УДК 004.02, 004.08, 004.09
DOI: https://doi.org/10.53920/ITS-2023-2-6

Ольга Іванівна ТКАЧЕНКО,
кандидат фізико-математичних наук, доцент,
доцент кафедри інформаційних технологій,
Державний університет інфраструктури та технологій
ORCID ID: 0000-0003-1800-618X
Максим Володимирович КОВАЛЬЧУК,
магістрант кафедри інформаційних технологій,
Державний університет інфраструктури та технологій
ORCID ID: 0009-0009-6749-618X

АВТОМАТИЗАЦІЯ РОЗГОРТАННЯ ХМАРНИХ ФУНКЦІЙ З ВИКОРИСТАННЯМ SERVERLESS ФРЕЙМВОРКУ: ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ

В наш час проблеми моніторингу працездатності та ефективності відстеження відповідного програмного коду, виявлення помилок та відлагодження програмного коду в безсерверному середовищі є достатньо складними проблемами, актуальність вирішення якиї не викликає сумнівів. Шляхи вирішення цих проблем для безсерверних функцій з використанням технології AWS є перспективними. Безсерверні функції активують оточення кожного разу при виклику, тому перший запуск безсерверної функції є досить повільним через необхідність ініціалізації середовища.

Розглядаються ключові аспекти розгортання та керування функціями в хмарних середовищах за допомогою Serverless фреймворку. Робиться акцент на значенні і практичних застосуваннях методів автоматизації в контексті сучасних хмарних обчислень та розробки безсерверних (Serverless) застосунків.

Метою роботи є аналіз та дослідження проблем, пов'язаних з використанням безсерверних функцій у хмарних середовищах, визначення можливостей використання сучасних принципів, технологій та інструментів, зокрема, Serverless фреймворку для автоматизації розгортання хмарних функцій та розробки відповідного програмного забезпечення.

У статті було розглянуто сутність підходу до автоматизації процесів розгортання хмарних функцій на основі використання Serverless фреймворку. Серед переваг використання Serverless фре-

ймворку було виділено, зокрема, такі, як зменшення завдань щодо інфраструктури, швидке та достатньо просте розгортання, еластичність, масштабованість та економія коштів, спрощені моніторинг і логування, більше часу безпосередньо на розробку.

Було надано загальний огляд Serverless фреймворку та наведено інструкцію (послідовність дій) щодо створення та розгортання за його допомогою безсерверних функцій в хмарному середовищі AWS. Було розглянуто приклади використання Serverless фреймворку при розв'язанні практичних задач, зокрема створення та розгортання функції для відправлення push-нотифікацій. Запропонований підхід може бути корисним при розробці застосунків з використанням хмарних технологій.

***Ключові слова**: хмарні обчислення, хмарне середовище, фреймворк, Serverless, безсерверна функція, push-нотифікації, хмарний провайдер, Node.js, AWS.*

Olha TKACHENKO
PhD of physical and mathematical sciences, associate professor,
associate professor at the department
of information technologies,
State University of Infrastructure and Technology
Maksym KOVALCHUK
Undergraduate at the department of information technologies,
State University of Infrastructure and Technology

AUTOMATION OF THE DEPLOYMENT OF CLOUD FUNCTIONS USING THE SERVERLESS FRAMEWORK: PROBLEMS AND PERSPECTIVES

Nowadays, the problems of monitoring the performance and effectiveness of tracking the corresponding software code, detecting errors and debugging the software code in a serverless environment are quite complex problems, the urgency of the solution of which is beyond doubt. Ways to solve these problems for serverless functions using AWS technology are promising. Serverless functions activate the environment each time they are called, so the first run of a serverless function is quite slow due to the need to initialize the environment.

Key aspects of deploying and managing features in cloud environments using the Serverless framework are covered. Emphasis

is placed on the importance and practical applications of automation methods in the context of modern cloud computing and the development of serverless applications.

