-пристроїв. Це дає змогу автоматизувати процес налаштування IoT-пристроїв, незалежно від їхнього географічного розташування, і легко оновлювати конфігурації при зміні вимог.
9. **SaltStack (Salt)** (saltstack.com) – інструмент для управління конфігураціями та оркестрування. SaltStack забезпечує швидке розгортання та управління конфігураціями для великих кількостей IoT-пристроїв. Підтримує агентні та безагентні сценарії, що підходить для різних IoT-архітектур. Salt також має можливості управління мережею та хмарною інфраструктурою.

Моніторинг та логування:

1. **AWS CloudWatch -** сервіс моніторингу від Amazon, який дозволяє відстежувати ресурси AWS, включаючи IoT-пристрої, і налаштовувати оповіщення на основі заданих метрик.
2. **Azure Monitor** (https://azure.microsoft.com/en-us/services/monitor/) – платформа моніторингу від Microsoft, яка забезпечує моніторинг ресурсів Azure, включаючи IoT-пристрої. Вона надає інструменти для збору та аналізу логів і метрик.
3. **Datadog** (https://www.datadoghq.com/) – хмарна платформа для моніторингу і аналітики, яка підтримує IoT. Вона дає змогу відстежувати IoT-пристроїв, використання ресурсів та інші показники.
4. **ELK Stack** (Elasticsearch, Logstash, Kibana, https://www.elastic.co/what-is/elk-stack) – це популярний стек програмно-алгоритмічних засобів для збору, опрацювання та візуалізації логів. Logstash використовується для збору і опрацювання даних IoT-простроїв та систем, Elasticsearch для зберігання та пошуку, а Kibana для візуалізації.
5. **Fluentd** (https://www.fluentd.org/) – інструмент для збору, опрацювання та передачі логів, який може бути використаний для агрегації логів з IoT-пристроїв і подальшого їх опрацювання.
6. **Grafana** (https://grafana.com/blog/2015/04/20/grafana-2.0-released/) – інструмент для візуалізації даних, який часто використовується разом з Prometheus. Він надає можливості для створення інтерактивних панелей для моніторингу та візуалізації метрик IoT-пристроїв.
7. **Nagios** (https://www.nagios.org/) – інструмент для моніторингу систем і мереж, який дає можливість контролювати стан IoT-пристроїв і отримувати сповіщення про проблеми.
8. **Prometheus** (https://prometheus.io/) – система моніторингу і збору метрик, яка підтримує запис метрик у часі, що полегшує процедури спостереження стану та оповіщення IoT-пристроїв.

9. **Splunk** (https://www.splunk.com/) – комерційна платформа для збору, аналізу і візуалізації даних з різних джерел, яка підходить для моніторингу IoT-застосунків.

10. **Zabbix** (https://www.zabbix.com/) – платформа для моніторингу, яка підтримує різноманітні типи даних і надає можливості для контролю IoT-пристроїв та спостереження їхнього стану в режимі реального часу.

Контейнери та оркестрування:

1. **Balena** (https://www.balena.io/) – платформа для управління IoT-пристроями та контейнерами, яка забезпечує просте розгортання і оновлення програмного забезпечення на віддалених пристроях.

2. **Docker** (https://www.docker.com/products/docker-desktop/) – найпопулярніший інструмент для контейнеризації, який дає змогу упаковувати застосунки та їх залежності в контейнери, що робить їх легкими для розгортання в будь-якому середовищі.

3. **Docker Compose** (https://docs.docker.com/compose/) – інструмент для визначення та запуску багатоконтейнерних Docker-застосунків, що полегшує конфігурацію та управління залежностями між контейнерами.

4. **Helm** (https://helm.sh/) – менеджер пакетів для Kubernetes, який спрощує процеси управління застосунками в контейнерах шляхом використання шаблонів для їх розгортання.

5. **Istio** (https://istio.io/) – платформа для управління сервісами в середовищах Kubernetes, яка забезпечує реалізацію функцій безпеки, моніторингу та управління трафіком IoT-пристроїв та систем.

6. **Kubernetes** (https://kubernetes.io/) – система оркестрування контейнерів, яка автоматизує процеси розгортання, масштабування та управління контейнеризованими застосунками. Вона особливо корисна для IoT, оскільки дозволяє ефективно управляти великими кількостями контейнерів на різних пристроях.

