УДК 004.75
DOI https://doi.org/10.32782/tnv-tech.2025.2.6

ФУНКЦІЯ ЯК СЕРВІС У ПРИВАТНІЙ ХМАРІ ПІДПРИЄМСТВА

Гіоргізова-Гай В. Ш. *– кандидат технічних наук,*
доцент кафедри системного проектування
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID ID: 0000-0001-6224-3532

У статті проводиться аналіз переваг та обмежень технології FaaS, тенденцій розвитку хмарних технологій, що сприяють її застосуванню у власній хмарній інфраструктурі організацій, та додаткових переваг, які вона може в ній надати. Також проводиться порівняння найбільш поширених FaaS фрейворків з відкритим вихідним кодом. Технологія FaaS модель хмарних обчислень, що пропонує платформу для виконання функціонально завершених фрагментів коду, які ініціюються подіями. Платформа забезпечує автоматичне масштабування та керування обчислювальними ресурсами, абстрагує низькорівневі інфраструктурні рішення від розробників. FaaS спрямована на підвищення ефективності роботи програмістів, зменшення експлуатаційних витрат та економію коштів при оплаті тільки за спожиті ресурси провайдера. Ця модель може забезпечити переваги при рішенні широкого кола задач. Популярність і використання у виробництві FaaS зростає. Однак послуги публічних провайдерів не можуть задовольнити всі потреби користувачів. Подолати обмеження публічних хмар дозволяють моделі приватної і гібридної хмари, вибір яких для побудови корпоративної хмарної інфраструктури спостерігається як стійка тенденція у світі. Серед інших тенденцій, що сприяють застосуванню платформ FaaS у власних хмарах організацій: поширення контейнерного підходу, мікросервісних та подієво-орієнтованих архітектур при побудові додатків, орієнтація на платформу оркестрації та розгортання контейнерів Kubernetes, поширення пропозицій Managed Kubernetes від хмарних провайдерів. Застосування фрейворків FaaS добре поєднується з цими технологіями та додатково дозволяє більш ефективно використовувати ресурси кластеру Kubernetes, долати обмеження, властиві послугам FaaS провайдерів, уникати прив'язки до одного постачальника, розгортати додатки в будь-якій хмарній інфраструктурі. Серед існуючих відкритих FaaS платформ з підтримкою Kubernetes для більш детального порівняння були обрані OpenFaas, OpenWhisk та Knative, які є зрілими рішеннями, мають широке застосування та активну підтримку. Знання їх характеристик та особливостей допоможе організаціям у виборі найбільш відповідної їх цілям та бізнес-завданнямв.

Ключові слова: *розподілені системи, приватна хмара, FaaS, Kubernetes, OpenFaas, OpenWhisk, Knative.*

Hiorhizova-Hai V. S. Function as a service in the enterprise's private cloud

The article analyzes the advantages and limitations of FaaS technology, trends in the development of cloud technologies that contribute to its use in the organization's own cloud infrastructure, and additional benefits that it can provide. A comparison of the most common open source, self-hosted FaaS frameworks is also provided. FaaS technology is a cloud computing model that provides a platform for executing functionally complete pieces of code that are triggered by events. The platform provides automatic scaling and management of computing resources, abstracting low-level infrastructure decisions away from developers. FaaS is aimed at increasing the efficiency of programmers, reducing operating costs and saving money when paying for provider resources on demand. This model can provide benefits for a wide range of tasks. FaaS is growing in popularity and use in production. However, public providers cannot meet all user needs. Overcoming the limitations of public clouds is possible with private and hybrid cloud models, the choice of which for building corporate cloud infrastructure is observed as a stable trend in the world. Other trends driving the adoption of FaaS platforms in organizations' own clouds include the rise of container-based, microservice-based, and event-driven architectures in application development, a focus on the Kubernetes container orchestration and deployment platform, and the rise of Managed Kubernetes offerings from cloud providers. The use of FaaS frameworks fits well with these technologies and additionally allows for more efficient use of Kubernetes cluster resources, overcom the limitations inherent in public FaaS services,

avoid provider-lock, and deploying applications in any cloud infrastructure. Among the existing Open Source self-hosted FaaS frameworks with Kubernetes support, OpenFaas, OpenWhisk and Knative were selected for a more detailed comparison, which are mature solutions, have a wide range of users and active support. Knowing their characteristics and features will help organizations in choosing the platform that best suits their goals and business tasks.

Key words: *distributed systems, private cloud, FaaS, Kubernetes, OpenFaas, OpenWhisk, Knative.*

Постановка проблеми. Безсерверні технології в першу чергу асоціюються з послугами відомих хмарних провайдерів [1–3]. А можливості, які може надати застосування FaaS в приватній хмарній інфраструктурі, організаціями оцінені ще не повною мірою, зокрема в Україні. Проте, використання послуг публічних провайдерів не можуть задовольнити всі потреби користувачів. Тоді як фреймворки FaaS з самостійним розміщенням у приватній чи гібридній хмарі компанії наряду з перевагами, властивими цій технології, можуть подолати ряд обмежень аналогічних публічних рішень. Аналізу цих питань та порівнянню особливостей найбільш застосовуваних FaaS фреймворків з відкритим вихідним кодом та підтримкою популярної платформи Kubernetes, присвячена дана стаття.

