УДК 004.4

ПІДХІД ДО АВТОМАСШТАБУВАННЯ КЛАСТЕРІВ KUBERNETES НА ОСНОВІ ПЕРСОНАЛЬНИХ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ КОРИСТУВАЧІВ

Ковальов А.В., Алєксєєв М.О.
Навчально-науковий інститут телекомунікаційних систем
НТУУ «КПІ» ім. Ігоря Сікорського, Україна
E-mail: andreyjester32@gmail.com, alexeyev@its.kpi.ua

APPROACH TO AUTOSCALING KUBERNETES CLUSTERS BASED ON PERSONAL COMPUTATIONAL RESOURCES OF USERS

The approach to Kubernetes cluster autoscaling based on users' personal computing resources is relevant, as it allows optimizing the utilization of existing devices and reducing server hardware costs amid growing computational power demands. Currently, existing solutions are primarily designed for cloud environments and static clusters, including Kubernetes Autoscaler, Cluster API, metal3-io, and other node management automation tools. However, these solutions have certain drawbacks, such as a high dependence on cloud providers, limited flexibility in integrating heterogeneous resources, and complexity in managing short-lived computing agents.

This work focuses on developing an approach that enables automatic scaling of Kubernetes clusters by leveraging users' personal devices, utilizing Cluster API and metal3-io capabilities for node registration and lifecycle management. The efficiency of the proposed solution has been experimentally validated through the deployment of a prototype, demonstrating successful scaling from a single worker node to multiple nodes without significant failures. This confirms the potential for real-world implementation of the proposed approach.

Сучасні телекомунікаційні сервіси та застосунки часто працюють у середовищі контейнерів Kubernetes, що став де-факто стандартом для оркестрації контейнерів. Важливою перевагою Kubernetes є здатність масштабувати застосунки та інфраструктуру під змінне навантаження (еластичність). Горизонтальне масштабування виконується як на рівні контейнерів (подів), так і на рівні вузлів кластера. Для автоматизації масштабування кластерів у Kubernetes використовується компонент Cluster Autoscaler – інструмент, що автоматично змінює кількість вузлів у кластері залежно від поточних потреб у ресурсах. Зокрема, коли ресурсів не вистачає для запуску нових контейнерів (подів), Cluster Autoscaler знаходить відповідну групу вузлів і додає новий вузол; якщо ж деякі вузли тривалий час недовантажені, він може видалити зайві вузли, оптимізуючи витрати ресурсів. Однак типова реалізація автомасштабування кластерів розрахована передусім на інтеграцію з хмарними провайдерами. Cluster Autoscaler взаємодіє з API хмарного середовища для створення або видалення віртуальних машин, які виконують роль вузлів. Подібна схема добре працює в публічних або приватних хмарах, де доступні ресурси можуть бути швидко виділені через API. Натомість у середовищі без хмарного провайдера (наприклад, на власному апаратному