7. **OpenShift** (https://www.openshift.com/) – платформа на базі Kubernetes, що забезпечує управління контейнерами з додатковими можливостями CI/CD, управління безпекою та інтеграції з іншими сервісами.

8. **Rancher** (https://rancher.com/) – платформа для управління Kubernetes, яка спрощує процеси розгортання і супроводу контейнеризованих IoT-застосунків, забезпечуючи централізоване управління.

Інструменти для управління версіями:

1. **Bitbucket** (https://bitbucket.org/) – популярний сервіс для хостингу Git-репозиторіїв, який надає функціональні можливості інтеграції з CI/CD та управління проектами через Jira.

2. **Git** (https://git-scm.com/) – найпопулярніша система контролю версій, яка дає змогу розробникам відстежувати зміни в коді, співпрацювати над проектами та управляти різними версіями застосунків. GitHub, GitLab і Bitbucket є популярними хостинг-сервісами для Git-репозиторіїв.

3. **GitHub** (https://github.com/) – Платформа для хостингу Git-репозиторіїв, яка також надає інструменти для управління проектами, безперервної інтеграції та доставки (CI/CD), а також обговорення та планування функцій.

4. **GitLab** (https://about.gitlab.com/) – платформа для управління Git-репозиторіями, яка інтегрує CI/CD, управління проектами і функції для моніторингу. GitLab дозволяє легко відстежувати зміни в IoT-застосунках.

5. **Semantic Versioning** (SemVer, https://semver.org/) – методологія версіювання, яка надає правила для управління версіями програмного забезпечення, зокрема для IoT-застосунків.

Інструменти для тестування:

1. **Appium** (https://appium.io/) – інструмент для автоматизованого тестування мобільних застосунків, включно з мобільними IoT-застосунками.

2. **Cucumber** (https://cucumber.io/) – інструмент для поведінкового тестування (BDD), що дає змогу писати тести у людинно сприйнятній формі, що може бути корисним для команд розробників, що працюють з IoT-пристроями та системами.

3. **Device Farm (AWS,** https://aws.amazon.com/device-farm/**) –** Хмарна платформа для тестування мобільних і веб-застосунків на реальних пристроях та системах, яка також може бути використана для тестування IoT-пристроїв.

4. **IoT Device Simulator** – інструменти, які можуть імітувати роботу IoT-пристроїв для тестування IoT-застосунків без необхідності використовувати фізичні пристрої. Функціональні можливості IoT Device Simulator реалізовані в інформаційно технологічних платформах AWS (https://aws.amazon.com/iot-device-simulator/), Azure (https://azure.microsoft.com/en-us/services/iot-hub/device-simulation/) та Google (https://cloud.google.com/iot/docs/simulator).
5. **JUnit** (https://junit.org/junit5/) – популярний фреймворк для юніт-тестування в Java. Використовується для тестування компонентів IoT-застосунків, написаних на Java.
6. **Katalon Studio** (https://www.katalon.com/) – інтегрований інструмент для автоматизації тестування, який підтримує веб-застосунки, мобільні застосунки та API, що може бути корисно для IoT.
7. **JMeter** (https://jmeter.apache.org/) – інструмент для тестування навантаження, який може використовуватися для перевірки продуктивності IoT-систем при високих навантаженнях.
8. **Mockito** (https://site.mockito.org/) – фреймворк для створення мок-об'єктів у Java. Використовується в поєднанні з JUnit для тестування ізоляції компонентів IoT.
9. **Postman** (https://www.postman.com/downloads/) – інструмент для тестування API, який дає можливість перевіряти взаємодію IoT-пристроїв із серверами через HTTP-запити.
10. **Robot Framework** (https://robotframework.org/) – фреймворк для автоматизації процесів тестування, який підтримує різні типи тестів, включаючи тестування IoT.
11. **Selenium** (https://www.selenium.dev/) – інструмент для автоматизованого тестування веб-застосунків. Може використовуватися для тестування інтерфейсів IoT-застосунків.
12. **TestComplete** (https://smartbear.com/product/testcomplete/overview/) – інструмент для автоматизованого тестування, який підтримує різні платформи, включаючи IoT-застосунки.

