УДК 004.41
DOI https://doi.org/10.32782/2521-6643-2025-2-70.10

Булгакова О. Ф., старший викладач кафедри комп'ютерних наук та інженерії програмного забезпечення
Університету митної справи та фінансів
ORCID: 0000-0001-9834-2970

Костенко В. В., старший викладач кафедри комп'ютерних наук та інженерії програмного забезпечення
Університету митної справи та фінансів
ORCID: 0000-0003-3847-2110

Булгаков Д. С., студент спеціальності 122 «Комп'ютерні науки»
Університету митної справи та фінансів
ORCID: 0009-0007-8667-3693

АВТОМАТИЗАЦІЯ РОЗГОРТАННЯ СЕРВЕРЛЕС-ЗАСТОСУНКІВ ІЗ ВИКОРИСТАННЯМ IAC: КЕЙС СИСТЕМИ «AGRO MONITOR»

У статті висвітлено результати дослідження та практичної реалізації автоматизації розгортання безсерверних (serverless) застосунків на основі концепції «інфраструктура як код» (Infrastructure as Code, IaC). Основна увага приділяється аналізу сучасних інструментів і технологій, що забезпечують автоматизоване управління інфраструктурою у хмарному середовищі, а також впровадженню serverless-застосунків із використанням IaC у реальному проєкті.

У першій частині статті розглянуто теоретичні основи безсерверної архітектури, яка дозволяє компаніям знижувати витрати на обслуговування інфраструктури, забезпечувати автоматичне масштабування та оплачувати лише фактично використані ресурси. Проаналізовано переваги та виклики застосування serverless-підходу, включно з проблемами взаємодії між компонентами, контролем версій, складнощами інтеграції та питаннями безпеки. Значну увагу приділено платформам функцій як сервісу (FaaS), зокрема AWS Lambda, Azure Functions, Google Cloud Functions та OpenFaaS.

Детально досліджено роль інфраструктури як коду у контексті автоматизації розгортання безсерверних рішень. Розглянуто два основних підходи до опису інфраструктури – декларативний (на прикладі Terraform та AWS CloudFormation) та імперативний IaC (Pulumi) і засоби керування конфігураціями (Ansible), охарактеризовано їхні особливості, переваги та обмеження. Визначено ключові переваги використання IaC для забезпечення стандартизації, відтворюваності, інтеграції з CI/CD-процесами, контролю версій та підвищення безпеки.

Практична частина статті присвячена автоматизації розгортання системи «Agro Monitor», що реалізована на основі безсерверної архітектури у хмарному середовищі Microsoft Azure. Застосовано інструмент Pulumi, який дозволяє описувати інфраструктуру мовами програмування та забезпечує гнучкість при створенні складних інфраструктурних сценаріїв. Описано архітектуру системи, основні компоненти, механізми взаємодії через API, а також процеси автоматичного масштабування, обробки даних та генерації звітів. Показано, що запропонований підхід забезпечує швидке розгортання нових середовищ, зниження витрат на підтримку, підвищення стабільності та розширюваність рішення.

Отримані результати показують ефективність поєднання безсерверної архітектури та інфраструктури як коду для створення сучасних інформаційних систем, що відзначаються високою гнучкістю, стабільністю та економічною доцільністю. Розроблене рішення може бути корисним для підприємств різних галузей, які прагнуть оптимізувати інфраструктуру та підвищити швидкість впровадження інноваційних IT-продуктів.

Ключові слова: *безсерверна архітектура, інфраструктура як код (IaC), Pulumi, автоматизація розгортання, хмарні сервіси, Azure, Terraform, CI/CD, DevOps, супутникові знімки, інформаційні технології.*

Bulhakova O. F., Kostenko V. V., Bulhakov D. S. Automation of serverless application deployment using IaC: case of 'Agro Monitor'

The article presents the results of research and practical implementation of serverless application deployment automation based on the Infrastructure as Code (IaC) concept. The primary focus is placed on analyzing modern tools and technologies that enable automated infrastructure management in cloud environments, as well as the deployment of serverless applications using IaC in a real-world project.

The first part of the article examines the theoretical foundations of serverless architecture, which allows companies to reduce infrastructure maintenance costs, ensure automatic scaling, and pay only for the actual use of computing resources. The

advantages and challenges of adopting the serverless approach are analyzed, including component interaction issues, version control, integration complexity, and security concerns. Particular attention is given to Function-as-a-Service (FaaS) platforms such as AWS Lambda, Azure Functions, Google Cloud Functions, and OpenFaaS.

The article thoroughly investigates the role of Infrastructure as Code in the context of automating the deployment of serverless solutions. Two primary approaches to infrastructure description are considered-declarative declarative (Terraform, AWS CloudFormation) and imperative (Pulumi), and configuration-management tools (Ansible) – with an overview of their characteristics, benefits, and limitations. The key advantages of IaC for ensuring standardization, infrastructure reproducibility, CI/CD process integration, version control, and enhanced system security are highlighted.

The practical section of the article focuses on the deployment automation of the «Agro Monitor» system, which is built on serverless architecture in the Microsoft Azure cloud environment. The Pulumi tool, which enables infrastructure definition using programming languages and provides flexibility in creating complex infrastructure scenarios, is applied. The system architecture, key components, API-based interaction mechanisms, as well as processes for automatic scaling, data processing, and report generation are described in detail. It is demonstrated that the proposed approach enables rapid deployment of new environments, reduces maintenance costs, enhances system stability, and improves scalability.