забезпеченні, on-premises або на периферійних пристроях) постає проблема: як автоматично додавати вузли кластера на фізичному обладнанні? У випадку bare-metal (фізичних серверів) відсутній універсальний хмарний API для створення нового вузла, тому необхідне альтернативне рішення для інтеграції з Kubernetes. З іншого боку, у багатьох організаціях та спільнотах є невикористані обчислювальні ресурси – персональні комп'ютери користувачів, робочі станції, локальні сервери, які простоюють або використовуються не на повну потужність. Виникає ідея залучити ці ресурси до спільної групи для обчислень. Концепція волонтерських обчислень показала, що пристрої користувачів можуть успішно виконувати наукові та інші завдання, об'єднуючись у розподілені обчислювальні мережі (наприклад, проєкт BOINC використовує потужності персональних комп'ютерів добровольців). Запропонований у цій статті підхід поєднує ідею використання зовнішніх (користувацьких) ресурсів із механізмами автомасштабування Kubernetes. Метою є створення гібридної інфраструктури, де кластер Kubernetes за потреби автоматично розширюється за рахунок вузлів, наданих користувачами, без залучення традиційних хмарних ресурсів. У роботі використано сучасні інструменти автоматизації керування кластером – Cluster API та суміжні проєкти. Cluster API – це проєкт спільноти Kubernetes, що забезпечує декларативне керування життєвим циклом кластерів (створення, масштабування, оновлення) за допомогою Kubernetes API. Одним із інфраструктурних провайдерів для Cluster API є Metal³ – набір засобів для керування фізичною (bare-metal) інфраструктурою з використанням Kubernetes. Metal³ інтегрується з Cluster API, дозволяючи йому виступати інструментом для створення вузлів на реальних машинах. У даній роботі Cluster API та Metal³ використовуються для автоматизованого приєднання/від'єднання фізичних хостів (серверів) до кластеру Kubernetes. Персональні комп'ютери користувачів розглядаються як такі «фізичні хости», що можуть динамічно ставати частиною спільного кластера. Запропонований підхід полягає в тому, що кластер Kubernetes може автоматично залучати додаткові вузли із зовнішньої групи обчислювальних ресурсів, які надаються користувачами. На відміну від стандартного сценарію, де нові вузли – це віртуальні машини у хмарі, тут нові вузли – це фізичні машини (або віртуальні машини на локальному обладнанні), що знаходяться поза початковим кластером. Кожен такий вузол належить окремому користувачу або організації та зазвичай не є частиною кластера постійно. Система повинна мати змогу реєструвати доступні вузли, додавати їх до кластера при зростанні навантаження та видаляти (вивільняти) коли потреба в додаткових ресурсах зникає. Рішення будується на архітектурі з використанням Cluster API. Передбачається наявність керувального кластеру (management Cluster), який виконує роль "контролера" для основного робочого кластеру (workload Cluster). У нашому випадку ці ролі можуть бути об'єднані: наприклад, один кластер може одночасно запускати робочі навантаження та містити контролери Cluster API (конфігурація кластера, в якій робочі навантаження та компоненти керування кластером працюють одночасно на одному кластері). Контролери Cluster API оперують спеціальними ресурсами Kubernetes для опису інфраструктури: Cluster, Machine, MachineSet, MachineDeployment тощо. З їх допомогою визначається склад кластера (включно з кількістю вузлів) у декларативний спосіб. Для інтеграції фізичних машин користувачів використано інфраструктурний провайдер Metal³. Проєкт Metal³ надає компонент Bare Metal Operator (BMO) (див. рис. 1), який керує об'єктами

типу BareMetalHost – вони представляють собою доступні фізичні вузли з інформацією про спосіб доступу для їхнього ввімкнення та розгортання (через BMC, IPMI тощо).

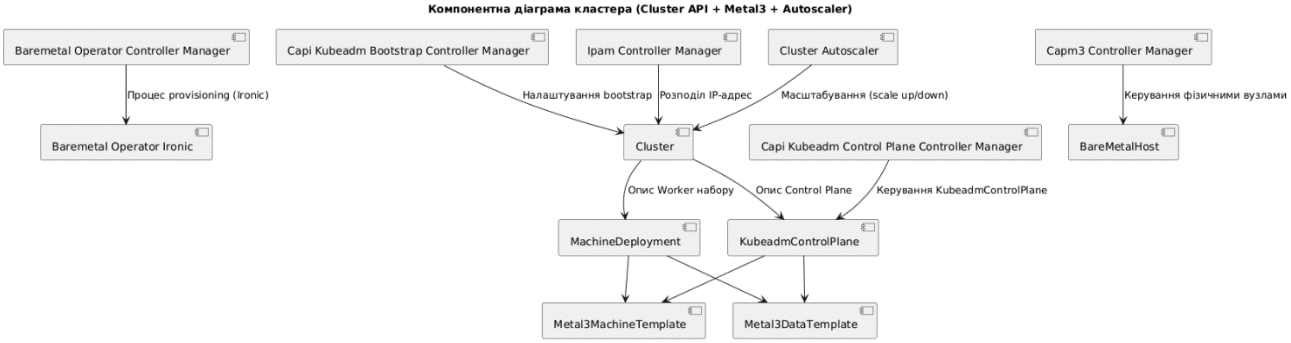

Рис. 1. Компонента діаграма кластеру.

