УДК 004.7 https://doi.org/10.31713/vt1202220

Сокол О. В., магістр, Цвєткова Т. П., к.т.н., доцент, Харів Н. О., старший викладач (Національний університет водного господарства та природокористування, м. Рівне, t.p.tsvetkova@nuwm.edu.ua, n.o.khariv@nuwm.edu.ua)

ПРОЄКТУВАННЯ ТА РОЗРОБКА АВТОМАТИЗОВАНИХ ПРОЦЕСІВ ВПРОВАДЖЕННЯ ВЕБСЕРВІСУ ЗА DEVOPS МЕТОДОЛОГІЄЮ

Розробка програмного забезпечення складається з багатьох етапів – від розробки, збирання та тестування до розгортання, випуску, доставлення користувачам та підтримки. З метою оптимізації та пришвидшення циклу розробки програмного продукту постає необхідність впровадження автоматизованих процесів, що забезпечує вищу надійність кожного етапу та пришвидшує весь цикл розробки програмного забезпечення. Автоматизація процесів розробки дозволяє створювати системи неперервної розробки та впровадження для ефективної роботи над програмним забезпеченням. З метою оптимізації процесів розробки та впровадження програмних продуктів у статті представлено проєктування та розробку автоматизованих процесів впровадження вебсервісу за DevOps методологією. Застосування методології DevOps дозволяє автоматизувати рутину роботу із збірки, тестування та впровадження вебзастосунку. Розглянуто методи та технології впровадження принципів DevOps підходу до розробки програмного забезпечення, з використанням яких здійснено проєктування та розробку автоматизованих процесів для безперервної інтеграції та розгортання вебсервісу на базі AWS EC2. Розроблені автоматизовані процеси тестувалися на прикладі розробленого сайту для бронювання кемпінгів та можуть застосовуватися для будь-якого вебпроєкту. Спроєктований та розроблений процес автоматизації забезпечив безперервну інтеграцію, автоматичне тестування, збірку та розгортання розробленого вебсервісу. Для цього на всіх етапах розробки та впровадження вебсервісу необхідно забезпечувати стабільну роботу системи, прозорість та зрозумілість системи для розробників. Отримані результати використання розроблених автоматизованих процесів позитивно впливають на швидкість командної розробки, що дозволяє розробляти якісні програмні продукти швидше та бути більш конкурентними на IT-ринку.

Ключові слова: **DevOps методологія; автоматизовані процеси; сервер; проєкт; вебсервіс; версія; CI/CD; Git; Bitbucket.**

Вступ. Розробка програмного забезпечення включає створення, проєктування, розгортання та підтримку програмного забезпечення, що є досить складним процесом та потребує залучення багатьох спеціалістів. Тому актуальною є автоматизація деяких етапів розробки програмного забезпечення: від розробки, збирання та тестування до розгортання, випуску, доставлення користувачам та підтримки, що забезпечує вищу надійність та пришвидшує весь цикл розробки програмного продукту. Адже впровадження автоматизованих процесів дозволяє створювати системи неперервної розробки та впровадження для ефективної роботи над вебресурсом, оптимізує командну роботу, що дозволяє розробляти якісні програмні продукти швидше та бути більш конкурентними на IT-ринку.

Постановка задачі. Розглядається задача застосування DevOps методології при проєктуванні, розробці та впровадженні вебсервісу для опимізації та пришвидченні циклу розробки. Головна мета проєкту полягає у мінімізації витрат часу на виконання рутинної роботи, пов'язаної з розробкою та впровадженням вебсервісу, на прикладі розробленого вебсайту бронювання готелів, шляхом їх автоматизації, впровадження CI/CD практики для полегшення процесів інтеграції та доставки коду.

Розробка автоматизованих процесів та їх впровадження. Для проєктування та розробки автоматизованих процесів впровадження вебсерервісу розглянемо застосування DevOps методології, яка дозволяє автоматизувати рутину роботу із збірки, тестування та впровадження вебзастосунку.

До появи методології DevOps процес розробки і подальшого розгортання програмного забезпечення являв собою чітку послідовність – розробка, тестування та введення в експлуатацію. Підхід DevOps дозволив розробникам і фахівцям з експлуатації працювати у більш тісній співпраці та менше залежати від проблем та особливостей діяльності один одного, а поява інструментів DevOps як спільної екосистеми зовсім стерло межі між розробкою і експлуатацією. Запропонований інструментарій дозволив стандартизувати процеси налаштування, розгортання та підтримки застосунків.