The purpose of the work is the analysis and research of problems related to the use of serverless functions in cloud environments, the determination of the possibilities of using modern principles, technologies and tools, in particular, the Serverless framework for automating the deployment of cloud functions and the development of appropriate software.

The article considered the essence of the approach to automating processes of deployment of cloud functions based on the use of the Serverless framework. Among the advantages of using the Serverless framework, such as reduction of infrastructure tasks, fast and fairly simple deployment, elasticity, scalability and cost savings, simplified monitoring and logging, more time directly for development were highlighted.

An overview of the Serverless framework was provided and a step-by-step guide for creating and deploying serverless features in the AWS cloud environment was provided. Examples of the use of the Serverless framework in solving practical problems were considered, in particular, the creation and deployment of a function for sending push notifications. The proposed approach can be useful in the development of applications using cloud technologies.

Keywords: *cloud computing, cloud environment, framework, Serverless, serverless function, push notifications, cloud provider, Node.js, AWS.*

Постановка проблеми. Serverless [1], як концепція, дає можливість розробникам концентруватися лише на функціональності їхніх додатків, не турбуючись про адміністрування серверів та внутрішньої інфраструктури. Проте, процес розгортання безсерверних функцій у хмарних середовищах залишається актуальною проблемою, яка потребує свого вирішення. Автоматизація розгортання безсерверних функцій у Serverless фреймворку [2] допомагає вирішити цю проблему, роблячи процес більш зрозумілим та ефективним.

Вирішення проблеми автоматизації розгортання безсерверних функцій передбачає, зокрема:

- аналіз можливостей та переваг автоматизації розгортання безсерверних функцій у хмарному середовищі;

- дослідження різних підходів, інструментів і методів, які можна використовувати для автоматизації цього процесу, включаючи IaC [3], Serverless фреймворк та інші.

Для демонстрації запропонованого підходу треба навести приклади практичного використання Serverless фреймворку для вирішення різноманітних задач.

Аналіз останніх досліджень і публікацій. В наш час безсерверні технології активно застосовуються в різних галузях, включаючи веброзробку, обробку даних, Інтернет речей (*Internet of things, IoT*) [4] та інші. Це свідчить про широкий спектр можливостей, які надає підхід з використанням Serverless.

Слід відмітити важливість розуміння того, як вирішувати проблеми, що виникають при використанні безсерверних хмарних функцій.

Дуже важливо забезпечити ефективне розгортання функцій при зростанні навантаження та автоматично масштабувати обчислювальні ресурси. Зокрема, в [5] було розглянуто можливості автоматизації розгортання та масштабування хмарних функцій.

Виявлення помилок та відлагодження програмного коду в безсерверному середовищі може бути складною проблемою. Тому проблема моніторингу працездатності та ефективності відстеження відповідного програмного коду завжди постає перед розробниками. Шляхи її вирішення для безсерверних функцій з використанням технології AWS Lambda Insights [6] були розглянуті в [7].

За замовчуванням, безсерверні функції активують оточення кожного разу при відповідному виклику, тому перший запуск безсерверної функції (Cold Start) [8] може бути досить повільним через необхідність ініціалізації спочатку середовища. Можливості оптимізації цього процесу та його прискорення розглянуті в [9].

В [10] розглядається конкретний технічний стек реалізації безсерверних функцій з використанням Node.js (серверного середовища виконання JavaScript-коду) [11], AWS Lambda (безсерверний обчислювальний сервіс від компанії Amazon) [12] та Serverless фреймворку.

Мета статті полягає в аналізі та дослідженні проблем, пов'язаних з використанням безсерверних функцій у хмарних середовищах, визначення можливостей використання сучасних принципів, технологій та інструментів, зокрема, Serverless фреймворку

для автоматизації (спрощення та оптимізації) розгортання хмарних функцій та розробки відповідного програмного забезпечення.

