УДК 004. 056.55:004.312.2 **doi: 10.26906/SUNZ.2025.2.175-179**

Д. А. Помелуйко[1], Н. С. Єрьоміна[1], С. В. Петров[2], Н. В. Іщенко[2], Волощук О. Б.[1]

[1] Харківський національний університет радіоелектроніки, Харків, Україна
[2] Харківський національний університет імені В.Н. Каразіна, Харків, Україна

МЕТОДИ МАСШТАБУВАННЯ KUBERNETES КЛАСТЕРУ В ХМАРНОМУ СЕРЕДОВИЩІ

Анотація. У статті розглянуто питання масштабування кластерів Kubernetes у хмарному середовищі, що є одним із ключових чинників забезпечення високої доступності мікросервісних додатків. Проаналізовано традиційні засоби та сучасні підходи, включно з механізмами вертикального та горизонтального масштабування. Розглянуто також такі методи, як передбачувальне масштабування, масштабування на основі подій та застосування спеціальних метрик. Проведено аналіз переваг і недоліків розглянутих підходів. Показано, як налаштування й оптимізація параметрів масштабування впливають на витрати на хмарні ресурси й продуктивність додатків. **Мета статті:** аналіз сучасних інструментів та підходів до масштабування Kubernetes-кластерів у хмарному середовищі AWS, виявлення переваг та недоліків кожного із наявних рішень. **Висновки.** Масштабування на основі подій є особливо корисним для навантажень з нерегулярною або імпульсною природою. Перспективними напрямами являється розробка динамічних механізмів зміни ресурсів контейнера під час виконання, одночасне використання горизонтального і вертикального масштабування, застосування гібридних підходів для мікросервісів, а також розширення досліджень на рівні інфраструктури. Вибір стратегії масштабування має базуватись на характеристиках робочих навантажень, бюджетних обмеженнях і загальних операційних цілях організації. Тонке налаштування конфігурацій масштабування та постійний моніторинг поведінки застосунків є критично важливими для досягнення ефективного використання ресурсів, підвищення продуктивності та зниження витрат у cloud-native середовищі.

Ключові слова: Kubernetes, масштабування кластеру, хмарне середовище, HPA, Horizontal Pod Autoscaler, VPA, Cluster Autoscaler, Karpenter, KEDA, Kubernetes Event Driven Scaling, Prometheus, EKS, AWS.

Вступ

Постановка проблеми. Зі зростанням попиту на динамічні та масштабовані застосунки традиційні методи розгортання та масштабування дедалі більше втрачають актуальність. Останніми роками мікросервісна архітектура стала провідним підходом до створення сучасних, гнучких та високодоступних застосунків. Проте керування мікросервісами, контейнерами та їхньою взаємодією є складним завданням, особливо у великих і складних системах.

Розвиток хмарних обчислень зумовив суттєві зміни у проєктуванні й експлуатації програмних систем. Поширення архітектури мікросервісів призвело до потреби автоматизації розгортання та обслуговування великої кількості взаємопов'язаних сервісів. Контейнерна оркестрація за допомогою Kubernetes стала стандартом для таких задач. Kubernetes сьогодні є одним із найпопулярніших інструментів для оркестрації контейнерів, який широко застосовується постачальниками хмарних сервісів. Він забезпечує керування життєвим циклом контейнеризованих застосунків. Для підтримання показників продуктивності застосунків і дотримання угоди про рівень обслуговування (SLA) упродовж часу, за умов змінних робочих навантажень або доступності ресурсів, ефективне масштабування відіграє ключову роль.

У Kubernetes реалізовано декілька вбудованих механізмів масштабування. Проте їхніх можливостей може не вистачати для належного пристосування до високодинамічних навантажень.

Метою роботи є аналіз сучасних інструментів та підходів до масштабування Kubernetes-кластерів у хмарному середовищі AWS, виявлення переваг та недоліків кожного із наявних рішень.

Основний матеріал

Із дедалі ширшим поширенням хмарно-орієнтованих архітектур і мікросервісного підходу проєктування, дедалі гостріше постає потреба ефективного масштабування застосунків, яка породжує унікальні виклики, що потрібно враховувати для забезпечення ефективності, надійності і помірної вартості рішень.