The results confirm the effectiveness of combining serverless architecture and Infrastructure as Code for developing modern information systems characterized by high flexibility, stability, and economic feasibility. The developed solution can be beneficial for enterprises in various industries seeking to optimize infrastructure and accelerate the deployment of innovative IT products.

Key words: *serverless architecture, Infrastructure as Code (IaC), Pulumi, deployment automation, cloud services, Azure, Terraform, CI/CD, DevOps, satellite imagery, information technologies.*

Постановка проблеми. Активний розвиток інформаційних технологій та посилення конкуренції в IT-сфері зумовлюють необхідність пошуку ефективних підходів до проєктування, розгортання та обслуговування програмного забезпечення. Сучасні компанії стикаються з потребою у прискоренні розробки та впровадження цифрових рішень, зниженні витрат на інфраструктуру, підвищенні надійності та забезпеченні масштабованості інформаційних систем.

Одним із ключових технологічних трендів є поширення хмарних технологій, які відкривають нові можливості для динамічного налаштування інфраструктури під конкретні бізнес-вимоги. У цьому контексті безсерверна архітектура (serverless architecture) привертає значну увагу завдяки здатності автоматично масштабувати ресурси, мінімізувати операційні витрати та забезпечувати оплату лише за фактичне використання обчислювальних потужностей. Проте впровадження serverless-підходу супроводжується низкою викликів, серед яких – складність інтеграції окремих компонентів, контроль життєвого циклу ресурсів, управління версіями та дотримання вимог безпеки.

У відповідь на ці виклики в IT-індустрії активно впроваджуються підходи до автоматизації управління інфраструктурою, зокрема концепція «інфраструктура як код» (Infrastructure as Code, IaC). Вона забезпечує можливість опису інфраструктури за допомогою коду, що сприяє стандартизації налаштувань, автоматизації процесів розгортання, повторюваності інфраструктурних рішень, інтеграції з CI/CD-процесами та підвищенню безпеки систем.

Інфраструктура як код (IaC) стала ключовою практикою відтворюваного, керованого та контрольованого розгортання хмарних систем. Поєднання IaC із безсерверними технологіями (FaaS/Backend-as-a-Service) дає змогу швидко масштабувати прикладні рішення та знижувати операційні витрати. Водночас емпіричні дослідження демонструють, що безсерверні платформи характеризуються різко нерівномірними профілями навантаження, значущою часткою «холодних стартів» і варіабельністю затримок, які безпосередньо впливають на продуктивність і вартість рішень [1-4]. Для керування якістю таких систем критичними стають метричний контроль процесів деплою, метрики якості IaC, автоматизоване тестування IaC-програм і спостережуваність (трасування викликів, моніторинг тригерів) [4, 7, 8, 12]. Паралельно з цим спільнота IaC пропонує таксономії тем і методики вимірювання якості інфраструктурного коду та автоматизованого тестування [6–8]. Окремим викликом є керування безпекою та дрейфом/дефектами узгодження станів, що підривають відтворюваність і відповідність політикам; сучасні роботи фіксують типові «запахи» безпеки, класифікують дефекти state reconciliation і демонструють ефективність GitOps-підходів у керованих середовищах [9-11].

На сьогоднішній день постає нагальна потреба у ґрунтовному аналізі існуючих інструментів і технологій для автоматизації розгортання безсерверних застосунків із використанням IaC, а також у апробації ефективності їх практичного застосування для побудови стабільних, масштабованих та економічно вигідних інформаційних систем.

Ця проблема є актуальною як для малих і середніх підприємств, які прагнуть мінімізувати витрати на інфраструктуру, так і для великих організацій, орієнтованих на високонавантажені сервіси з вимогами до надійності та швидкого масштабування.

Аналіз останніх досліджень та публікацій. У наукових і технічних колах все більше уваги приділяється дослідженню імперативних та декларативних підходів до IaC, а також вибору оптимальних інструментів

для їх реалізації в умовах serverless-архітектур. Особливий інтерес викликає платформа Pulumi, яка поєднує переваги класичних IaC-рішень із можливістю використання популярних мов програмування, що відкриває нові перспективи для інтеграції DevOps-практик.

Попри зростаючу популярність безсерверної архітектури у сучасних IT-системах, питання автоматизації її розгортання за допомогою концепції «інфраструктура як код» (IaC) дотепер залишається недостатньо висвітленим у науковій літературі.

Класичні праці з емпіричного аналізу serverless-платформ (USENIX ATC'18; ATC'20) заклали основу розуміння внутрішніх механізмів та реальних профілів робочих навантажень у хмарах [1, 2]. Подальші дослідження виявили варіабельність продуктивності в часі (зокрема діурнальні патерни) та підвищену частоту cold-start у певні періоди [3], а також показали вагомий внесок тригерів у хвости затримок; бенчмарк TriggerBench систематизує порівняння різних тригерів і провайдерів [4]. Систематичний огляд TOSEM (2023) консолідує 160+ досліджень серверлеса і виділяє відкриті проблеми продуктивності, переносимості, тестування та спостережуваності [5].