Подані вище групи інформаційно-технологічних інструментів дозволяють забезпечити безперервну інтеграцію та доставку, автоматизацію процесів, моніторинг та управління конфігураціями, що є критично важливими для успішного розроблення IoT-застосунків [10-13].

В процесі дослідження було також ідентифіковано мови програмування, які використовуються найчастіше [9]. На рисунку наведено розподіл частот практичного використання мов програмування в IoT-застосунках.

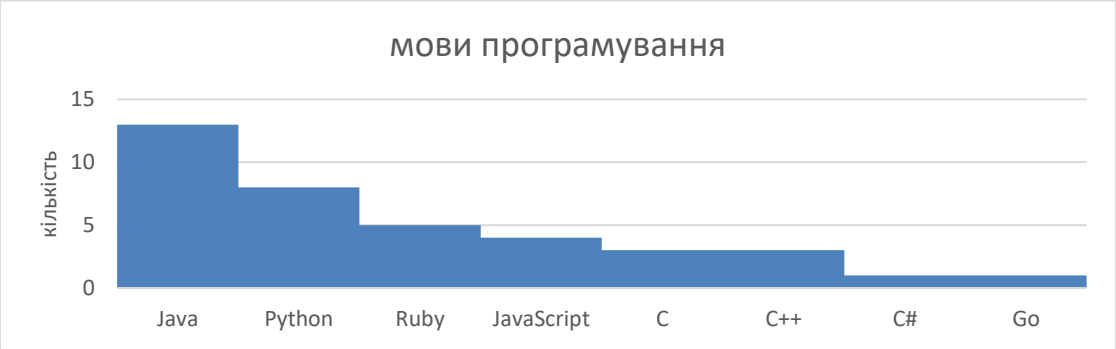

Рис. 1. Розподіл частот використання мов програмування в IoT-застосунках [9]

Впровадження спеціальних методів та практик методології DevOps сприяє забезпеченню належного рівня безпеки в IoT-застосунках. Розглянемо декілька ключових методів убезпечення DevOps процесів, які використовуються в IoT-застосунках.

1. *Безпечне програмування*

• Статичний та динамічний аналіз коду використовуються як інструменти для автоматизованого аналізу коду з метою виявлення вразливостей на ранніх стадіях розроблення.

• Використання безпечних бібліотек і фреймворків сприяє обиранню перевірених бібліотек з активною підтримкою і регулярними оновленнями безпекових можливостей.

2. *Ідентифікація і управління доступом*

• Аутентифікація та авторизація відбувається з використанням методів аутентифікації (наприклад, двофакторна аутентифікація) і забезпечує коректний контроль доступу до пристроїв і даних. Приклад аналогічного підходу наведено на Рис.2.

