УДК 004.4

КЕРУВАННЯ КЛІЄНТСЬКИМИ ОБЧИСЛЮВАЛЬНИМИ РЕСУРСАМИ З ДИНАМІЧНИМ ЖИТТЄВИМ ЦИКЛОМ У КОРПОРАТИВНІЙ МЕРЕЖІ

DOI 10 36994 / 2788- 5518- 2022-02-04-15

Смаглюк В.О. *Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна. amxleclerk@gmail.com*
Алєксєєв М.О.*, к.т.н. Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна. n.alexeyev@gmail.com*

Анотація: *Робота присвячена актуальному питанню ефективного використання обчислювальних ресурсів в системах з динамічною розподіленою архітектурою, які застосовують обчислювальні пристрої персональні або загального користування для потреб розподілених обчислень, а саме задачі ефективного керування життєвим циклом робочих вузлів кластерів контейнеризованих застосунків. На відміну від відомих існуючих рішень, системи, які розглядаються у роботі, мають відносно короткостроковий і недетермінований час життя своїх обчислювальних агентів і тому повинні мати спеціалізований функціонал для автоматичного під'єднання та відключення обчислювальних агентів від кластеру, оскільки сам процес приєднання нового робочого вузла до такої системи займає певний час, і потребує певної кваліфікації і рівня авторизації. Якщо ж ввести у систему додатковий адміністративний вузол, що буде вирішувати питання налаштування нових клієнтів, ми в рази збільшуємо оперативне навантаження на такий управляючий об'єкт. Необхідно надати прозорий інтерфейс (процес на декілька дій) для користувача і при цьому автоматизувати будь-які повторювані дії зі сторони оператора Kubernetes кластеру. Запропонований у роботі підхід базується на організації процесу керування робочими вузлами на зразок Server Discovery у середовищі Kubernetes кластеру. Перевагою запропонованого підходу є гнучкий та простий інтерфейс керування робочими вузлами з недетермінованим життєвим циклом у корпоративній мережі на основі підготовлених програм, що виконують операції з перевірки операційного середовища користувача, інсталяції необхідного програмного забезпечення для приєднання до обчислювальної системи або ж деінсталяції компонентів та відключення агента за вимогою користувача. У роботі розглянуто експериментальне середовище, проведено дослідження роботи системи у можливих сценаріях поведінки користувача та обґрунтована ефективність підходу.*

Ключові слова: *розподілена обчислювальна система, Kubernetes кластер, керування життєвим циклом обчислювальних вузлів.*

MANAGEMENT OF CLIENT COMPUTING RESOURCES WITH A DYNAMIC LIFE CYCLE IN A CORPORATE KUBERNETES CLUSTER NETWORK

Volodymyr Smahliuk. *National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine. amxleclerk@gmail.com*
Nikolay Alieksieiev*, Ph,D, Ass. Prof. National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine. n.alexeyev@gmail.com*

Abstract: *The work is dedicated to the topical issue of effective use of computing resources in systems with a dynamic distributed architecture, which use personal or public computing devices for the needs of distributed computing, in particular the paper is devoted to the task of effective management of the life cycle of working nodes of clusters of containerized applications. Unlike the known existing solutions, the systems considered in the work have a relatively short-term and non-deterministic lifetime of their computing agents and therefore must have specialized functionality for automatically joining and*

detaching computing agents from the cluster as far as the process of joining a new working node to such a system takes a certain amount of time and requires a certain qualification and level of authorization. If we inject an additional administrative node into the system, which will solve the issue of setting up new clients, we will increase the operational load on such a control object. It is necessary to provide a transparent interface (few clicks process) for the user and at the same time automate any repetitive actions on the part of the Kubernetes cluster operator. The approach proposed in the work is based on the organization of the process of managing working nodes such as Server Discovery in the Kubernetes cluster environment. The advantage of the proposed approach is a flexible and simple interface for managing working nodes with a non-deterministic life cycle in the corporate network based on prepared programs that perform operations of checking the user's operating environment, installing the necessary software to connect to the computer system, or uninstalling components and disabling the agent on user demand. The paper describes the experimental environment, conducts a study of the system's operation in possible user behavior scenarios and proves the effectiveness of the approach.