У галузі IaC мепінг-дослідження IST (2019) сформувало таксономію напрямів і підсилило фокус на дефектах та ризиках безпеки [6]. Новіші роботи пропонують метрики якості для Terraform-артефактів (TerraMetrics, ICPC'24) [7] і підходи до автоматизованого тестування IaC-програм на великих корпусах відкритого коду (IEEE TSE, 2024) [8]. З погляду безпеки «семеро гріхів» IaC-скриптів (ICSE'19) описують репертуар типових smells і детектування [9]. Проблематику дрейфу конфігурації та дефектів узгодження станів уточнено у FSE'24 (таксономія восьми категорій) [10], а в керованих середовищах (Kubernetes) ефективними демонструються GitOps-практики протидії дрейфу (IEEE UCC'24) [11]. Нарешті, огляди засобів трасування (JSS, 2023) [12] окреслюють можливості/обмеження observability-стеку, що прямо пов'язано з побудовою SLI/SLO та policy-as-code у конвеєрах перевірки.

Управління конфігураційним дрейфом і дефектами узгодження станів стає центральною темою сучасних робіт: у межах FSE'24 запропоновано таксономію восьми категорій дефектів state reconciliation та показано, що частина з них унікальна для IaC-систем [10]; в середовищах Kubernetes переважають GitOps-практики, що демонструють ефективність у виявленні/усуненні дрейфу (IEEE UCC'24) [11]. Нарешті, з боку спостережуваності актуальними є дослідження трасувальних інструментів та їх накладних витрат (JSS'23; сучасні роботи про overhead OpenTelemetry/Elastic) [12], що прямо корелює з потребою кількісно оцінювати SLI/SLO та інтегрувати policy-as-code у конвеєри перевірки.

Таким чином, наявна література підтверджує: (1) serverless-навантаження відзначається високою нерівномірністю викликів і тригерними латентностями; (2) зрілість IaC залежить від наявності метрик якості, автоматизованого тестування та практик керування дрейфом; (3) для реплікованої наукової оцінки необхідні відтворювані експерименти з прозорими трейсами/даними. На цьому тлі запропоновані в роботі практики (IaC-тестування, policy-as-code, SLO-орієнтована спостережуваність, порівняльні метрики деплою/вартості/латентностей) відповідають виявленим прогалинам і доповнюють сучасний стан знань.

Метою статті є обґрунтування доцільності та практичної ефективності застосування концепції «інфраструктура як код» (IaC) для автоматизації процесів розгортання безсерверних застосунків у хмарних середовищах, а також аналіз переваг, обмежень та можливостей використання сучасних інструментів IaC в умовах serverless-архітектур.

У межах дослідження акцентовано увагу на:

– критичному аналізі сучасних підходів до автоматизації розгортання безсерверних систем;

– виявленні існуючих обмежень і проблем, пов'язаних з інтеграцією IaC у DevOps-процеси для serverless-рішень;

– практичній апробації інструментів IaC (зокрема Pulumi) для забезпечення масштабованості, повторюваності та стабільності інфраструктури на прикладі побудови системи аграрного моніторингу у хмарному середовищі Microsoft Azure;

– оцінці доцільності використання імперативних інструментів IaC у порівнянні з традиційними декларативними підходами.

Особливу увагу приділено питанням інтеграції таких рішень із CI/CD-процесами, забезпечення безпеки інфраструктури та зниження витрат на розгортання й супровід хмарних застосунків.

Матеріали та методи. Об'єктом дослідження є процес управління інфраструктурою хмарних інформаційних систем, реалізованих на основі безсерверної архітектури (serverless). Предметом дослідження виступає методика автоматизації розгортання таких систем з використанням сучасних інструментів інфраструктури як коду (IaC).

Методологічна основа дослідження ґрунтується на системному підході до проєктування та розгортання інформаційних систем у хмарному середовищі, зокрема на принципах DevOps, CI/CD (Continuous Integration/Continuous Delivery), GitOps та модульного розділення інфраструктури на незалежні керовані компоненти. У межах дослідження було використано як декларативні, так і імперативні підходи до опису інфраструктури, що дозволило порівняти їх ефективність у різних сценаріях.

Виклад основного матеріалу. Для досягнення цілей дослідження було розроблено архітектуру реального інформаційного застосунку – системи агромоніторингу Agro Monitor, яка використовує подієво-орієнтовану serverless-модель і реалізована у хмарному середовищі Microsoft Azure. Система включає в себе багаторівневу інфраструктуру, що складається з функціональних блоків обробки супутникових зображень, аналітики рослинності (на основі NDVI), зберігання даних, API-шлюзів і механізмів асинхронної взаємодії компонентів.

В якості основного інструмента інфраструктури як коду обрано Pulumi – сучасну IaC-платформу, яка дозволяє описувати конфігурацію хмарних ресурсів із використанням повноцінних мов програмування (у даному випадку – TypeScript). Це забезпечило високу гнучкість опису логіки створення, модифікації та знищення інфраструктурних ресурсів, а також полегшило інтеграцію з зовнішніми бібліотеками, обробку умовних сценаріїв та винятків.

Інтеграція процесів автоматизації розгортання була реалізована через зв'язку Pulumi з GitHub Actions, що дозволило забезпечити повний CI/CD цикл: запуск автоматичних перевірок, конфігурування середовищ, контроль версій змін в інфраструктурі та моніторинг стану хмарних ресурсів. Окрім цього, для конфігурації параметрів доступу та безпеки застосовувалися засоби управління секретами (Azure Key Vault).