Досягнення цієї мети забезпечується вирішенням, зокрема, наступних завдань:

- визначення сутності безсерверного підходу, його основних переваг і недоліків у порівнянні зі статичним сервером, а також можливих сфер використання;
- визначення складових компонентів Serverless фреймворку (зокрема, визначення того, з яких частин складається цей фреймворк та конфігураційні yaml файли);
- дослідження проблем автоматизації розгортання хмарних функцій та шляхів їх вирішення;
- розробка покрокової інструкції для розгортання хмарних функцій за допомогою Serverless фреймворку;
- Тестування запропонованого підходу при вирішенні практичних задач за допомогою використання Serverless фреймворку.

Мета і завдання статті спрямовані на просування інноваційних підходів та технологій, які можуть підтримати стійкий розвиток хмарних технологій та забезпечити високий рівень розробленого програмного забезпечення, яке вирішує конкретні практичні проблеми.

Виклад основного матеріалу дослідження. Безсерверні функції — нова парадигма в хмарних обчисленнях, яка привернула значну увагу в останні роки завдяки своїм обіцянкам спростити розгортання та керування застосунками.

Функція, як послуга (FaaS) [13], – це підхід, при якому розробники можуть розгортати та запускати окремі функції (зв'язні елементи коду, так звані «одиниці» коду) без необхідності «напряму» керувати серверною інфраструктурою.

На відміну від традиційного серверного підходу, який передбачає надання та підтримку серверів для статичного розміщення застосунків, безсерверні обчислення повністю абстрагують рівень інфраструктури, дозволяючи розробникам зосередитися виключно на написанні коду.

При використанні такої моделі, обчислювальні ресурси автоматично розподіляються та масштабуються по принципу on demand [14] для виконання коду у відповідь на певні події, ситуаційні стани чи тригери.

(Під тригером будемо розуміти певну подію чи предмет, при спрацьовуванні якого, починає відбуватися дія безпосередньо або побічно пов'язана з ним, тобто це те, що стає так би мовити спусковим гачком).

Розробники пишуть програмні коди для функцій, що розгортаються та виконуються в ефемерних контейнерах без збереження стану.

Ці функції можуть тригеритись (відповідати певними діями, спонукати на реакцію) на виконання різних подій, такими речами, як HTTP-запити, модифікування в базі даних або повідомлення з черги подій.

Не менш важливою перевагою є те, що безсерверні функції дотримуються принципу pay-as-you-go, згідно з яким плата користувачам виставляється на основі фактичного часу виконання та ресурсів, спожитих їхніми функціями.

У традиційній інфраструктурі невикористані ресурси та потужності серверів можуть бути занадто дорогими. Безсерверні ж функції не потребують додаткових витрат тоді, коли вони не виконуються. Це особливо вигідно для програм із непередбачуваним робочим навантаженням.

Тим не менш, безсерверні функції мають і певні недоліки, зокрема, один з яких – автоматизація їх керування та розгортання.

Для вирішення цієї проблеми було створено Serverless фреймворк [15, 16]. Він пропонує уніфікований файл конфігурації, який об'єднує в собі всі параметри для визначення ресурсів та розгортання функцій.

Все це допомагає уніфікувати процес розгортання, а також спростити керування функціями, ресурсами і тригерами подій в рамках одного проєкту.

Фреймворк Serverless підтримує різних хмарних провайдерів, зокрема таких, як AWS [10, 12], Azure [27], Google Cloud [28] тощо.

Розробники можуть написати програмний код один раз і розгорнути його в кількох хмарних середовищах, підвищуючи портативність і зменшуючи прив'язаність (і, відповідно, залежність) до конкретного провайдера.

У випадку, коли треба динамічно розширити базовий функціонал фреймворку, використовується велика екосистема існуючих сучасних плагінів [15]. Вони підтримуються широкий спектр випадків, зокрема:

- оптимізація коду;
- підвищення безпеки;
- автоматизація розгортання;
- інтеграція з різними сторонніми службами.

