Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Kubernetes-кластер на основі Raspberry PI*

Виконав: студент *IV* курсу, групи *CI-41*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

		Крайник О.В.
	(підпис)	(прізвище та ініціали)
Керівник		*Луцків А.М.*
	(підпис)	(прізвище та ініціали)
Нормоконтроль		*Тиш Є.В.*
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		*Осухівська Г.М.*
	(підпис)	(прізвище та ініціали)
Рецензент		*Гащин Н.Б.*
	(підпис)	(прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
« ___ » _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Крайнику Олексію Володимировичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Kubernetes-кластер на основі Raspberry PI*

Керівник роботи *Луцків Андрій Мирославович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *28* » *лютого* 2023 року № *4.7-238*

2. Термін подання студентом завершеної роботи *22.06.2023 р.*

3. Вихідні дані до роботи *особливості kubernetes, способи і засоби організації кластерів, тип мікроконтролера Raspberry PI*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз принципів та особливостей організації kubernetes-кластерів
2. Проектування апаратно-програмної інфраструктури kubernetes-кластера
3. Реалізація kubernetes-кластера на Raspberry PI
4. Безпека життєдіяльності, основи охорони праці. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. *Організація структури kubernetes..*
2. *Організація кластерів на віртуальних машинах.*
3. *Архітектура kubernetes-кластера.*
4. *Характеристики Raspberry PI.*
5. *Сценарії взаємодії користувача з кластером.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	*Гурик О.Я., к.т.н., доц. каф. МТ*		
основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Розробка і затвердження технічного завдання*	*28.02-13.03.2023*	
2.	*Аналіз технічного завдання*	*15.03-02.04.2023*	
3.	*Визначення вимог до апаратного та програмного забезпечення kubernetes-кластера*	*03.04-18.04.2023*	
4.	*Проектування структури kubernetes-кластера*	*19.04-04.05.2023*	
5.	*Налаштування параметрів kubernetes-кластера та сценаріїв використання*	*04.05-12.05.2023*	
6.	*Розробка інструкцій із встановлення та налаштування параметрів безпеки kubernetes-кластера*	*12.05-29.05.2023*	
7.	*Безпека життєдіяльності, основи охорони праці*	*01.06-05.06.2023*	
8.	*Оформлення кваліфікаційної роботи*	*05.06-12.06.2023*	
9.	*Попередній захист кваліфікаційної роботи*	*12.06-17.06.2023*	
10.	*Захист кваліфікаційної роботи*	*19.06-24.06.2023*	

Студент _____ *Крайник Олексій Володимирович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Луцків Андрій Мирославович*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Kubernetes-кластер на основі Raspberry PI // Кваліфікаційна робота на здобуття освітнього ступеня бакалавр // Крайник Олексій Володимирович // ТНТУ, спеціальність 123 «Комп'ютерна інженерія»// Тернопіль, 2023 // с.– 63 , рис. – 49 , табл. –4, аркушів А1 – 5, бібліогр. – 21.

Ключові слова: kubernetes, кластер, мікроконтролер, Raspberry PI.

Під час виконання кваліфікаційної роботи проведено аналіз завдань щодо організації kubernetes-кластера на основі Raspberry PI та досліджено особливості функціональних властивостей та структури відкритої платформи Kubernetus.

На основі результатів аналізу спроектовано архітектуру кластеру, до складу якого входить три мінікомп'ютери Raspberry PI 3 Model B. Один з цих вузлів налаштовано як master, що дає змогу забезпечити керування іншими хостами та виконанням тестових docker-контейнерів.

В процесі організації kubernetes-кластера використано додаткове комутаційне обладнання, зокрема Mikrotik RB760iGS (hEX S), що забезпечує доступ до мережі Інтернет та відповідає за безпеку доступу до елементів кластера, та некерований комутатор DGS-1005P, основна задача якого полягає у забезпеченні зв'язку між хостами кластера та маршрутизатором.

На кожному хості кластера встановлено операційну систему Ubuntu 18.04, а процес автоматизованого розгортання інфраструктури забезпечує інструмент Ansible.

ABSTRACT

Raspberry PI-based Kubernetes-cluster // Bachelor's thesis // Krainyk Oleksii // TNTU, speciality 123 «Computer engineering»// Ternopil, 2023 // p.– 63 , fig. – 49 , tab. – 4, posters A1 – 5, ref. – 21.

Keywords: kubernetes, cluster, microcontroller, Raspberry PI.

During the performance of the qualification work, an analysis of tasks related to the organization of a kubernetes cluster based on Raspberry PI was carried out, and the features of the functional properties and structure of the open platform Kubernetes were investigated.

Based on the results of the analysis, a cluster architecture was designed, which includes three Raspberry PI 3 Model B minicomputers. One of these nodes is configured as a master, which allows for the management of other hosts and the execution of test docker containers.

In the process of organizing a kubernetes cluster, additional switching equipment was used, in particular Mikrotik RB760iGS (hEX S), which provides access to the Internet and is responsible for the security of access to cluster elements, and an unmanaged switch DGS-1005P, the main task of which is to ensure communication between the cluster hosts and the router.

Ubuntu 18.04 operating system is installed on each host of the cluster, and the automated infrastructure deployment process is provided by the Ansible tool.

ЗМІСТ

ВСТУП

Сучасне зростання об’єму інформації вимагає від фахівців у сфері інформаційних технологій застосування і розробки нових інструментів їх опрацювання. Одним із шляхів вирішення цих задач є використання методів і технологій паралельної і розподіленої обробки даних як у хмарних сервісах, так і засобами наявної апаратно-програмної інфраструктури. Це означає також необхідність організації обчислювальних кластерів, управління яким повинно бути гнучким в контексті розподілу апаратних ресурсів та оптимізації виконання програмних додатків.

Одним з нових напрямів розвитку інформаційних технологій є напрям DevOps, реалізація якого базується на процесі, що називають «контейнеризація». У цьому процесі програмний компонент, його середовище та залежності поміщаються в ізольований контейнер.

Контейнерізація покращує швидкість розгортання, виправлення та масштабування. У наші дні віртуальні машини та хмарні обчислення значно скоротили простої та несправні програмних додатків. Але запуск гостьових операційних систем на них вимагає багато обчислювальних ресурсів.

Контейнери застосовують «процесори виконання», які спільно використовують операційну систему хоста машин, на яких вони розгорнуті. Механізми виконання забезпечують швидший час запуску, покращену ефективність сервера та менші розміри сховища.

Більшість механізмів виконання займають кілька мегабайт, тоді як віртуальним машинам потрібно 4-8 гігабайт пам’яті. Таким чином, контейнери портативні, добре масштабовані, безпечні, підтримують кілька хмарних платформ і є дружніми до DevOps.

Kubernetes – це система оркестрації контейнерів з відкритим кодом для керування, масштабування та автоматизації розгортання програмного забезпечення.

Kubernetes допомагає організаціям із DevOps, оскільки він поєднує фазу розробки та обслуговування програмних систем для підвищення гнучкості.

За допомогою інтерфейсу користувача Kubernetes розробники можуть переглядати, отримувати доступ, розгортати, оновлювати та оптимізувати екосистеми контейнерів. По суті, контейнеризація є ефективнішим і результативним способом реалізації DevOps, ніж монолітна програма. Kubernetes створює та керує контейнерами на хмарних серверних системах, допомагає командам DevOps зменшити навантаження на інфраструктуру, дозволяючи їм працювати на різних машинах/середовищах без збоїв.

Тому актуальною задачею на сьогодні є побудова kubernetes-кластера на базі Raspberry PI, що дозволить підвищити масштабованість і керованість програмними додатками з однієї сторони, а з іншої – дозволить зекономити кошти у порівнянні з фінансовими витратами на використання хмарних сховищ.

РОЗДІЛ 1 АНАЛІЗ ПРИНЦИПІВ ТА ОСОБЛИВОСТЕЙ ОРГАНІЗАЦІЇ KUBERNETES-КЛАСТЕРІВ

1.1 Аналіз завдань при організації kubernetes-кластера

Перш, ніж перейти до аналізу вимог щодо організації kubernetes-кластера потрібно проаналізувати сферу його застосування в контексті загальних практик DevOps, що є сукупністю практик і рекомендацій, орієнтованих на збільшення продуктивності процесу імплементації програмного забезпечення. DevOps допомагає скоротити час та одночасно підвищити якість виконання процесів при створенні ПЗ зі стабільною та послідовною доставкою коду.

Завдяки підвищеній швидкості DevOps дозволяє організаціям обслуговувати своїх клієнтів і ефективно досягати поставлених цілей. Це також покращує якість програмних продуктів, зменшує помилки та покращує загальну продуктивність. Основними перевагами DevOps є швидкість, безпека, надійність, CI/CD, масштабованість і покращена співпраця. На рис. 1.1 показано фази виконання операцій при розробці програмного забезпечення із застосуванням Kubernetes.

Рисунок 1.1 – Kubernetes у DevOps

Kubernetes-кластер на основі Raspberry PI призначений для забезпечення швидкості та зручності масштабованості програмних додатків, автоматизації процесу розгортання та керування виконанням.

Важливою перевагою застосування Kubernetes, особливо у випадку оптимізації розроблення програмного забезпечення для «наземної» інфраструктури (аналог хмари), є забезпечення побудови платформи з можливостями планування запуску контейнерів. Запуск контейнерів відбувається у на базі фізичних кластерів або віртуальних машин (ВМ). При організації kubernetes-кластера апаратна інфраструктура складається з трьох мінікомп'ютерів Raspberry PI.

У даному випадку, kubernetes планується використовувати як засіб автоматизованого розгортання та оркестрації контейнерів, що забезпечує функції масштабування, балансування навантаження, організації безпеки та інші.

У більш ширшому розумінні, організація такого типу кластеру дає змогу забезпечити повноту реалізації програмного забезпечення з можливістю ефективного використання інфраструктури із застосуванням контейнерів у відповідному середовищі. Враховуючи те, що kubernetes — це засіб автоматизації операційних завдань, то за допомогою нього можна робити багато того ж, що й інші платформи додатків або системи керування, але для користувацьких контейнерів.

На kubernetes- кластер покладаються функції по типу тих, які забезпечує хмарна інфраструктура, тобто програмні додатки можна запускати наче у хмароподібному середовищі, отримувати доступ до них ззовні, а програмному забезпеченню, що зберігає стани надається можливість використовувати рівень зберігання даних.

Мета побудови та налаштування kubernetes-кластера на основі Raspberry PI полягає у забезпеченні можливості реалізації дешевого інструменту масштабування інфраструктури та управління навантаженням на вузли кластера,

а також організації безпечного доступу з мережі Інтернет для управління як самим кластером, так і його вузлами.

Для досягнення мети щодо організації kubernetes-кластера у кваліфікаційній роботі визначено наступні задачі::

– обґрунтувати вибір апаратного та програмного забезпечення для побудови kubernetes-кластера;

– дослідити технічні характеристики апаратних компонентів кластера;

– провести налаштування комунікаційної інфраструктури для доступу до мережі Інтернет;

– зареєструвати доменне ім’я, як точку входу для безпечного доступу до ресурсів кластера ;

– організувати налаштування головного (master) вузла та інших вузлів (slave) кластера;

– забезпечити налаштування безпеки доступу до кластера як з мережі Інтерент так і з мережі кластера;

– провести тестування kubernetes-кластера шляхом запуску ngnix сервера.

Основна задача kubernetes-кластера полягає в автоматизації процесів розгортання інфраструктури при проведенні обчислень, а також автоматизації управління виконанням контейнерів. Окрім цього, розробники мають можливість організації та побудови хмарно-орієнтованих додатків при використанні Kubernetes в якості платформи, яка підтримує шаблони.

Шаблони представляють собою сукупність інструментів, які застосовуються при створенні програм і служб на базі принципів контейнеризації. Основні функції Kubernetes:

– оркестрація контейнерів на багатьох вузлах;

– оптимізація використання апаратних ресурсів при виконанні і запуску користувацького програмного забезпечення;

– контроль та автоматизоване розгортання необхідного програмного забезпечення;

– ефективність функціонування програмного забезпечення, що передбачає збереженням стану;

– масштабування програм, які містяться у контейнерах та необхідних їм ресурсів у режимі реального часу;

– декларативне управління сервісами та системними службами для гарантування того, що розгорнуте програмне забезпечення завжди працює так, як заплановано;

– перевірка працездатності і самовідновлення користувацьких програм за допомогою авторозміщення, автоперезапуску, автореплікації та автомасштабування.

Однак Kubernetes покладається на інші проекти, щоб повністю надавати ці організовані послуги. Додавши інші проекти з відкритим вихідним кодом, можна повністю використати потужність Kubernetes. Ці необхідні частини включають (серед іншого):

– реєстр через такі проекти, як Docker Registry;

– мережу за допомогою таких проектів, як OpenvSwitch та інтелектуальну граничну маршрутизацію;

– телеметрію через такі проекти, як Kibana, Hawkular і Elastic;

– безпеку за допомогою таких проектів, як LDAP, SELinux, RBAC і OAUTH з багаторівневими рівнями;

– автоматизацію з додаванням playbook Ansible для встановлення та керування життєвим циклом кластера;

– сервіси через багатий каталог популярних шаблонів програм.

При проектуванні Kubernetes-кластера доцільно використовувати Ansible для автоматизованого розгортання програмного забезпечення та доступу до Microtik-маршрутизатора. Безпека доступу ззовні забезпечується шляхом використання CloudFlare, а всередині кластера – через налаштування маршрутизатора.

Зв’язок між компонентами кластеру забезпечуються як на програмному, так і на апаратному рівні. Апаратним рівнем визначено проводове з’єднання за допомогою Ethernet кабеля категорії 5е.

На рівні програмного забезпечення забезпечується передача даних за допомогою транспортного рівня і протоколу TCP/IP. З’єднання з мережею Інтернет керованого маршрутизатора забезпечується на основі оптоволоконної лінії зв’язку.

Діагностика kubernetes-кластера на основі Raspberry PI передбачає тестування коректності його функціонування за визначеним розкладом або у випадку виникнення нештатних ситуацій, які характеризуються виходом з ладу апаратного забезпечення або програмних збоїв.

Перспективи розвитку kubernetes-кластера передбачають масштабованість апаратних компонентів кластера, зокрема нарощування кількості Raspberry PI, як вузлів для виконання обчислень. Окрім цього, передбачається додавання функціональності шляхом додаткової оптимізації розподілу навантаження на вузли кластера та візуалізації навантаженості вузлів в реальному часі.