DevOps методологія орієнтована на командну роботу, де розглядаються всі аспекти життєвого циклу програмного забезпечення: від програмного коду до експлуатації продукту кінцевим користувачем (рис. 1). Згідно з рис. 1, основними процесами розробки програмного забезпечення за DevOps методологією є наступні [2; 7]:

- Code (код) – розробка та аналіз, контроль версій та злиття коду;
- Build (збірка) – безперервна інтеграція різних збірок;
- Test (тест) – інструменти безперервного тестування, що повідомляють про бізнес-ризики;

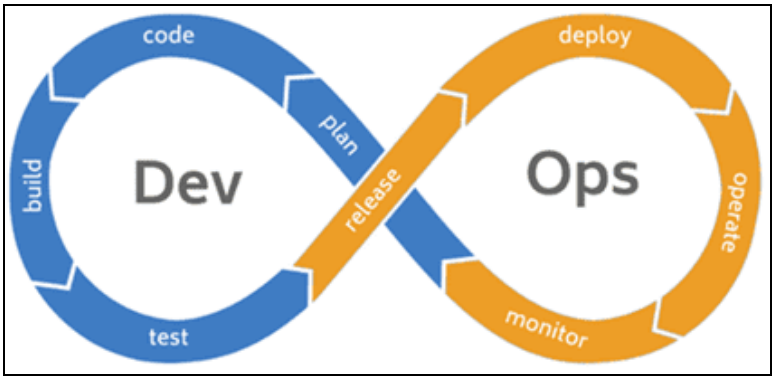


Рис. 1. Основні процеси розробки програмного забезпечення за DevOps

- Operate (робота з пакетами) – репозиторій артефактів, попереднє встановлення застосунків;
- Release (випуск) – управління змінами, офіційне затвердження випуску, автоматизація випуску;
- Deploy (розгортання конфігурації) – управління інфраструктурою як кодом;
- Monitor (моніторинг) – моніторинг продуктивності застосунків, досвід роботи з кінцевим користувачем.

Дані процеси описують весь цикл розробки програмного забезпечення, які необхідно оптимізувати за допомогою впровадження DevOps методології на кожному кроці розробки.

Протягом усього життєвого циклу застосунки в DevOps застосовують на різних етапах конкретні методики, які автоматизують і прискорюють конкретний етап або дозволяють створити цілісний продуктивний процес, охоплюючи відразу декілька етапів. Розглянемо основні підходи до розробки

програмного забезпечення, які використовуються в DevOps методології.

Безперервна інтеграція і доставка (CI/CD) дозволяє розробникам оперативно додавати зміни в основний код. Програмний код відразу автоматично тестується, що підвищує стабільність роботи програми. При безперервній доставці нові версії регулярно автоматично розгортаються на сервер, за рахунок цього можливі більш часті оновлення проєкту. Тобто CI/CD дозволяє автоматизувати всі дії – від фіксації коду до розгортання його на сервер, щоб скоротити час і трудозатрати на рутинні операції і прискорити процес розгортання коду з мінімальними ризиками.

Керування версіями дозволяє відслідковувати всі виправлення та зміни коду, що допомагає спростити процес його аналізу та відновлення в разі помилок. Системи управління версіями дозволяють команді розробників спільно створювати код, забезпечуючи об'єднання його змін в файлах, вирішення конфліктів і, в разі необхідності, повернутися до попередніх версій.

Гнучка розробка програмного забезпечення. Дуже важливою є спільна робота різних команд, отримання оперативного зворотного зв'язку від користувачів та адаптація до коротких циклів випуску програмного забезпечення, що дозволяє враховувати побажання клієнтів і швидко вносити зміни до застосунків.

Інфраструктура як код. За даною методикою системні ресурси описуються як код, що допомагає здійснювати розгортання середовища розробки, тестування та експлуатацію оперативно, передбачувано та надійно, знижує ризики помилок, пов'язаних з людським фактором.

Управління конфігурацією дозволяє керувати станом ресурсів (серверів, віртуальних машин, баз даних) в системі. За допомогою певних інструментів DevOps-інженер може розгортати зміни контрольовано та систематично, при цьому зменшивши до мінімуму ризики зміни конфігурації, а також відстежувати актуальний стан системи і будь-які зміни чи відхилення в конфігурації від потрібного її стану.