Під час реалізації було застосовано наступні технічні інструменти:

– Microsoft Azure – хмарна платформа, що забезпечує обчислювальні, сховищні та подієві сервіси.
– Pulumi (TypeScript) – імперативний IaC-фреймворк для мультихмарних середовищ.
– Azure Functions – серверлес-сервіси для виконання обчислень.
– Azure Blob Storage та Cosmos DB – для зберігання супутникових знімків і структурованих даних.
– Event Grid та Logic Apps – для оркестрації подій.
– GitHub Actions – для CI/CD-автоматизації.
– Visual Studio Code та Node.js – середовище розробки та виконання інфраструктурного коду.

Для моделювання архітектури застосовувались стандартні підходи до багатошарового поділу системи (data layer, logic layer, presentation layer), що забезпечило масштабованість та незалежність компонентів. Весь процес розгортання був відтворюваним, з можливістю швидкого оновлення середовищ або повного знищення інфраструктури без залишків.

Таким чином, у межах дослідження було реалізовано повний цикл створення, розгортання та управління безсерверною інформаційною системою із використанням імперативного підходу IaC, інтегрованого у DevOps-процеси.

У процесі розробки та автоматизації розгортання системи «Agro Monitor» були використані сучасні технічні інструменти, які забезпечили повний життєвий цикл serverless-застосунку – від проєктування архітектури до реалізації CI/CD-процесів.

Хмарна платформа Microsoft Azure слугувала базовим середовищем для розміщення всіх компонентів системи. Було задіяно низку її сервісів, зокрема:

– Azure Functions для обробки подій та реалізації серверлес-обчислень;
– Azure Blob Storage для зберігання великих обсягів супутникових зображень;
– Cosmos DB як високопродуктивну NoSQL-базу для структурованих аналітичних даних;
– Event Grid для маршрутизації подій між компонентами;
– Azure Key Vault для зберігання секретів, токенів доступу та конфіденційної інформації.

Azure забезпечив надійну, масштабовану та безпечну інфраструктурну основу для системи з гнучким керуванням доступом і можливістю централізованого моніторингу.

IaC-фреймворк Pulumi був використаний як основний інструмент автоматизації. У рамках проєкту інфраструктура застосунку (включаючи Azure Functions, Blob Storage, Cosmos DB, App Services тощо) описувалась у вигляді TypeScript-коду.

Завдяки цьому:

– було досягнуто високої гнучкості при конфігурації інфраструктури (використання циклів, умов, функцій);
– код інфраструктури став частиною загального репозиторію з логікою застосунку;
– забезпечено повну відтворюваність розгортання через використання state-файлів Pulumi та логування всіх змін.
– Функції Azure стали основним обчислювальним середовищем для реалізації серверлес-логіки. Вони:
– реагували на події з Event Grid (наприклад, надходження нового знімка до Blob Storage);
– запускали обробку NDVI-зон, створення аналітичних карт та генерацію звітів;
– масштабувалися автоматично відповідно до навантаження.

Завдяки серверлес-підходу, вдалося мінімізувати витрати на постійне утримання інфраструктури та забезпечити високу відмовостійкість.

Blob Storage використовувався для зберігання великих файлів супутникових зображень (у форматах GeoTIFF, JPEG, PNG).

Cosmos DB зберігав метадані знімків, інформацію про ділянки, оброблені NDVI-значення, агрономічні параметри.

Ці сервіси були інтегровані з Azure Functions через SDK, що дозволило реалізувати подієво-орієнтований підхід.

Event Grid слугував маршрутизатором подій у системі: завантаження нового зображення, оновлення даних, запити від користувача тощо. У разі складніших сценаріїв обробки подій було задіяно Logic Apps, які дозволяли безкодовим способом визначати реакції на певні тригери (наприклад, зберегти NDVI-звіт у PDF і надіслати його електронною поштою).

Для забезпечення повного CI/CD-процесу було створено робочі процеси (workflows) у GitHub Actions, які:

– автоматично запускали перевірку змін у коді;
– виконували linting і тестування;
– запускали Pulumi-скрипти для оновлення інфраструктури;
– реєстрували нові функції Azure Functions та оновлювали конфігурації.

Це дозволило досягти швидкого, контрольованого та безпечного оновлення системи при кожному коміті.

Як середовище розробки використовувалася Visual Studio Code з плагінами для Pulumi, TypeScript і Azure CLI. Node.js був платформою для виконання Pulumi-скриптів, а також середовищем для тестування логіки обробки функцій до їхнього деплою в Azure.

Ці інструменти діяли в єдиній екосистемі, забезпечуючи автоматизоване, масштабоване й контрольоване розгортання інфраструктури та застосунку, що повністю відповідало концепціям DevOps та хмарної інженерії.

Реалізація. У результаті дослідження було розроблено і впроваджено інформаційну систему «Agro Monitor», яка призначена для автоматизованої обробки супутникових знімків сільськогосподарських угідь з метою виявлення зон стресу рослинності та підтримки прийняття рішень агрономами. Система реалізована у хмарному середовищі Microsoft Azure із використанням безсерверної архітектури, що забезпечує масштабованість, високу доступність та ефективне використання ресурсів.