Аналіз останніх досліджень і публікацій. Функція як сервіс (Function as a Service, FaaS) – модель хмарних обчислень, що пропонує платформу для виконання невеликих фрагментів коду (функцій), які ініціюються подіями. Платформа забезпечує автоматичне масштабування та керування обчислювальними ресурсами, абстрагує низькорівневі інфраструктурні рішення від розробників, що дозволяє їм зосереджуватись на коді самих функцій. Джерелами подій можуть виступати HTTP-запити, повідомлення в черги повідомлень, зміни або поява даних. Також функції можна запускати за таймером.

Деякі важливі характеристики FaaS:

1. Обчислювальні ресурси розподіляються динамічно в міру необхідності платформою.

2. Ціноутворення на основі споживання: користувач платить тільки за обчислювальні ресурси, використовувані під час виконання коду функції.

3. Обчислювальні ресурси масштабируются за запитом в залежності від трафіку, без необхідності настройки розробника.

Переваги послуг FaaS

Для операторів хмарних систем застосування FaaS вигідно ефективнішим споживанням ресурсів – функції запускаються тільки при виникненні необхідності і відразу після обробки події завершують свою роботу, тобто на відміну від мікросервісов не вимагають постійної наявності запущених оточень, які споживають ресурси на холостому ходу.

Для користувачів:

1) Більш швидкий випуск продукції на ринок. Використання безсерверної моделі дозволяє розробникам покращити бізнес-логіку без розгортання всієї програми, менший і, отже, більш зрозумілий обсяг коду, можливість писати кожну функцію на іншій мові програмування дозволяє розробникам швидко оновлювати код та експериментувати. Підхід також дозволяє командам спільно використовувати кінцеві точки і паралельно виконувати свою роботу. Розробники мобільних додатків можуть створювати бізнес-логіку, не володіючи експертними знаннями про роботу сервера.

2) Можливість мінімізувати виконання експлуатаційних завдань і доступна масштабованість сприяють покращанню рівня надання послуги (SLA) сервісам

додатків. Провайдери хмарних послуг, докладають безліч зусиль для забезпечення доступності, високої продуктивності та масштабованості додатків своїх користувачів.

3) Економія витрат на обслуговування нерегулярних і стрибкоподібних навантажень. Клієнтам не потрібно платити за підтримку віртуальних серверів в робочому стані під час простоюють. Крім того, будь-яка оптимізація продуктивності FaaS-додатку (скорочення часу виконання дій певних функцій) автоматично знижує вартості послуги без змін в інфраструктурі.

Отже, технологія FaaS спрямована на те, щоб зробити розгортання додатків більш швидким, гнучким, економічнім и масштабованім. Але на ряду з перевагами технології властиві обмеження [4, с. 82; 5].

Обмеження технології FaaS

1) Додаток, який виконує багато ініціалізацій і критичній до затримок, не буде добре працювати з FaaS через затримки запуску, необхідну на ініціалізацію нового екземпляра функції, після тривало часу її не активності, так званий «холодний старт». Він може сягати 3сек, а іноді й більше. Однак, не для всіх задач це критично. Також існує багато підходів для скорочення цього часу, наприклад, в [6].

2) Кожна функція є незалежною. Всі взаємодії здійснюється по мережі. Примірники функцій не мають власної пам'яті, отже, вони вимагають наявності загального сховища для зберігання стану. Тому, будь-який стан, якого потребує додаток, має бути перенесений в базу даних або файлову систему, які реалізуються сервісами бекенд як послуга (Back end as a Service, BaaS) та оплачуються на постійній основі.

3) При постійному передбачуваному робочому навантаженні, коли функція постійно активна, оплата за кількість запитів стає економічно невигідною. Дешевше орендувати ресурси на постійній основі та заощадити витрати в довгостроковій перспективі, налаштувавши та оптимізувавши сервери, віртуальні машини чи кластер контейнерів відповідно до очікуваного робочого навантаження.

4) Існує досить жорстка прив'язка до постачальника хмарних послуг. Кожна платформа безсерверних обчислень має свої API і свої шаблони побудови, розгортання, моніторингу додатків. Зміна постачальника означає суттєву переробку додатку.

5) Передача управління частиною додатку третій стороні також породжує свої недоліки. Це може вилитися в деяку втрату контролю, наприклад, непередбачені апгрейди API. Також не всі користувачі готові довіряти конфіденційні дані третій стороні.