Безперервний моніторинг передбачає відстеження в реальному часі продуктивності та працездатності програми. Для цього автоматично збираються певні показники телеметрії, метаданих, а також налаштовується оповіщення про відхилення у роботі програми. Ця інформація допомагає вирішувати виникаючі проблеми

дуже швидко і допомагає зрозуміти, що можна поліпшити у майбутніх циклах розробки [1; 4–6].

Розглянемо застосування DevOps підходу при проєктуванні, розробці та впровадженні вебсервісу для оптимізації командної роботи, забезпечені якості, надійності кінцевого продукту та пришвидшенні циклу розробки програмного забезпечення. Створені автоматизовані процеси впроваджено у розроблений вебсервіс бронювання готелів. Головною метою є мінімізація витрат часу на виконання рутинної роботи, пов'язаної з розробкою та впровадженням вебсервісу, шляхом автоматизації процесів інтеграції та доставки коду. Для вирішення поставленої задачі необхідно виконати наступні завдання:

1. Налаштувати локальне середовище розробки проєкту за допомогою Docker контейнерів.

2. Створити сервіс розгортання проєкту на сервер BitBucket Pipeline, який буде виконувати наступні функції:

– збірка тестової версії проєкту;

– запуск автоматизованого тестування проєкту (Unit-тестування);

– збірка проєкту для завантаження на тестовий (prod) та мастер (dev) сервери;

– копіювання готової аплікації на файлове сховище Amazon S3;

– ініціалізація запуску розгортання проєкту на dev та prod сервери;

– налаштування серверів для проєкту.

3. Створити віртуальні машини на Amazon EC2 для хостингу проєкту.

4. Налаштувати систему керування базами даних Amazon RDS для зберігання бази даних.

5. Підключити файлове сховище Amazon S3 для зберігання аплікації проєкту.

6. Забезпечити універсальність алгоритму для легкої адаптації під інші вебзастосунки.

Результатом виконання всіх завдань є реалізація алгоритму (рис. 1), який описує весь процес проєктування, розробки, впровадження та підтримки вебсервісу із застосуванням DevOps методології. На всіх етапах розробки та впровадження вебсервісу потрібно забезпечити стабільну роботу системи, прозорість та зрозумілість системи для розробників.

DevOps методологія містить підхід до роботи з системою контролю версій програмного забезпечення GIT. Git Workflow – це рекомендація щодо використання Git для виконання роботи послідовно та продуктивно, яка спонукає розробників ефективно та послідовно використовувати Git. Основна ідея цих рекомендацій полягає у тому, що існують master-гілка, dev-гілка та feature's гілки. Розробники працюють в межах своєї feature-гілки, синхронізуючись з dev-гілкою. Після потрапляння в dev, feature's гілки видаляються. Час від часу, коли проєкт на dev-гілці стає стабільним, dev синхронізується з master-гілкою і отримуємо наступну робочу версію продукту (рис. 2) [5; 6].

Початок створення автоматизованого CI/CD процесу – це завантаження проєкту на мастер-гілку, яке ініціалізує запуск саме процесу безперервної інтеграції (Continuous Integration), який виконує збірку та запуск автоматичних тестів. У разі успішного проходження тестування починається етап безперервної доставки (Continuous Delivery), під час якого проєкт копіюється на сервер, створюється резервна копія старої версії, яка замінюється новою версією.