Перед початком масштабування адміністратор реєструє персональні машини користувачів у системі як об’єкти BareMetalHost (наприклад, через додавання їх MAC-адрес, IP для управління та облікових даних доступу до їх BMC). Таким чином формується група потенційних вузлів, які готові до приєднання. Альтернативно, якщо безпосереднього доступу для автоматичного розгортання ОС на вузлі немає, може застосовуватися підхід агентського типу: на машині користувача заздалегідь встановлено необхідне програмне забезпечення (середовище виконання контейнерів, наприклад, containerd або CRI-O, що відповідають за запуск та керування контейнерами), і вона реєструється як готовий хост у кластері управління (аналогічно концепції Bring Your Own Host, BYOH). У даній роботі у реалізації зроблено акцент на використанні Metal³ та Bare Metal Operator для повної автоматизації процесу додавання вузла. Коли навантаження на кластері зростає (наприклад, з’являються нові поди, які не були розподілені планувальником Kubernetes (scheduler), тобто не можуть бути розміщені через нестачу ресурсів на існуючих вузлах), спрацьовує Cluster Autoscaler. Він працює як окремий компонент у кластері і постійно відслідковує невідпрацьовані запити на ресурси. У даній роботі Cluster Autoscaler налаштований у режим роботи, в якому замість API хмарного провайдера використовується інтерфейс прикладного програмування Cluster API для управління ресурсами (прапор --cloud-provider=Clusterapi при запуску). У такій конфігурації при необхідності масштабування Cluster Autoscaler не звертається до зовнішнього провайдера напряму, а створює новий об’єкт Machine (або збільшує репліку MachineDeployment) через API Kubernetes. Контролер Cluster API виявляє, що потрібен новий вузол, і ініціює процес його створення за допомогою провайдера Metal³. На стороні Metal³ Bare Metal Operator шукає вільний зареєстрований BareMetalHost, щоб призначити його новому Machine. Вибраний хост автоматично переходить у процес підготовки: через інтеграцію з OpenStack Ironic йому завантажується образ операційної системи і виконується конфігурування Kubernetes-вузла (наприклад, через cloud-init запуск kubeadm join). Після завершення цього процесу новий вузол підключається до кластеру Kubernetes (відображається в списку kubectl get nodes) і стає готовим для розміщення обчислювальних завдань, реалізованих у вигляді подів Kubernetes (див. рис. 2).

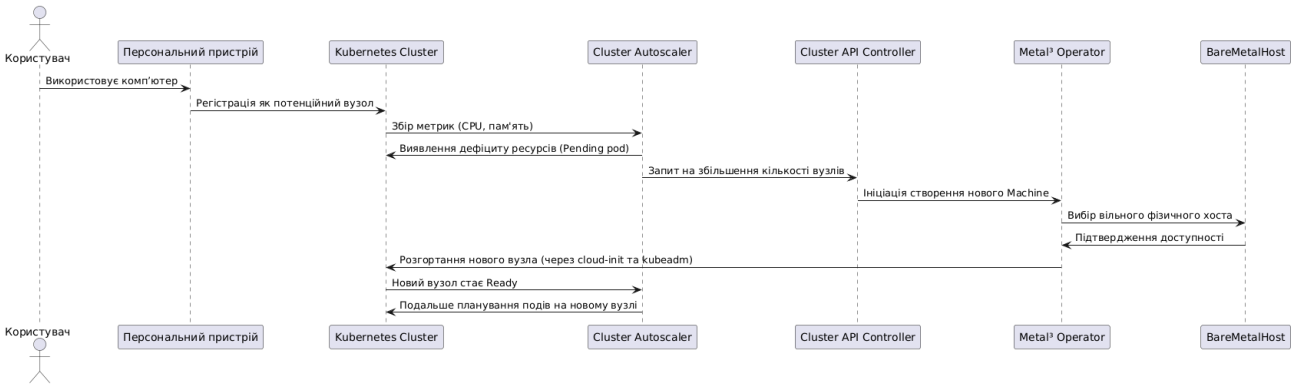

Рис. 2. Схема процесу реєстрації робочих вузлів.

Таким чином, кластер збільшується на один вузол. Весь цикл – від детектування нестачі обчислювальних ресурсів до моменту повного підключення та готовності нового вузла виконувати навантаження – виконується автоматично. Аналогічно, у разі тривалого простою ресурсів, Cluster Autoscaler може вирішити видалити зайвий вузол: тоді через Cluster API відповідний об’єкт Machine буде видалено, що призведе до від’єднання фізичного вузла (вузол повертається до початкового стану зареєстрованого об'єкта BareMetalHost (стану готовності для наступного підключення). Варто зазначити, що використання Metal³ у зв’язці з Cluster API фактично надає засоби Kubernetes для управління апаратними вузлами як об’єктами кластеру. Такий підхід узгоджується з концепцією Kubernetes-native infrastructure, коли інфраструктура (сервери) управляється так само, як і застосунки, через декларативні API. У даній роботі це дозволило досягти мети – автоматизувати залучення персональних машин.