Перехід від монолітної до розподіленої архітектури збільшує складність керування застосунком. У розподілених системах багато сервісів та компонентів взаємодіють у межах різних вузлів і середовищ, що створює додаткові складнощі. Нижче наведено ключові проблеми, з якими зазвичай стикаються під час масштабування застосунків:

- Комунікація та координація. Різні сервіси мають безперешкодно взаємодіяти один із одним, що стає складнішим зі збільшенням кількості сервісів і вузлів.
- Керування станом. Підтримка цілісності стану системи в розподіленому середовищі є важливим аспектом, особливо коли йдеться про події, що відбуваються одночасно, та потребують узгодження оновлень між різними сервісами.
- Спостереження та налагодження. Пошук і виправлення помилок у розподіленій системі складніші, ніж у монолітній.
- Ефективний розподіл обчислювальних ресурсів. Один із важливих аспектів масштабування. Зі зростанням навантаження, збільшується й споживання ресурсів, що може призвести як до нестачі ресурсів, так і до їх перевитрати.
- Непередбачуваність трафіку. Багато застосунків стикаються із проблемою непередбачуваності трафіку, викликаного сезонними процесами та яви-

щами, рекламними кампаніями чи раптовою популярністю.

- Балансування навантаження. Зі збільшенням кількості копій застосунку Kubernetes використовує сервіси й механізми балансування навантаження для розподілу трафіку між подами. Однак, якщо трафік дуже нерівномірний, балансування може бути неідеальним, спричиняючи диспропорцію використання ресурсів.

Хоча Kubernetes здатен суттєво спростити ці аспекти завдяки своїм вбудованим засобам масштабування, серед яких головними є:

- Horizontal Pod Autoscaler (HPA) для масштабування кількості подів;
- Vertical Pod Autoscaler (VPA) для зміни розміру ресурсів подів;
- Cluster Autoscaler (CA) для зміни кількості вузлів в кластері.

Ці інструменти також мають й свої недоліки. Так, за даними одного дослідження, понад 50 % організацій використовують HPA, тоді як менш ніж 1 % – VPA що обумовлено складністю конфігурації та можливими конфліктами при спільному використанні з HPA [1]. CA надійно масштабує вузли, але працює з попередньо визначеними групами вузлів і застосовує менш ефективний підхід [2], що може призводити до надлишку або недостачі обчислювальних ресурсів в кластері.

Масштабування також часто супроводжується затримками, необхідними для запуску нових вузлів чи розгортання подів. Ця проблема відома як проблема холодного запуску (cold start problem). Вона полягає в наступних складнощах:

- запуск вузла в AWS (EC2 instance) триває від кількох десятків секунд до декількох хвилин;
- завантаження контейнерних образів може займати чимало часу, в залежності від розміру образу та пропускної здатності мережі;
- ініціалізація застосунку всередині контейнера (з'єднання з БД, кешування, зовнішні залежності).

Для зменшення впливу цих проблем використовують різні механізми, зокрема метод завчасного прогрівання вузлів (warm pools), зменшення образу контейнерів (використання alpine базових образів), а також масштабування на основі подій (KEDA), що дозволяє швидше реагувати на зміни в системі.

Під горизонтальним масштабуванням Kubernetes кластеру розуміють збільшення (чи зменшення) кількості подів (Pods) або робочих вузлів (Nodes). Це дає змогу оперативно реагувати на коливання навантаження: чим більше копій, тим вище загальна пропускна здатність сервісу. Стандартний Horizontal Pod Autoscaler (HPA) передбачає ручне налаштування певних порогових значень, як-от завантаження CPU, мінімальна та максимальна кількість подів. Деякі дослідження пропонують поліпшити цей механізм, залучаючи додаткові метрики (наприклад, аналіз трафіку [3] або час відгуку [4]). Крім того, є дослідження, що додають у HPA машинне навчання чи евристичний аналіз [5, 6].

Вертикальне масштабування передбачає зміну обсягу виділених контейнерам ресурсів. У Kubernetes вертикальне масштабування реалізується за допомогою Vertical Pod Autoscaler (VPA). Воно дає змогу подам ефективніше використовувати доступні ресурси, однак потребує перезапуску подів при зміні конфігурації. Через таке обмеження вертикальне масштабування користується меншим менше уваги.

Гібридне масштабування поєднує горизонтальне та вертикальне масштабування. Наявні рішення [7] зазвичай виконуються поступово: спочатку визначають оптимальний обсяг ресурсів для контейнерів за допомогою вертикального масштабування, а потім динамічно змінюють кількість контейнерів методом горизонтального масштабування.