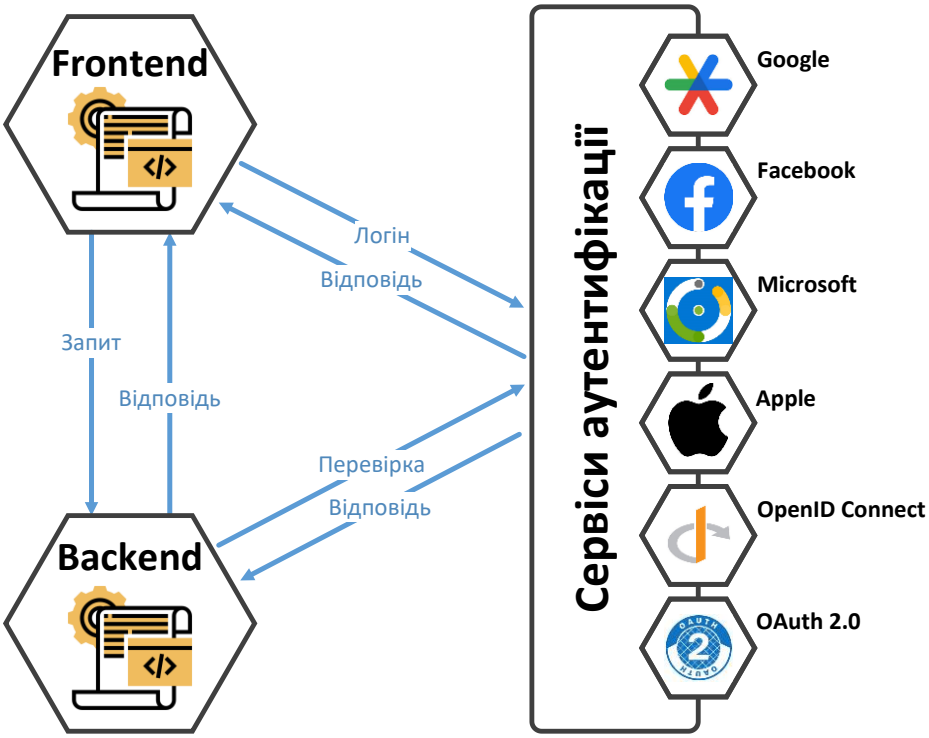

Рис. 2. Приклад реалізації управління доступом

• Ідентифікація пристроїв застосовується до кожного IoT-пристрою, щоб мати можливість контролювати і обмежувати доступ IoT пристроїв до мережі.

3. *Шифрування даних*

• Шифрування в спокої та в процесах передачі забезпечується як під час їх зберігання на пристроях, так і під час передачі між пристроями та серверами.

• Використання VPN і TLS як захищених каналів зв'язку, такі як VPN і протоколи TLS, для захисту даних під час передачі.

4. *Моніторинг та логування*

• Централізоване логування забезпечує збір і аналіз логів з усіх IoT-пристроїв і серверів у централізованій системі для виявлення аномалій та інцидентів безпеки.

• Моніторинг у реальному масштабі часу забезпечує оперативне реагування на можливі атаки. Приклад подібного наведено на Рис.3.

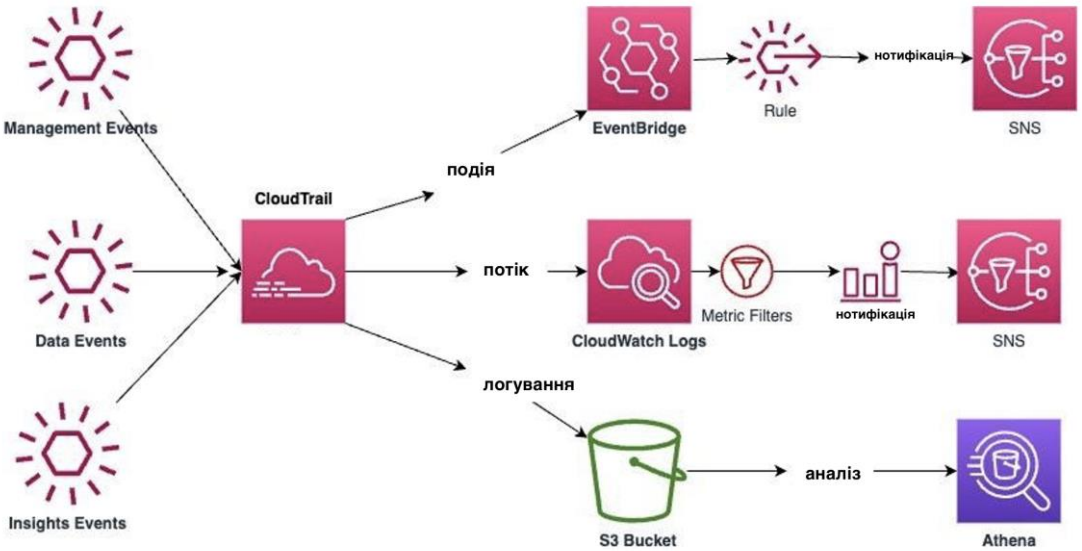

Рис 3. Архітектура системної реалізації функцій монітрингу і логування у хмарному провайдері Амазон [14]

5. *Оновлення та патчинг*

• Регулярні оновлення ПЗ забезпечують можливість регулярного оновлення програмного забезпечення на IoT-пристроях для усунення вразливостей.

• Автоматичне розповсюдження патчів здійснюється з метою мінімізації часу між виявленням вразливості та моментом її усуненням.

6. *Контейнеризація та ізоляція*

• Контейнери та віртуалізація використовуються для ізоляції різних компонентів системи, що зменшує ризик поширення атак.

• Мікросервісна архітектура сприяє розподілу IoT-застосунків на окремі мікросервіси, що дозволяє краще контролювати і захищати кожен компонент.