6) Час виконання примірника функції зазвичай обмежений (5–15 хв.). Це означає, що підхід FaaS не підходить для додатків, що вимагають тривалої фонової обробки даних, або обробки великого обсягу даних. Також є обмеження на обсяг пам'яті і довжину коду. Зазвичай провайдер може збільшити ліміти, але за додаткову плату.

Архітектура FaaS має свої особливості, недостатнє розуміння яких може ускладнити підтримку додатку. Може бути складно побачити вичерпну структуру сервісу, визначити, як функції інтегруються між собою, зрозуміти причини відмови. Проте сьогодні існують надійні методики проектування, за допомогою яких можна створити стійкі FaaS додатки, або інтегрувати функції в наявні архітектури. У кожного постачальника є детальний опис власних шаблонів проектування та інструменти для тестування, розгортання і моніторингу додатків, які постійно вдосконалюються.

Знання особливостей FaaS архітектури дозволяє використати її переваги і оминути можливі проблеми при створенні додатків.

Задачі, для яких застосування FaaS може дати потенційні переваги

Додатки з пульсуючим навантаженням. FaaS обчислення зекономлять витрати на обслуговування нерегулярних робочих навантажень, які носить сезонний характер, наприклад, в електронній комерції.

Додатки без фіксації стану. Безсерверна архітектура ідеально підходить для асинхронних додатків без фіксації стану, в яких дані клієнтів не зберігаються між сеансами. Наприклад, так працюють чат-боти, запити до сховищ даних, планувальники завдань та програми Інтернету речей.

Машинне навчання. Вже навчені моделі на основі API машинного навчання можна оформлювати у вигляді функцій і викликати в міру необхідності.

Мобільні та Web додатки. Сьогодні існує два найпоширеніші види веб – додатків: керовані сервером і керовані клієнтом (одно сторінковий додаток SPA). В обох сценаріях FaaS архітектура може надати необхідну бізнес-логіку на запит програмного забезпечення проміжного шару або REST API. FaaS додатки можна використовувати в якості серверної частини мобільних додатків. Мобільний пристрій ініціює події, після чого виконується безсерверний код для виконання запитів.

Обробка потокових даних. Безсерверні програми можуть обробляти величезні обсяги потокових даних, що надходять із сотень тисяч джерел, з низькою затримкою та високою пропускною здатністю. Прикладами можуть слугувати додатки, що обробляють потоки даних від різноманітних пристроїв Інтернету речей (IoT) для моніторингу продуктивності, запобігання виникнення можливих проблем, надання актуальних споживацьких пропозицій користувачам за даними їх геолокації та ін.

Аналітика даних. Ще однією важливою областю застосування FaaS є аналітика даних, що надає компаніям інформацію в реальному часі по різних аспектах власного бізнесу для його подальшої оптимізації. Наприклад, компанії можуть відстежувати та здійснювати підтримку рівнів сервісного обслуговування додатків та обладнання, слідкувати на настроями споживачів їх продукції, постійно аналізуючи дані клієнтської активності та публікації користувачів у соціальних мережах. Фінансові установи використовують потокові дані для відстеження змін на фондових біржах, для контролю ризиків, виявлення шахрайських операцій із банківськими картками.

Пакетна обробка і рідко виконувані операції. Існують задачі, для яких функції можна використовувати як тимчасові допоміжні служби, або як сервіси, що застосовуються для тестового середовища. За потреби вони легко масштабуються і не споживають ресурси, поки не використовуються. Програми для пакетної обробки періодично виконують однотипну обробку даних у великих обсягах, наприклад, резервне копіювання, фільтрацію та сортування, обробку зображень і відео (зміни їх розміру, кодування, вирізання фону та ін.). Прикладами інших допоміжних і рідко виконуваних операцій, що виконуються за таймером, можуть бути завдання моніторингу, аудиту безпеки віртуальних машин.

Поєднання сервісів. За допомогою FaaS функцій можна з'єднати між собою інші сервіси, наприклад, Git – репозиторій з внутрішніми програмами, чат-бот в Slack с Jira та з календарем.

Викладення основного матеріалу. Аналізуючи сучасний стан застосування хмарних технологій [7–12] можна виділити ряд наступних тенденцій.

1) Використання публічних хмарних сервісів, в тому числі FaaS, продовжує зростати у всіх організаціях світу. За даними звіту Flexera від 2024 року FaaS

використовували у виробництві 48 % опитуваних організацій, а планували та експериментували – 33 % [7].

2) Однак, наряду з такими перевагами публічних хмар, як масштабованість, економічність, висока доступність і широке охоплення, швидке розгортання та підвищення продуктивності праці розробників, досвід тривалого використання виявив такі їх недоліки, як обмеженість безпеки та контролю, складність з дотриманням нормативних вимог, високі затримки, обмеженість налаштувань, залежність від провайдера та неконтрольовані зростання витрат. В умовах глобальної економічної нестабільності останніх років проблеми витрат на хмарні послуги продовжують хвилювати організації (71 % опитуваних за звітом Flexera від 2025 року [8]).