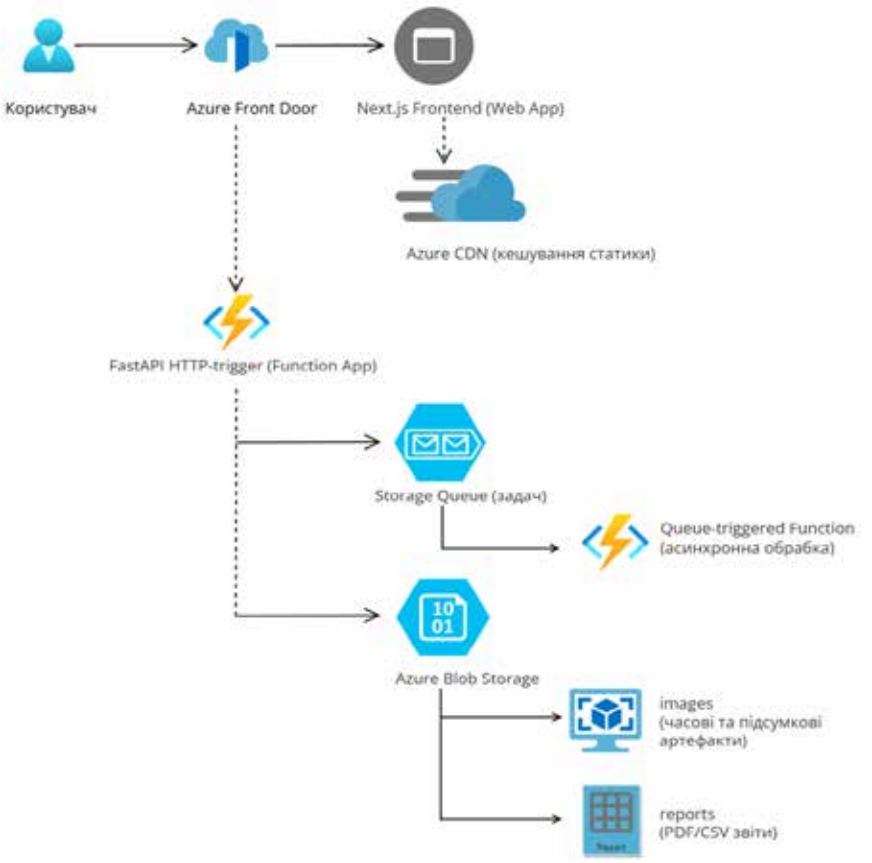

Рис. 1. Схема взаємодії основних компонентів системи «Agro Monitor»

Архітектура системи побудована на принципах багаторівневої моделі, кожен з рівнів якої виконує окрему функцію:

– Рівень даних відповідає за зберігання та обробку великих масивів супутникових знімків, використовуючи Azure Blob Storage для неструктурованих даних (зображень) та Cosmos DB – для зберігання структурованої аналітичної інформації (NDVI-значення, координати ділянок, історичні дані).

– Обчислювальний рівень реалізовано за допомогою Azure Functions, які обробляють події, що надходять через Event Grid. Тут відбувається обчислення NDVI, класифікація рослинного покриву, генерація карт та звітів.

– Рівень оркестрації та інтеграції забезпечується зв'язкою Event Grid та Logic Apps, які автоматично запускають відповідні обчислення при завантаженні нових знімків чи зміні параметрів користувача.

– Інфраструктурний рівень описаний у вигляді коду за допомогою Pulumi (TypeScript), що дозволяє здійснювати автоматичне створення, оновлення та знищення ресурсів у хмарному середовищі. Інфраструктура є відтворюваною, контрольованою та масштабованою.

– Рівень автоматизації реалізований за допомогою GitHub Actions, що забезпечує безперервну інтеграцію та доставку змін (CI/CD): кожна зміна у коді інфраструктури чи логіки викликає відповідний процес перевірки, тестування та оновлення середовища.

Рис. 2. Файлова структура репозиторію проєкту «Agro Monitor»

Візуально структура системи відображена на рисунку 1, що демонструє взаємодію компонентів і подієво-орієнтовану модель. Архітектура побудована таким чином, щоб підтримувати принцип незалежності сервісів і масштабування за запитом.

Організація репозиторію проєкту представлена на рисунку 2 і демонструє логічну ізоляцію інфраструктурного коду, логіки обробки зображень та CI/CD-налаштувань. Такий підхід забезпечує зручність супроводу та розширення системи.

На рисунку 3 показано приклад сформованого звіту на основі аналізу NDVI: система автоматично генерує візуалізацію зон стресу рослин з поясненнями, що можуть бути використані у польових умовах агрономами. Рисунок 4 ілюструє приклади супутникових зображень на різних етапах обробки – від сирого знімка до NDVI-мапи та зони вегетаційного стресу.