7. *Безперервна інтеграція та безперервне постачання (CI/CD)*

• Автоматизоване тестування безпеки у CI/CD процесах використовується для регулярної перевірки коду на вразливості.

• Інфраструктура як код (IaC) використовують для забезпечення узгодженості та автоматизації налаштувань безпеки.

8. *Планування і реагування на інциденти*

• План реагування на інциденти розробляють, щоб швидко і ефективно діяти у випадку атаки.

• Постійне навчання та тренування регулярно проводять для команди з метою підвищення їхньої обізнаності та готовності до дій у випадку інцидентів.

Впровадження цих методів дозволяє створити багаторівневу систему безпеки, яка забезпечує надійний захист IoT-застосунків на всіх етапах їх життєвого циклу.

Детальніше проаналізуємо як саме методологія DevOps сприяє масштабованості IoT-систем. Автоматизоване розгортання дозволяє швидко і безпечно розгортати нові версії програмного забезпечення на великій кількості IoT-пристроїв, що значно полегшує масштабування систем. Інфраструктура як код використовується для зручного створення, конфігурування і керування інфраструктурою через код, що полегшує масштабування інфраструктури в хмарі або на локальних серверах. CI/CD процеси забезпечують автоматизоване тестування коду, що гарантує його якість і готовність до масштабування. Це в свою чергу дозволяє швидко випускати оновлення і реалізовувати нові функції, що забезпечує гнучкість і адаптивність IoT-систем до зростання кількості пристроїв та користувачів. Розподіл навантаження з використанням мікросервісної архітектури дозволяє розподіляти навантаження між різними сервісами, що полегшує масштабування окремих компонентів системи. Кожен мікросервіс можна масштабувати незалежно від інших, що забезпечує гнучкість і ефективність в управлінні ресурсами. Контейнери забезпечують легке і швидке розгортання застосунків, що спрощує масштабування. Використання оркестраторів контейнерів, таких як Kubernetes, дозволяє автоматизувати управління контейнерами і забезпечити їх масштабованість. Централізований моніторинг забезпечує постійний контроль за станом IoT-пристроїв і застосунків, дозволяючи оперативно виявляти і усувати проблеми.

Аналіз журналів сприяє збору і аналізу логів, розумінню поведінки системи і прийняттю обґрунтованих рішень щодо її масштабування. Використання систем оркестрування, таких як Kubernetes, дозволяє автоматично масштабувати застосунки на основі заданих правил і метрик. Автоматичне додавання нових екземплярів сервісів для опрацювання збільшеного навантаження забезпечує ефективне горизонтальне масштабування. Хмарні провайдери дозволяють динамічно виділяти ресурси відповідно до поточного навантаження, що забезпечує масштабованість без значних витрат. Сервернес обчислення дозволяють масштабувати опрацювання подій і функцій без необхідності управління серверами, що забезпечує високу гнучкість.

Тісна співпраця між командами розробки та експлуатації сприяє швидкому вирішенню проблем і адаптації до змін. Використання методології Agile та ітеративного розвитку дозволяє постійно вдосконалювати IoT-системи і забезпечувати їх масштабованість.

Впровадження цих методів і практик DevOps дозволяє IoT-системам ефективно адаптуватися до зростання кількості підключених пристроїв, збільшення обсягу даних і змін вимог з боку бізнесу.

Висновки. У результаті проведених досліджень виокремлено шість груп інструментів методології DevOps, які сприяють підтримці якості програмного забезпечення, підвищенню ефективності та прискоренню процесів доставки оновлень або нових функцій на пристрої в інформаційних системах, що використовують IoT технології. Інфраструктура як код (IaC) є ще однією ключовою практикою методології DevOps, яка відіграє ключову роль в системах базованих

на інформаційних технологіях IoT. Особливо зазначимо, що IaC є процесом управління та надання обчислювальної інфраструктури за допомогою машинозчитуваних файлів визначення, а не реальної фізичної конфігурації обладнання або інтерактивних інструментів конфігурації. У контексті систем, базованих на технологіях IoT IaC, дозволяє спростити процеси налаштування та управління складною інфраструктурою, необхідною для IoT-застосунків. За допомогою IaC розробники можуть швидко та надійно відтворювати налаштування інфраструктури, що полегшує масштабування IoT-застосунків по мірі зростання кількості підключених пристроїв.