Key words: *distributed computing system, Kubernetes cluster, management of nodes life cycle.*

Вступ

Для підвищення ефективності використання незайнятих ресурсів обчислювальних вузлів та зниження загального рівня їх простоювання і водночас, для вирішення нагальних завдань з розміщення та виконання програмних компонентів корпоративних інформаційно-обчислювальних систем раніше у роботі було вирішено створювати кластери із залученням персональних пристроїв користувачів корпоративної мережі, а також парку комп'ютерної техніки та загальнодоступних обчислювальних пристроїв у корпоративній мережі. Також було прийняте рішення щодо використання контейнеризації прикладних програм та програмної технології Kubernetes для оркестрації робочих навантажень на їх основі, оскільки цей підхід дозволяє використовувати апаратні і програмні обчислювальні ресурси різні за походженням і характеристиками, а також дає змогу підключати і від'єднувати їх у процесі експлуатації [1]. Вузол Kubernetes (node, "нода") або робочий вузол у даному контексті це фізична або віртуальна машина, яка бере участь у кластері Kubernetes, і яку можна використовувати для запуску робочих навантажень (pod, "под") у вигляді контейнеризованих програм. Проте, процес долучення та від'єднання робочих вузлів потребує виконання певних операцій у ручному режимі. Користувач, що хоче надати обчислювальні потужності свого пристрою для виконання прикладних програм у корпоративному кластері, повинен самостійно підготувати систему, а саме:

- перевірити сумісність клієнтської системи з цільовою кластерною мережею;
- встановити необхідне клієнтське програмне забезпечення за вимогами його операційної системи;
- ініціалізувати запит на приєднання до Kubernetes кластеру.

За необхідності користувач повинен мати можливість від'єднатися від обчислювальної системи і безпечно очистити будь-які артефакти, що могли залишитися після налаштування зв'язку з цільовою системою, включаючи клієнтське програмне забезпечення. Також, необхідно врахувати сценарії, за яких клієнт більше не доступний для розміщення на ньому прикладних програм (чи то через проблеми з мережею, чи через санкціоновані дії клієнта спрямовані на розірвання зв'язку з системою) і реагувати у відповідності з конкретними причинами.

Враховуючи це, можна виділити три основні можливі стани клієнта:

1) готовність – клієнт готовий ініціалізувати запит на приєднання;
2) активний режим – клієнт є частиною розподіленої системи та надає свої незайняті ресурси для прикладних обчислень;
3) неактивний режим – клієнт більше не надає свої ресурси або зв'язок з ним втрачено, але він досі числиться у списку робочих вузлів.

Для вирішення цих проблем можна ввести додатковий керуючий вузол (адміністратор), який буде налаштовувати цільові системи.

При невеликій кількості потенційних клієнтів адміністратор Kubernetes кластеру може сам проконтролювати переходи між станами та виконувати конфігураційні задачі, а саме: ініціалізацію програмного забезпечення, аналіз відповідності наданого пристрою до вимог цільової системи або видалення його. Проте, зі збільшенням кількості робочих пристроїв N буде зростати операційне навантаження P на адміністративний вузол за певним експоненціальним законом $P = exp(N)$, що призводить до складностей коректного налаштування та підтримки таких систем.

Якщо додатково врахувати, що система є динамічною, тоді час життя робочих пристроїв не обов’язково буде збільшуватися за лінійним законом, що є справедливим для системи зі статичними пристроями, постійно під’єднаними до системи. Навіть навпаки: збільшення частоти переходу між можливими станами призводить до додаткових операційних навантажень, оскільки треба відслідковувати та вести звітність про кожен з клієнтів: у якому стані вони знаходяться, а також які дії потрібно виконати, щоб безпечно перевести пристрій з одного в інший стан.
Досвідчені користувачі можуть самі налаштовувати свою систему за потреби, але у загальному випадку, клієнт повинен мати зручний інтерфейс для під’єднання до цільової системи без додаткових операційних навантажень зі свого боку.

Виконання досліджень

Враховуючи усе вище згадане, необхідно вирішити дві основні проблеми для зменшення кількості адміністративної роботи або, за можливості, уникнути будь-якої додаткової взаємодії з ним.