Вимоги надійності до kubernetes-кластера передбачають безперебійність його роботи як при виконанні обчислень, так і у випадку режиму очікування. Це означає готовність та налаштовуваність усіх компонент до визначеного режиму функціонування. Окрім цього, важливим показником надійності є захищеність кластеру з точки зору зовнішнього програмного доступу та з локальної комп’ютерної мережі.

У випадку виникнення збою у роботі компонентів, відновлення працездатності не повинно займати багато часу, тобто кластер в цілому та усі його компоненти повинні бути ремонтопридатними.

Основна функція, яка покладається на kubernetes-кластер полягає у забезпеченні розгортання інфраструктури при виконанні обчислень або запуску користувацьких додатків на виконання з використанням Raspberry PI. Основне завдання, яке необхідно реалізувати у кваліфікаційній роботі – забезпечити можливість запуску сервера nginx, на якому може бути розгорнуте

користувацьке програмне забезпечення з оптимальним розподілом задач на виконання окремими вузлами Raspberry PI та захищеним публічним доступом з мережі Інтернет.

1.2 Аналіз функціональності kubernetes

Kubernetes представляє собою портативну, здатну до масштабованості платформу з відкритим вихідним кодом, що забезпечує управління контейнерами, службами та відповідними навантаженнями, які вони формують. Це дозволяє полегшити процес налаштування та автоматизації при використанні декларативного підходу. Він має велику екосистему, що швидко розвивається.

Сервіси, підтримка та інструменти Kubernetes широко доступні. Kubernetes поєднує понад 15 років досвіду Google у створенні робочих навантажень у масштабі з найкращими у своєму роді ідеями та практиками спільноти. Для того, щоб зрозуміти чому використання Kubernetes є важливим та необхідним на сьогодні, потрібно проаналізувати технології організації кластерів, які використовувались до цього часу [1].

На рис. 1.2 показано підходи, які використовуються при розгортанні програмного забезпечення.

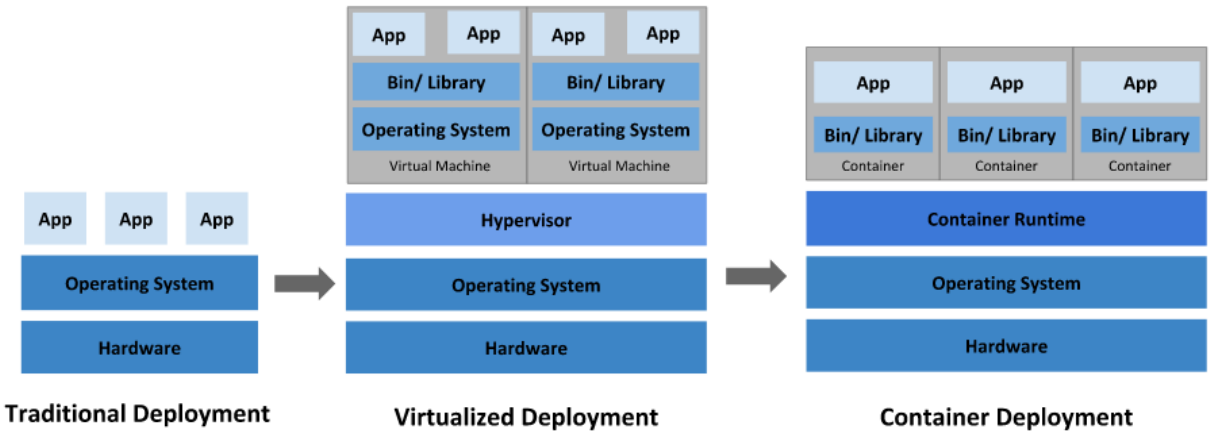

Рисунок 1.2 – Підходи до розгортання ПЗ

З самого початку, при виникненні необхідності функціонування програмного забезпечення на кількох комп'ютерах, тобто їх розгортанні на кластері, розробникам доводилось виконувати їх запуск на фізичних серверах. При цьому була відсутня можливість встановлення або визначення ресурсів, які необхідні програмному забезпечення для ефективного функціонування.. Відповідно така ситуація породжувала ряд проблем з розподілом ресурсів, які вимагали розробки нових рішень. Доволі часто виникали ситуації, коли на одному фізичному сервері потрібно виконувати декілька програмних додатків, які вимагають різних апаратних ресурсів. Однак, запустивши на виконання одну, яка використовує левову частку, наприклад, оперативної пам'яті чи потужності процесора, запустити інші було неможливо. Виходом з цієї ситуації було застосування іншого апаратного сервера, а це в свою чергу значно підвищувало вартість експлуатації програмного забезпечення при зростанні неефективності розподілу фізичних ресурсів [2].

Наступний етап розвитку у сфері інформаційних технологій характеризується застосуванням технологій віртуалізації. Такі технології забезпечують можливість одночасного запуску та функціонування декількох virtual machines на одному апаратному сервері. Перевагою технології віртуалізації є те, що вона дає змогу забезпечити ізольованість програмного додатку між різними віртуальними машинами та заданий рівень безпеки. Це зумовлено тим, що одночасно працюючі програми не мають можливості використання даних одна одної. Як наслідок, такий спосіб розподілу ресурсів на основі віртуалізації є більш ефективним у порівнянні з попереднім і дає змогу підвищити рівень масштабованості [2-4].

Окрім наведених вище переваг віртуалізації, ще одна полягає у забезпеченні можливості представлення набору фізичних ресурсів у вигляді кластера віртуальних машин. Кожен елемент кластеру представляється у вигляді повнофункціональної машини, що забезпечує роботу усіх компонентів, які включають операційну систему, що функціонує на основі програмно визначених апаратних пристроїв.

Наступний етап – розгортання контейнерів, що є розвитком технології віртуалізації. Функціонально контейнери подібні до віртуальних машин, однак основна відмінність полягає у тому, що вони володіють більш послабленими властивостями ізоляції при спільності використання операційної системи програмними додатками. Це є перевагою над віртуальними машини, оскільки контейнери є більш легкими. Проводячи аналогію з віртуальними машинами, до складу кожного контейнера входить власна файлова система, виділена частина потужності центрального процесора, оперативної пам'яті, простору зберігання даних тощо. Контейнери відокремлюються від базової інфраструктури, і можуть переноситися між хмарними сервісами чи окремими дистрибутивами операційної системи.

До переваг застосування контейнерів у порівнянні з віртуальними машинами можна віднести:

– гнучкість побудови та розгортання програмного забезпечення, що проявляється у підвищенні ефективності та спрощенні формування образів контейнерів;

– підтримка принципів безперервності розробки, інтеграції та розгортання, що дає змогу забезпечити надійність і доволі часте формування і розгортання образу контейнера.

– розділення розробки і операційних функцій – передбачає створення образів контейнерів потрібного програмного забезпечення у процесі його випуску, таким чином розділяючи реалізацію бізнес-логіки додатку від інфраструктури.

– забезпечення моніторингу – дозволяє не тільки відображати інформацію та показники рівня операційної системи, але й продуктивність програмного забезпечення.

– підтримка послідовності процесів під час реалізації програмного забезпечення – формує враження про однотипність функціонування як на локальній машині, так у хмарі.

– переносимість та підтримка міграції – забезпечує можливість однакового функціонування і поведінки на різних операційних системах як в хмарі так і на локальних машинах.

– управління з орієнтацією на програмний додаток – забезпечує зростання рівня абстракції, починаючи від старту операційної системи на віртуальних апаратних пристроях до безпосереднього запуску програмного додатку, використовуючи при цьому логічні ресурси.

– підтримка принципів мікросервісної архітектури – програмне забезпечення розглядається як окремі незалежні частини, які можна швидко розгортати та виконувати управління ними, а не у вигляді моноліту, який функціонує на одній машині.

– ізольованість ресурсів – забезпечує очікувану продуктивність роботи програмного додатку.

– ефективність використання ресурсів.

У результаті розгортання Kubernetes одержується кластер, до складу якого входить сукупність «робочих машин» – вузлів, призначених для запуску контейнеризованих програмних додатків.

Характерною особливістю кластера є те, що він повинен містити хоча б один робочий хост. На вузлах кластера поміщаються модулі, які є компонентами, що формують робоче навантаження програмного додатку.

Для управління вузлами і модулями у кластері використовується компоненти контролю планування. У виробничих середовищах рівень керування зазвичай працює на кількох комп’ютерах, а кластер зазвичай запускає кілька вузлів, забезпечуючи відмовостійкість і високу доступність. Типова схема організації Kubernetes показана рис. 1.3.

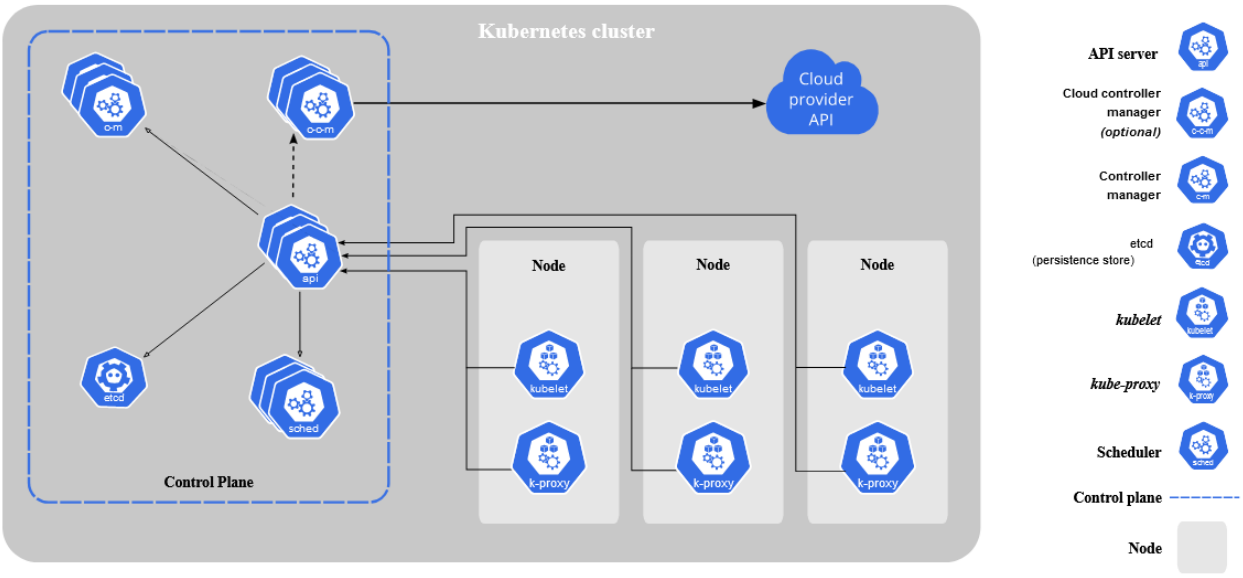

Рисунок 1.3 – Схема організації Kubernetes

Компоненти рівня керування відповідають за прийняття глобальних рішень, які стосуються кластера, а також виявляють події у ньому і реагують на них. Компоненти керування плануванням можна запускати з будь-якої машини кластера. Проте з метою забезпечення зручності і простоти налаштування сценаріїв, усі такі компоненти керування краще запускати на одній машині, але водночас не рекомендується на ній виконувати контейнерів користувача.

Для забезпечення доступу до прикладного програмного інтерфейсу Kubernetes застосовується kube-apiserver сервер API. Даний сервер виступає в ролі фронт-енду при керуванні Kubernetes, а найбільш важливою його реалізацією є kube-apiserver. Даний компонент створено з метою масштабування по горизонталі, тобто передбачається більша кількість екземплярів, які розгортаються. Існує можливість запуску декількох екземплярів kube-apiserver та балансування трафіку між ними [5].

Etcd представляє собою послідовне та високодоступне сховище ключових значень, що застосовується в якості резервного місця зберігання для усіх даних кластера. У випадку, коли Kubernetes використовує etcd як резервне сховище, переконайтеся, що у вас є план резервного копіювання даних.

Kube-планувальник є компонентом системи керування плануванням, який виконує функції моніторингу за новоствореними контейнерами без призначеного вузла та вибирає вузол для їх запуску.

Для запуску процесів Kubernetes використовує kube-controller-manager. Даний компонент виконує запуск процесів контролера. Кожен контролер представляється окремим процесом, однак для зниження складності усі наявні контролери компілюються в єдиний двійковий файл, а їх виконання проходить в одному процесі. У kubernetes визнчено декілька типів контролерів:

– контролер вузлів – основна функція контролерів полягає у маркуванні та генерації реакції у випадку, коли вузли виходять з ладу.

– контролер завдань – стежить за об’єктами Job, які представляють одноразові завдання, а потім створює модулі для виконання цих завдань до кінця.

– контролер EndpointSlice – заповнює об’єкти EndpointSlice (для забезпечення зв’язку між службами та модулями).

– контролер ServiceAccount – створєю облікові записи ServiceAccount за замовчуванням для нових просторів імен.

– хмарний контролер-менеджер – вбудовує спеціальну хмарну логіку управління кластером.

Хмарний контролер-менеджер дозволяє підключити кластер до API провайдера хмарних послуг та відокремлює компоненти, які взаємодіють із цією хмарною платформою, від компонентів, які мають відношення лише до кластера. Якщо Kubernetes використовується локально, наприклад в навчальному середовищі на власному ПК, то кластер не має диспетчера хмарного контролера. У випадку побудови kubernetes-кластера на основі Raspberry PI не передбачається використання цього типу контролера. Провівши аналіз вимог до організації кластера на основі Raspberry PI та дослідивши принцип функціонування та організації kubernetes далі необхідно перейти до безпосередньої його реалізації та налаштування на відповідному обладнанні.

РОЗДІЛ 2 ПРОЕКТУВАННЯ АПАРАТНО-ПРОГРАМНОЇ ІНФРАСТРУКТУРИ KUBERNETES-КЛАСТЕРА

2.1 Проектування архітектури інфраструктури kubernetes-кластера

У кваліфікаційній роботі потрібно створити кластер kubernetes, використовуючи пристрої Raspberry Pi як основу обчислювальної платформи.

Kubernetes – популярна система для автоматизації розгортання, оркестрації контейнерів з функціями:

– масштабування,

– балансування навантаження,

– керування томами,

– організацією безпеки та інші.

Зважаючи на такий широкий спектр функціональності його можна вважати kubernetes екосистемою.

У роботі на апаратному рівні буде використовуватися тільки дешеве обладнання, яке є в будь-якому магазині комп’ютерної техніки: маршрутизатор, комутатор і сервер зберігання.

Оскільки kubernetes покриває 90% потреб у програмному забезпеченні, все ще можна використовувати деякий ansible для автоматизації; сучасні маршрутизатори, комутатори та сервери зберігання даних працюють на базі операційної системи, схожої на UNIX, тому її також варто застосувати у проекті.