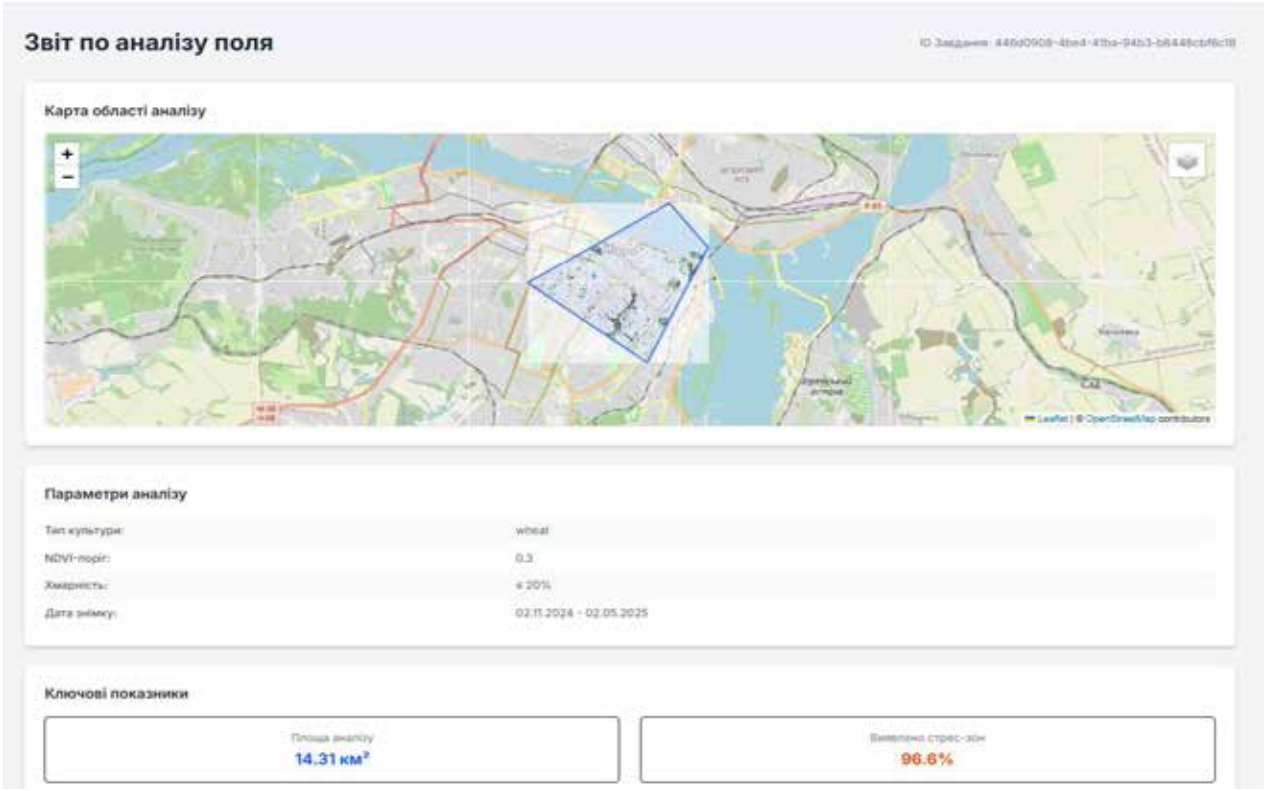

Рис. 3. Вигляд результатів роботи системи «Agro Monitor»

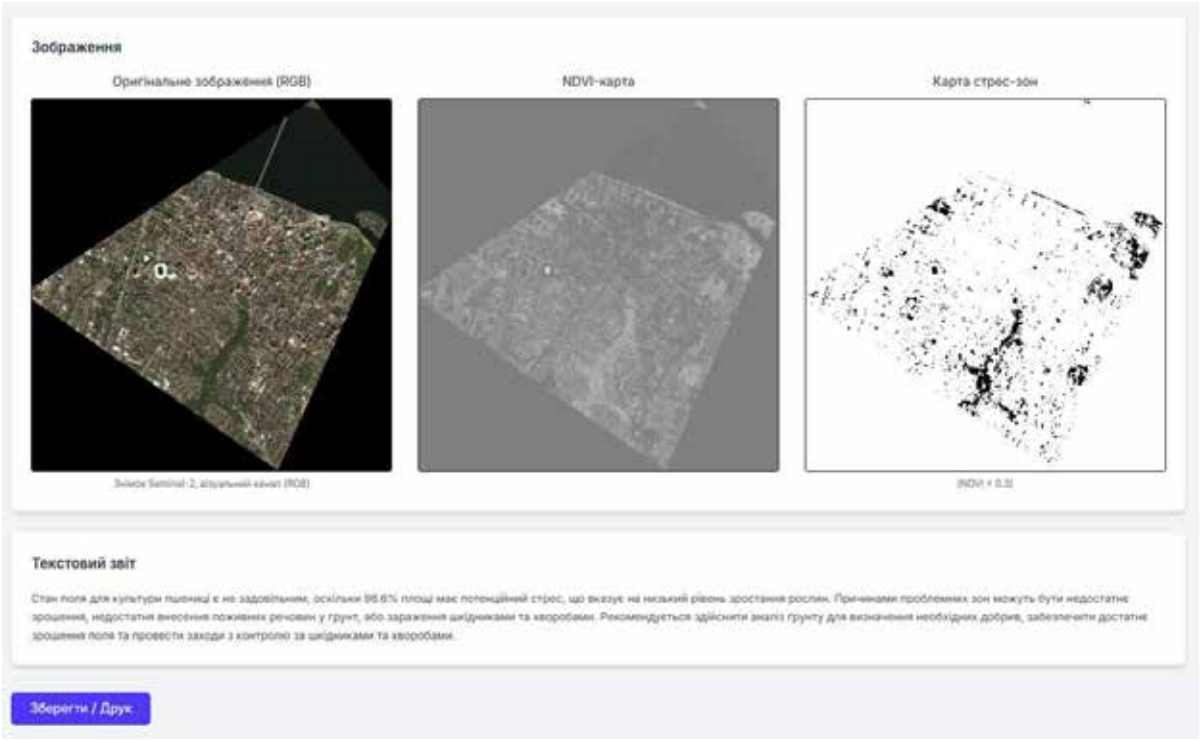

Рис. 4. Вигляд супутникових знімків, опрацьованих за допомогою системи «Agro Monitor»

Таким чином, система «Agro Monitor» не лише демонструє ефективність автоматизованого підходу до обробки агроданих, але й підтверджує практичну реалізованість принципів IaC, DevOps та serverless у реальних галузевих рішеннях.