Розробники можуть легко додавати та налаштовувати плагіни відповідно до потреб на проекті.

Фреймворк Serverless забезпечує інтерфейс командного рядка (*Serverless CLI*) [16], який спрощує керування проєктами. Розробники, щоб розгортати функції, викликати їх і керувати різними аспектами своїх безсерверних програм, можуть використовувати такі команди, як

- *sls deploy*;
- *sls invoke*.

Розглянемо, як за допомогою фреймворку можна створити та розгорнути безсерверну хмарну функцію. Для розробленого програмного продукту, що демонструє в статті переваги запропонованого підходу, було використано мову Node.js (JavaScript) [11] як основну мову програмування і AWS [10, 12, 22] як хмарний провайдер.

Розглянемо процес створення і розгортання безсерверної хмарної функції:

- Інсталювати Node.js і NPM [17];
- Інсталювати Serverless фреймворк як глобальний модуль. Це можна зробити за допомогою команди:

npm i-g serverless

- Ініціалізувати новий проєкт, виконавши команду:

serverless create –template aws-nodejs–path <serverless-project-name>

Налаштувати основний конфігураційний файл *serverless.yml* у каталозі розробляємого проєкту, включивши функції, тригери подій та усі необхідні ресурси. Детально властивості конфігураційного файлу описані в [18].

У конфігураційному файлі (рис. 1):

– поле *service* визначає назву проєкту;
– поле *provider* визначає тип хмарного провайдера і середу виконання коду;
– поле *functions* визначає перелік функцій.

```
service: my-serverless-project

provider:
  name: aws
  runtime: nodejs14.x

functions:
  hello:
    handler: handler.hello
```

Рис. 1. Конфігураційний файл для хмарної функції

Джерело: розробка авторів

- Створити файл *handler.js*;
- Експортувати асинхронну функцію *hello* (рис. 2). Функція має один вхідний параметр event, який містить інформацію про вхідну подію, а саме:
  – контекст виконання;
  – тип події;
  – властивості, передані клієнтом в функцію напряму для обробки.

```
// handler.js

module.exports.hello = async (event) => {
  // write your code here
};
```

Рис. 2. Асинхронна обробка подій в середовищі Node.js

Джерело: розробка авторів

- Розгорнути безсерверну службу, виконавши наступні команди:
  – *serverless print;*
  – *serverless deplo.*

Команда *serverless print* надає усі зміни без розгортання служби, що сприяє перевірці коректності конфігурації перед розгортанням безсерверної служби.

Команда *serverless deplo* ініціює виконання таких дій, як:

– упакування авторського програмного коду;
– створення необхідних ресурсів в AWS;
– розгортання відповідної хмарної функції.

- Викликати функцію: serverless *invoke -f hello*.

Безсерверні функції є універсальним засобом, що широко застосовується на практиці. Зокрема, використання зазначених функцій доцільно використовувати у таких випадках, як ті, що наведені нижче.

- Обробка логіки автентифікації та авторизації користувачів у вебзастосунках. Вони можуть:
  – створювати та верифікувати токени авторизації;
  – застосовувати політики контролю доступу;
  – інтегруватися зі сторонніми постачальниками токенів (наприклад, таких як, AWS Cognito [19]).
- Планування задач, які мають виконуватися через певні проміжки часу, зокрема, таких як:
  – регулярне резервне копіювання даних;
  – створення звітів;
  – синхронізація даних між системами.
- Надсилання користувачам сповіщень в режимі реального часу через різні канали, зокрема, такі як:
  – електронна пошта;
  – SMS;
  – push-повідомлення;
  – платформи обміну повідомленнями.
- Швидка обробка та аналіз безперервних потоків даних з пристроїв, що підтримують Інтернет речей [4], ініціюючи відповіді на основі даних, що отримуються від різноманітних приладів (датчиків, лічильників, контролерів тощо), чи подій пристрою.