Контейнери та мікросервіси - це ще дві практики методології DevOps, які можуть значно покращити розроблення інформаційних систем на базі технологій IoT. Контейнер - це автономний виконуваний пакет, який включає все необхідне для запуску застосунка, включаючи код, середовище виконання, системні інструменти, системні бібліотеки та налаштування. Мікросервіси - це архітектурний стиль, який структурує застосунок як набір невеликих, слабо пов'язаних сервісів. Ще одним важливим аспектом в процесах розроблення та операційного супроводу систем на базі технологій IoT є віповідність таких системних рішень регуляторним вимогам. Автоматизовані процеси тестування та розгортання дозволяють швидко впроваджувати зміни, необхідні для відповідності новим стандартам та регламентам. Це особливо важливо для галузей, де безпека даних та конфіденційність є пріоритетом, зокрема, таких як охорона здоров'я та фінанси. І в цьому контексті практики DevOps є незамінними системними IT рішеннями. Загалом проведені авторами статті дослідження підтвердили високу актуальність наукових розвідок щодо впливів, які генерують практики методології DevOps, на екосистеми практичних застосунків, що базуються на інформаційних технологіях IoT. Водночас дослідження підсвітило і ряд питань, які належить вирішувати у майбутньому. Мова зокрема йде про інструменти точного фахового вибору тих чи інших програмних продуктів, які реалізують базові практики методології DevOps. Йдеться зокрема про автоматизацію процедур багатопараметричного оцінювання характеристик широкого спектру програмних інструментів, що убезпечували б швидке та якісне ухвалення практичних рішень по тих чи інших практиках DevOps. Підсумовуючий висновок, виконаного авторами дослідження, можна сформулювати як високу затребуваність в колах IT фахівців інтегрованого системного взаємопроникаючого подання практик методології DevOps та якнайширших класів інформаційних систем, базованих на розлогих мережевих структурах, що функціонують за принципами інформаційних технологій класу IoT.

Список бібліографічного опису

1. Kitchenham B., Charters S. Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report EBSE-2007-01. URL: https://www.cs.auckland.ac.nz/~norsaremah/2007%20Guidelines%20for%20performing%20SLR%20in%20SE%20v2.3.pdf (дата звернення: 08.11.2024).
2. Dave D. M., Bhanushali A. Continuous integration and continuous deployment (CI/CD) for AI-enabled IoT systems. URL: https://www.researchgate.net/publication/380694991_Continuous_Integration_and_Continuous_Deployment_CICD_for_AI-Enabled_IoT_Systems (дата звернення: 08.11.2024).
3. Dave D. M. Enhancing IoT security: The role of security information and event management (SIEM) systems. URL: https://www.researchgate.net/publication/380695139_Enhancing_IoT_Security_ (дата звернення: 08.11.2024).
4. Кальницька А. Ю., Захарченко Д. О. Використання методології DevOps та хмарних технологій для підвищення рівня автоматизації інформаційних систем спортивних організацій // Автоматизація, електроніка та робототехніка. Стратегії розвитку та інноваційні технології до 90-річчя ХНУРЕ. Харків, 2020. С. 27–29.
5. Kim G., Willis J., Debois P., Humble J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. Portland: IT Revolution Press, 2016.
6. Khalyly B., Belangour A., Erraissi A., Banane M. Meta-model approach of applied DevOps on Internet of Things ecosystem // IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS). 2020. P. 1–6. https://doi.org/10.1109/ICECOCS50124.2020.9314552.
7. Khalyly B., Belangour A., Banane M., Erraissi A. A new metamodel approach of CI/CD applied to Internet of Things Ecosystem // Conference: 2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS). 2020. P. 1–6. DOI: 10.1109/ICECOCS50124.2020.9314485.
8. Díaz J., Pérez-Martínez J., López-Peña M., Mena G., Yague A. Self-service cybersecurity monitoring as enabler for DevSecOps // IEEE Access. 2019. Vol. 7. P. 100283–100295. https://doi.org/10.1109/ACCESS.2019.2930000.
9. Botta A., De Donato W., Persico V., Pescapé A. Integration of cloud computing and Internet of Things: A survey // Future Generation Computer Systems. 2016. Vol. 56. P. 684–700. https://doi.org/10.1016/J.Future.2015.09.021.
10. Pereira I. M., de Senna Carneiro T. G., Figueiredo E. Understanding the context of IoT software systems in DevOps // arXiv. 2021. https://doi.org/10.48550/arXiv.2104.10147.
11. Maayan G. D. A DevOps guide to IoT technology. URL: https://devops.com/a-devops-guide-to-iot-technology (дата звернення: 08.11.2024).