Засоби масштабування Kubernetes кластеру наступні:

- Horizontal Pod Autoscaler;
- Vertical Pod Autoscaler (VPA);
- Cluster Autoscaler (CA);
- Karpenter;
- Масштабування на основі подій;
- Масштабування на основі спеціальних метрик;
- Методи передбачувального масштабування.

Horizontal Pod Autoscaler дозволяє автоматично регулювати кількість реплік подів у Deployment або ReplicaSet на основі показників використання процесора, пам'яті або інших зазначених метрик. Контролер періодично проводить порівняння фактичних значень показників із цільовими значеннями і розраховує бажану кількість копій для підтримки заданого рівня навантаження [9]. Deployment або ReplicaSet в свою чергу відповідає за підтримання бажаної кількості копій подів. Такий процес забезпечує оптимальне використання ресурсів та відповідність наявних ресурсів потребам застосунків за різних типів навантаження. Налаштування HPA включає задання бажаного рівня завантаження процесора чи інших показників, а також визначення мінімальної і максимальної кількості копій подів. У табл. 1 наведено порівняння основних типів метрик HPA, їх переваги, недоліки та типові сценарії використання.

Vertical Pod Autoscaler (VPA) забезпечує автоматичне коригування запитів і лімітів на ресурси (CPU та пам'ять) для контейнерів у подах. На основі аналізу історичних даних про споживання ресурсів та поточного використання VPA рекомендує або автоматично призначає найоптимальніші значення ресурсів для кожного поду [10]. Основна мета використання VPA полягає в підвищенні ефективності використання ресурсів кластера, а також у покращенні продуктивності додатків завдяки динамічному регулюванню ресурсів. VPA особливо ефективний у сценаріях, коли вимоги до ресурсів є нестабільними або коли початкове налаштування ресурсів виконується вручну. Використання VPA дозволяє уникнути ситуацій недостатнього або надмірного виділення ресурсів, що призводить до зниження витрат на обслуговування кластерів. Цей підхід до масштабування є корисним для додатків із періодичними або поступовими змінами навантаження, які не потребують великої кількості ресурсів одночасно.

Таблиця 1 – **Основні типи метрик для масштабування за допомогою HPA**

Тип метрики	Опис	Сценарій використання	Переваги	Недоліки
CPU utilization	Відсоток завантаження CPU	Універсальні контейнери	Простота налаштування, вбудована метрика	Не завжди є показником завантаженості контейнеру
Memory utilization	Відсоток використовуваної оперативної пам'яті (ОП)	Контейнери з інтенсивним використанням ОП	Попередження помилок OutOf Memory, вбудована метрика	Можливе надлишкове виділення ресурсів
Спеціальні метрики	Специфічні показники контейнеру	Спеціалізовані робочі навантаження	Висока точність масштабування	Складність в налаштуванні
Зовнішні метрики	Показники зі зовнішніх джерел	Хмарні сервіси, черги	Масштабування за зовнішніми умовами	Залежність від зовнішніх систем

VPA може використовуватися одночасно з HPA, який відповідає за зміну кількості копій подів, тоді як VPA регулює ресурси всередині цих подів. Така інтеграція дозволяє більш точно контролювати масштабування контейнерів, ефективно використовуючи ресурси та оптимізуючи продуктивність застосунків залежно від їхніх потреб [8].

Cluster Autoscaler (CA) є важливим компонентом Kubernetes, що автоматично змінює розмір кластера шляхом додавання або видалення вузлів залежно від наявності ресурсів для розміщення подів. Він складається з головного циклу, який періодично перевіряє стан кластера та шукає поди, які неможливо розмістити через нестачу ресурсів. Потім CA взаємодіє з API відповідного хмарного провайдера для управління групами вузлів та динамічного створення чи видалення вузлів кластеру. Кожен хмарний провайдер має параметри, які впливають на роботу Cluster Autoscaler. Наприклад, типи та вартість віртуальних серверів, а також можливості API для управління групами вузлів відрізняються у провайдерів таких, як AWS, GCP чи Azure [11]. Тому налаштування Cluster Autoscaler може потребувати додаткових коригувань з метою досягнення максимальної ефективності та економічності кластера залежно від конкретного хмарного середовища.