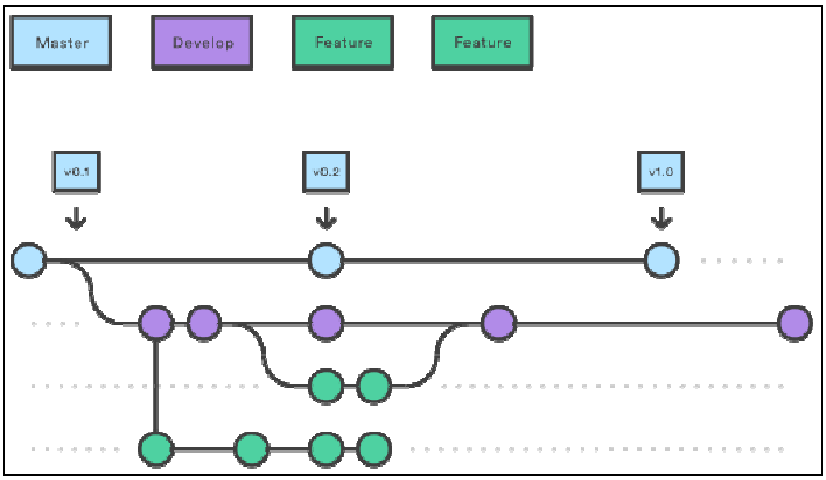


Рис. 2. Схема Git Workflow

Загальна блок-схема автоматизованої CI/CD системи має вигляд (рис. 3):

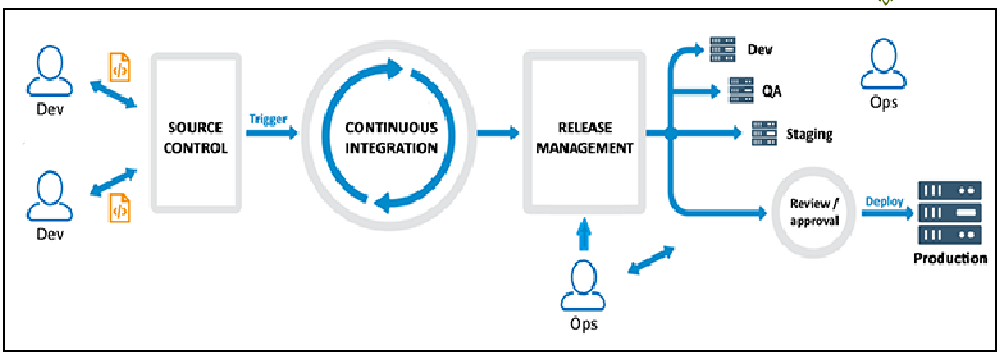


Рис. 3. Загальна CI/CD блок-схема

До реалізації автоматизованих процесів впровадження вебпроєкт розгортався шляхом виконання певного набору операції, які зводилися до наступного:

– завантаження останніх змін;
– збірка проєкту локально, який призводив до створення артефакту;
– локальний запуск, перевірка справності роботи;
– перенесення артефакту на сервер;
– заміна артефакту новою версією;
– перенесення нового артефакту у відповідне місце;
– видалення старого артефакту.

Таким чином, даний процес можна відобразити у вигляді наступної схеми (рис. 4):

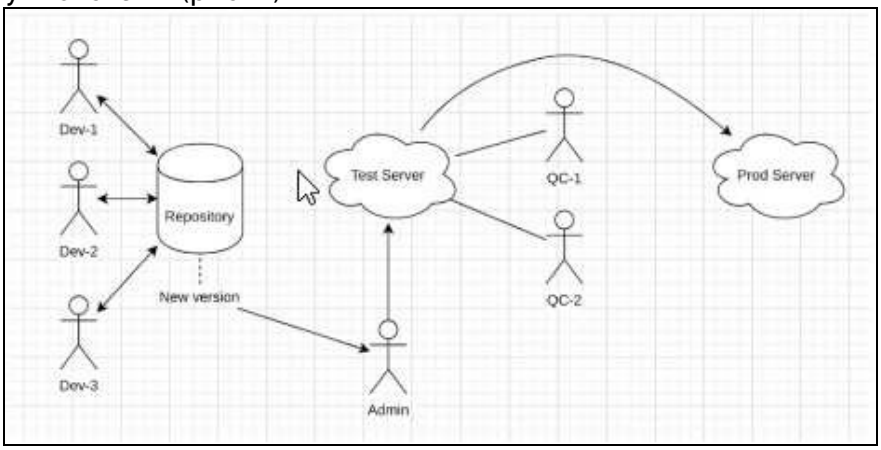


Рис. 4. CI/CD процеси до виконання автоматизації

Загалом процес розробки вимагає багато розробників, які, в свою чергу, взаємодіють з загальною кодовою базою, яка в певний