Першою з проблем є ведення актуальної таблиці станів робочих пристроїв. Загалом такий механізм називається Service Discovery, де у ролі сервіса виступає робочий вузол (“нода”). Service Discovery створений для того, щоб з мінімальними витратами можна підключити новий сервіс до нашого оточення.
У нашому випадку це питання вирішується за допомогою інтегрування певної бази даних. У середовищі Kubernetes кластеру цю функцію виконує програмне забезпечення etcd.
База даних etcd є послідовним розподіленим сховищем у форматі ключ-значення. Це ПЗ використовується як окрема служба координації в розподілених системах. Воно призначене для зберігання невеликих обсягів даних, які можуть повністю поміститися в оперативній пам’яті [2]. Цей компонент зберігає всю інформацію про робочі навантаження, які представлені у системі, враховуючи облік нод. За допомогою інтерфейсу Kubernetes API ця інформація періодично оновлюється, щоб відповідати реальному станові системи.
Таким чином, etcd веде інформацію про два стани нод: активний (*Ready*) та неактивний (*NotReady*) (рис. 1).

```
NAME                 STATUS    ROLES   AGE     VERSION
master.example.com   Ready     master  5h      v1.23
node1.example.com    NotReady  compute 5h      v1.23
node2.example.com    Ready     compute 5h      v1.23
```

Рис. 1. Приклад відображення станів робочих вузлів (“нод”) Kubernetes кластеру

Якщо у etcd немає інформації про певну з нод, тоді вважається, що вона перебуває у стані готовності до ініціалізації. Навіть у випадку, коли пристрій користувача було вилучено із кластеру, але програмні компоненти досі встановлено на його машині, він знову має пройти етап ініціалізації запиту на приєднання до розподілювальної системи.
Друга проблема полягає в тому, як зменшити операційне навантаження на адміністративний вузол під час підготовки клієнтської системи до приєднання до кластерної мережі. Для вирішення цього питання пропонується використовувати автоматизований пакет (скрипт), який перевірить систему на відповідність вимог цільової кластерної мережі, інсталює необхідні програмні пакети та приєднає клієнтський пристрій до Kubernetes.
Ідея полягає в тому, щоб надати доступне (в рамках корпоративної мережі) посилання на єдиний виконувальний файл, який в залежності від сценарію буде мати різний алгоритм дій. Бажаний сценарій можна буде обрати додаючи різні опції під час запуску скрипту.
Розглянемо більш детально кожен із сценаріїв, які реалізує програмний код, та порядок дій для кожного з них.

1. Реєстрація робочих вузлів:
    a. Ініціалізація запиту на приєднання (запит на адресу http://cluster.local/script)
    b. Отримання скрипту на встановлення компонентів
    c. Запуск скрипту з флагом *--init*

i. Перевірка системи (os-release) та вимог цільової системи до ресурсів пристрою;
ii. Перевірка встановлених пакетів;
iii. Встановлення пакеті, якщо таких не було знайдено в системі.

d. Ініціалізація з’єднання до кластеру
e. З’єднання встановлено

Блок-схему наведеного алгоритму зображено на рис. 2. По завершенню сценарію клієнтський пристрій переходить зі стану “готовності до ініціалізації” в стан “активний режим”.