Karpenter контролер автоматично оптимізує тип, розмір та кількість вузлів, необхідних для виконання робочих навантажень. На відміну від CA, що працює із заздалегідь визначеними статичними групами вузлів, Karpenter забезпечує вищий рівень гнучкості завдяки динамічному підбору найоптимальніших типів віртуальних серверів і оперативному масштабуванню у режимі реального часу. Контролер реагує на зміни у кластері протягом кількох секунд, тоді як реакція CA зазвичай є повільнішою. Це особливо важливо для робочих навантажень, які потребують оперативного масштабування у відповідь на різкі зміни навантаження. Karpenter створений з метою максимально ефективного використання доступних ресурсів завдяки вибору економічно вигідних доступних типів віртуальних серверів [12]. Він забезпечує розгортання лише тих ресурсів, які необхідні для запуску робочих навантажень. Це робить Karpenter особливо важливим для організацій, які прагнуть зменшити витрати на хмарну інфраструктуру. При використанні Karpenter немає необхідності керувати окремими групами вузлів вручну, адже він автоматично налаштовує тип та розмір серверу відповідно до поточних потреб навантаження. Це значно спрощує процес налаштування, роблячи його більш зрозумілим і доступним для розробників.

Масштабування на основі подій можливе за допомогою Kubernetes Event-driven Autoscaling (KEDA) – контролеру, що дозволяє масштабувати контейнери відповідно до подій, таких як повідомлення в чергах, запис у базах даних або власні спеціальні метрики [13]. Цей інструмент ефективний для сценаріїв, де навантаження є нерегулярним, наприклад, у разі безсерверних функцій чи задач обробки даних, які залежать від подій. Масштабування на основі подій ефективно використовується в реальних умовах, таких як мікросервісна архітектура, IoT та пакетна обробка даних. Наприклад, система обробки платежів може масштабуватись залежно від довжини черги транзакцій, а сервіс обробки зображень — відповідно до кількості завантажень зображень. Такі додатки мають значні переваги від цього типу масштабування через їхню непередбачувану природу навантажень.

Масштабування на основі спеціальних метрик можливе за допомогою Prometheus адаптера – інструменту, що дозволяє використовувати метрики Prometheus для масштабування Kubernetes. Він транслює запити Prometheus у формат, зрозумілий для Custom Metrics API, завдяки чому HPA може використовувати спеціальні метрики для масштабування [14]. Це дає змогу адаптувати політики масштабування до специфічних особливостей додатків, таких як швидкість запитів, довжина черг або інші бізнес-показники. Custom Metrics API розширює можливості масштабування Kubernetes, дозволяючи використовувати не лише показники CPU чи пам'яті, але й довільні метрики, специфічні для конкретних застосунків.

Розширені стратегії використання спеціальних метрик для масштабування передбачають застосування декількох метрик одночасно, використання швидкості зміни показників або відсоткових метрик для більш точного прогнозування необхідних ресурсів. Подібні стратегії потребують детального аналізу поведінки додатків та точного налаштування, але дозволяють значно підвищити ефективність процесу масштабування.

Методи передбачувального масштабування. Передбачувальне масштабування, яке ґрунтується на методах машинного навчання, використовує накопичені дані про споживання ресурсів для прогнозування майбутніх потреб мікросервісів. Для цього застосовуються регресійний аналіз, прогнозування

часових рядів та нейронні мережі [4]. Такі моделі дозволяють аналізувати історичні дані та зовнішні фактори, що впливають на навантаження, з метою прогнозування майбутніх ресурсних потреб і, таким чином, зменшення часу реакції на зміну навантаження.

У табл. 2 наведено порівняльну характеристику методів передбачувального масштабування.

Таблиця 2 – **Методи передбачувального масштабування**

Модель	Техніка	Переваги	Недоліки	Складність
Лінійна регресія	Прогноз часових рядів	Простота, прозорість	Тільки лінійні залежності	Низька
ARIMA	Аналіз часових рядів	Враховує тренди та сезонність	Вимагає стаціонарності даних	Середня
Нейронні мережі	Глибоке навчання	Виявляє складні закономірності	Необхідність великих наборів даних, низька інтерпретованість	Висока
Prophet	Адитивна модель	Обробляє сезонність та пропуски даних	Можливе надмірне спрощення складних залежностей	Середня

Моделі аналізу часових рядів, такі як ARIMA, експоненційне згладжування та Prophet, дозволяють виявляти тренди, сезонність та циклічні зміни у споживанні ресурсів. Ці методи ефективно використовуються для виявлення залежностей у поведінці навантажень, що дає змогу передбачати короткострокові та довгострокові потреби у ресурсах для Kubernete -кластерів.

Незважаючи на значні переваги, існують певні труднощі в реалізації передбачувального масштабування. Зокрема, складно забезпечити точність моделей при зміні характеру навантажень, а також балансувати між точністю прогнозів та часом їх обчислення.