3) Подолати обмеження публічних хмар дозволяють моделі приватної і гібридної хмари, вибір яких для побудови корпоративної хмарної інфраструктури спостерігається як стійка тенденція у світі [7, 9, 10, 11].

Інфраструктура приватної хмари може бути побудована на базі локальних ресурсів підприємства, або орендованих ресурсів провайдерів хмарних послуг, або їх комбінації. Приватна хмара налаштовується відповідно до вимог бізнес-завдань та безпеки конкретної організації. Вона забезпечує повний контроль над даними, додатками та інфраструктурою, можливість вибору специфічного обладнання, гнучкість налаштувань, що дозволяє забезпечити високу продуктивність додатків. Хоча початкове впровадження потребує значних фінансових витрат, в подальшому в добре спланованих під задачі організації і робочі навантаження та автоматизованих приватних хмарах досягається економія коштів порівняно з оплатою на вимогу в публічних хмарах.

А модель гібридної хмари, яка пропонує інтегровану інфраструктуру, що включає приватну, публічну хмару та локальну інфраструктуру в різних комбінаціях, дозволяє організаціям скористатися перевагами різних типів хмар. В єдиній інфраструктурі забезпечується можливість спільного розміщення та надання послуг, уніфікований моніторинг і управління. Гібридний підхід дозволяє зменшити ризики, швидко реагувати на мінливі потреби бізнесу та оптимізувати витрати, вибираючи найбільш ефективну платформу для кожного робочого навантаження. Організації отримують гнучкість та інновації публічних хмар, виконуючи в них некритичні робочі навантаження. Тоді як локальні ресурси використовуються для зберігання конфіденційної інформації відповідно до нормативних вимог та для робочих навантажень, які потрібують низької затримка виконання. Програми приватної хмари можуть звертатися за додатковими ресурсами публічної хмари під час тимчасових сплесків навантаження для економії коштів на оплату надлишкових виділених ресурсів під час їх простою.

4) З поширенням хмарних технологій у світі дедалі активніше розвивається контейнерний підхід до побудови програмного забезпечення, орієнтація на мікросервісну та подієво-орієнтовану архітектуру додатків. Використання контейнерів постійно зростає і за даними CNCF [13] знаходиться на рівні близько 84 %. Для оркестрації та розгортання контейнерів фактичним стандартом стала платформа Kubernetes (K8s). За даними CNCF [13] у 2024 році в середовищі розробки та виробництва K8s використовували понад 70 % респондентів, а серед розробників мікросервісів ця цифра сягала 95 %. Причому 54 % використовували K8s у гібридних та багато хмарних інфраструктурах.

З ростом популярності Kubernetes зростає і популярність платформ Managed Kubernetes as a Service (KaaS), які спрощують роботу з K8s, міграцію на хмарні

платформи та між провайдерами, а також дозволяють використовувати різну інфраструктуру для розгортання Kubernetes-кластерів (локальна інфраструктура, приватні та публічні хмари). Послуги KaaS пропонують як великі провайдери Google (GKE), Amazon (EKS), Azure (AKS), Alibaba (ASK), так і регіональні. Наприклад, в Україні послуги KaaS пропонують De Novo, OneCloudPlanet [14, 15].

Переваги застосування FaaS в приватній хмарній інфраструктурі

Відповідно до розглянутих загальних тенденцій розвитку хмарних технологій застосування FaaS у приватній хмарі може надати наступні переваги.

Підвищення ефективності роботи програмістів

Платформи FaaS пропонують просту модель програмування при розгортанні програм у Kubernetes, яка абстрагує його складні концепції та завдання. Вони можуть забезпечити авто масштабування, налаштування прогресивного розгортання (синьо – зелені та канаркові розгортання), обробку події з багатьох джерел та підтримку багатьох мов програмування. Все це підвищує ефективність, знижує витрати та прискорює цикли розробки, отже, випуск продукції компанії на ринок пришвидшується.

Ефективне управління і використання обчислювальних ресурсів

За рахунок оптимізації і ущільнення обчислень при перенесенні частини функціональності на FaaS можна без втрати загальної продуктивності досягти економії обчислювальних ресурсів, наприклад, Kubernetes-кластеру. Це стосується як частини операцій основних додатків, так і всіх сервісів, для яких тримати постійно виділені ресурси недоцільно. Наприклад, тимчасові допоміжні служби, рідко виконувані операції, завдання за таймером, послуги, що застосовуються для тестового середовища.