період часу містить відповідну версію проєкту. Ця нова версія може бути перенесена на сервер, тобто, відбудеться процес розгортання даної аплікації. Сервер для початку повинен бути тестовим – для відлагодження та перевірки програми. Коли нова версія програми потрапляє на тестовий сервер, відділ тестування випробовує, перевіряє на помилки даний продукт. Після успішної перевірки, якщо не було виявлено жодних дефектів, проєкт розгортається в межах production середовища, тобто такого, яке використовується реальними користувачами. Якщо забрати з загального процесу людину, яка займається адмініструванням, то виникає проблема в тому, що весь цей процес стає неможливим. Тобто, є певні ризики щодо певних непередбачених ситуацій, які можуть виникнути через велику залежність від однієї людини. Можливі наступні шляхи вирішення даної проблеми:

- найняти ще одного адміністратора;
- автоматизувати частково ці процеси, якими займається даний робітник.

У випадку застосування спроектованих автоматизованих процесів до розробки програмного забезпечення додаємо до схеми DevOps спеціаліста, який налаштує автоматичну збірку та доставку проєкту до кінцевого користувача, і схема CI/CD процесу представлена на рис. 5.

Для тестування розроблених автоматизованих процесів можна використати будь-який вебпроєкт. Проведемо дане тестування на прикладі спеціально розробленого сайту для бронювання кемпінгів.

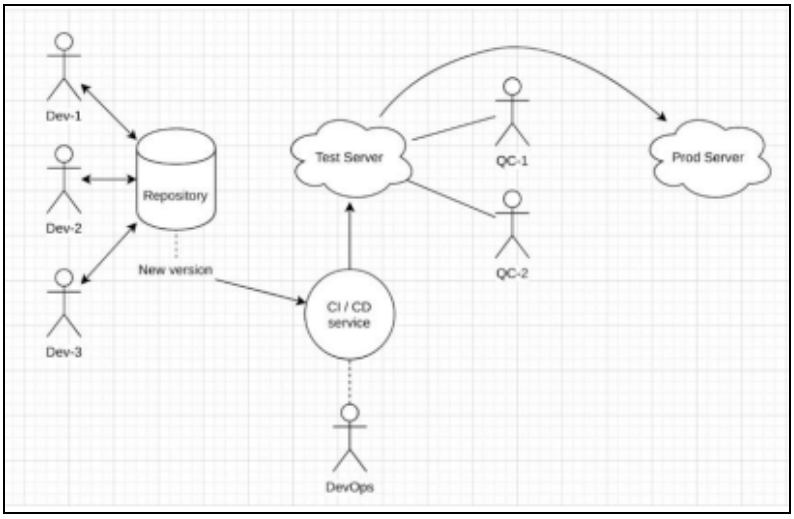


Рис. 5. CI/CD процеси після автоматизації

Спочатку відкриємо локальну версію проєкту та додамо надпис «Тест CI/CD» на головну сторінку сайту (рис 6) [3].

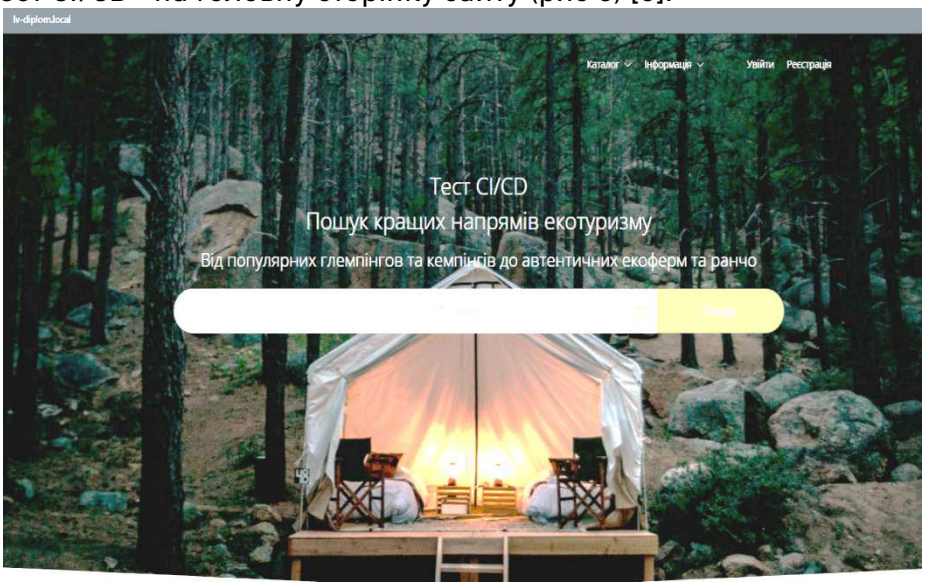


Рис. 6. Головна сторінка вебсервісу на локальному комп'ютері

Після завантаження змін на Git автоматично запускається створений раніше набір команд для сервісу Bitbucket Pipeline, який автоматично виконує розгортання проєкту на сервер. Результат виконання даного файлу наведено на рис. 7.

