Володимир Малахівський, Ольга Мокрицька*
[1]НЛТУ України, Львів, Україна.ORCID:0009-0004-7510-707X,23malakhivskyi.v@nltu.lviv.ua
[2]НЛТУ України, Львів, Україна,ORCID:0000-0002-2887-9585,olha.v.mokrytska@lpnu.ua

Розроблення та оптимізація Serverless застосунку на платформі.NET з використанням сучасних методів та інструментів

Анотація. У даній роботі розглянуто створення серверлесного застосунку з використанням платформи.NET, а також оптимізацію його продуктивності та ефективності за допомогою сучасних методів і інструментів. Робота охоплює дослідження переваг і викликів Serverless-архітектури, її інтеграцію з хмарними сервісами, такими як AWS Lambda та Azure Functions, а також впровадження CI/CD (Continuous Integration/Continuous Deployment) для автоматизації процесів тестування та розгортання застосунку. Здійснено оцінку впливу використаних технологій на зниження витрат, підвищення продуктивності і забезпечення масштабованості застосунку.

Ключові слова – Serverless,.NET, AWS Lambda, Azure Functions, CI/CD, моніторинг.

У цій роботі досліджено можливості та особливості використання Serverless-підходу на платформі.NET для створення високопродуктивних і ресурсоощадних застосунків. Використовуючи Amazon Web Services (AWS) Lambda, було протестовано різні конфігурації для визначення оптимальних налаштувань масштабованості та продуктивності. CI/CD (Continuous Integration/Continuous Delivery/Deployment) – це практика та набір процесів для автоматизації розробки, тестування та розгортання програмного забезпечення. CI/CD можна уявити як конвеєр, де новий код подається на одному кінці, тестується на кількох етапах (вихідний код, збірка, тестування, постановка та виробництво), а потім публікується як готовий до виробництва код[1]. Давайте розглянемо детально кожну складову CI/CD:

* науковий керівник

- **Continuous Integration (CI) – Безперервна інтеграція.** це програмна практика, яка вимагає частого внесення коду до спільного репозиторію[2], в основну гілку (main або master);
- **Continuous Delivery (CD) – Безперервна доставка.** Continuous Delivery – це процес підготовки програмного забезпечення до випуску у виробниче середовище (production) Отже, щоб воно було завжди готове до розгортання;
- **Continuous Deployment – Безперервне розгортання.** Continuous Deployment автоматизує весь процес розгортання нових версій програмного забезпечення на продакшен. Це наступний крок після CD, у якому кожен успішний тест автоматично запускає розгортання у виробниче середовище.

  Приклад коду для GitLab CI/CD (.gitlab-ci.yml):

```
stages:
- build
- deploy
variables:
IMAGE_NAME: "registry.gitlab.com/username/repository-name/root-image" # Замість registry.gitlab.com використайте вашу реєстрацію
build:
stage: build
image: docker:latest
services:
- docker:dind
script:
- echo "Building Docker image..."
- docker build -t $IMAGE_NAME:$CI_COMMIT_REF_NAME.
- docker push $IMAGE_NAME:$CI_COMMIT_REF_NAME
only:
- main
- tags
deploy:
stage: deploy
script:
- echo "Deploying Docker image..."
# Додайте команду для деплою, наприклад, виклик через SSH або kubectl для Kubernetes
- docker pull $IMAGE_NAME:$CI_COMMIT_REF_NAME
- docker run -d $IMAGE_NAME:$CI_COMMIT_REF_NAME
only:
- tags
```