Розглянемо використання *Serverless* фреймворку при написанні та подальшому розгортанні безсерверної функції для надсилання APNS-сповіщень [20] на мобільні пристрої користувачів.

Для цього насамперед потрібно, використовуючи отриману інформацію, сформувати та надіслати нотифікацію в AWS SNS-topic [21] для відповідного конкретного користувача.

На рис. 3 зображено конфігураційні файли проєкту, що розглядається.

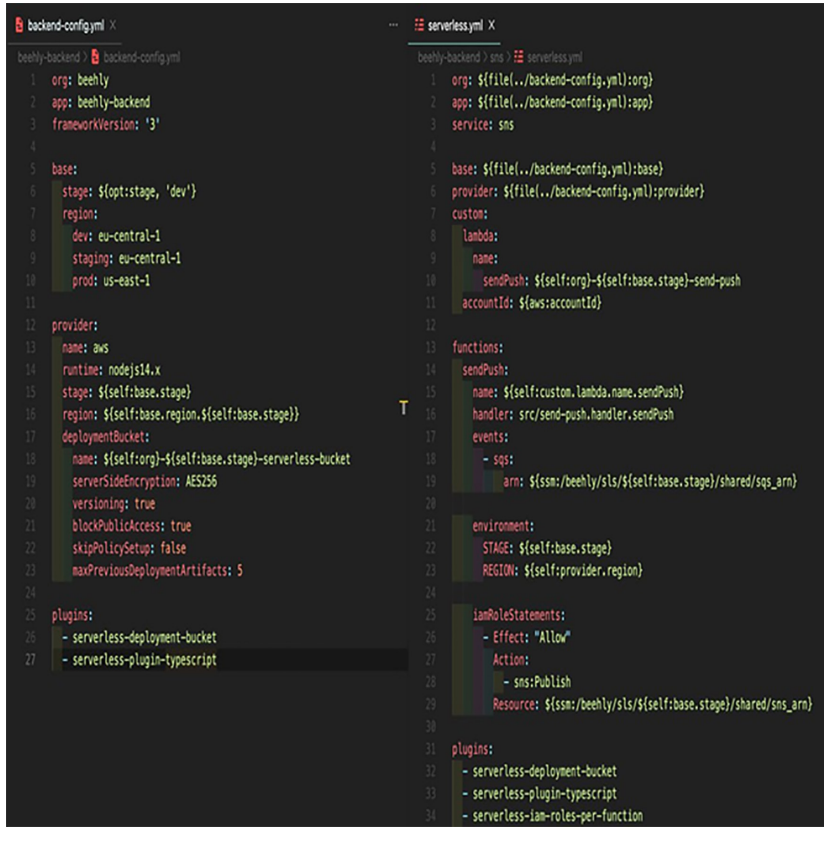

Рис. 3. Конфігураційні файли функції для надсилання push-нотифікацій на мобільні пристрої користувачів

Джерело: розробка авторів

Ці конфігураційні файли проєкту містять, зокрема:

- Файл *backend-config.yml*, який є загальним для усього проєкту і я якому зазначені:
  - *region* (регіони) для розгортання функцій (відповідно до кожного *environmental (оточення, оточуючого середовища)*);

  - тип провайдера;
  - використовувана мова програмування;
  - загальні плагіни;
  - інформація про deployment bucket – місце, куди буде завантажено AWS CloudFormation стек [22] кожної із заданих функцій (хмарних функцій).
- Файл *serverless.yml* є специфічним для даної функції. В ньому вказуються, зокрема:
  - назва функції;
  - посилання на відповідний обробник в TypeScript-файлі;
  - плагіни;
  - змінні *environmental*;
  - унікальний ідентифікатор AWS SQS-черги [23] (з якої надходитимуть дані про події);
  - AWS IAM-роль [24] (для того, щоб функція мала можливість публікувати нотифікації в SNS-topic).