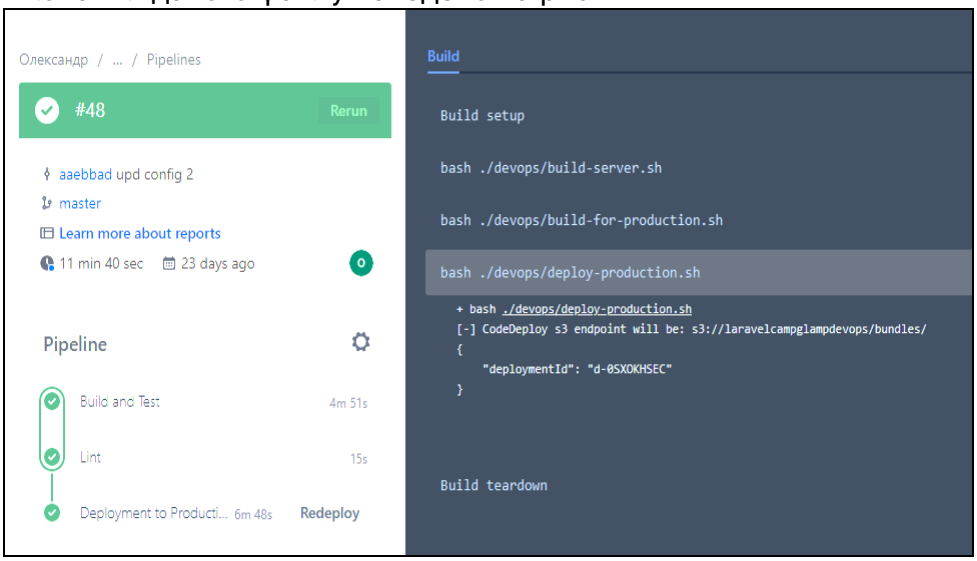
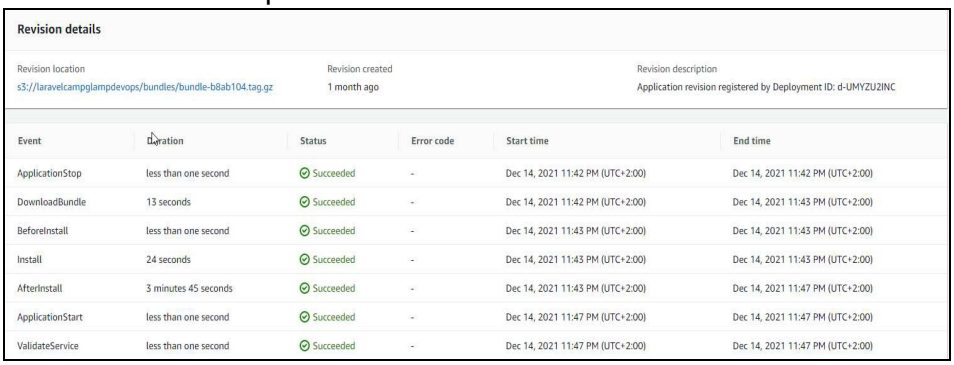


Рис. 7. Виконання розгортання проєкту на сервер Bitbucket Pipeline

Також даний файл ініціалізує запуск сервісу AWS CodeDeploy, що копіює завантажену сервісом BitBucket Pipeline аплікацію із сховища Amazon S3 у вебдиректорію, результат виконання якого можна побачити на рис. 8.