Зрештою, кластер kubernetes повинен працювати як хмарна інфраструктура, де можна запускати програми у хмароподібному середовищі, отримувати до них доступ з мережі Інтернет, програми зі збереженням стану матимуть можливість використовувати рівень зберігання даних.

Перше на чому потрібно зосередитися стосується таких на основних моментів:

– вимоги до пристрою Raspberry Pi;

– підготовка вузла kubernetes;

– налаштування кластера.

В якості комутаційного обладнання пропонується використати маршрутизатор Microtik hEX S(RB760iGS). За $69,00 він пропонує широкий вибір функцій, які зазвичай доступні лише на маршрутизаторах високого класу від таких постачальників, як Cisco, Juniper та інших. Один порт маршрутизатора використовується для підключення до Інтернет-провайдера. Інший порт з'єднує маршрутизатор з комутатором.

Комутатор D-Link DGS-1005 P коштує менше 50 доларів, він має п'ять портів Gigabit Ethernet. Чотири порти підтримують живлення через Ethernet (PoE). Порт 5 використовується для підключення до маршрутизатора. Такий тип зв’язку називається висхідним зв’язком. Інші чотири порти можна використовувати для підключення до пристроїв Raspberry Pi.

Завдяки PoE на портах пристрої Raspberry Pi не потребують живлення через USB. Оскільки кластер Kubernetes ділиться мережею з іншим обладнанням, то доцільно ізолювати його у VLAN. Це не обов’язково, однак є питанням зручності. Основними компонентами інфраструктури, частиною якої є kubernetes-кластер:

– Mikrotik RB760iGS — маршрутизатор, що забезпечує кластер Kubernetes підключенням до Інтернету, комутацією VLAN, сервером DHCP, NAT, динамічним DNS та іншими

– D-Link DGS-1005P — некерований комутатор з 4 портами PoE, який забезпечує Raspberry Pi комутацією живлення та канального рівня з маршрутизатором і вузлами Kubernetes

– Raspberry Pi 3 Model B+ — міні-комп’ютер із 64-розрядним чотирьохядерним процесором 1,4 ГГц, 1 ГБ оперативної пам’яті, Gigabit Ethernet Raspberry Pi PoE

– HAT — розширення Raspberry Pi, яке дозволяє живити пристрій через Ethernet microSDHC MIREX (class10) 8GB — SD-карта з операційною системою для Raspberry Pi.

– microSDHC MIREX (class10) 8GB — SD-карта з операційною системою для Raspberry Pi

На рис. 2.1 показано схему організації kubernetes-кластера, який реалізується у кваліфікаційній роботі.

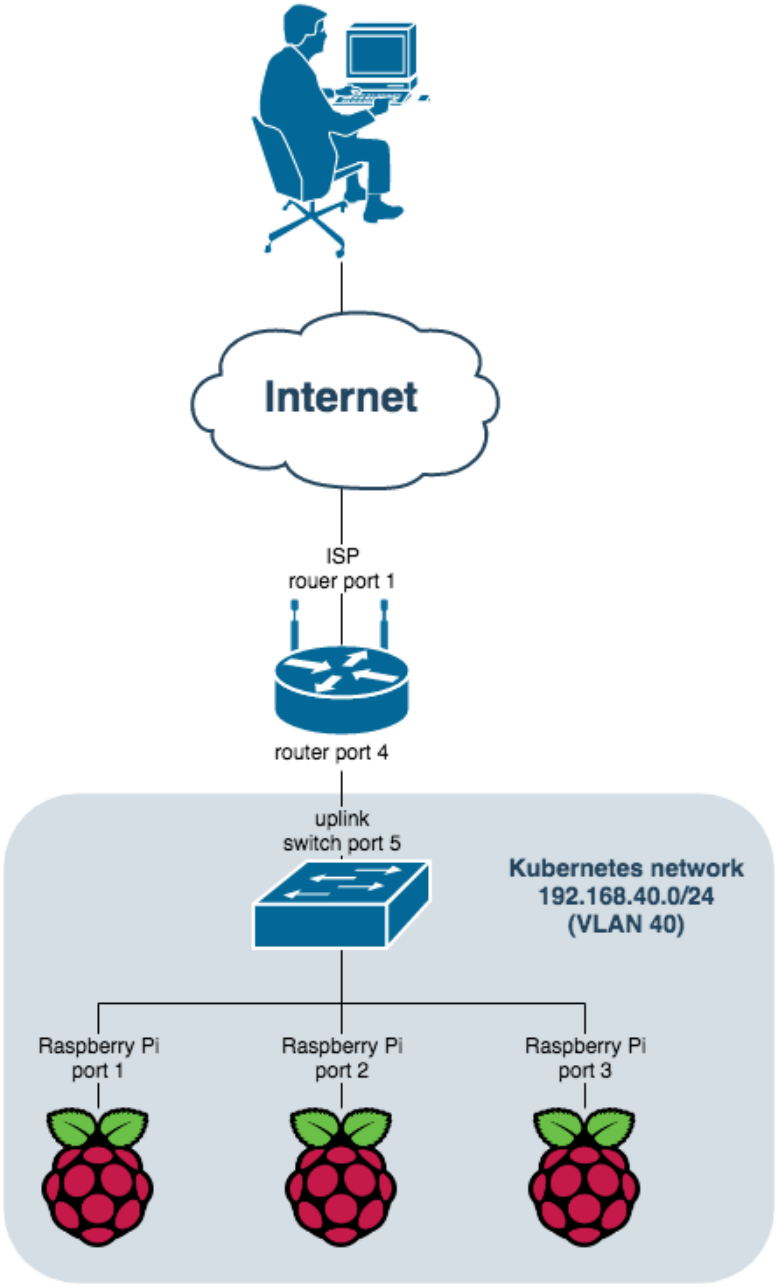

Рисунок 2.1 – Організація комунікаційної інфраструктури з kubernetes-кластером

На рис. 2.2 показано підключення між маршрутизатором, некерованим комутатором та трьома Raspberry PI, на яких буде розгорнути обчислювальний кластер.

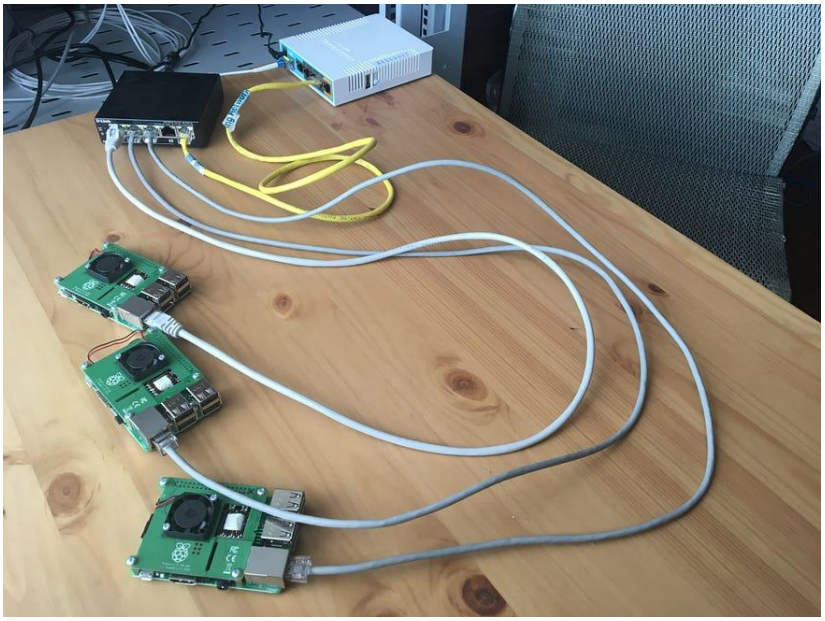

Рисунок 2.2 – Схема організації фізичного з’єднання пристроїв

На рис. 2.2 маршрутизатор – пристрій у білому корпусі, комутатор — у чорному. Жовтий кабель – висхідна лінія зв’язку (перемикач на підключення до маршрутизатора). Сірі кабелі з’єднують пристрої Raspberry Pi із комутатором.

2.2 Аналіз особливостей hEX S (RB760iGS)

RB760iGS (hEX S) представляє собою маршрутизатор, що містить 5 портів з підтримкою швидкості передачі даних 1 Гб/с і призначений для застосування у випадку, коли не потрібно використовувати безпровідне з’єднання.

На відміну від hEX (RB750Gr3) маршрутизатор hEX S додатково володіє слотом SFP, має підтримку PoE out на 5-му порті та забезпечує більшу функціональність. На рис. 2.3 показано зовнішній вигляд маршрутизатора RB760iGS [7].

Рисунок 2.3 – Маршрутизатор RB760iGS

Mikrotik RB760iGS працює на основі процесора MT7621A з тактовою частотою 880 МГц, до складу якого входить два фізичних ядра з чотирма логічними потоками. Максимальний розмір оперативної пам'яті становить 256 МБ. Такі характеристики забезпечують високу продуктивність за будь-якого сценарію використання, будь то гігабітне підключення або навіть робота з IPsec.

Наявність слота SFP 1.25G дозволяє здійснювати пряме підключення до оптоволокна без необхідності використання додаткового конвертера. Оскільки сам пристрій не оснащений підтримкою бездротової мережі, наявність підтримки PoE Out на інтерфейсі ether5 дозволяє розширити функціонал за рахунок застосування окремої точки безпровідного доступу, що живиться від пасивного PoE. Подібна комбінація дозволяє скоротити кількість дротів до мінімуму [8].

Маршрутизатор RB760iGS також підтримує PoE і може бути заживлений за стандартом Passive PoE або 802.11at/af. Але і на цьому функціонал пристрою не обмежується, адже у розпорядженні hEX S містить інтерфейс з відповідними портами USB2.0 та слот microSD.

Інтерфейс USB можна використовувати для 3G/4G/LTE модемів або зовнішнього накопичувача, у той час як на карті microSD можна налаштувати збереження системних журналів, розмістити файли HotSpot або взагалі, використовувати для The Dude.

Здатність підтримки USB та microSD є доволі високою перевагою на тлі відносно невеликого обсягу постійної пам'яті. Як і в інших подібних пристроях, RB760iGS володіє лише 16 МБ flash-пам'яті.

Діапазон вхідної напруги збільшений до 57 В, що дозволить застосовувати блоки з напругою 48В або використовувати 802.11af/at. До того ж, зі збільшенням напруги, можна підвищити вихідну потужність PoE. Це особливо актуально з урахуванням підтримки PoE Out [8].

Кнопка скидання, індикатори PWR та USR, слот microSD були прибрані з передньої панелі, їхнє місце зайняв слот SFP. Самі ж індикатори USR і PWR були перенесені на верхню панель, тепер вони розташовані поруч із індикаторами інтерфейсів. Слот microSD перекочував на бічну поверхню, поряд із портом USB, туди ж перекочувала кнопка скидання. Mikrotik опублікували 2 діаграми, які роз'яснюють схему підключення інтерфейсів hEX S (рис. 2.4. та рис. 2.5).

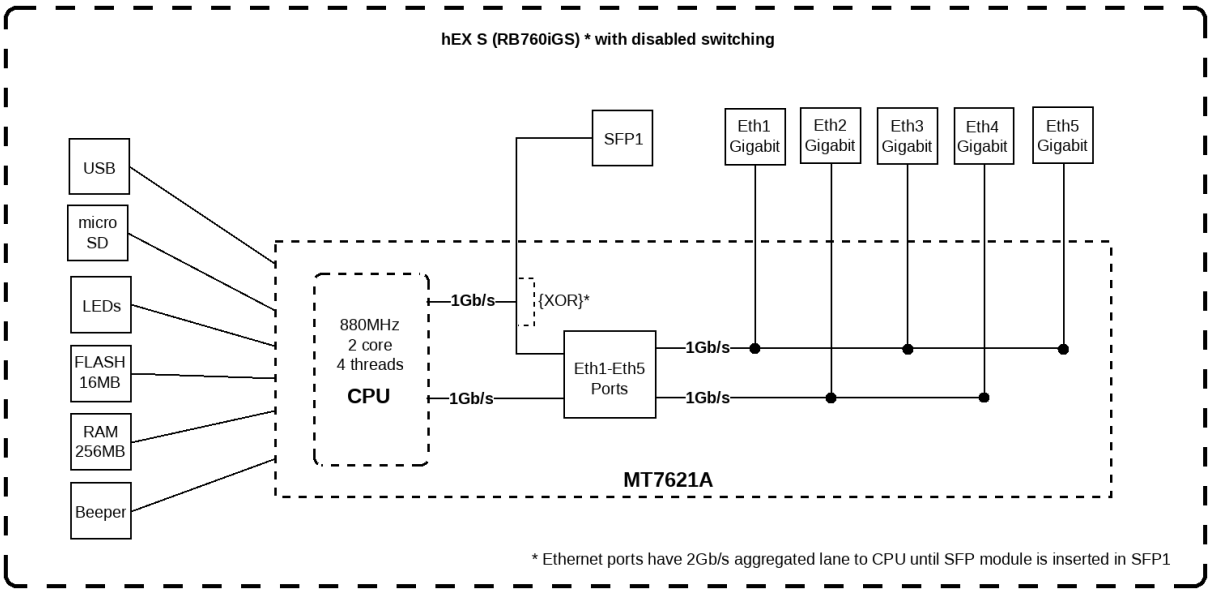

Рисунок 2.4 – Діаграма підключення інтерфейсів RB760iGS з відключеною комутацією

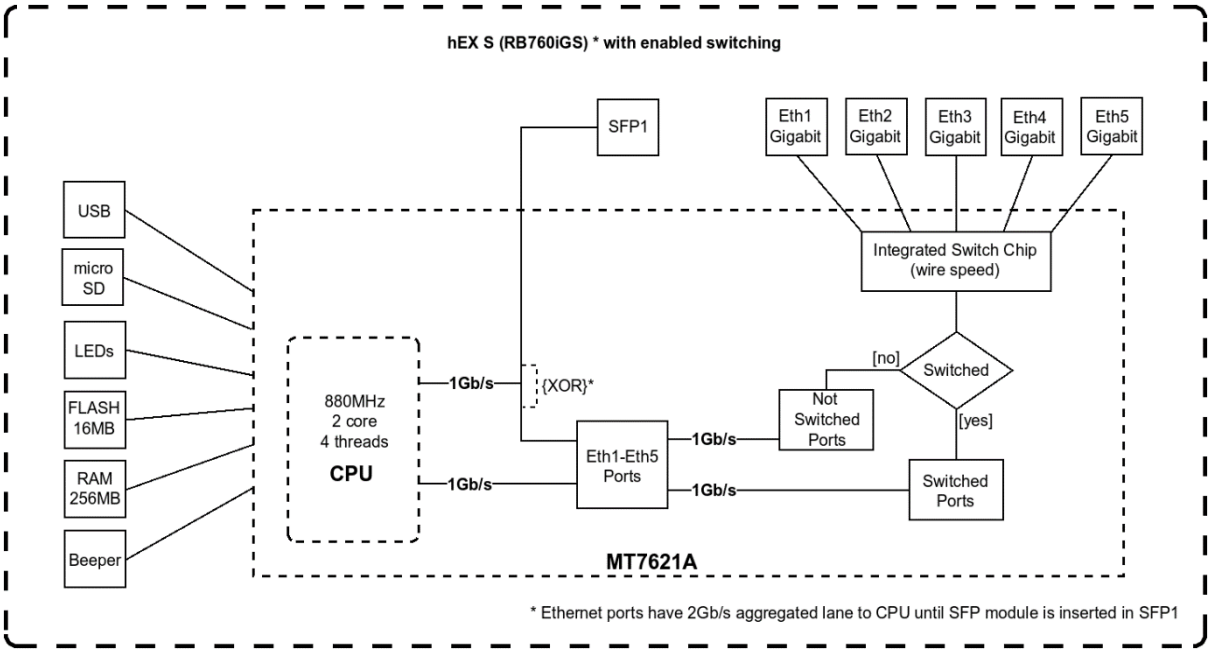

Рисунок 2.4 – Діаграма підключення інтерфейсів RB760iGS з підключеною комутацією

Як видно з рис. 2.4, при використанні модуля SFP група портів Ethernet втрачає половину пропускної здатності - вона зменшується до 1 Гбіт. Потрібно звернтуи увагу на те, що за відсутності SFP (і відключеній комутації), порти 1,3,5 і 2, 4 підключені окремо і, швидше за все, ядра процесора, так само як RB3011 з IPQ-8086. На рис. 2.5 показано функціональну блок-схему RB760iGS [8].