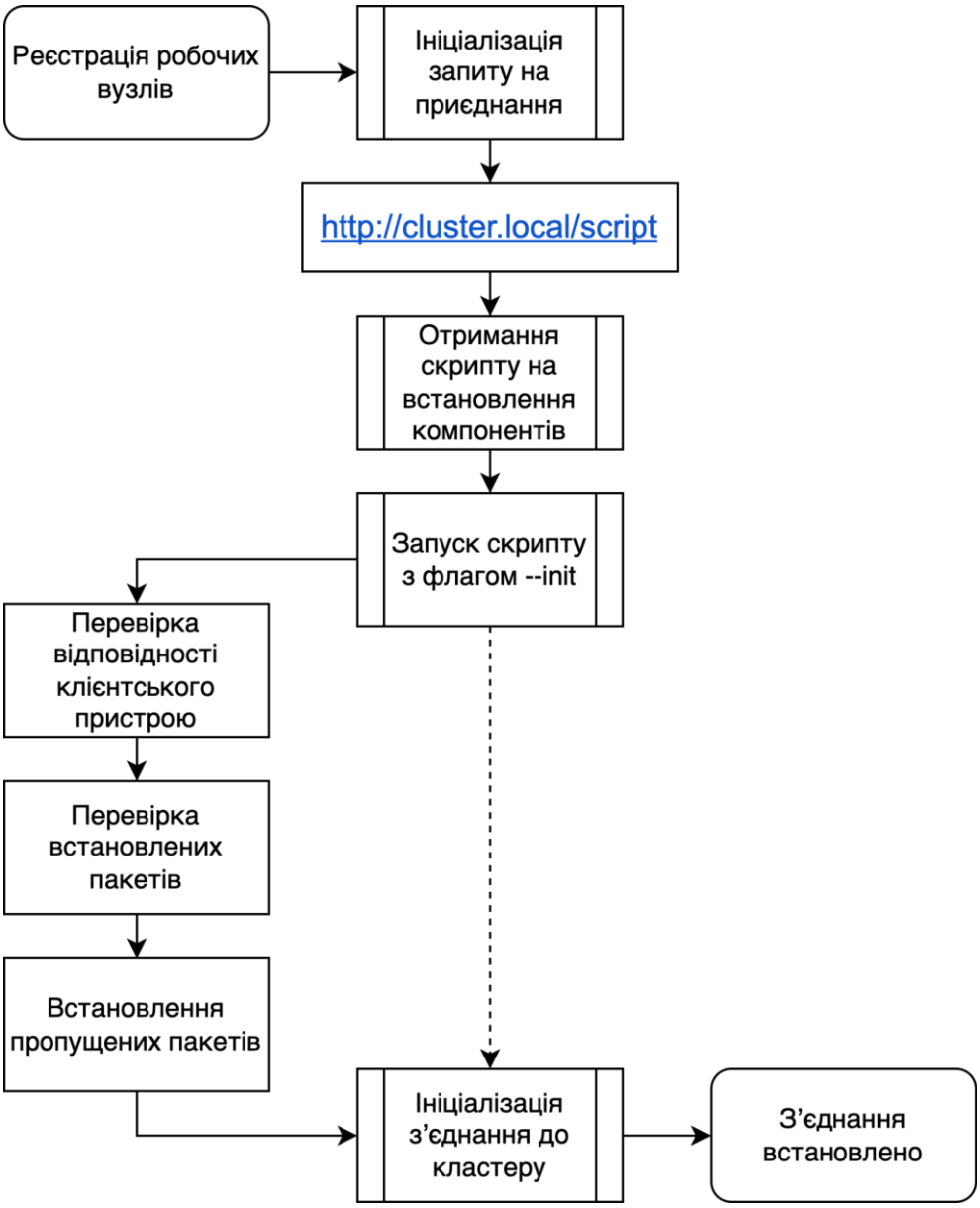

Рис. 2. Блок-схема процесу реєстрації робочих вузлів

2.Вилучення робочих вузлів:

f. Ініціалізація запиту на вилучення (запит на адресу http://cluster.local/script)
g. Отримання скрипту на видалення компонентів
h. Запуск скрипту з флагом *--reset*
i. Перевірка системи (os-release)
ii. Перевірка встановлених пакетів
iii. Видалення пакетів
i. Команда на роз’єднання від кластеру

j. Сервер від’єднано від кластеру

Блок-схему наведеного алгоритму зображено на рисунку 3. По завершенню сценарію клієнтський пристрій переходить зі стану “активний режим” в стан “готовність до ініціалізації”, не зважаючи на те, що приєднання не планується.