Запропоноване рішення істотно відрізняється від класичних підходів до розгортання інформаційних систем, зокрема у сфері агромоніторингу. Насамперед, варто відзначити переваги автоматизованого підходу із застосуванням інфраструктури як коду (IaC) у порівнянні з традиційними методами ручного налаштування серверів і сервісів. Реалізація системи «Agro Monitor» за допомогою Pulumi продемонструвала значне скорочення часу, необхідного для розгортання інфраструктури, а також забезпечила повну відтворюваність конфігурацій, що є критично важливим при масштабуванні або міграції рішень.

З точки зору практичної значущості, реалізоване рішення має високий рівень адаптованості до інших хмарних середовищ. Хоча проєкт було реалізовано в Microsoft Azure, архітектурна модель і обрані інструменти є мультихмарними за своєю природою. Зокрема, використання Pulumi як імперативного засобу IaC дозволяє за потреби перенести систему до AWS, Google Cloud чи інших провайдерів з мінімальними модифікаціями, що робить рішення універсальним та масштабованим.

Разом з тим, проєкт має певні обмеження. Основним з них є залежність від конкретної хмарної платформи (у даному випадку – Azure), що обумовлює необхідність враховувати специфіку її сервісів, моделей безпеки та тарифікації. Крім того, використання Pulumi потребує певного рівня знань із програмування, що створює додаткову навчальну криву для фахівців, які звикли до декларативних підходів (наприклад, Terraform або CloudFormation).

Особливі умови воєнного стану в Україні також вплинули на реалізацію дослідження. Було зафіксовано періоди ускладненого доступу до інтернету та обмеженого фізичного доступу до локальних ресурсів, що вимагало адаптації командної роботи до дистанційного формату. У таких умовах важливу роль відіграв централізований доступ до хмарної інфраструктури та можливість її керування з будь-якої геолокації.

У межах перспектив подальших досліджень розроблену архітектуру планується адаптувати до інших галузей, де потрібна обробка великих обсягів даних у реальному часі – зокрема в екологічному моніторингу, урбаністиці, безпеці. Також передбачається порівняльне тестування цього підходу у середовищах інших хмарних провайдерів, що дозволить визначити оптимальні конфігурації та виявити вузькі місця в кросплатформному розгортанні.

Висновки та перспективи подальших досліджень. У межах дослідження було розроблено і впроваджено методику автоматизації розгортання інформаційних систем із безсерверною архітектурою на прикладі системи агромоніторингу «Agro Monitor». Основною технологічною основою рішення виступив імперативний підхід до інфраструктури як коду з використанням платформи Pulumi у середовищі Microsoft Azure.

Суттєвими результатами є:

– реалізація повністю автоматизованого процесу створення, конфігурації та оновлення компонентів системи;
– зменшення часу розгортання інфраструктури до кількох хвилин;
– забезпечення повторюваності, масштабованості та зручності супроводу системи;
– глибока інтеграція інфраструктурного коду в CI/CD-процеси за допомогою GitHub Actions.

Отримані результати пояснюються ефективністю поєднання serverless-архітектури, імперативної IaC-моделі та подієво-орієнтованого підходу до обробки даних. Застосування Microsoft Azure дозволило реалізувати високопродуктивну систему, адаптовану до аграрних задач реального часу.

Практична користь полягає у створенні рішення, що може бути масштабовано на інші хмарні платформи та галузі, а також використане для підвищення ефективності обробки даних в умовах динамічного навантаження.

У подальших дослідженнях доцільно:

– розширити застосування методики на інші типи застосунків (екологія, безпека, міська аналітика);
– протестувати систему в мультихмарному середовищі;
– провести кількісне порівняння із декларативними підходами IaC.

Таким чином, запропоноване рішення підтверджує ефективність і практичну доцільність використання імперативної IaC-платформи Pulumi для побудови сучасних serverless-систем.

Список використаних джерел:

1. Wang L.; Li M.; Zhang Y.; Ristenpart T.; Swift M. Peeking Behind the Curtains of Serverless Platforms. Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC ’18). Boston, MA, USA, 11–13 July 2018. С. 133–145.

2. Shahrad M., Fonseca R., Zhang I., et al. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC ’20). 2020. С. 205–218.

3. Schirmer T., Japke N., Greten S., Pfandzelter T., Bermbach D. The Night Shift: Understanding Performance Variability of Cloud Serverless Platforms. The 1st Workshop on SErverless Systems, Applications and MEthodologies (SESAME ’23), Rome, Italy, 8 May 2023. Стаття 7. DOI: 10.1145/3592533.3592808.

4. Scheuner J., Bertilsson M., Grönqvist O., et al. TriggerBench: A Performance Benchmark for Serverless Function Triggers. 2022 IEEE International Conference on Cloud Engineering (IC2E), Monterey, CA, USA, 28 Sept. 2022. DOI: 10.1109/IC2E55432.2022.00018.