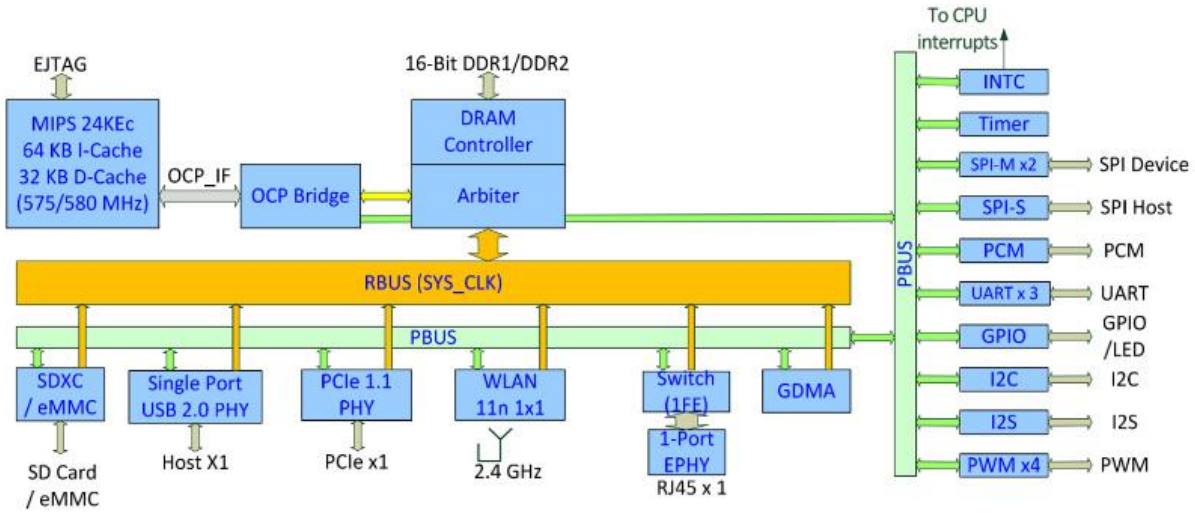

Рисунок 2.5 – Функціональна блок-схема RB760iGS

Виділити для SFP шину 1 Гбіт крок цілком логічний, але наскільки виправдано забирати частину шини у Ethernet-портів – невідомо. Якщо подивитися на схему MT7621A, можна зробити висновок, що для SFP можна виділити окрему шину, задіявши додатковий передавач з інтерфейсом PCIe або RGMII.

2.3 Технічні особливості некерованого комутатора DGS-1005P

DGS-1005P оснащено екологічними технологіями, такими як IEEE 802.3az Energy-Efficient Ethernet (EEE), визначення стану з'єднання та визначення довжини кабелю. EEE регулює енергоспоживання комутатора на основі використання мережі, знижуючи вартість експлуатації під час періодів бездіяльності. Виявлення статусу з'єднання виключає живлення портів, якщо з'єднання не виявлено, заощаджуючи енергію, коли підключені пристрої не працюють або відключаються. Виявлення довжини кабелю регулює вихідну потужність порту на базі довжини кабелю, зменшуючи вимоги до живлення комутатора. Компактний дизайн DGS-1005P дозволяє розміщувати його у більшості місць, у тому числі в місцях, де мало простору. На рис. 2.6 показано зовнішній вигляд комутатора DGS-1005P [9].

Рисунок 2. 6 – Комутатор DGS-1005P

Форм-фактор настільного комп'ютера забезпечує все необхідне для створення нової мережі або розширення існуючої, а безвентиляторна конструкція забезпечує безшумну роботу навіть у чутливих до шуму зонах, таких як робочі станції та конференц-зали.

Основні технічні характеристики комутатора [9] наведено у табл. 2.1.

Таблиця 2.1 – Технічні характеристики DGS-1005P

Характеристика	Значення характеристики
Flash-пам’ять	1 КБ
Інтерфейси	5 портів 10/100/1000Base-T (4 порти з підтримкою PoE)
Індикатори	Power
	PoE Max
	Link/Activity/Speed (на порт)
	PoE OK/PoE Fail (на порт PoE)
Роз’єм живлення	Роз’єм для підключення адаптера живлення (постійний струм)
Функціональність	
Стандарти і функції	IEEE 802.3 10Base-T
	IEEE 802.3u 100Base-TX
	IEEE 802.3ab 1000Base-T
	IEEE 802.3az Energy Efficient Ethernet
	Управління потоком IEEE 802.3x
	Автоматична узгодженість швидкості і режиму дуплексу
	Автоматичне визначення MDI/MDIX на всіх портах
	IEEE 802.1p QoS (8 черг)

Продовж. табл. 2.1

Характеристика	Значення характеристики
Швидкість передачі даних	Ethernet: 10 Мбіт/с (напівдуплекс) / 20 Мбіт/с (повний дуплекс)
	Fast Ethernet: 100 Мбіт/с (напівдуплекс) / 200 Мбіт/с (повний дуплекс)
	Gigabit Ethernet: 2000 Мбіт/с (повний дуплекс)
Продуктивність	
Комутаційна матриця	10 Гбіт/с
Метод комутації	Store-and-forward
Макс. швидкість перенаправлення 64-байтних пакетів	7,44 Mpps
Розмір таблиці MAC-адрес	2K записів
Буфер пакетів	192 КБ
Jumbo-фрейм	9 216 байт
PoE	
Стандарт PoE	IEEE 802.3af
	IEEE 802.3at
Порти з підтримкою PoE	Порти 1-4
Потужність PoE	60 Вт (макс. 30 Вт на порт PoE)

2.4 Особливості технічних характеристик Raspberry Pi при організації kubernets-кластера

Raspberry PI є одним з найбільш популярних мінікомп'ютерів, які використовуються як при організації простих IoT рішень, так і при побудові складних комп'ютерних систем. У кваліфікаційній роботі передбачається організація обчислювального kubernetes-кластера на базі трьох Raspberry Pi

Model B. Характерною особливістю цього мінікомп'ютера є те, що його габаритні розміри відповідають розмірам банківської карти, його можна під'єднувати до практично будь-якого пристрою виводу інформації, а також підключати типові пристрої вводу – клавіатура та миша з USB інтерфейсом. Raspberry PI є хорошим програмно-апаратним пристроєм при вивченні основ програмування, прототипуванні проектів в електроніці, застосуванні в якості персонального комп'ютера при формуванні електронних таблиць, роботі з текстом, організації доступу до мережі Інтернет та ін. На рис. 2.7 показано зовнішній вигляд Raspberry PI 3 з наявними інтерфейсами та портами [10].

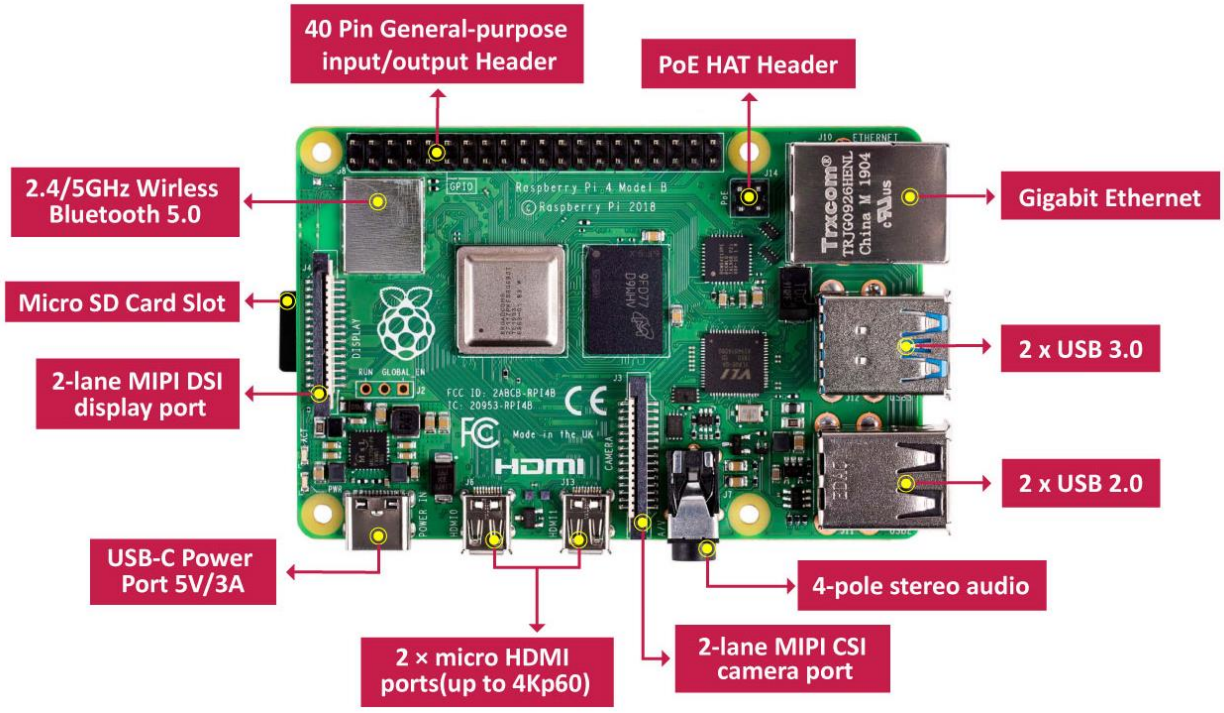

Рисунок 2.7 – Типовий вигляд Raspberry PI

Зважаючи на різноманітність застосування мінікомп'ютерів та їх еволюцію, існує багато варіантів вибору версій Raspberry PI. Основні технічні характеристики різних моделей даного мінікомп'ютера приведено у табл. 2.2. При проектуванні та організації kubernetes-кластера пропонується використати версію Raspberry PI 3 Model B.

Таблиця 2.2 – Різновиди Raspberry PI

Модель	SoC	Тактова частота, МГц	RAM	Технології Ethernet	Wireless	Blue-tooth
Raspberry Pi Model A+	BCM2835	700	512МБ	-	-	-
Raspberry Pi Model B+	BCM2835	700	512МБ	100Base-T	-	-
Raspberry Pi 2 Model B	BCM2836/7	900	1 ГБ	100Base-T	-	-
Raspberry Pi 3 Model B	BCM2837 A0/B0	1200	1 ГБ	100Base-T	802.11n	4.1
Raspberry Pi 3 Model A+	BCM2837 B0	1400	512МБ	-	802.11 ac/n	4.2
Raspberry Pi 3 Model B+	BCM2837 B0	1400	1 ГБ	1000Base-T	802.11 ac/n	4.2
Raspberry Pi 4 Model B	BCM2711	1500	2 ГБ	1000Base-T	802.11 ac/n	5.0
Raspberry Pi 4 Model B	BCM2711	1500	4 ГБ	1000Base-T	802.11 ac/n	5.0
Raspberry Pi 4 Model B	BCM2711	1500	8 ГБ	1000Base-T	802.11 ac/n	5.0
Raspberry Pi Zero	BCM2835	1000	512МБ	-	-	No
Raspberry Pi Zero W	BCM2835	1000	512МБ	-	802.11n	4.1
Raspberry Pi 400	BCM2711	1800	4ГБ	1000Base-T	802.11 ac/n	5.0

Інтерфейс GPIO у Raspberry PI забезпечує зв'язок між його складовими компонентами, а відповідні виводи призначені для підключення зовнішніх периферійних пристроїв. Загалом інтерфейс GPIO може забезпечувати функціональність за трьома групами:

– забезпечення периферійних пристроїв живленням необхідного номіналу – може подаватися напруга з рівнями 3,3 В або 5 В в залежності від типу пристрою, який підключається;

– заземлення – дає змогу заземлити пристрій;

– передавання сигналів – забезпечує обмін сигналами управління між Raspberry PI та периферійними пристроями.

Особливістю інтерфейсу GPIO є те, що один і той самий вивід може бути як вхідним, так і вихідним контактом, а логіка їх функціонування визначається на програмному рівні [11]. Схема та призначення контактів GPIO показана на рис. 2.8.