Як вже згадувалось раніше, підвищенню ефективності використання ресурсів і запобіганню їх простоїв сприяє виконання високонавантажених процедур за допомогою функцій. Це досягається за рахунок автоматичного масштабування функцій і розпаралелювання обробки даних. Також, на відміну від універсальної системи масштабування платформи Kubernetes, масштабування у FaaS фреймворках можна налаштувати більш гнучко і точно під будь-яке робоче навантаження.

В моделі гібридної хмари економічно обгрунтований розподіл робочого навантаження дозволяє оптимізувати витрати як на підтримку власної інфраструктури, так і на використання послуг публічних провайдерів. Якщо навантаження в приватній хмарі перевищує певний поріг, екземпляри функцій можуть запускатися в публічній хмарі, що особливо ефективно при тимчасових стрибках навантаження. Ще однин варіант гібридного підходу – запуск частини додатка, яка критична до безпеки даних, на власних ресурсах, а частини – у вигляді функцій в публічній хмарі.

Уникнення обмежень, властивих послугам FaaS публічних провайдерів

Загалом, в приватній хмарі на відміну від публічної можна забезпечити спеціальну безпеку, повний контроль і володіння даними, підвищити швидкість виконання додатка за рахунок тонкого налаштування приватної інфраструктури. А встановлені FaaS платформи дозволяють подолати обмеження публічних послуг на час виконання, обсяг пам'яті, довжину коду функцій та «холодний старт» за рахунок відповідних налаштувань.

Безсерверні послуги публічних провайдерів не означають повну відсутність операційних завдань з боку клієнта. Клієнт продовжує відповідати за моніторинг, розгортання, безпеку, мережеву взаємодію своїх додатків. А встановлені FaaS платформи надають більше можливостей для моніторингу, трасування та роботи з журналами, ніж їх публічні аналоги.

Стратегія застосування відкритого програмного забезпечення та широко підтримуваних платформ, як Kubernetes, дозволяє уникати прив'язки до одного хмарного провайдера, розгортати власні додатки в будь-якій хмарній інфраструктурі та гнучко адаптуватися до зміни потреб компанії.

FaaS фреймворки з відкритим вихідним кодом

Існує чимало відкритих FaaS платформ, кожна з яких пропонує певну цікаву функціональність [11]. Але певні з них, наприклад, Fission та Kubeless мало використовуються, а Nuclio не підтримує автомаштабування. Тому для більш детального порівняння були обрані три платформи, які на даний час широко застосовуються і активно підтримуються [12, 13, 14]. Всі вони можуть встановлюватися на Kubernetes-кластер.

Вибір FaaS фреймворку залежить від вимог конкретного застосування. Але є ряд загальних характеристик, на які варто звертати увагу при виборі. До них можна віднести наступне:

- Особливості функціональності (підтримувані мови програмування, типи тригерів, можливість теплого старту і масштабування до нуля, можливість налаштування або обмеження таких параметрів як час виконання, обсяг пам'яті і довжина коду функцій).
- Ресурсоємність (об'ємність і складність побудови платформи).
- Кількість додаткового коду для функцій.
- Складність освоєння і використання (якість документації, навчальних матеріалів, прикладів).
- Наявність засобів автоматизації встановлення.
- Підтримка засобів моніторингу та відладки FaaS додатку.
- Підтримка спільноти, випуск нових релізів.
- Популярність серед користувачів.

Всі розглянуті платформи мають підтримку спільноти, користуються довірою крупних компаній та підтримкою хмарних провайдерів, мають хорошу документацію та багато навчальних матеріалів. Всі активно впроваджують новий функціонал в напрямку підтримки засобів CI/CD, інтеграції з іншими сервісами, підтримки робочого оточення. Всі надають можливості ведення журналів логів та моніторингу, в тому числі за допомогою таких популярних інструментів як Prometheus та Grafana. Фреймворки підтримують широкий набір мов програмування та тригерів функцій, а OpenWhisk та Knative ще дозволяють створювати власні тригери. Через те, що OpenFaaS та Knative будують docker image із кодом функцій, вони не мають обмеження на розмір коду функцій.

Проте кожний фреймворк має свої особливості. *OpenFaaS* завоював найбільшу популярність у користувачів та створив велику активну спільноту завдяки своїй орієнтації на простоту та зручністю розгортання, використання та управління великою кількістю функцій. Він забезпечує велику екосистему інтеграцій з популярними інструментами робочого процесу. Підтримується система шаблонів для скорочення типового коду з великою кількістю готових шаблонів у сховищі. Масштабування OpenFaaS можна настроювати відповідно до конкретних шаблонів робочого навантаження. Підтримується масштабування до нуля, але з «холодним» стартом.

OpenFaaS може бути розгорнутий в будь-якому середовищі, що підтримує Docker. Наприклад, дистрибутив OpenFaaS Edge може бути розгорнутий без Kubernetes на одному хості або віртуальній машині, включаючи Arm/Raspberry Pi. Це корисно для IoT проектів та тих, що вимагають розгортання у кількох середовищах або гібридних хмарах.