Revision details

Revision location	Revision created	Revision description
s3://laravelcampglampdevops/bundles/bundle-b8ab104.tag.gz	1 month ago	Application revision registered by Deployment ID: d-UMYZU2INC

Event	Duration	Status	Error code	Start time	End time
ApplicationStop	less than one second	Succeeded	-	Dec 14, 2021 11:42 PM (UTC+2:00)	Dec 14, 2021 11:42 PM (UTC+2:00)
DownloadBundle	13 seconds	Succeeded	-	Dec 14, 2021 11:42 PM (UTC+2:00)	Dec 14, 2021 11:43 PM (UTC+2:00)
BeforeInstall	less than one second	Succeeded	-	Dec 14, 2021 11:43 PM (UTC+2:00)	Dec 14, 2021 11:43 PM (UTC+2:00)
Install	24 seconds	Succeeded	-	Dec 14, 2021 11:43 PM (UTC+2:00)	Dec 14, 2021 11:43 PM (UTC+2:00)
AfterInstall	3 minutes 45 seconds	Succeeded	-	Dec 14, 2021 11:43 PM (UTC+2:00)	Dec 14, 2021 11:47 PM (UTC+2:00)
ApplicationStart	less than one second	Succeeded	-	Dec 14, 2021 11:47 PM (UTC+2:00)	Dec 14, 2021 11:47 PM (UTC+2:00)
ValidateService	less than one second	Succeeded	-	Dec 14, 2021 11:47 PM (UTC+2:00)	Dec 14, 2021 11:47 PM (UTC+2:00)

Рис. 8. Результати роботи сервісу AWS CodeDeploy

Як видно з рис. 8, всі етапи розгортання проєкту пройшли успішно, отже, всі зміни повинні бути вже на віддаленому сервері. Перейдемо за адресою http://35.178.225.200 (дата останнього доступу: 30.11.2021) та перевіримо (рис. 9).

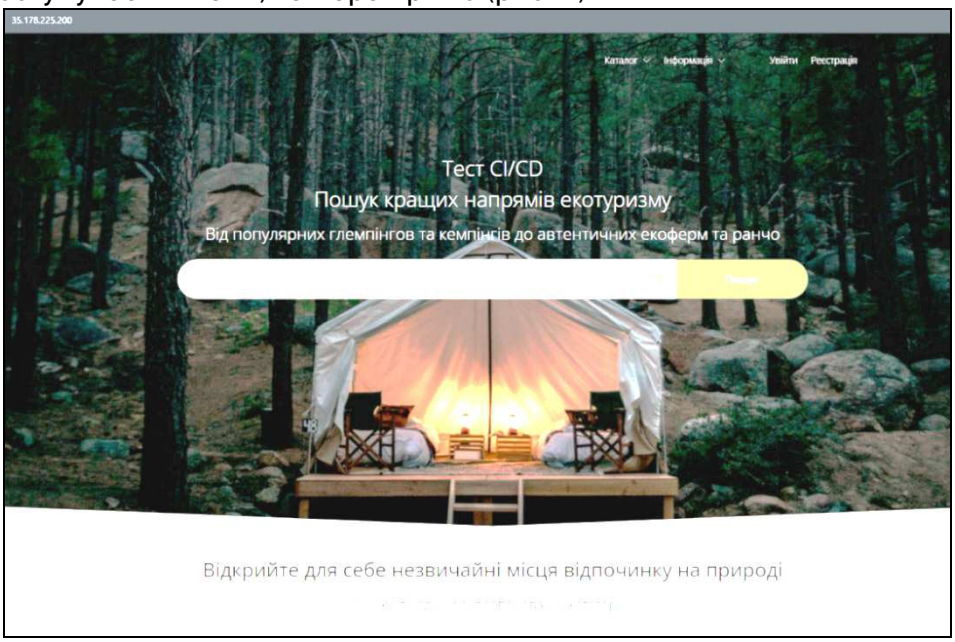


Рис. 9. Головна сторінка вебсервісу на віддаленому сервері

Надпис «Тест CI/CD» з'явився на мастер-сервері, отже, все налаштовано правильно та працює без помилок, що свідчить що спроєктований та розроблений процес автоматизації забезпечує безперервну інтеграцію, автоматичне тестування, збірку та розгортання вебсервісу.

Висновок. DevOps методологія дозволяє здійснювати автоматизоване розгортання та покращення надійності коду, підвищити продуктивність команди, зменшити бізнес-ризики та кінцеву ціну для замовника. Результати використання розроблених автоматизованих процесів позитивно впливають на швидкість командної розробки. Для порівняння один staging-реліз (випуск програмного забезпечення) в місяць перетворився у 4, що є кращим показником, ніж було раніше, при цьому впевненість у працездатності та правильності роботи вебпроєкту залишилась на хорошому рівні.