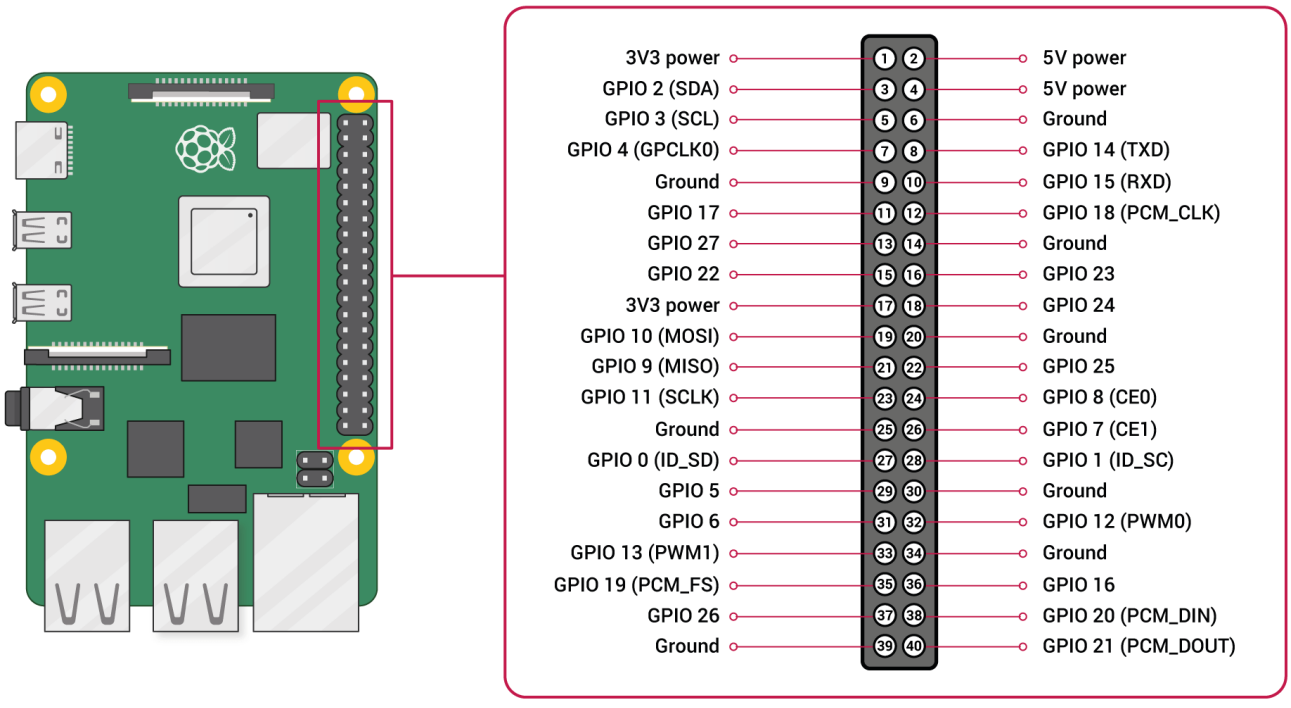

Рисунок 2.8 – Виводи GPIO

Незважаючи на те, що будь-який вивід інтерфейсу GPIO у Raspberry PI може працювати як на вхід так і на вихід, все ж існує пара виводів особливого

призначення. Це стосується виводів промаркованих як BCM0 і BCM1 (виводи №27 і №28), що застосовуються для підключення плат розширення. Їх не варто використовувати як контакти для під'єднання інших модулів. Окрім цього, важливо обережно застосовувати живлення через інтерфейс GPIO, оскільки максимальна сила струму, яку можуть витримувати контакти, становить 50 мА. Деталізована схема цифрового інтерфейсу Raspberry PI та потенційні можливості щодо використання продемонстровані на рис. 2.9.

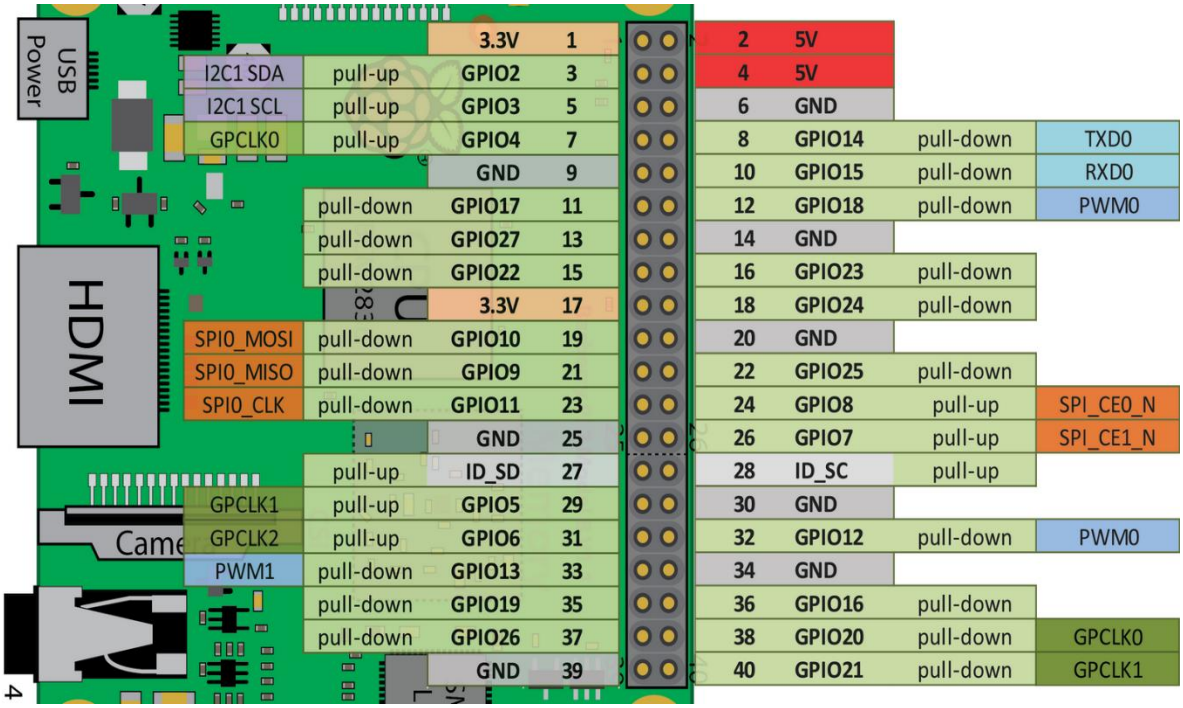

Рисунок 2.9 – Деталізована схема виводів цифрового інтерфейсу Raspberry PI

Управління виводами GPIO забезпечується на програмному рівні Raspberry PI. Мови програмування, які реалізують алгоритми роботи з пристроями та Raspberry PI можуть бути як об'єктно-орієнтованими, наприклад, C++ або PHP, а можуть бути, наприклад, Python, C або навіть Basic.

Схему електричну принципову інтерфейсу GPIO продемнстровано на рис. 2.10.

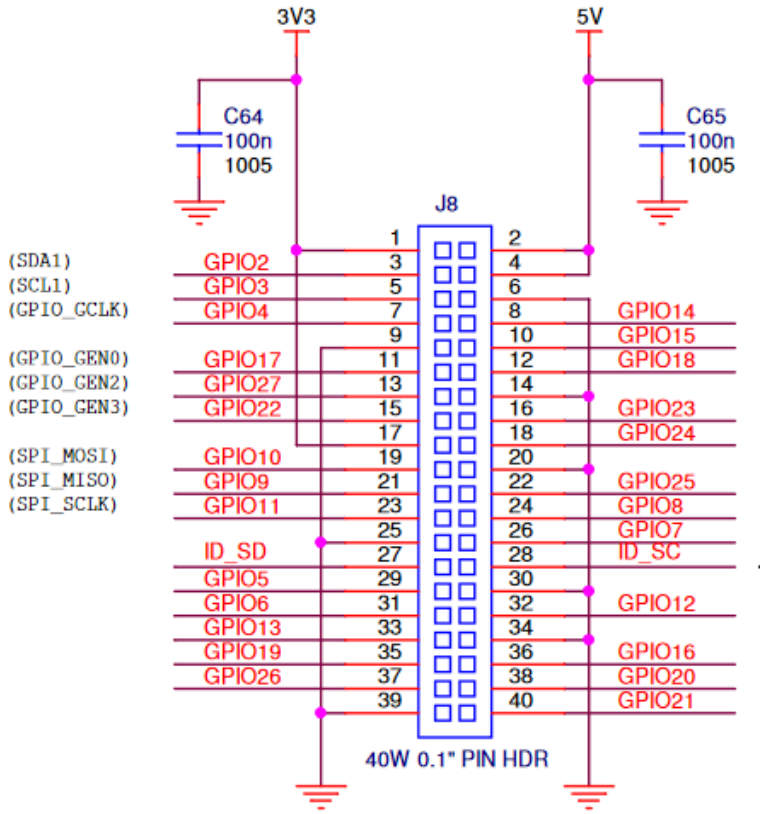

Рисунок 2.10 – Схема електрична принципова GPIO Raspberry PI Model B

До основних технічних характеристик Raspberry PI 3 Model B належать:

– чотириядерний 64-бітний процесор Broadcom BCM2837 на базі ARMv7 Quad Core Processor вбудований в однокристальний Single Board Computer, що працює з частотою 1.2GHz

– оперативна пам’ять – 1GB RAM;

– вбудований WiFi модуль BCM 43143;

– вбудований Bluetooth Low Energy;

– інтерфейс GPIO з 40 виводами;

– 4xUSB 2.0 порти;

– аудіо вихід і композитний відеопорт (CVP);

– HDMI порт (Full size) – для підключення монітора або дисплея;

– порт CSI – для під’єднання Raspberry Pi камери;

– DSI порт – для підключення сенсорного дисплею;

– Micro SD порт – для інсталяції операційної системи і зберігання даних;

– оновлений блок живлення Micro USB з підтримкою 2,4 Amps.

2.5 Програмне забезпечення kubernetes-кластера

Почнемо зі списку всіх програмних компонентів, які будуть використані при побудові kubernetes-кластера. До його складу входять:

– Ubuntu 18.04 – операційна система;

– Docker Engine – середовище виконання контейнера;

– Kubelet – «агент вузла», який працює на кожному вузлі кластера;

– Kubeadm – інструмент для створення кластерів;

– Kubernetes kubectl – інтерфейс командного рядка для запуску команд у кластерах.

Kubernetes має значно більше компонентів, зокрема, kube-apiserver, kube-controller-manager, kube-proxy та незліченну кількість додатків.

Для отримання детальної інформації можна звернутися до офіційної документації Kubernetes Components. У даному випадку kubeadm встановить їх після ініціалізації кластера Kubernetes.

Ubuntu 18.04 має збірку ARM 64 і навіть готовий образ ОС для Raspberry Pi. Google створює пакети deb за допомогою Kubernetes для ARM 64 [13].

У даному випадку, буде використовуватися Docker Engine для середовища виконання контейнера. Незважаючи на те, що є альтернативи в rkt, cri-o та інших. Однак, якщо придивитися ближче, то можна побачити, що Kubernetes використовує контейнер. Kubelet є агентом вузла. Він спілкується з kube-apiserver на головному вузлі та, з іншого боку, із середовищем виконання локального контейнера, щоб запускати контейнери Docker як головні команди. На рис. 2.11 показано компоненти вузла Kubernetes.

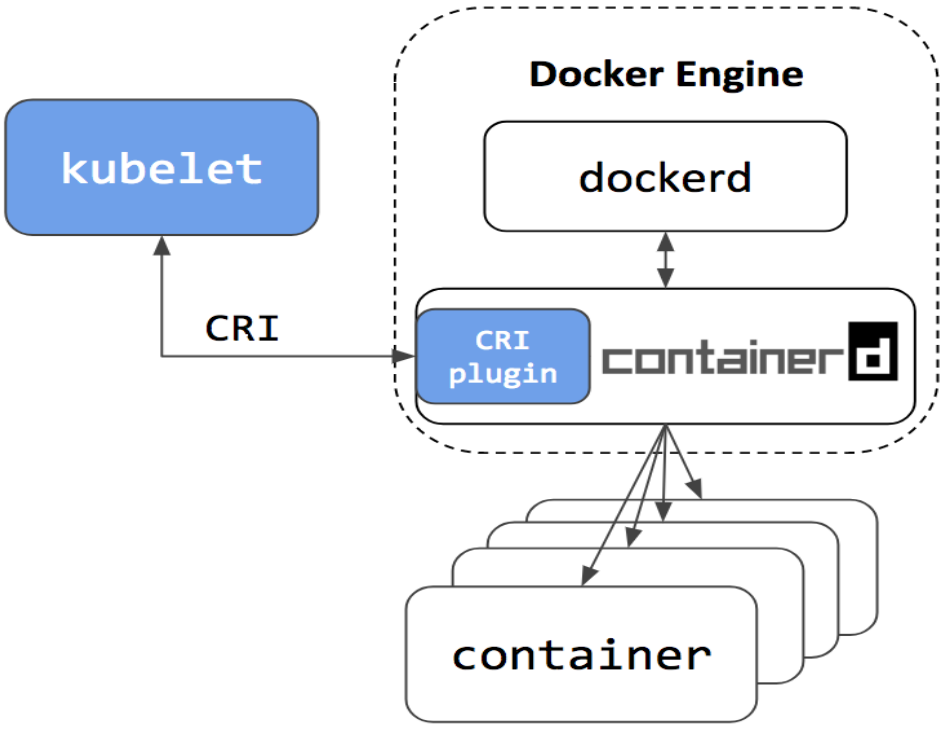

Рисунок 2.11 – Компоненти вузла Kubernetes

Kubectl – це інтерфейс командного рядка для запуску команд у кластерах Kubernetes. Він може створювати/читати/оновлювати/видаляти ресурси в kubernetes. Приклад застосування kubectl показано на рис. 2.12.

```
$ kubectl create -f deployment.yaml
$ kubectl create secret generic db-pass --from-file=./password.txt
$ kubectl get pods
$ kubectl --namespace=kube-system edit deployment coredns
$ kubectl delete node node1.kube.local
```

Рисунок 2.12 – Приклад застосування kubectl

Kubeadm – це інструмент для створення кластерів Kubernetes. У простому випадку – команда init kubeadm створює головний вузол kubernetes; Команда kubeadm join приєднує вузол до кластера, асоціюючи його з головним вузлом. Наступні кроки щодо організації kubernetes-кластера представлені у розділі 3.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ KUBERNETES-КЛАСТЕРА НА RASPBERRY PI

3.1 Базові налаштування кластера

Перед тим, як безпосередньо розпочати налаштування kuberenets-кластера необхідно, щоб на SD-карту була встановлена операційна система Ubuntu 18.04, а сама карта вставлена у пристрій і всі необхідні кабелі підключено, як показано на рис. 2.2. Для автоматизації розгортання інфраструктури кластера пропонується використовувати інструмент ansible.

На перший погляд Kubernetes – складна система. На щастя, вся складність прихована за кількома надійними та добре оптимізованими компонентами; наприклад kubelet і середовище виконання контейнерів.