Іншою проблемою є можливість появи аномалій, які можуть вплинути на точність прогнозів. Вирішення цих проблем потребує глибокого аналізу та постійного уточнення використовуваних моделей і алгоритмів прогнозування.

Висновки

У статті детально проаналізовано стандартні та передові методи масштабування Kubernetes, зокрема горизонтальне та вертикальне масштабування подів, вузлів кластера, використання спеціальних метрик, а також нові підходи до масштабування. Проведене дослідження показало, що хоча традиційні методи вертикального та горизонтального масштабування залишаються основою, нові стратегії, такі як передбачувальне масштабування, масштабування на основі подій та використання спеціальних метрик, забезпечують додаткову гнучкість і чутливість до змін у навантаженні.

Кожен із розглянутих підходів має свої переваги й обмеження з точки зору складності впровадження, ефективності використання ресурсів та вимог до налаштування. Горизонтальне масштабування добре підходить для безстанових додатків, тоді як вертикальне є ефективнішим для ресурсомістких сервісів. Впровадження методів машинного навчання та аналізу часових рядів істотно підвищує точність передбачень в керуванні ресурсами, однак вимагає надійних моделей прогнозування та якісних даних. Масштабування на основі подій, наприклад за допомогою KEDA, є особливо корисним для навантажень з нерегулярною або імпульсною природою.

Перспективними напрямами є розробка динамічних механізмів зміни ресурсів контейнера під час виконання, одночасне використання горизонтального і вертикального масштабування, застосування гібридних підходів для мікросервісів, а також розширення досліджень на рівні інфраструктури.

Зрештою, вибір стратегії масштабування має базуватись на характеристиках робочих навантажень, бюджетних обмеженнях і загальних операційних цілях організації.

Тонке налаштування конфігурацій масштабування та постійний моніторинг поведінки застосунків є критично важливими для досягнення ефективного використання ресурсів, підвищення продуктивності та зниження витрат у cloud-native середовищі.

СПИСОК ЛІТЕРАТУРИ

1. Kubernetes Autoscaling and Best Practices for Implementations. StormForge. URL: https://stormforge.io/kubernetes-autoscaling (дата звернення: 09.04.2025).
2. Phuc L.H., Phan L.-A., Kim T. Traffic-Aware Horizontal Pod Autoscaler in Kubernetes-Based Edge Computing Infrastructure.
3. Choi B., Park J., Lee C., Han D. pHPA: A Proactive Autoscaling Framework for Microservice Chain. APNet 2021: Proceedings of the 5th Asia-Pacific Workshop on Networking. 2021.
4. Toka L., Dobreff G., Fodor B., Sonkoly B. Machine Learning-Based Scaling Management for Kubernetes Edge Clusters. IEEE Transactions on Network and Service Management. 2021.
5. Rudrabhatla C.K. A Quantitative Approach for Estimating the Scaling Thresholds and Step Policies in a Distributed Microservice Architecture. IEEE Access. 2020
6. Balla D., Simon C., Maliosz M. Adaptive Scaling of Kubernetes Pods. NOMS 2020: IEEE/IFIP Network Operations and Management Symposium. Apr. 2020. P. 1–5.
7. Khaleq A., Ra I. Intelligent Autoscaling of Microservices in the Cloud for Real-Time Applications. IEEE Access. 2021.
8. Baresi L., Hu D., Quattrocchi G., Terracciano L. KOSMOS: Vertical and Horizontal Resource Autoscaling for Kubernetes. ICSOC 2021: International Conference on Service-Oriented Computing. 2021

9. Nguyen T.T., Yeom Y.J., Kim T., Park D.H., Kim S. Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration. Sensors. 2020.
10. Rossi F. Auto-Scaling Policies to Adapt the Application Deployment in Kubernetes. ZEUS 2020: Proceedings of the Workshop on Services and Applications over Decentralized Systems.
11. Cluster Autoscaler on AWS. Kubernetes Autoscaler Documentation. GitHub. URL: https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/cloudprovider/aws/README.md (дата звернення: 09.04.2025).
12. Tarn E., Saha R. Harness the Power of Karpenter to Scale, Optimize & Upgrade Kubernetes. AWS re:Invent 2023. Presentation CON331. URL: https://d1.awsstatic.com/events/Summits/reinvent2023/CON331_Harness-the-power-of-Karpenter-to-scale-optimize-and-upgrade-Kubernetes.pdf (дата звернення: 09.04.2025).
13. KEDA Concepts. KEDA Documentation. URL: https://keda.sh/docs/2.17/concepts/ (дата звернення: 09.04.2025).
14. Venkataraman V. Using Prometheus Adapter to Autoscale Applications Running on Amazon EKS. AWS Cloud Operations Blog. 28.09.2021. URL: https://aws.amazon.com/blogs/mt/automated-scaling-of-applications-running-on-eks-using-custom-metric-collected-by-amazon-prometheus-using-prometheus-adapter/ (дата звернення: 09.04.2025).