Розроблені автоматизовані процеси можуть повторно застосовуватися для аналогічних проєктів. Адаптація даного рішення чи використання принципів підходу до проєктування та розробки програмного забезпечення є можливими та легко інтегрується у вже існуюче різного роду програмне забезпечення та не вимагає значних зусиль на їх реалізацію. Крім того, алгоритм розробленої системи автоматизованого процесу є досить простим для розробників та дає можливість застосовувати його в невеликих проєктах з малочисельною командою.

1. Вольф Е. Continuous delivery. Практика неперервних апдейтів. Вид-во «МІФ», 2016. 78 с. **2.** Девіс Д. Філософія DevOps. Мистецтво управління IT. Вид-во «ВВМ», 2018. 92 с. **3.** Хоган Б. HTML5 та CSS3. Веб-розробка за стандартами нового покоління : 4-те вид. Київ : Видавничий дім «Слово», 2010. 260 с. **4.** Gene Kim. The DevOps Handbook. Packt Publishing, 2019. 302 с. **5.** Joseph Muli. Beginning DevOps with Docker. Packt Publishing, 2019. 215 с. **6.** Rafal Leszko. Continuous Delivery with Docker and Jenkins: Delivering software at scale. Packt Publishing Limited, 2018. 140 с. **7.** Складові DevOps. Synopsys : вебсайт. URL: https://www.synopsys.com/glossary/what-is-devops.html (дата звернення: 30.05.2022).

REFERENCES:

1. Volf E. Continuous delivery. Praktyka neperervnykh apdeitiv. Vyd-vo «MIF», 2016. 78 s. **2.** Devis D. Filosofiia DevOps. Mystetstvo upravlinnia IT. Vyd-vo «VVM», 2018. 92 s. **3.** Khohan B. HTML5 ta CSS3. Veb-rozrobka za standartamy novoho pokolinnia : 4-te vyd. Kyiv : Vydavnychyi dim «Slovo», 2010. 260 s. **4.** Gene Kim. The DevOps Handbook. Packt Publishing, 2019. 302 c. **5.** Joseph Muli. Beginning DevOps with Docker. Packt Publishing, 2019. 215 c. **6.** Rafal Leszko. Continuous Delivery with Docker and Jenkins: Delivering software at scale. Packt Publishing Limited, 2018. 140 c. **7.** Skladovi DevOps. Synopsys : vebsait. URL: https://www.synopsys.com/glossary/what-is-devops.html (data zvernennia: 30.05.2022).

Sokol O. V., Master, [1]**Tsvietkova T. P., Candidate of Engineering (Ph.D.), Associate Professor,** [2]**Khariv N. O., Senior Lecturer** (National University of Water and Environmental Engineering, Rivne,
[1]t.p.tsvetkova@nuwm.edu.ua, [2]n.o.khariv@nuwm.edu.ua)

DESIGN AND DEVELOPMENT OF AUTOMATED PROCESSES OF WEB SERVICE IMPLEMENTATION USING DEVOPS METHODOLOGY

Software development consists of many stages – from development, assembly and testing to deployment, release, delivery to users and support. In order to optimize and accelerate the software development cycle, implementation of automated processes which provide higher reliability on each stage and speeds up the entire software development cycle is required. Automation of development processes allows to create continuous development and implementation systems for effective work on the software. In order to optimize the processes of development and implementation of software products, this article presents the design and development of automated web service implementation processes according to DevOps methodology. The application of DevOps methodology allows to automate routine work on the assembly, testing and deployment of web application. The methods and technologies of DevOps approach implementation are considered and used for design and development of automated processes for continuous integration and deployment of a web service based on AWS EC2. Developed automated processes were tested on the example developed site for booking campsites and

can be used for any web project. The automation process designed and developed in this paper ensured continuous integration, automatic testing, assembly and deployment of the developed web service. For this purpose, the stable operation of the system, transparency and system clarity for developers must be ensured on each stage of development and implementation of the web service. The results of using the developed automated processes have a positive effect on the speed of cooperative development, which allows to develop quality software products faster and to be more competitive in the IT market.

Keywords: **DevOps methodology; automated processes; server; project; web service; version; CI/CD; Git; Bitbucket.**