З іншого боку, необхідно приділити додаткову увагу плануванню мережі. Під час реєстрації головного (master) сервера вузол kubernetes надає йому деяку назву. Ім’я має бути унікальним і доступним для розпізнавання (система доменних імен). У хмарних інфраструктурах, таких як Amazon Web Services, Google Cloud Platform, Microsoft Azure та інших – це рідна функція.

Будь-яка віртуальна машина має унікальне назву хоста, яке можна розпізнати; наприклад, «`ip-12-34-56-78.us-west-2.compute.internal`», який розпізнається як IP-адреса 12.34.56.78.

У даному випадку пристрої Raspberry Pi за замовчуванням мають схожі імена хостів, що потрібно виправити. Є кілька способів зробити це. Найдосконалішою була б інтеграція сервера DHCP (Dynamic Host Configuration Protocol) із сервером DNS (Domain Name System), але це може бути надто складним для налаштування, яке повинно відповідати критеріям простоти.

Отже, щоб вирішити цю проблему, необхідно виконати наступні дії:

– оновити назву кожного хоста за допомогою інструмента командного рядка hostnamectl (наприклад, «*hostnamectl set-hostname node1.kube.local*»)

– додати кожен запис для кожного хоста у файлі /etc/hosts.

Приклад, визначення хостів Raspberry Pi показано на рис. 3.1 [14].

```
192.168.40.102 master1.kube.local
192.168.40.103 node1.kube.local
192.168.40.104 node2.kube.local
```

Рисунок 3.1 – Визначення хостів кластера

У випадку автоматизованого налаштування та розгортання за допомогою ansible, завдання виглядають таким чином, як показано на рис. 3.2.

```
- name: Update hosts file
  lineinfile:
    dest: /etc/hosts
    line: "{{ hostvars[item].ansible_host }} {{ item }}"
    regexp: '.*{{ item }}$'
  with_items: "{{ groups.all }}"

- name: Update host name
  hostname:
    name: "{{ inventory_hostname }}"
```

Рисунок 3.2 – Фрагмент скрипта автоматизованого визначення хостів kubernetes-кластера за допомогою ansible

Групи керування Linux (cgroups) є важливою частиною контейнерів та екосистеми kubernetes. Без належної конфігурації все працюватиме не так, як очікується. З якоїсь причини підсистема групи керування пам’яттю вимкнена за замовчуванням у збірці Raspberry Pi Ubuntu 18.04. Щоб виправити це потрібно додати наступний рядок, як показано на рис. 3.3.

```
cgroup_enable=cpuset cgroup_memory=1 cgroup_enable=memory
```

Рисунок 3.3 – Команда увімкнення групи контролю оперативною пам’яттю

Команду, показану на рис. 3.3 необхідно запустити у рядку cmd ядра Linux на кожному пристрої Raspberry PI. У Ubuntu 18.04 файл /boot/firmware/cmdline.txt встановлює необхідний параметр для Raspberry Pi. Для автоматизації цього процесу сформовано наступне завдання у вигляді скрипта ansible, як показано на рис. 3.4. Після перезавантаження пристроїв, kubernetes має всі необхідні налаштування контрольних груп.

```
- name: Enable memory control group subsystem
  replace:
    # https://wiki.ubuntu.com/ARM/RaspberryPi#Raspberry_Pi_packages
    path: /boot/firmware/cmdline.txt
    regexp: '(rootwait)$'
    replace: '\1 cgroup_enable=cpuset cgroup_memory=1
cgroup_enable=memory'
```

Рисунок 3.3 – Скрипт автоматизації увімкнення контрою керування пам’яттю Raspberry PI

Kuberntes не має багато програмних залежностей. По суті, все, що потрібно, це увімкнути офіційний репозиторій kubernetes apt (https://apt.kubernetes.io/). Після цього потрібно встановити пакети лише для головного (master) вузла: kubelet, kubeadm, docker, kubectl

Тепер, коли все підготовлено, необхідно увійти на головний вузол за допомогою безпечної оболонки (ssh) і виконати команду, яка наведена на рис. 3.4.

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

Рисунок 3.4 – Ініціалізація конфігурації кластера

Процес виконання команди, показаної на рис. 3.4 може зайняти деякий час, можливо, близько 15 хвилин. Протягом цього часу kubeadm збирає образи докерів для основних компонентів kubernetes, генерує сертифікати PKI та запускає служби.

Якщо команда виконана успішно, далі потрібно скопіювати конфігурацію адміністратора kubernetes у файл конфігурації kubelet, щоб увімкнути запуск команд у кластері Kubernetes (рис. 3.5).

```
mkdir ~/.kube
sudo cp /etc/kubernetes/admin.conf ~/.kube/config
sudo chown $(id -u):$(id -g) ~/.kube/config
```

Рисунок 3.5 – Додавання даних до файлу конфігурація kubernetes-кластера

У випадку успішного додавання даних до конфігураційного файлу, то відповідь kubernetes матиме відповідь, як показано на рис. 3.6 [15].

```
$ kubectl get node
NAME                STATUS     ROLES    AGE   VERSION
master1.kube.local  NotReady   master   1h    v1.14.2
```

Рисунок 3.6 – Приклад відповіді kubernetes на успішне додавання даних до файлу конфігурації

Коли налаштовано головний вузол кластера (master) і виконано його запуск, то перетворення Raspberry Pi на вузол kubernetes стає тривіальною операцією. На першому кроці необхідно отримати команду приєднання на головному вузлі за допомогою команди, що показана на рис. 3.7.

```
$ kubeadm token create --print-join-command
```

Рисунок 3.7 – Команда приєднання до головного вузла

Результат виконання команди (рис. 3.7) показано на рис. 3.8.

```
kubeadm join 192.168.40.102:6443 --token xh0b2k.56yrq79emc79ad7n
--discovery-token-ca-cert-hash
sha256:d4868076fbf3aa8dec5faa7add744350c9c0b3d9ba10a6ef09c08b42d57028
70
```

Рисунок 3.8 – Результат додавання вузлів до kubernetes-кластера

Останнім етапом створення kubernetes-кластера є виконання команди на кожному з його вузлів. Після цього потрібно виконати перевірку статусу вузлів, виконавши команду kubectl get node на головному вузлі (рис. 3.9).

```
$ kubectl get node
NAME                STATUS     ROLES    AGE   VERSION
master1.kube.local  NotReady   master   11h   v1.14.2
node1.kube.local    NotReady   <none>   10h   v1.14.2
node2.kube.local    NotReady   <none>   10h   v1.14.2
```

Рисунок 3.9 – Виконання команди перевірки вузлів кластера

Як видно з рис. 3.9, усі вузли підключено, але їхній статус NotReady, оскільки мережа контейнера не встановлена.

Контейнерна мережа є однією з найважливіших тем у налаштуванні кластера kubernetes. Залежно від обраної моделі мережі можна застосовувати або не застосовувати мережеві політики, масштабувати кластер до тисячі вузлів, увімкнути шифрування. Однак потрібно розуміти, що неможливо перемкнути мережу контейнерів на працюючому кластері, тому слід ретельно його планувати.

Одним з варіантів для вирішення питань налаштування політик безпеки, здатності до масштабування є використання плагіну Calico CNI. Calico має кілька переваг перед іншими мережевими плагінами. Зокрема, він має збірки ARM 64, може вмикати мережеві політики в kubernetes та є ресурсоефективним. Однак складність налаштування даного плагіну не забезпечило коректність

встановлення характеристик, тому застосовано просте та надійне рішення — Flannel, що є новим рішенням у мережах контейнерів для kubernetes. Застосування Flannel показано на рис. 3.10.

```
$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel
/v0.11.0/Documentation/kube-flannel.yml
```

Рисунок 3.10 – Підключення плагіну Flannel

У результаті перевірки коректності налаштування вузлів kubernetes-кластера та контейнерів Flannel, як видно з рис. 3.11, є працездатним.

```
$ kubectl --namespace="kube-system" get pod --selector="app=flannel"
NAME                          READY   STATUS    RESTARTS   AGE
kube-flannel-ds-arm64-6lnd4   1/1     Running   0          10h
kube-flannel-ds-arm64-g8ltj   1/1     Running   0          10h
kube-flannel-ds-arm64-ntr2j   1/1     Running   0          10h

$ kubectl get node
NAME                 STATUS   ROLES    AGE   VERSION
master1.kube.local   Ready    master   11h   v1.14.2
node1.kube.local     Ready    <none>   10h   v1.14.2
node2.kube.local     Ready    <none>   10h   v1.14.2
```

Рисунок 3.11 – Перевірка коректності налаштування kubernetes з плагіном Flannel

Модулі Flannel CNI втсановлені, а вузли kubernetes перебувають у стані готовності. Кластер Kubernetes готовий до першого розгортання.

3.2 Розгортання kubernetes-кластера

Існує багато способів розгорнути програмні додатки в kubernetes. У наведеному нижче фрагменті коду представлено зразок термінального сеансу, який створює веб-сервер nginx у кластері та відкриває його порт для локальної

мережі. Коротше кажучи, команда kubectl create deployment nginx створює контролер розгортання в кластері.

Планувальник Kubernetes створює пакет на основі визначення розгортання. Коли модуль nginx перебуває в стані Running, веб-сервер nginx готовий обслуговувати запити HTTP. Команда kubectl create service nginx створює службу kubernetes. Служба надає nginx pods як мережеву службу.

```
$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created

$ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   0/1     1            0           16s

$ kubectl create service nodeport nginx --tcp=80:80
service/nginx created

$ kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
nginx-65f88748fd-7hb6x   1/1     Running   0          41s

$ kubectl describe service nginx
Name:                     nginx
Namespace:                default
Labels:                   app=nginx
Annotations:              <none>
Selector:                 app=nginx
Type:                     NodePort
IP:                       10.102.102.220
Port:                     80-80  80/TCP
TargetPort:               80/TCP
NodePort:                 80-80  31681/TCP
Endpoints:                10.244.2.8:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Рисунок 3.12 – Скрипт розгортання веб-сервера nginx на kubernetes-кластер

У даному випадку створено тип служби NodePort, що означає, що kubernetes надаватиме службу на рівні вузлової машини. Щоб отримати доступ

до служби з локальної мережі, потрібно під'єднати kube до IP-адреси будь-якого вузла та використати TCP-порт NodePort служби (випадково призначений kubernetes із діапазону 30000–32767). Для цього, в якості тесту, відкрито IP-адресу node1 192.168.40.103 і службу nginx NodePort 31681. Результат налаштування щодо відкриття веб-сервера показано на рис. 3.13.

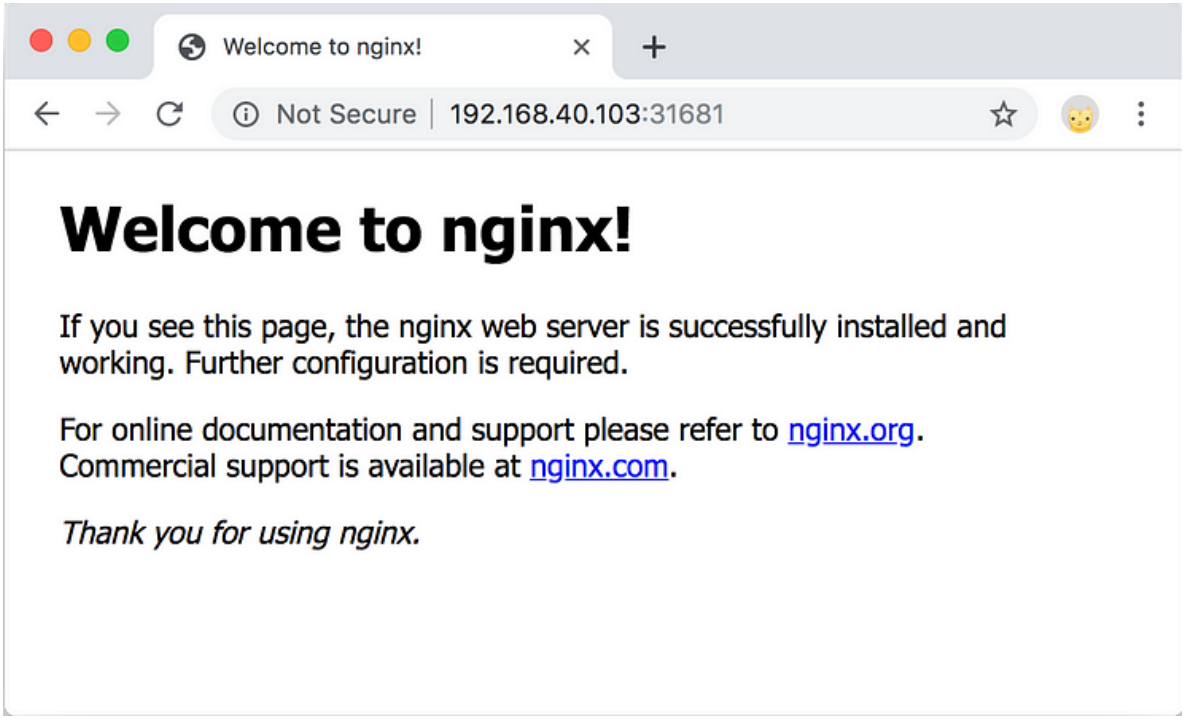

Рисунок 3.13 – Результат розгортання і запуску nginx-сервера

На рис. 3.13 показано, що доступ до веб-сервера nginx одеражано у кластері kubernetes. Це підтверджує, що кластер kubernetes, мережа контейнерів і програми в kubernetes на Raspberry Pi працюють належним чином. Наступні кроки полягають у правильності надання ресурсів kubernetes в мережі Інтернет, увімкнення рівня постійного зберігання для модулів і пошуку шляхів оптимізації ресурсів пристроїв.

3.3 Налаштування інтернет-шлюзу доступу до kubernetes-кластера

Одним із завдань при створенні kubernetes-кластера є забезпечення інфраструктури, яка дозволить отримувати доступ до додатків у kubernetes через Інтернет. Згодом він служитиме тим самим цілям, що й AWS Elastic Load Balancer, GCP Load Balancer, Azure Load Balancer та будь-який інший балансувальник навантаження в хмарних інфраструктурах.

Вирішити цю проблему лише одним компонентом неможливо, тому буде використано кілька речей: інтернет-шлюз, динамічний DNS, мережа доставки контенту.

Термін «Інтернет-шлюз» може означати будь-що. Щоб все було зрозуміло, потрібно дати означення, яке б ідеально відповідало меті створення kubernetes-кластера.

Інтернет-шлюз – апаратно-програмна система, призначена для забезпечення підключення до мережі від публічного сегмента Інтернету до приватної мережі на транспортному рівні (модель OSI, рівень 4). Є кілька способів зробити це: трансляція мережевих адрес (NAT), застосування різних проксі-серверів (такі як HAProxy, nginx, envoy та багато інших).

Для використання проксі-сервера знадобиться спеціальне мережеве обладнання, яке приймає підключення з Інтернету та проксі до вузлів kubernetes. Таким чином, використання NAT є вибором при забезпеченні доступу з мережі Інтернет до ресурсів кластера. Оскільки використовується багатофункціональний маршрутизатор Mikrotik, то його відносно легко налаштувати навіть для складного сценарію.

Далі буде використано налаштування маршрутизатора Mikrotik, однак цього можна досягти на багатьох інших пристроях. У другому розділі виконано наступні налаштування:

– IP-адреси вузлів kubernetes – 192.168.40.103 і 192.168.40.104;

– nginx надає службу на порт вузла 31681.

Згідно сценарію, показаного на рис. 3.14, користувач надсилає HTTP-запити через Інтернет; інтерфейс протоколу «точка-точка» (PPP) з’єднує

маршрутизатор з Інтернетом через глобальну мережу (WAN); маршрутизатор має з'єднання з вузлом kubernetes у локальній мережі (LAN).

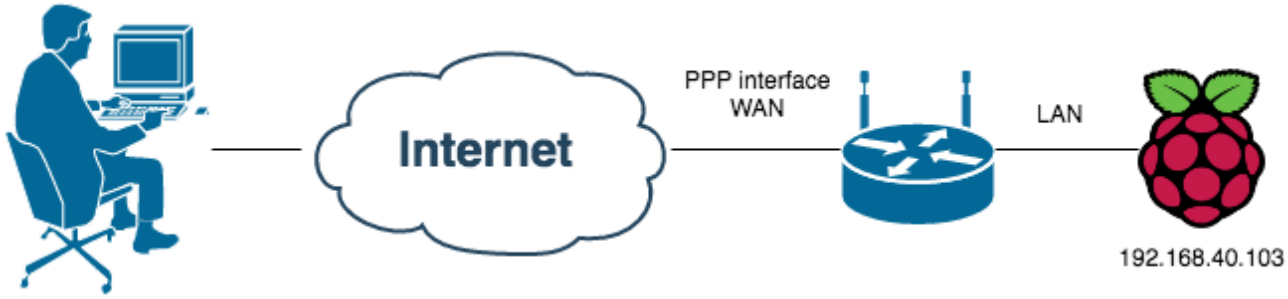

Рисунок 3.14 – Сценарій взаємодії користувача з кластером через мережу Інтернет

Виходячи з організації, показаної на рис. 3.14, перш за все потрібно вказати маршрутизатору яким чином обробляти мережевий трафік. На рис. 3.15 наведено команду для трансляції мережевих адрес для TCP-пакетів із портом призначення 80, який надходить на PPP-інтерфейс, щоб пакети далі перенаправлялися на IP-адресу 192.168.40.103 на порт 31681.