Синтаксис Serverless фреймворку дозволяє динамічно вказувати значення в конфігураціних файлах, отримуючи їх, наприклад, зі сховища параметрів AWS Systems Manager [25].

На рис. 4 зображено асинхронний обробник подій для визначеної хмарної функції, написаний мовою програмування TypeScript.

Цей обробник здійснює *автоматичний збір даних* (вхідних повідомлень) з SQS-черги *та їхнє подальше структурування,* потім формує push-нотифікації згідно із заданим форматом і відправляє команди на публікацію нотифікацій у SNS-topic.

Для того, щоб відправити push-нотифікацію, потрібно лише створити і надіслати повідомлення зі стороннього сервісу в SQS-чергу (із заданим в конфігураційному файлі *arn*).

Це спричинить виконання хмарної функції, яка, в свою чергу, створить потрібну нотифікацію, що надійде користувачеві на мобільний пристрій.

Слід зазначити, що для того, щоб SNS мав можливість надсилати APNS сповіщення, потрібно надати ключ або сертифікат застосунку, обидва з яких можна отримати зі свого облікового запису розробника Apple.

```
const client = new SNSClient({ region: process.env.REGION });

export async function sendPush(event) {
  const records = event.Records;
  const commands: PublishCommand[] = [];

  for (const record of records) {
    const parsed = parseQueueMessage(record);
    if (parsed) {
      const { deviceEndpoint, payload, ...rest } = parsed;
      const applePushNotificationPayload: ApplePushNotificationPayload = {
        aps: { alert: payload, ...rest },
      };
      const pushNotificationPayload: PushNotification = {
        [DEFAULT_PUSH_NOTIFICATION_KEY]: DEFAULT_MESSAGE,
        [APPLE_PUSH_NOTIFICATION_BODY_KEY]: JSON.stringify(
          applePushNotificationPayload
        ),
        [APPLE_SANDBOX_PUSH_NOTIFICATION_BODY_KEY]: JSON.stringify(
          applePushNotificationPayload
        ),
      };
      commands.push(
        new PublishCommand({
          TargetArn: deviceEndpoint,
          Message: JSON.stringify(pushNotificationPayload),
          MessageStructure: 'json',
        })
      );
    }
  }
  await Promise.all(commands.map((command) => client.send(command)));
  return event;
}
```

Рис. 4. TypeScript код функції для надсилання push-нотифікацій на мобільні пристрої користувачів

Джерело: розробка авторів

Висновки та пропозиції. У статті було розглянуто сутність підходу до автоматизації процесів розгортання хмарних функцій на основі використання Serverless фреймворку. Серед переваг використання Serverless фреймворку слід виділити, зокрема, такі:

- Зменшення завдань щодо інфраструктури, завдяки чому розробники можуть уникнути необхідності управління серверами, резервування ресурсів та налаштування моніторингу; все це забезпечує хмарний постачальник, а розробники можуть займатися лише програмним кодом свого вебдодатку.

- Швидке та достатньо просте розгортання, що особливо корисно для розробників, які мають за мету швидкий випуск продукту на ринок або перевірку нових функцій продукту.
- Еластичність, масштабованість та економія коштів (Serverless автоматично масштабується відповідно до навантаження), що обумовлює те, що користувач сплачує тільки за використані ресурси (виконані функції), має високий рівень доступності до ресурсів та має можливість зменшити витрати на відповідну інфраструктуру.
- Спрощені моніторинг і логування, що спрощує відстеження розробниками ходу виконання програмного коду.
- Збільшення часу безпосередньо на розробку, бо Serverless надає розробникам можливість зосередитися на функціональності та інноваціях свого програмного продукту (вебдодатку), замість витрати часу на вирішення відповідних потрібних адміністративних завдань.

Крім того в статті надано загальний огляд Serverless фреймворку та наведено інструкцію (послідовність дій) щодо створення та розгортання за його допомогою. безсерверних функцій в хмарному середовищі AWS.