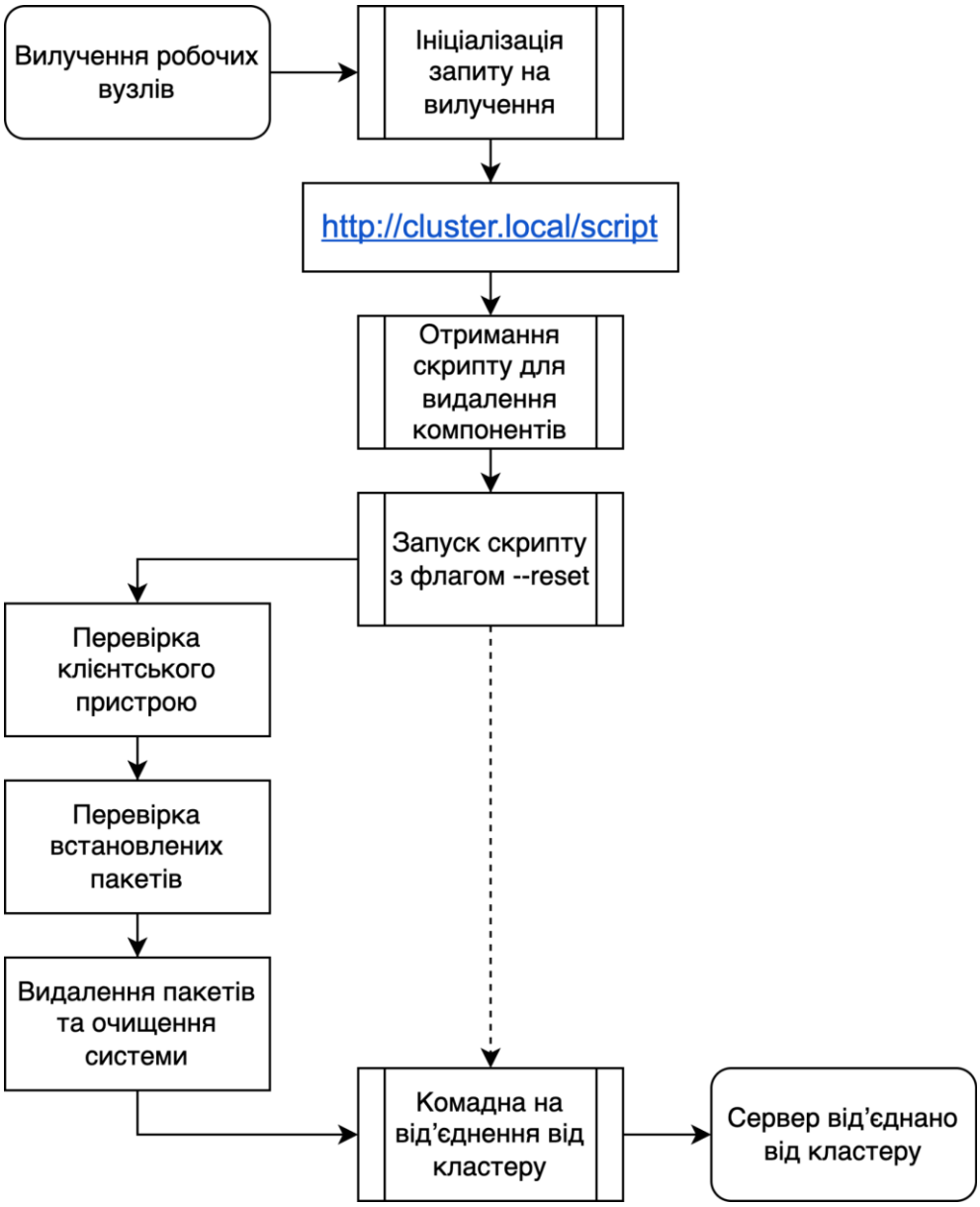

Рис. 3. Блок-схема процесу від’єднання робочих вузлів

Наведені два сценарії контролюються спеціальним програмним кодом, що отримує клієнт із відповідної адреси. Проте, може бути ситуація, коли було втрачено зв’язок із робочим вузлом. У такому випадку, робочий вузол переводиться у стан “неактивний режим”. Коли вузол вимикається або виходить з ладу, він переходить у стан NotReady, тобто його не можна використовувати для запуску подів (розподілених прикладних програм). Після цього всі поди, запущені на вузлі, стають недоступними [3]. Проте, варто наголосити, що робочі навантаження досі закріплено за проблемним вузлом.

Поширеними причинами *Node NotReady* помилки Kubernetes є

- брак ресурсів на вузлі, проблему з kubelet (агентом, який дозволяє Kubernetes сервісам отримувати доступ до вузла і та контролювати його роботу);
- помилку, пов’язану з kube-proxy (мережевий агент на вузлі);
- проблему з мережею в цілому.