У рамках даної роботи розгорнуто експериментальну інфраструктуру, реалізовано експериментальний сценарій із залученням персональних обчислювальних ресурсів користувачів, що емулюються за допомогою віртуальних машин. Використано кластер Kubernetes версії 1.30, у якому встановлено контролери Cluster API (версія v1beta1) та провайдер Metal³. Було зареєстровано декілька (2 шт.) віртуальних BareMetalHost як доступних вузлів – вони грали роль "персональних комп’ютерів" в експерименті. Кожному хосту призначено параметри BMC (емуляція через virtual BMC) та підготовлено завантажувальний образ з Kubernetes (CoreOS-based). Cluster Autoscaler розгорнуто як под (контейнер) у цьому ж кластері і налаштовано на автоматичну ідентифікацію групи вузлів (об'єктів Machine), придатних для масштабування. Параметри (або межі) масштабування кластера, що визначають мінімальну та максимальну допустиму кількість вузлів встановлені: мінімальна кількість вузлів = 1, максимальна = 2 (1 початковий + 1 додатковий). Моніторинг стану кластера здійснювався засобами Kubernetes (метрики використання CPU/RAM) та логів Cluster Autoscaler. На рис. 3 представлено етапи процесу автоматичної реєстрації нового вузла до Kubernetes-кластера за допомогою Metal³ та Cluster API:

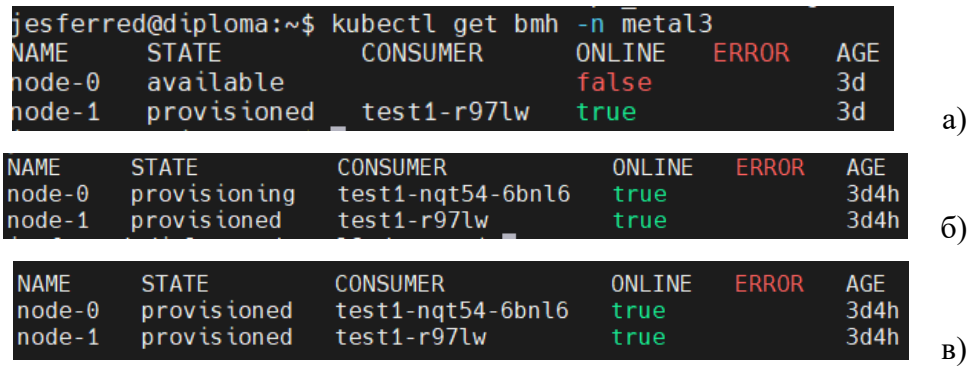

Рис. 3. Процес реєстрації нового вузла
а) Вузол готовий до приєднання (available) – це стан, коли вузол (BareMetalHost) зареєстрований у системі, але ще не приєднаний до робочого кластера;
б) Вузол приєднується (provisioning) – виконується автоматичний процес конфігурації, встановлення ОС, запуск необхідних сервісів, вузол перебуває у процесі інтеграції з кластером;
в) Вузол приєднаний (provisioned) – вузол повністю інтегровано в кластер, він готовий приймати навантаження та запускати поди Kubernetes.

Висновки. У даній роботі представлено новий підхід до автомасштабування кластерів Kubernetes, який базується на залученні персональних обчислювальних ресурсів користувачів. Сформовано архітектуру рішення з використанням Cluster API та Metal³, що дозволяє автоматично додавати/видаляти вузли кластера на основі зовнішніх (нехмарних) ресурсів. Реалізовано прототип та проведено експерименти, які підтвердили ефективність підходу: кластер успішно масштабується від 1 до 2 вузлів і назад без ручного втручання, забезпечуючи необхідні ресурси під час пікового навантаження.

Отримані результати свідчать, що спільне використання Kubernetes з інструментами керування інфраструктурою (Cluster API, Bare Metal Operator) відкриває можливість створення динамічних гібридних кластерів, де частина ресурсів надається добровільно учасниками або використовується за принципом спільноти.

Література

1. Смаглюк В.О., Алєксєєв М.О. Керування клієнтськими обчислювальними ресурсами з динамічним життєвим циклом у корпоративній мережі. Інфокомунікаційні та комп'ютерні технології, 2022, №2(04), с. 134–142.
2. Kubernetes Cluster Autoscaler – Офіційна документація. Kubernetes Documentation. – URL: https://Kubernetes.io/docs/concepts/Cluster-administration/node-autoscaling/
3. Metal³ Project – Metal3.io. Офіційний вебсайт проекту Metal³. – URL: https://metal3.io/
4. Cluster API Provider BYOH (Bring Your Own Host). GitHub Repository – vmware-tanzu/Cluster-api-provider-bringyourownhost. – URL: https://github.com/vmware-tanzu/Cluster-api-provider-bringyourownhost/
5. CNCF Blog – Kubernetes Cluster API reaches production readiness with v1.0. Cloud Native Computing Foundation, 6 Oct 2021. – URL: https://www.cncf.io/blog/2021/10/06/Kubernetes-Cluster-api-reaches-production-readiness-with-version-1-0/
6. BOINC: A Platform for Volunteer Computing. Berkeley Open Infrastructure for Network Computing (BOINC) Overview. – URL: https://boinc.berkeley.edu/