Було також розглянуто приклади використання Serverless фреймворку при розв'язанні практичних задач, зокрема створення та розгортання функції для відправлення push-нотифікацій на мобільні пристрої користувачів.

ЛІТЕРАТУРА

1. Serverless. URL: https://www.cloudflare.com/learning/serverless/what-is-serverless (дата звернення: 11.10.2023).

2. Serverless Framework. URL: https://www.serverless.com (дата звернення: 11.10.2023).

3. Infrastructure as Code. URL: https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac (дата звернення: 12.10.2023).

4. Internet of Things. URL: https://www.ibm.com/topics/internet-of-things (дата звернення: 12.10.2023).

5. Serverless Deployment: Best Practices and Examples. URL: https://www.techmagic.co/blog/how-serverless-deployment-works-best-practices-principles-examples (дата звернення: 14.10.2023).

6. AWS Lambda Insights. URL: https://docs.aws.amazon.com/Amazon-CloudWatch/ latest/monitoring/Lambda-Insights.html (дата звернення: 13.10.2023).

7. Using Amazon CloudWatch Lambda Insights to Improve Operational Visibility. URL: https://aws.amazon.com/blogs/aws/using-amazon-cloudwatch-lambda-insights-to-improve-operational-visibility (дата звернення: 13.10.2023).

8. AWS Lambda Cold Start. URL: https://blog.octo.com/cold-start-warm-start-with-aws-lambda (дата звернення: 12.10.2023).

9. How to Improve Serverless Function Cold Start Performance. URL: https://vercel.com/guides/how-can-i-improve-serverless-function-lambda-cold-start-performance-on-vercel (дата звернення: 12.10.2023).

10. Deploying a Simple Serverless Node.js Application on AWS. URL: https://hackernoon.com/deploying-a-simple-serverless-nodejs-application-on-aws-lambda-functions (дата звернення: 14.10.2023).

11. Node.js. URL: https://nodejs.org/en/docs (дата звернення: 15.10.2023).

12. AWS Lambda. URL: https://aws.amazon.com/lambda (дата звернення: 15.10.2023).

13. Function as a Service. URL: https://itglobal.com/ru-ru/company/glossary/faas (дата звернення: 16.10.2023).

14. Scaling on demand. URL: https://www.simpleservers.co.uk/scale-on-demand (дата звернення: 14.10.2023).

15. Serverless Framework Plugins. URL: https://www.serverless.com/plugins (дата звернення: 15.10.2023).

16. Serverless Framework CLI. URL: https://www.serverless.com/framework/ docs/providers/aws/cli-reference/deploy (дата звернення: 12.10.2023).

17. Node Package Manager. URL: https://www.npmjs.com. (дата звернення: 12.10.2023).

18. Serverless.yml reference. URL: https://www.serverless.com/framework/docs/ providers/aws/guide/serverless.yml (дата звернення: 18.10.2023).

19. AWS Cognito. URL: https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/Welcome.html (дата звернення: 15.10.2023).

20. Apple Push Notifications. URL: https://developer.apple.com/notifications. (дата звернення: 12.10.2023).

21. AWS SNS. URL: https://aws.amazon.com/sns (дата звернення: 11.10.2023).

22. AWS CloudFormation. URL: https://aws.amazon.com/cloudformation (дата звернення: 14.10.2023).

23. AWS SQS. URL: https://www.javatpoint.com/aws-sqs (дата звернення: 17.10.2023).

24. AWS IAM. URL: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html (дата звернення: 17.10.2023).

25. AWS Systems Manager. URL: https://aws.amazon.com/systems-manager/features (дата звернення: 17.10.2023).

26. Mobile device endpoint. URL: https://heimdalsecurity.com/blog/mobile-device-management (дата звернення: 16.10.2023).

27. What is Azure? URL: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/ (дата звернення: 16.10.2023).

28. Google Cloud overview. URL: https://cloud.google.com/docs/overview (дата звернення: 16.10.2023).

REFERENCES

1. Serverless, available at: https://www.cloudflare.com/learning/serverless/what-is-serverless (Accessed 11 October 2023).