```
/ip firewall nat
add chain=dstnat action=dst-nat \
 in-interface=all-ppp protocol=tcp dst-port=80 to-
addresses=192.168.40.103 to-ports=31681
```

Рисунок 3.15 – Налаштування переадресації з порта 80 на порт 31681

Іншими словами, якщо запит HTTP (порт 80 за замовчуванням) досягне загальнодоступної IP-адреси маршрутизатора, то він перешле запит на вузол 1 kubernetes у службі nginx (через порт вузла). Однак ту є очевидний недолік. Трафік пересилається лише до одного з вузлів kubernetes, тому балансування навантаження не працює. Тому необхідно організувати інтернет-шлюз з балансуванням навантаження.

Балансування навантаження передбачає, що трафік поділяється на кілька потоків, і кожен потік пересилається на відповідну цільову серверну частину. У

номенклатурі Mikrotik метод називається класифікатором з’єднань. Зіставник за класифікатором з’єднань (PCC) дозволить розділити трафік на рівні потоки з можливістю зберігати пакети з певним набором параметрів в одному конкретному потоці. На рис. 3.16 показано схему організації з балансувальником навантаження на інтернет-шлюзі.

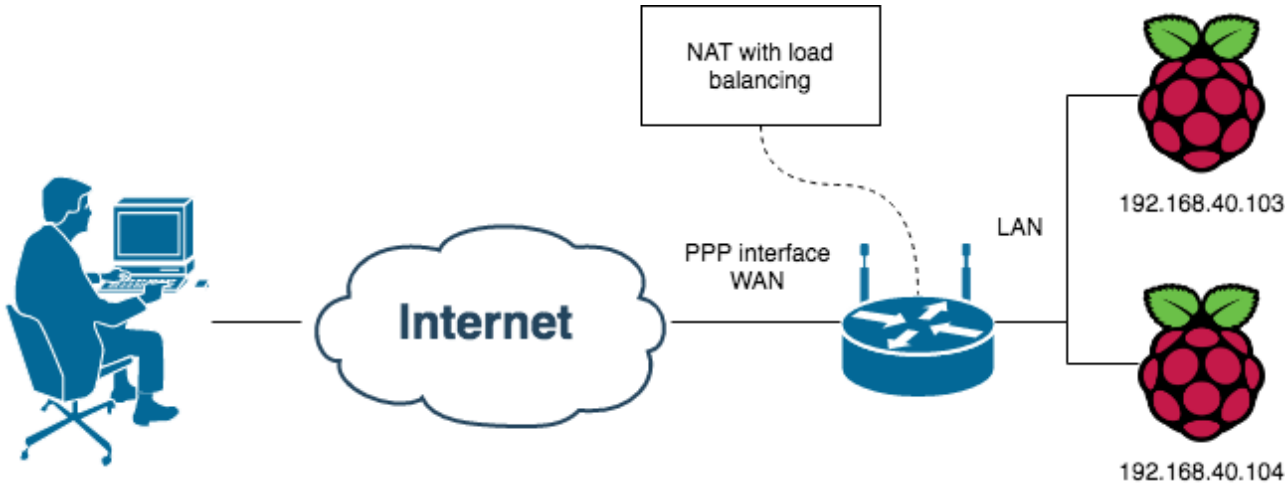

Рисунок 3.16 – Інтернет-шлюз з балансувальником навантаження

Зважаючи на принцип організації, який показаний на рис. 3.16, конфігурація маршрутизатора повинна містити дві частини:

– класифікація трафіку з метою накладання маркерів;
– трансляція мережевих адрес на основі маркерів.

На рис. 3.17 наведено налаштування маршрутизатора для класифікації трафіку і трансляції мережевих адрес.

У конфігурації налаштовано те, що маршрутизатор позначатиме сегмент трафіку маркерами kube_node_1 і kube_node_2. Далі трафік із позначкою kube_node_1 спрямовується до вузла kubernetes 1 (IP-адреса 192.168.40.103), а трафік із позначкою kube_node_2 спрямовується до вузла kubernetes 2 (IP-адреса 192.168.40.104).

Після цього класифікатор приймає числове представлення адреси та порту джерела пакету TCP. Потім це значення ділиться на знаменник, а остача від ділення порівнюється із заданою остачею. Якщо ці значення рівні, то пакет буде зафіксовано. Наприклад зразок TCP-пакета надходить з IP-адреси 192.0.2.87, а

операційна система вибирає 59831 (зазвичай тимчасовий порт). Тоді числове представлення 192.0.2.87 дорівнює 3221226071.

Потім, як сказано в документації, це значення ділиться на вказаний знаменник, а остача порівнюється з указаним залишком. У даному випадку знаменник дорівнює 2.

```
/ip firewall mangle
add chain=prerouting action=mark-connection \
  in-interface=all-ppp protocol=tcp dst-port=80 \
  new-connection-mark=kube_node_1 per-connection-classifier=src-
address-and-port:2/0 \
  comment="Load balancer. Service nginx. Mark connections to kube
node 1"
add chain=prerouting action=mark-connection \
  in-interface=all-ppp protocol=tcp dst-port=80 \
  new-connection-mark=kube_node_2 per-connection-classifier=src-
address-and-port:2/1 \
  comment="Load balancer. Service nginx. Mark connections to kube
node 2"

/ip firewall nat
add chain=dstnat action=dst-nat \
 connection-mark=kube_node_1 to-addresses=192.168.40.103 protocol=tcp
to-ports=31681 \
 comment="Load balancer. Service nginx. DST NAT to kube node 1"
add chain=dstnat action=dst-nat \
  connection-mark=kube_node_2 to-addresses=192.168.40.104
protocol=tcp to-ports=31681 \
  comment="Load balancer. Service nginx. DST NAT to kube node 2"
```

Рисунок 3.17 – Конфігурація налаштування Mikrotik для доступу до кластера

У випадку цілочисельного ділення на 2 одержують в остачі 0, тому в даному експерименті класифікатор позначатиме цей пакет як kube_node_1. В якості демонстраційної публічної IP-адреси використовується 198.51.100.19. Таким чином організовано інтернет-шлюз із необхідними властивостями, який приймає трафік з Інтернету та виконує балансування навантаження на вузли kubernetes.

3.4 Налаштування динамічного DNS

Будь-який інтернет-провайдер має пул загальнодоступних IP-адрес і під час кожного сеансу VPN (PPPoE) маршрутизатор отримує одну з них. Це означає, що через динамічний характер публічних IP-адрес неможливо передбачити, яка IP-адреса буде під час наступного сеансу VPN.

Якщо припустити, що повторне підключення VPN може відбутися будь-коли, то можуть виникати проблеми. Для доступу до веб-ресурсів, які містяться у kubernetes-кластері, можна використовувати деякий домен, наприклад, kube-test.info.

Технологія динамічного DNS дозволяє створити запис DNS, який вказує на поточну публічну IP-адресу маршрутизатора. Маршрутизатор Mikrotik має вбудований динамічний DNS. Після ввімкнення цієї опції одержується статичне ім’я DNS для загальнодоступної IP-адреси маршрутизатора. Налаштування щодо увімкнення DNS показано на рис. 3.18.