Таблиця 1

Порівняння відкритих фреймворків (березень 2025 року)

Фреймворк	**OpenFaas**	**OpenWhisk**	**Knative: Serving + Eventing**
Розробник	OpenFaaS Ltd	Apache Software Foundation	Google
Популярність (кількість зірок на Github)	25,5 тис	6,6 тис	7,2 тис (5,7 + 1,5)
Кількість учасників на Github	160	203	508 (288 + 220)
Підтримка компаній	> 70 (VMware, HPE, DigitalOcean, Intel, Cognite та інші)	19 (Adobe, IBM, RedHat та інші)	30 (Google, Red Hat, IBM, VMware та інші)
Рік останнього релізу	2024 р.	2024 р.	2025 р.
Допоміжні технології	Alertmanager/ Prometheus, Nats	CouchDB, Kafka, Nginx, Docker	Istio, Zipkin, Statsd, Elasticsearch, Fluentd, та інші
Побудова Docker image	Так	Ні	Так
Автоматичне створення Docker image	Так	Ні	Ні
Наявні мовні шаблони	7 (Go, Java, Python, Node.js,C#, PHP, Dockerfile)	8 (Go,.Net, Java, JavaScript, PHP, Python, Ruby, Swift)	7 (Node.js, Python, Go, Rust, Quarkus, Spring Boot, TypeScript)
Установка за допомогою helm chart	Так	Так	Ні (є за допомою оператора)
Система збору логів	faas-netes, faasd, openfaas-loki. Також є можливість створити власний log provider	Logback через slf4j	Fluent Bit (додаткове налаштування)
Підтримка CI/CD	+	+	+
Моніторинг (Система збору метрик, візуалізація)	Prometheus, Grafana	Kamon (StatsD, Grafana), Kamon (Prometheus, Grafana–додаткове налаштування)	Prometheus, Grafana; OpenTelemetry Collector (додаткове налаштування)
Документація	+	+	+
Кількість додаткового коду для функцій	–	–	+

Закінчення таблиці 1

Фреймворк	**OpenFaas**	**OpenWhisk**	**Knative: Serving + Eventing**
Типи тригерів	CLI, Cron, HTTP, Async/NATS, Kafka, RabbitMQ, Postgresql, AWS SNS/SQS, JetStream queue worker	CLI, Cron, HTTP, Cloudant events, Kafka events, database events, підтримка власних тригерів	CLI, HTTP, Cron, Broker triggers, KLR (Knative lambda runtime для AWS lambda) CloudEvent + filtering Knative Eventing Broker та Trigger Custom Resources
Керування	утиліта faas-cli та веб-інтерфейс.	утиліта wsk CLI	утиліта knCLI
Теплий старт / масштабування до 0	+/+	+/-	-/+(є обмеження)
Налаштування для часу виконання, обсягів пам'яті і довжини коду функцій	Немає обмежень на довжину коду, інше +	Все +	Немає обмежень на довжину коду, інше +
Орієнтований хмарний провайдер	DigitalOcean	IBM-Cloud	Google-Cloud

Довгий час OpenFaaS був безкоштовним. Проте тепер безкоштовною є лише обмежена версія для особистого чи експериментального використання. Основну орієнтацію розробники спрямували на виробничу сферу, для якої пропонується підтримка підвищених стандартів безпеки, багатокористувацького середовища, виділених асинхронних черг, графічного процесору, Istio та інше.

OpenWhisk як і OpenFaaS є надійною, перевіреною часом платформою. Незважаючи на складнішу порівняно OpenFaaS внутрішню реалізацію, OpenWhisk є досить простим у використанні і встановленні на Kubernetes кластер. Також він значно компактніше, ніж Knative. Виклик функцій на платформі OpenWhisk не потребує зборку та завантаження на Docker Hub образів контейнерів функцій.

Фреймворк пропонує численні інтеграції з різними популярними зовнішніми сервісами та системами за допомогою пакетів: Kafka, бази даних, Push-повідомлення з мобільних додатків, повідомлення Slack та RSS-канали. Конвеєри розробки можуть скористатися перевагами інтеграції з GitHub, JIRA або легко підключитися до служб даних. За допомогою OpenWhisk можна об'єднувати функції у багаті композиції та підтримувати як синхронні так і асинхронні виклики для створення безсерверних масштабованих API з функцій. В OpenWhisk є можливість налаштування обмеження ресурсів, які використовуються функціями. Проте в ньому немає можливості масштабування до нуля, що не раціонально для рідко виконуваних функцій.

Як і OpenFaaS, OpenWhisk підримує різні варіанти розгортання контейнерів у локальних та хмарних інфраструктурах: Kubernetes і OpenShift та Docker Compose. В останній час платформа OpenWhisk підвищела свою продуктивність,

додала підтримку нових мов та ряд інших покращень. Також вона є повністю безкоштовною, що тепер надає їй додаткові переваги.