5. Wen J., Chen Z., Jin X., Liu X. Rise of the Planet of Serverless Computing: A Systematic Review. *ACM Transactions on Software Engineering and Methodology*. 2023. Т. 32. № 5. Стаття 131. DOI: 10.1145/3579643.

6. Rahman A., Mahdavi-Hezaveh R., Williams L. A Systematic Mapping Study of Infrastructure as Code Research. *Information and Software Technology*. 2019. Т. 108. С. 65–77. DOI: 10.1016/j.infsof.2018.12.004.

7. Begoug M., Chouchen M., Ouni A. TerraMetrics: An Open Source Tool for Infrastructure-as-Code (IaC) Quality Metrics in Terraform. Proceedings of the 2024 IEEE/ACM 32nd International Conference on Program Comprehension (ICPC) – Tool Demonstrations. 2024. DOI: 10.1145/3643916.3644439.

8. Sokolowski D., Spielmann D., Salvaneschi G. Automated Infrastructure as Code Program Testing. *IEEE Transactions on Software Engineering*. 2024. Т. 50. № 6. С. 1585–1599. DOI: 10.1109/TSE.2024.3393070.

9. Rahman A., Parnin C., Williams L. The Seven Sins: Security Smells in Infrastructure as Code Scripts. Proceedings of the 41st International Conference on Software Engineering (ICSE ’19). 2019. С. 164–175. DOI: 10.1109/ICSE.2019.00033.

10. Hassan M. M., Salvador J., Santu S. K. K., Rahman A. State Reconciliation Defects in Infrastructure as Code. Proceedings of the ACM on Software Engineering. 2024. Т. 1 (FSE). Стаття 83. DOI: 10.1145/3660790.

11. Shrestha R., Ali A. A. N. Configuration Management in Kubernetes Environments: A GitOps Approach. 2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC 2024), Sharjah, UAE, 16–19 Dec. 2024. С. 497–502. DOI: 10.1109/UCC63386.2024.00077.

12. Janes A., Li X., Lenarduzzi V. Open Tracing Tools: Overview and Critical Comparison. *Journal of Systems and Software*. 2023. Т. 204. Стаття 111793. DOI: 10.1016/j.jss.2023.111793.

References:

1. Wang, L., Li, M., Zhang, Y., Ristenpart, T., & Swift, M. (2018). Peeking behind the curtains of serverless platforms. In Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC ’18) (pp. 133–145). Boston, MA, USA. Retrieved from: https://www.usenix.org/conference/atc18/presentation/wang-liang

2. Shahrad, M., Fonseca, R., Zhang, I., et al. (2020). Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC ’20) (pp. 205–218). Retrieved from: https://www.usenix.org/conference/atc20/presentation/shahrad

3. Schirmer, T., Japke, N., Greten, S., Pfandzelter, T., & Bermbach, D. (2023). The night shift: Understanding performance variability of cloud serverless platforms. In The 1st Workshop on SErverless Systems, Applications and MEthodologies (SESAME ’23). https://doi.org/10.1145/3592533.3592808

4. Scheuner, J., Bertilsson, M., Grönqvist, O., et al. (2022). TriggerBench: A performance benchmark for serverless function triggers. 2022 IEEE International Conference on Cloud Engineering (IC2E). https://doi.org/10.1109/IC2E55432.2022.00018

5. Wen, J., Chen, Z., Jin, X., & Liu, X. (2023). Rise of the planet of serverless computing: A systematic review. *ACM Transactions on Software Engineering and Methodology*, 32(5), Article 131. https://doi.org/10.1145/3579643

6. Rahman, A., Mahdavi-Hezaveh, R., & Williams, L. (2019). A systematic mapping study of infrastructure as code research. *Information and Software Technology*, 108, 65–77. https://doi.org/10.1016/j.infsof.2018.12.004

7. Begoug, M., Chouchen, M., & Ouni, A. (2024). TerraMetrics: An open source tool for infrastructure-as-code (IaC) quality metrics in Terraform. Proceedings of the 2024 IEEE/ACM 32nd International Conference on Program Comprehension (ICPC) – Tool Demonstrations. https://doi.org/10.1145/3643916.3644439

8. Sokolowski, D., Spielmann, D., & Salvaneschi, G. (2024). Automated infrastructure as code program testing. IEEE Transactions on Software Engineering, 50(6), 1585–1599. https://doi.org/10.1109/TSE.2024.3393070

9. Rahman, A., Parnin, C., & Williams, L. (2019). The seven sins: Security smells in infrastructure as code scripts. Proceedings of the 41st International Conference on Software Engineering (ICSE ’19), 164–175. https://doi.org/10.1109/ICSE.2019.00033

10. Hassan, M. M., Salvador, J., Santu, S. K. K., & Rahman, A. (2024). State reconciliation defects in infrastructure as code. Proceedings of the ACM on Software Engineering, 1 (FSE), Article 83. https://doi.org/10.1145/3660790

11. Shrestha, R., & Ali, A. A. N. (2024). Configuration management in Kubernetes environments: A GitOps approach. 2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC 2024), 497–502. https://doi.org/10.1109/UCC63386.2024.00077

12. Janes, A., Li, X., & Lenarduzzi, V. (2023). Open tracing tools: Overview and critical comparison. *Journal of Systems and Software*, 204, 111793. https://doi.org/10.1016/j.jss.2023.111793

Дата надходження статті: 01.09.2025
Дата прийняття статті: 10.10.2025
Опубліковано: 30.12.2025