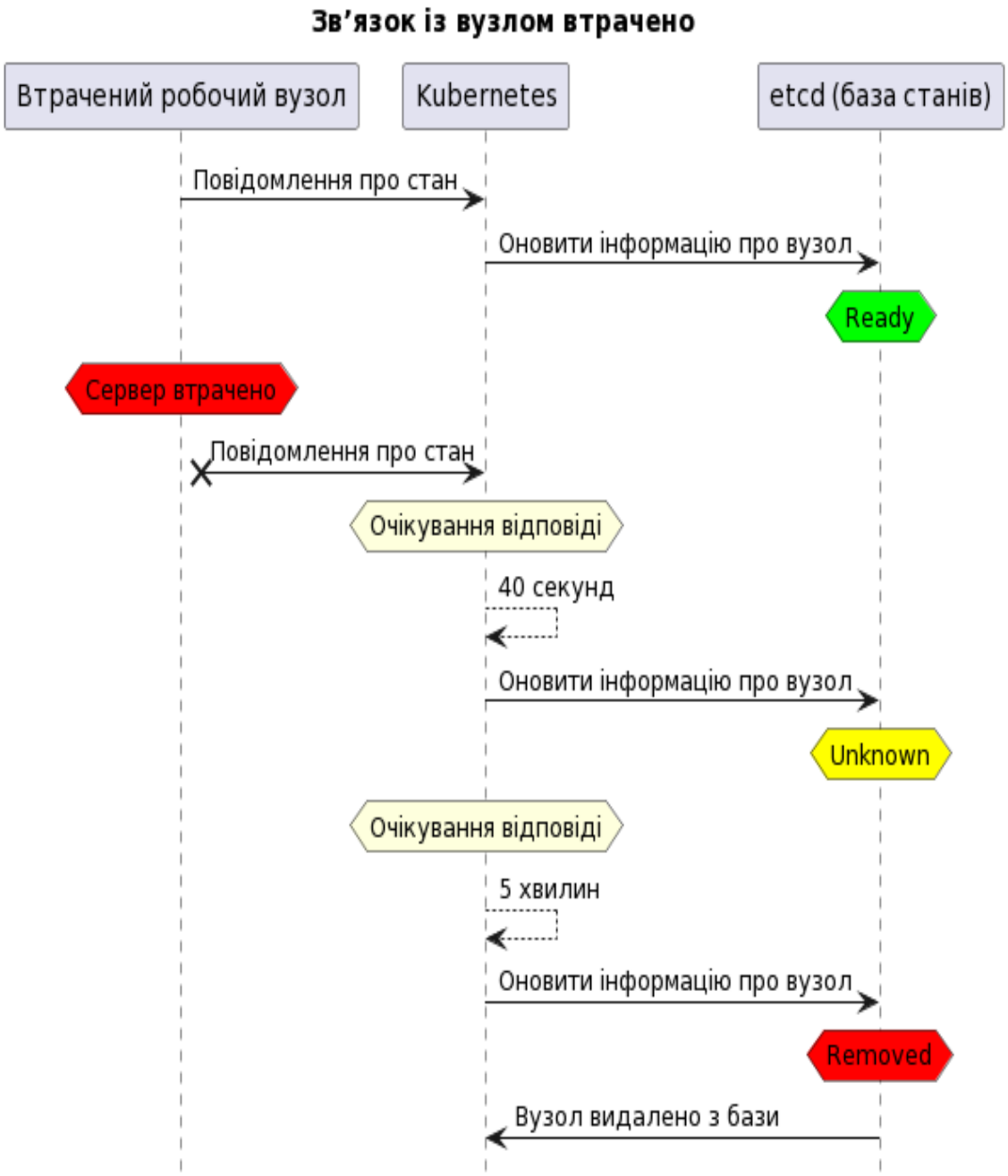

Рис. 4. Діаграма послідовностей у випадку втрати зв’язку з робочим вузлом

Якщо Kubernetes не може зв’язатися з вузлом, він чекає за замовчуванням 40 секунд, а потім встановлює статус вузла на *Unknown* або невідомий. Якщо вузол залишається недоступним відбувається ініціювання виселення за допомогою API для всіх подів на недоступному вузлі. За замовчуванням контролер вузла чекає 5 хвилин між позначенням вузла як невідомого та поданням першого запиту на вилучення [4]. Якщо недоступний вузол не повернувся до мережі, його буде видалено із внутрішньої бази (див. рис. 4), що у термінах даної статті означає переведення у стан “готовності до ініціалізації”.

У випадку повторного приєднання втраченого робочого вузла після того, як його було видалено з Kubernetes кластеру, клієнт повинен повністю пройти процес ініціалізації описаний вище (див. рис. 2).