Knative був створений Google як програмне забезпечення проміжного шару поверх Kubernetes для спрощення створення, розгортання та керування безсерверними робочими навантаженнями. На основі Knative створено сервіс Google Cloud Run, і вони сумісні. Платформа Knative є безкоштовною. Вона складається з двох проектів: Serving, який відповідає за швидке розгортання безсерверних контейнерів, автоматичне масштабування, деталі роботи в мережі та відстежування ревізій, та Eventing, який відповідає за універсальну підписку стандартним способом CloudEvents, доставку і управління подіями.

Knative розширює Kubernetes додатковими API та компонентами і забезпечує уніфікований спосіб створення, розгортання та керування як безперервно працюючими контейнерними сервісами, так і безсерверними додатками. Він спрощує та стандартизує такі завдання як розгортання і оновлення додатків, прив'язка запущених служб до зростаючої екосистеми джерел подій, маршрутизація трафіку та автоматичне масштабування додатків (в тому числі до нуля). Knative підтримує різні шаблони розробки та інтеграцій, багатокористувацьку середу та ізоляцію в кластері Kubernetes.

Проте порівняно з OpenFaaS та OpenWhisk створення функцій в Knative потребує більшої кількості додаткового коду, а інфраструктурою контейнерів потрібно керувати самостійно. Встановлення платформі є не простим завданням, а також вимагає суттєвих апаратних ресурсів як у локальній інфраструктурі так і у хмарі провайдера, що веде до відповідних витрат на їх оплату. Оскільки Knative – це більше, ніж просто FaaS, його широкий функціонал може бути складним та збитковим для проектів, задачі яких можуть бути вирішені за допомогою OpenFaaS та OpenWhisk.

Платформа Knative особливо ефективна для гнучкої розробки складних контейнерних додатків з мікросервісною та подійно-керованою архітектурою, що можуть працювати в будь яких типах хмар. Knative знаходить все більше прихильників, особливо серед крупних компаній та хмарних провайдерів, які з огляду на складність його використання для кінцевих користувачів пропонують рішення Managed Knative [15].

Висновки. Сучасні компанії впевнено йдуть шляхом цифрової трансформації для отримання ринкових переваг. На цьому шляху вони все більше орієнтуються на контейнеризацію, мікросервісні і подієво-орієнтовані архітектури додатків, приватні та гібридні хмари у корпоративних інфраструктурі. Використання технології FaaS є добрим доповненням до цього вибору, що дозволить компаніям підвищити ефективність використання ресурсів і витрат, уникнення обмежень, властивих послугам FaaS в публічних хмарах, запобігти прив'язки до одного провайдера та підвищення продуктивність роботи команди програмістів. А знання переваг і обмежень кожної з розглянутих FaaS платформ дозволить обрати ту, що краще відповідає бізнес-завданням і цілям конкретної організації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. AWS. AWS Lambda. URL: https://aws.amazon.com/ru/lambda / (Дата звернення 28.03.2025)

2. Microsoft. Azure Functions. URL: https://azure.microsoft.com/en-us/products/functions (Дата звернення 02.04.2025)

3. Google Cloud. Cloud Run functions. https://cloud.google.com/functions?hl=ru (Дата звернення 28.03.2025)

4. Brendan Burns, Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. – "O'Reilly Media, Inc.", 2018. 166 p.

5. Castro, P., Ishakian, V., Muthusamy, V., Slominski, A. The rise of serverless computing. *Commun. ACM*. 2019. Vol. 62, № 12. P. 44–54. doi:10.1145/3368454

6. Angela Razzell. Best Practices for AWS Lambda Container Reuse. 2019. URL:https://medium.com/capital-one-tech/best-practices-for-aws-lambda-container-reuse-6ec45c74b67e (Дата звернення 28.03.2025)

7. Flexera. State of the Cloud Report.. URL: https://www.flexera.com/blog/finops/cloud-computing-trends-flexera-2024-state-of-the-cloud-report/ (Дата звернення 28.03.2025)

8. Flexera. 2025 IT Priorities Report. URL: https://info.flexera.com/ITV-REPORT-IT-Priorities?lead_source=Organic%20Search (Дата звернення 28.03.2025)

9. CNCF. CNCF 2023 Annual Survey. April 2024. URL: https://www.cncf.io/reports/cncf-annual-survey-2023/ (Дата звернення 28.03.2025)

10. Mordor Intelligence. Hybrid Cloud Market Size - Industry Report on Share, Growth Trends & Forecasts Analysis (2024-2029). URL: https://www.mordorintelligence.com/industry-reports/hybrid-cloud-market (Дата звернення 28.03.2025)

11 В. Ш. Гіоргізова-Гай, Б. А. Кірюша. Хмарні послуги в корпоративній інфраструктурі. *Вчені записки Таврійського національного університету ім. В. І. Вернадського. Серія: Технічні науки.* 2024. № 2. Т. 35(74). С. 32–39.