* науковий керівник

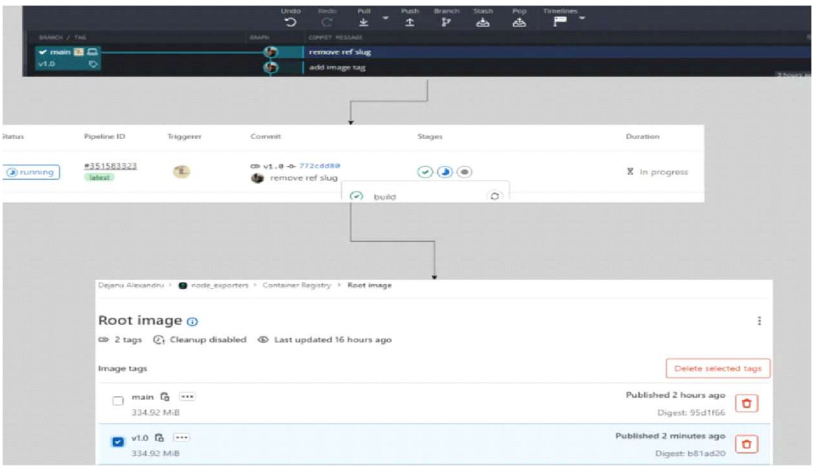

Рисунок 1. Інтерфейс CI/CD конвеєра

На рисунку 1, зображено інтерфейс CI/CD конвеєра, який використовує графічний інтерфейс для моніторингу етапів виконання конвеєра:

- основну гілку (main) та тег (v1.0): У верхній частині видно граф, який показує дві гілки – основну (main) і тег (v1.0). Це вказує на те, що зміни відбуваються в основній гілці або гілці, позначеній тегом;
- Pipeline ID та статус виконання: У секції "Status" показано, що конвеєр (Pipeline ID: 351583323) перебуває в статусі running (виконується). Це означає, що конвеєр наразі працює та виконує один або кілька етапів;
- Commit та Triggerer: Видно інформацію про коміт (v1.0 - 772cdd88) та особу, яка запустила цей конвеєр. Це коміт, який містить певні зміни з описом remove ref slug;
- етапи (Stages): Показано етапи виконання конвеєра. На даний момент виконуються або виконані етапи build – етап зборки додатка. Біля нього видно статус, який свідчить, що він наразі виконується (піктограма зі стрілкою);
- Container Registry: У нижній частині скріншота показано кореневий Docker-образ ("Root image") з двома тегами – main і v1.0. Це може свідчити про те, що конвеєр автоматично створює Docker-образ і завантажує його в реєстр контейнерів з тегами, які відповідають основній гілці та версії додатка (v1.0).

* науковий керівник

Кожен з цих тегів має розмір (334.92 MiB), і вказано час останньої публікації для кожного тега. Є можливість видалити обрані теги, що забезпечує легке керування версіями контейнерів у реєстрі.

Висновок. Дослідження продемонструвало, що Serverless-архітектура на платформі.NET із використанням сучасних хмарних сервісів та автоматизованих процесів CI/CD забезпечує високу продуктивність і стабільність додатків. Використання серверлесного підходу дає змогу мінімізувати витрати та швидко адаптуватися до змін у навантаженні, а автоматизація тестування і розгортання підвищує швидкість розвитку продукту. Застосовані методи забезпечили значні покращення у швидкодії, надійності та економічності додатка, що є перспективним для подальшого застосування в сучасних веб та бізнес-додатках.

Список використаних літературних джерел

[1] Continuous Integration and Continuous Delivery for 5G Networks on AWS Retrieved from https://docs.aws.amazon.com/whitepapers/latest/cicd_for_5g_networks_on_aws/cicd-on-aws.html

[2] GitHub Actions Documentation. «Automating.NET Core Builds with GitHub Actions.Retrieved from https://docs.github.com/en/actions/about-github-actions/about-continuous-integration-with-github-actions

Development and optimisation of a Serverless application on the.NET platform using modern development methods and tools

Malahivskyi Volodymyr, Mokrytska Olha

This paper discusses the creation of a serverless application using the.NET platform, as well as the optimisation of its performance and efficiency using modern methods and tools. The work includes the study of the benefits and challenges of serverless architecture, its integration with cloud services such as AWS Lambda and Azure Functions, as well as the implementation of CI/CD (Continuous Integration/Continuous Deployment) to automate the testing and deployment of the application. The impact of the technologies used on reducing costs, increasing productivity and ensuring the scalability of the application was assessed.

* науковий керівник