2. Serverless Framework, available at: https://www.serverless.com. (Accessed 11 October 2023).

3. Infrastructure as Code, available at: https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac (Accessed 12 October 2023).

4. Internet of Things, available at: https://www.ibm.com/topics/internet-of-things (Accessed 12 October 2023).

5. Serverless Deployment: Best Practices and Examples, available at: https://www.techmagic.co/blog/how-serverless-deployment-works-best-practices-principles-examples (Accessed 14 October 2023).

6. AWS Lambda Insights, available at: https://docs.aws.amazon.com/AmazonCloudWatch/ latest/monitoring/Lambda-Insights.html. (Accessed 13 October 2023).

7. Using Amazon CloudWatch Lambda Insights to Improve Operational Visibility, available at: https://aws.amazon.com/blogs/aws/using-amazon-cloudwatch-lambda-insights-to-improve-operational-visibility (Accessed 13 October 2023).

8. AWS Lambda Cold Start, available at: https://blog.octo.com/cold-start-warm-start-with-aws-lambda (Accessed 12 October 2023).

9. How to Improve Serverless Function Cold Start Performance, available at: https://vercel.com/guides/how-can-i-improve-serverless-function-lambda-cold-start-performance-on-vercel (Accessed 12 October 2023).

10. Deploying a Simple Serverless Node.js Application on AWS, available at: https://hackernoon.com/deploying-a-simple-serverless-nodejs-application-on-aws-lambda-functions (Accessed 14 October 2023).

11. Node.js, available at: https://nodejs.org/en/docs (Accessed 15 October 2023).

12. AWS Lambda, available at: https://aws.amazon.com/lambda (Accessed 15 October 2023).

13. Function as a Service, available at: https://itglobal.com/ru-ru/company/glossary/faas (Accessed 16 October 2023).

14. Scaling on demand, available at: https://www.simpleservers.co.uk/scale-on-demand (Accessed 14 October 2023).

15. Serverless Framework Plugins, available at: https://www.serverless.com/plugins (Accessed 15 October 2023).

16. Serverless Framework CLI, available at: https://www.serverless.com/framework/ docs/providers/aws/cli-reference/deploy (Accessed 12 October 2023).

17. Node Package Manager, available at: https://www.npmjs.com (Accessed 12 October 2023).

18. Serverless.yml reference, available at: https://www.serverless.com/framework/docs/ providers/aws/guide/serverless.yml (Accessed 18 October 2023).

19. AWS Cognito, available at: https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/Welcome.html (Accessed 15 October 2023).

20. Apple Push Notifications, available at: https://developer.apple.com/notifications (Accessed 12 October 2023).

21. AWS SNS, available at: https://aws.amazon.com/sns. (Accessed 11 October 2023).

22. AWS CloudFormation, available at: https://aws.amazon.com/cloudformation (Accessed 14 October 2023).

23. AWS SQS, available at: https://www.javatpoint.com/aws-sqs. (Accessed 17 October 2023).

24. AWS IAM, available at: https://docs.aws.amazon.com/IAM/latest/UserGuide/ id_roles.html (Accessed 17 October 2023).

25. AWS Systems Manager, available at: https://aws.amazon.com/systems-manager/features (Accessed 17 October 2023).

26. Mobile device endpoint, available at: https://heimdalsecurity.com/blog/mobile-device-management (Accessed 16 October 2023).

27. What is Azure? available at: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/ (Accessed 16 October 2023).

28. Google Cloud overview, available at: https://cloud.google.com/docs/overview (Accessed 16 October 2023).

СТАТТЯ НАДІЙШЛА ДО РЕДАКЦІЇ 25.09.2023