12. Data Dog. The State of serverless. 2023. URL: https://www.datadoghq.com/state-of-serverless/ (Дата звернення 28.03.2025)

13. CNCF. Kubernetes in 2025: are you ready for these top 5 trends and predictions? URL: https://www.cncf.io/blog/2025/01/22/kubernetes-in-2025-are-you-ready-for-these-top-5-trends-and-predictions/ (Дата звернення 28.03.2025)

14. De Novo. Платформа Kubernetes за моделлю «як сервіс» в колективній хмарі. URL: https://denovo.ua/kaas (Дата звернення 02.04.2025)

15. OneCloudPlanet. Managed Kubernetes. URL: https://onecloudplanet.com/managed-kubernetes (accessed 03.04.2025)

16. LinuxLinks. 8 Best Free and Open Source Functions-as-a-Service. URL: https://www.linuxlinks.com/best-free-open-source-faas/ (Дата звернення 28.03.2025)

17. Офіційний сайт OpenFaas. URL: https://www.openfaas.com/ (Дата звернення 28.03.2025)

18. Офіційний сайт OpenWhisk. URL: https://openwhisk.apache.org/ (Дата звернення 28.03.2025)

19. Офіційний сайт Knative. URL: https://knative.dev/ (Дата звернення 28.03.2025)

REFERENCES:

1. AWS. AWS Lambda. URL: https://aws.amazon.com/ru/lambda/ (accessed 28.03.2025)

2. Microsoft. Azure Functions. URL: https://azure.microsoft.com/en-us/products/functions (accessed 02.04.2025)

3. Google Cloud. Cloud Run functions. https://cloud.google.com/functions?hl=ru (accessed 28.03.2025)

4. Brendan Burns, Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. – "O'Reilly Media, Inc.", 2018. 166 p.

5. Castro, P., Ishakian, V., Muthusamy, V., Slominski, A. The rise of serverless computing. *Commun. ACM* 62, 12 (2019), 44–54. doi:10.1145/3368454

6. Angela Razzell. Best Practices for AWS Lambda Container Reuse. 2019. URL:https://medium.com/capital-one-tech/best-practices-for-aws-lambda-container-reuse-6ec45c74b67e (accessed 28.03.2025)

7. Flexera. State of the Cloud Report. URL: https://www.flexera.com/blog/finops/cloud-computing-trends-flexera-2024-state-of-the-cloud-report/ (accessed 28.03.2025).

8. Flexera. 2025 IT Priorities Report. URL: https://info.flexera.com/ITV-REPORT-IT-Priorities?lead_source=Organic%20Search (accessed 28.03.2025)

9. CNCF. CNCF 2023 Annual Survey. April 2024. URL: https://www.cncf.io/reports/cncf-annual-survey-2023/ (accessed 28.03.2025)

10. Mordor Intelligence. Hybrid Cloud Market Size – Industry Report on Share, Growth Trends & Forecasts Analysis (2024–2029). URL: https://www.mordorintelligence.com/industry-reports/hybrid-cloud-market (accessed 28.03.2025)

11. Hiorhizova-Hai, V. S. Kyriusha, B. A. (2024). Khmarni posluhy v korporatyvniy ynfrastrukturi [Cloud services in corporate infrastructure]. *Vcheni zapysky Tavriys'koho natsional'noho universytetu im. V. I. Vernads'koho. Seriya: Tekhnichni nauky – Scientific notes of the V. I. Vernadsky Tavrichesky National University. Series: Technical Sciences*, 2(35), 32-39 [in Ukrainian].

12. Data Dog. The State of serverless. 2023. URL: https://www.datadoghq.com/state-of-serverless/ (accessed 28.03.2025).

13. CNCF. Kubernetes in 2025: are you ready for these top 5 trends and predictions? URL: https://www.cncf.io/blog/2025/01/22/kubernetes-in-2025-are-you-ready-for-these-top-5-trends-and-predictions/ (accessed 28.03.2025).

14. De Novo. Platforma kubernetes za modellyu “yak servis” v kolektyvniy khmari [Kubernetes platform as a service in a collective cloud]. URL: https://denovo.ua/kaas (accessed 02.04.2025).

15. OneCloudPlanet. Managed Kubernetes. URL: https://onecloudplanet.com/managed-kubernetes (accessed 03.04.2025)

16. LinuxLinks. 8 Best Free and Open Source Functions-as-a-Service. URL: https://www.linuxlinks.com/best-free-open-source-faas/ (accessed 28.03.2025).

17. OpenFaas. URL: https://www.openfaas.com/ (accessed 28.03.2025).

18. OpenWhisk. URL: https://openwhisk.apache.org/ (accessed 28.03.2025).

19. Knative. URL: https://knative.dev/ (accessed 28.03.2025).