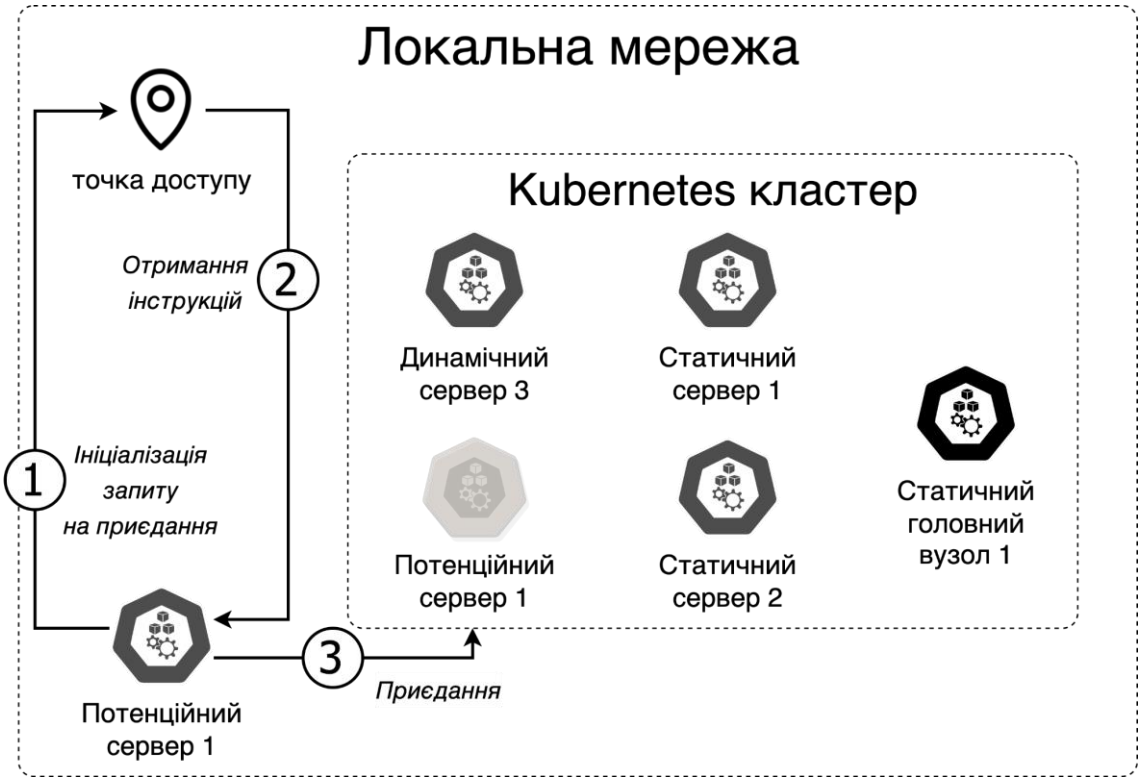

Рис. 5. Схема експериментального макету

Експериментальний макет (див. рис. 5) містить один головний вузол, два робочі статичні вузли, один динамічний вузол (у стані “активний режим”), потенційний динамічний вузол (у стані “готовність до ініціалізації”) та точку доступу (в рамках корпоративної мережі) з виконувальним програмним кодом. Повна інформація про зареєстровані робочі вузли наведена на рисунку 6.

```
NAME                STATUS   ROLES    AGE   VERSION
master.static.com   Ready    master   12h   v1.23
node1.static.com    Ready    compute  12h   v1.23
node2.static.com    Ready    compute  12h   v1.23
node3.dynamic.com   Ready    compute  10h   v1.23
```

Рис. 6. Список робочих вузлів Kubernetes кластеру

Для приєднання нового вузла до кластеру достатньо завантажити код та запустити його з опцією *--init* (рис. 7).

Відповідно, після успішного завершення скрипта ми побачимо новий робочий вузол node4.dynamic.com (рис. 8), запис про який збережено у ectd.

Процес вилучення відбувається під час запуску програмного скрипта з опцією *--reset* (рис. 9). Для чистоти експерименту, було запущено запит на видалення для node3.dynamic.com.