12. Apprecode. DevOps in the creative industries: Streamlining content creation workflows. URL: https://apprecode.com/blog/devops-in-iot-accelerating-innovation-in-the-internet-of-things (дата звернення: 08.11.2024).
13. Michalowski M. Using DevOps practices to enhance IoT security. URL: https://www.iotforall.com/using-devops-practices-to-enhance-iot-security (дата звернення: 08.11.2024).
14. Softprom. AWS CloudTrail: Monitoring and logging. New demo! URL: https://softprom.com/ua/aws-cloudtrail-monitoring-ta-vedennya-jurnaliv-loguvannya-nove-demo (дата звернення: 08.11.2024).

References

1. Kitchenham B., Charters S. Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report EBSE-2007-01. URL: https://www.cs.auckland.ac.nz/~norsaremah/2007%20Guidelines%20for%20performing%20SLR%20in%20SE%20v2.3.pdf (дата звернення: 08.11.2024).
2. Dave D. M., Bhanushali A. Continuous integration and continuous deployment (CI/CD) for AI-enabled IoT systems. URL: https://www.researchgate.net/publication/380694991_Continuous_Integration_and_Continuous_Deployment_CICD_for_AI-Enabled_IoT_Systems (дата звернення: 08.11.2024).
3. Dave D. M. Enhancing IoT security: The role of security information and event management (SIEM) systems. URL: https://www.researchgate.net/publication/380695139_Enhancing_IoT_Security_ (дата звернення: 08.11.2024).
4. Kalnytska A. Yu., Zakharchenko D. O. Use of DevOps methodology and cloud technologies to increase the level of automation of information systems of sports organizations // Automation, electronics and robotics. Development strategies and innovative tec.
5. Kim G., Willis J., Debois P., Humble J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. Portland: IT Revolution Press, 2016.
6. Khalyly B., Belangour A., Erraissi A., Banane M. Meta-model approach of applied DevOps on Internet of Things ecosystem // IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS). 2020. P. 1–6. https://doi.org/10.1109/ICECOCS50124.2020.9314552.
7. Khalyly B., Belangour A., Banane M., Erraissi A. A new metamodel approach of CI/CD applied to Internet of Things Ecosystem // Conference: 2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS). 2020. P. 1–6. DOI: 10.1109/ICECOCS50124.2020.9314485.
8. Díaz J., Pérez-Martínez J., López-Peña M., Mena G., Yague A. Self-service cybersecurity monitoring as enabler for DevSecOps // IEEE Access. 2019. Vol. 7. P. 100283–100295. https://doi.org/10.1109/ACCESS.2019.2930000.
9. Botta A., De Donato W., Persico V., Pescapé A. Integration of cloud computing and Internet of Things: A survey // Future Generation Computer Systems. 2016. Vol. 56. P. 684–700. https://doi.org/10.1016/J.Future.2015.09.021.
10. Pereira I. M., de Senna Carneiro T. G., Figueiredo E. Understanding the context of IoT software systems in DevOps // arXiv. 2021. https://doi.org/10.48550/arXiv.2104.10147.
11. Maayan G. D. A DevOps guide to IoT technology. URL: https://devops.com/a-devops-guide-to-iot-technology (дата звернення: 08.11.2024).
12. Apprecode. DevOps in the creative industries: Streamlining content creation workflows. URL: https://apprecode.com/blog/devops-in-iot-accelerating-innovation-in-the-internet-of-things (дата звернення: 08.11.2024).
13. Michalowski M. Using DevOps practices to enhance IoT security. URL: https://www.iotforall.com/using-devops-practices-to-enhance-iot-security (дата звернення: 08.11.2024).
14. Softprom. AWS CloudTrail: Monitoring and logging. New demo! URL: https://softprom.com/ua/aws-cloudtrail-monitoring-ta-vedennya-jurnaliv-loguvannya-nove-demo (дата звернення: 08.11.2024).