Received (Надійшла) 23.02.2025
Accepted for publication (Прийнята до друку) 21.05.2025

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Помелуйко Денис Андрійович – студент кафедри кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Denys Pomeluiko – student, Department of Electronic Computing, Kharkiv National University of Radio Electronics; Kharkiv, Ukraine;
e-mail**:** denys.pomeluiko@nure.ua; ORCID Author ID: https://orcid.org/0009-0005-6994-280X.

Єрьоміна Наталія Сергіївна – кандидат технічних наук, старший викладач кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Nataliia Yeromina - PhD, Senior Lecturer of the Department of Electronic Computing, Kharkiv National University of Radio Electronics; Kharkiv, Ukraine;
e-mail: nataliia.yeromina@nure.ua ;ORCID Author ID: https://orcid.org/0000-0002-0463-2342;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57194558513.

Петров Сергій Валерійович – кандидат технічних наук, доцент, доцент кафедри електротехніки та електроенергетики, Харківський національний університет імені В.Н. Каразіна, Харків, Україна;
Serhii Petrov – PhD, Docent, Associate Professor of the Department of Electrical Engineering and Power Engineering, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine;
e-mail: psv_topol@ukr.net; ORCID Author ID: http://orcid.org/0000-0001-8933-9649 ;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57194556925 .

Іщенко Наталія Володимирівна – асистент кафедри автоматизації, метрології та енергоефективних технологій Харківський національний університет імені В.Н. Каразіна, Харків, Україна;
Nataliia Ishchenko – assistant of the Department of Automation, Metrology and Energy Efficient Technologies, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine
e-mail: natalidocenko158@gmail.com; ORCID Author ID: https://orcid.org/0009-0004-3899-3965.

Волощук Олена Борисівна – кандидат технічних наук, доцент, доцент кафедри ЕОМ, Харківського національного університету радіоелектроніки, Харків, Україна;
Olena Voloshchuk – PhD, Ass.prof, Ass.prof of the Department of electronic computers, Kharkiv National University of radio electronics, Kharkiv, Ukraine;
e-mail: olena.voloshchuk@nure.ua; ORCID Author ID: https://orcid.org/0000-0002-5912-4126;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57207762084.

Kubernetes cluster scaling methods in cloud environments

Denys Pomeluiko, Nataliia Yeromina, Serhii Petrov, Nataliia Ishchenko, Olena Voloshchuk

Abstract. The article explores Kubernetes cluster scaling in cloud environments, a critical factor in ensuring high availability for microservice applications. It analyzes traditional techniques and modern approaches, including vertical and horizontal scaling mechanisms. Methods such as predictive autoscaling, event-driven scaling, and the use of custom metrics are also considered. The advantages and disadvantages of these approaches are discussed. Furthermore, the paper demonstrates how the configuration and optimization of scaling parameters impact cloud resource costs and application performance. **Purpose of the article** is to analyse modern tools and approaches to scaling Kubernetes clusters in the AWS cloud environment, to identify the advantages and disadvantages of each of the available solutions. **Conclusions**. Event-based scaling is especially useful for loads with irregular or impulsive nature. Promising directions include the development of dynamic mechanisms for changing container resources at runtime, the simultaneous use of horizontal and vertical scaling, the use of hybrid approaches for microservices, and the expansion of research at the infrastructural level. The choice of scaling strategy should be based on the characteristics of workloads, budgetary constraints, and the organisation's overall operational goals. Fine-tuning scaling configurations and continuously monitoring application behaviour are critical to achieving efficient resource utilisation, increased productivity and reduced costs in a cloud-native environment.

Keywords: Kubernetes, cluster scaling, cloud environment, HPA, Horizontal Pod Autoscaler, VPA, Cluster Autoscaler, Karpenter, KEDA, Kubernetes Event Driven Scaling, Prometheus, EKS, AWS.