Kubernetes кластер реагує на архітектурні зміни. Робочий вузол більше не є частиною обчислювальної мережі, як можна побачити на рис. 10.

```
bash-3.2$ sh ./script.sh --init
# System information
Target OS: Ubuntu 22.04.1 LTS
System resources check: passed


# Kubernetes information
Target Kubernetes version: 1.23.13
Installation kubernetes packages: kubeadm, kubelet, kubectl
Successfully installed kubernetes packages.


# Docker information
Target Docker version: 20.10.8
Installation docker packages: docker-ce, docker-ce-cli, containerd.io
Successfully installed docker packages.


# Joining the cluster
Successfully joined the cluster.
```

Рис. 7. Процес ініціалізації нового динамічного вузла до Kubernetes кластеру

```
NAME                STATUS    ROLES   AGE     VERSION
master.static.com   Ready     master  13h     v1.23
node1.static.com    Ready     compute 13h     v1.23
node2.static.com    Ready     compute 13h     v1.23
node3.dynamic.com   Ready     compute 11h     v1.23
node4.dynamic.com   Ready     compute 6m      v1.23
```

Рис. 8. Список робочих вузлів Kubernetes кластеру після додавання нового вузла

```
bash-3.2$ sh ./script.sh --reset

# System information
Target OS: Ubuntu 22.04.1 LTS

# Kubernetes information
Target Kubernetes version: 1.23.13
Removing kubernetes packages: kubeadm, kubelet, kubectl
Successfully removed kubernetes packages.

# Docker information
Target Docker version: 20.10.8
Removing docker packages: docker-ce, docker-ce-cli, containerd.io
Successfully removed docker packages.

# Reset the cluster
Successfully rested the cluster.
```

Рис. 9. Процес вилучення динамічного вузла до Kubernetes кластеру

```
NAME                STATUS    ROLES   AGE     VERSION
master.static.com   Ready     master  13h     v1.23
node1.static.com    Ready     compute 13h     v1.23
node2.static.com    Ready     compute 13h     v1.23
node4.dynamic.com   Ready     compute 20m     v1.23
```

Рис. 10. Список робочих вузлів Kubernetes кластеру після видалення динамічного вузла

На рис.11 зображено поточну кластерну мережу відносно доступних робочих вузлів. Видаленого серверу 3 більше не існує у мережі.

Рис. 11. Кінцевий стан системи

Програмний код запропонованого рішення буде розміщено у публічному доступі за посиланням: https://github.com/VolodymyrSmahliuk/k8s-dynamic-node-operator

Висновки

1. Завдяки програмному коду вдалося мінімізувати оперативне навантаження на адміністратора Kubernetes кластеру, стосовно управління життєвим циклом робочих вузлів та їх підготовкою до приєднання у систему.
2. Запропонований метод дозволяє автоматизувати процес реєстрації та вилучення робочих вузлів до Kubernetes кластеру, що дає можливість побудувати динамічну обчислювальну систему з робочими вузлами з короткостроковим і недетермінованим часом життя своїх обчислювальних агентів.

Література

1. Смаглюк, В. О., Алєксєєв, М. О. Використання обчислювальних потужностей мобільних пристроїв та пристроїв інтернету речей у корпоративній мережі. *Збірник матеріалів Міжнародної науково-технічної конференції «Перспективи телекомунікацій»*, 2021. С. 238-240.
2. Docker Documentation: Docker architecture. 2021. URL: https://docs.docker.com/get-started/overview/#docker-architecture.
3. Etcd Documentation: “What is etcd?”, 2022. URL: https://etcd.io/docs/v3.5/faq/#what-is-etcd.
4. Luksa, Marko. *Kubernetes in action*. Simon and Schuster, 2017.
5. Kubernetes Documentation: Kubernetes Nodes: Conditions, 2022. URL: https://kubernetes.io/docs/concepts/architecture/nodes/#condition.

References

1. Smahlyuk, V. O., Alyeksyeyev, M. O. Vykorystannya obchyslyuval'nykh potuzhnostey mobil'nykh prystroyiv ta prystroyiv internetu rechey u korporatyvniy merezhi. [Quotation of the number of pressures of mobile devices and Internet speeches at corporate]. *Zbirnyk materialiv Mizhnarodnoyi naukovo-tekhnichnoyi*
2. *konferentsiyi «Perspektyvy telekomunikatsiy*», 2021.S, 238-240.
Docker Documentation: Docker architecture. 2021. URL: https://docs.docker.com/get-\
started/overview/#docker-architecture.
4. Etcd Documentation: “What is etcd?”, 2022. URL: https://etcd.io/docs/v3.5/faq/#what-is-etcd.
5. Luksa, Marko. *Kubernetes in action*. Simon and Schuster, 2017.
Kubernetes Documentation: Kubernetes Nodes: Conditions, 2022. URL:
https://kubernetes.io/docs/concepts/architecture/nodes/#condition