```
/ip cloud set ddns-enabled=yes
/ip cloud print
          ddns-enabled: yes
  ddns-update-interval: none
           update-time: yes
        public-address: 198.51.100.19
              dns-name: 529c0491d41c.sn.mynetname.net
                status: updated
```

Рисунок 3.18 – Налаштування увімкнення DDNS

Перші 12 байтів (529c0491d41c) імені DNS є серійним номером маршрутизатора. Для перевірки того, чи можна відкрити nginx у kubernetes-кластері за допомогою імені DNS, яке надане Mikrotik, необхідно в браузері у стрічці адресу вписати його назву. Результат перевірки продемонстровано на рис. 3.19.

Рисунок 3.19 – Результат перевірки доступу до кластера за DNS

Наступний крок полягає у налаштуванні безпеки доступу до kubernetes-кластера.

3.5 Налаштування безпеки kubernetes-кластера

Для забезпечення безпеки доступу до кластера пропонується скористатися технологією Content Delivery Network. Мережа доставки вмісту (CDN) дає кілька переваг. Найважливішими для підвищення ефективності функціонування кластеру є: підтримка HTTPS із коробки та підвищення безпеки.

Під підвищенням безпеки мається на увазі, що Cloudflare приховує сервер за своїми загальнодоступними IP-адресами. Отже, немає простого способу сказати, який тип бекенду прихований за доменом і де знаходиться сервер. Cloudflare пропонує безкоштовний план, що знижує вартість організації кластера. Існує альтернатива CDN, наприклад, letencrypt для HTTPS і пенаправлення запису DNS безпосередньо на динамічний DNS CNAME.

Після того, як домен додано до Cloudflare і виконано початкове налаштування, створюється запис DNS для динамічного домену DNS 529c0491d41c.sn.mynetname.net (рис. 3.20). Режим проксі ввімкнено, тому існують всі переваги безпеки від Cloudflare.

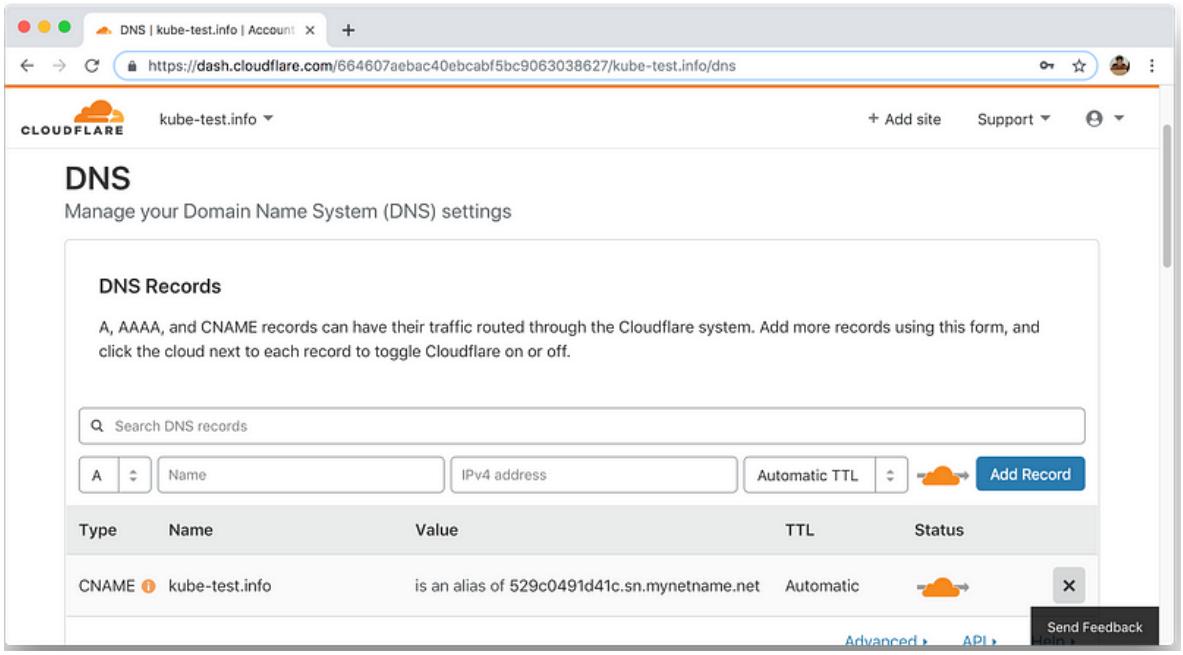

Рисунок 3.20 – Додавання DNS до Cloudflare

Оскільки бекенд може спілкуватися лише через звичайний HTTP, то потрібно вказати Cloudflare проксі HTTPS-запитів від клієнтів до порту HTTP на бекенді (рис. 3.21). Це називається режимом гнучкого SSL.

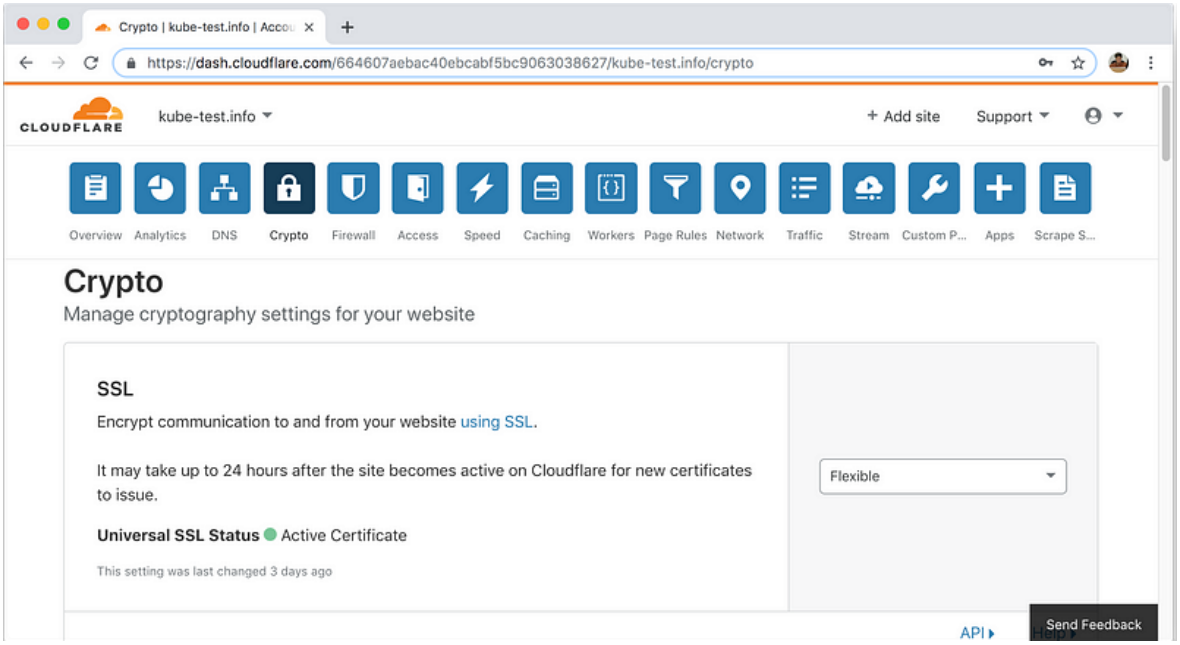

Рисунок 3.21 – Налаштування режиму гнучкого SSL

У результаті проведеного налаштування організовано безпечний доступ до kubernetes-кластера з kube-test.info, що підтверджується результатом, показаним на рис. 3.22.

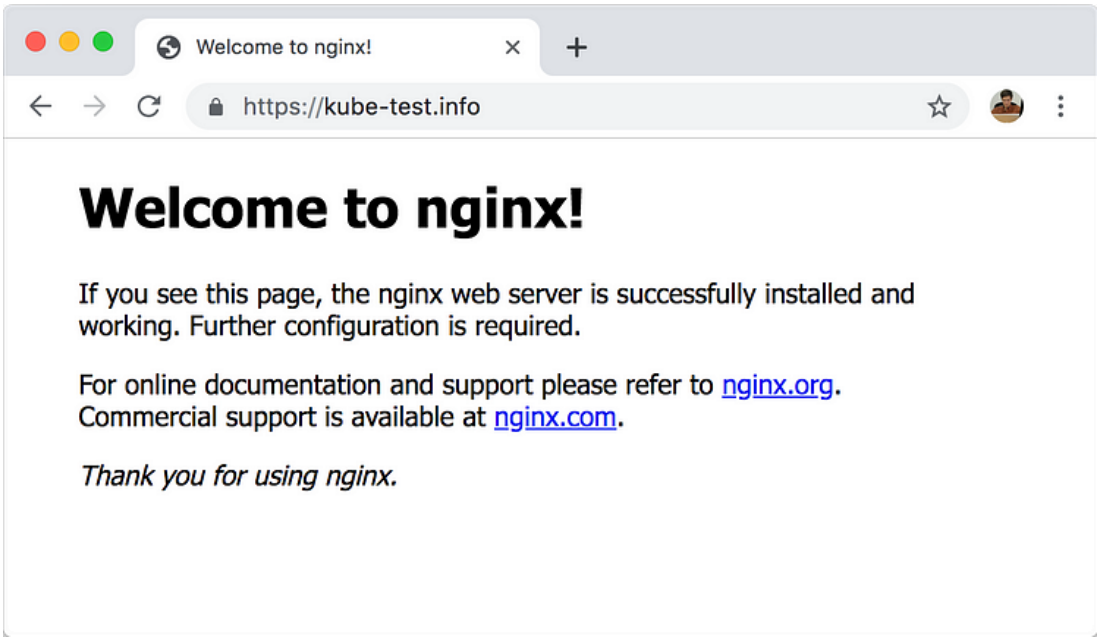

Рисунок 3.22 – nginx прихований за Cloudflare

Таким чином, у кваліфікаційній роботі проведено ряд заходів щодо організації kubernetes-кластера на базі трьох Raspberry PI, налаштовано головний і похідні вузли для виконання обчислень та запуску необхідних програмних додатків. Реалізовано балансувальник навантаження та організовано безпеку доступу до ресурсів як з мережі Інтернет, так і в межах локальної комп'ютерної мережі.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Менеджмент безпеки

Правове забезпечення безпеки життєдіяльності в Україні орієнтовано на державну політику щодо забезпечення життєдіяльності населення у техногеннобезпечному й екологічному чистому світі [19-21].

Екологічно чистий світ можливий лише при відсутності загрози з боку природних об'єктів чи при умові недопущення виникнення джерел техногенної безпеки. Із зазначених позицій основне місце посідає законодавство у галузі регулювання відносин з охорони здоров'я людини та навколишнього середовища, безпеки в надзвичайних та повсякденних ситуаціях, тобто безпеки життєдіяльності. Ці відносини регулюються нормативними актами різної юридичної сили: конституцією, законами, урядовими підзаконними актами, галузевими інструкціями вимог і правил безпеки життєдіяльності та відповідними актами місцевих органів влади [20].

Суспільство і держава відповідальні перед сучасним і майбутніми поколіннями за рівень здоров'я і збереження генофонду народу України, забезпечують пріоритетність охорони здоров'я в діяльності держави, поліпшення умов праці, навчання, побуту і відпочинку населення, розв'язання екологічних проблем, вдосконалення медичної допомоги і запровадження здорового способу життя.

Об'єктом управління БЖД є стан умов, параметрів і норм життєдіяльності на визначеній території або об'єкті. Головний напрямок у керуванні БЖД – створення безпечних умов життєдіяльності на всіх стадіях повного циклу функціонування системи "людина – навколишнє середовище".

Ефективне виконання будь-яких завдань організації управління та нагляду за безпекою життєдіяльності базується на безумовному дотриманні відповідних принципів [20]:

– системності, що передбачає застосування єдиних взаємоузгодженихпідходів до правового регулювання взаємовідносин у сфері компетенції Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи, служби Цивільної оборони;

– плановості, який полягає виключно у плановому порядку створення нормативно-правових актів;

– спадкоємності, який передбачає повне використання раніше напрацьованих матеріалів з вітчизняного та зарубіжного досвіду;

– економічності, який передбачає обов'язкове економічне обґрунтування всіх нормативно-правових актів;

– ієрархічності, який передбачає одночасну узгоджену розробку пакета нормативно-правових актів на державному та місцевому рівнях.

Контроль за дотриманням законодавства щодо безпеки життєдіяльності в Україні здійснюють різні державні та громадські організації. Серед них державні органи загальної, спеціальної та галузевої компетенції. До першої групи органів належать Кабінет Міністрів, виконавчі комітети місцевих рад народних депутатів, місцеві адміністрації.

Управління, контроль та нагляд за безпечною життєдіяльністю в Україні здійснюють [20]:

– Кабінет Міністрів України;
– Національна рада з питань безпечної життєдіяльності населення;
– Державна служба гірничого нагляду та промислової безпеки;
– Державна служба з надзвичайних ситуацій України;
– Державна інспекція ядерного регулювання України;
– Державний департамент пожежної безпеки;
– Заклади санітарно-епідеміологічної служби МОЗ.

Кабінет Міністрів України забезпечує:

– реалізацію державної політики в галузі охорони праці;

– затверджує національну програму щодо поліпшення стану безпеки, гігієни праці і виробничого середовища;

– визначає функції міністерств, інших центральних органів державної виконавчої влади щодо створення безпечних і нешкідливих умов праці та нагляду за охороною праці;

– визначає порядок створення і використовування державного, галузевих і регіональних фондів охорони праці.

З метою координації діяльності органів державного управління безпекою громадян та охороною праці створюється Національна рада з питань безпечної життєдіяльності населення, яку очолює віце-прем'єр-міністр України.

Національна рада організовує свою діяльність на підставі Положення про національну раду з питань безпечної життєдіяльності населення, затвердженого постановою Кабінету Міністрів України від 15 вересня 1993 р. № 733.

Запобігання виникнення небезпечних та несприятливих умов життєдіяльності передбачає підготовку і реалізацію комплексу правових, соціально-економічних, політичних, організаційно-технічних, санітарно-гігієнічних і інших заходів, спрямованих на регулювання умов і параметрів норм життєдіяльності, проведення оцінки рівня ризику, своєчасне реагування на зміну умов життєдіяльності на основі даних моніторингу, експертизи, контролю і прогнозу і недопущення переростання цих змін у небезпечні та несприятливі умови.

Ліквідація негативних наслідків передбачає скоординовані дії всіх структурних органів системи управління БЖД по реалізації заздалегідь розроблених планів, уточнених в умовах конкретного характеру і рівня надзвичайної ситуації з метою виключення загрози здоров'ю і життю людей і надання невідкладної допомоги по-страждалим.

Основні завдання та функції державної системи управління:

– планування робіт;

– розробка, прийняття і відміна нормативних актів;

– професійний відбір;

– реєстрація і облік;

– експертиза;

– ліцензування та сертифікація;

– управління фондами;

– узгодження і видача дозволів, наукове забезпеченя, міжнародне співробітництво;

– забезпечення безпеки обладнання, будівель та споруд;

– забезпечення санітарно-гігієнічних умов праці, санітарно-побутового обслуговування, лікувально-профілактичного і медичного обслуговування;

– розслідування та облік нещасних випадків, захворювань, аварій;

– пропаганда культури безпеки.

4.2 Естетичне оформлення та ергономічне дослідження робочого місця оператора

Ергономічна організація робочого місця користувача ЕОМ враховує як специфіку діяльності, що виконується, так і забезпечує комфортні умови перебування людини.

Тому основними ергономічними завданнями щодо організації робочого місця є наступні [21]:

– забезпечення просторових параметрів робочого місця, які відповідають антропометричним характеристикам користувача;

– раціональне розташування елементів робочого місця відносно користувача на підставі поглибленого кількісного та якісного аналізу діяльності, яка виконується;

– оптимізацію умов робочого середовища.

На рисунку 4.1 наведено робоче місце користувача ЕОМ та позначено основні ергономічні та просторові параметри його складових.

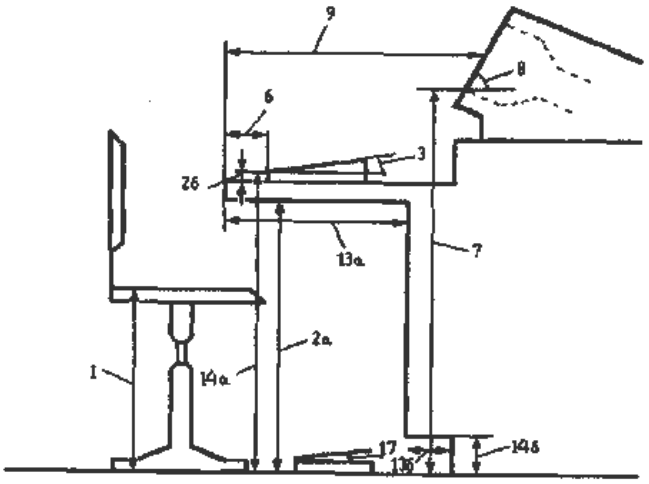

Рисунок 4.1 – Робоче місце користувача ЕОМ

Основні просторові параметри робочого місця користувача ЕОМ приведені в таблиці 4.1.

Таблиця 4.1 – Просторові параметри робочого місця

Умовні позначення	Параметри	Спосіб вимірювання параметра	Значення параметра
1	Висота сидіння	Від підлоги до верхньої площини сидіння	400-500 мм
2	Висота клавіатури (від рівня підлоги)	Від підлоги до нижнього ряду клавіатури	600-700 мм
2а	Висота клавіатури (від рівня стола)	Від базової поверхні до нижнього ряду клавіатури	20 мм
3	Кут нахилу клавіатури	Від горизонтальної площини	$7\text{-}15^0$
4	Ширина основної клавіатури	Визначається оптимальною зоною моторного поля	До 400 мм
5	Глибина основної клавіатури	Визначається оптимальною зоною моторного поля	До 200 мм
6	Відстань від клавіатури до краю стола	Від переднього краю стола до клавіатури	Понад 80 - 100 мм

Продовж. табл. 4.1

Умовні позначення	Параметри	Спосіб вимірювання параметра	Значення параметра
7	Висота екрана	Від підлоги до нижнього краю екрана	950-1050 мм
8	Кут нахилу екрана	Від вертикальної площини	15^0
9	Відстань від екрана до краю стола	Від переднього краю стола до екрана	500-700 мм
10	Висота поверхні для запису	Від підлоги	870-860 мм
11	Площа поверхні для запису	Визначається оптимальною зоною моторного поля	600 х 400 мм 900 х 600 мм
12	Кут нахилу поверхні для запису	Від горизонтальної площини	$0 – 10^o$
13	Глибина простору для ніг на рівні колін	Від переднього краю стола	Понад 400 мм
13а	Глибина простору для ніг на рівні	Від підлоги	Понад 600 мм
14	Висота простору для ніг на рівні колін	Від переднього краю стола	Понад 600 мм
14а	Висота простору для ніг на рівні ступень	Від підлоги	Понад 1 00 мм
15	Ширина простору для ніг на рівні колін		Понад 500 мм
15а	Ширина простору для ніг на рівні		Понад 250 мм
16	Висота підставки для ніг	Від підлоги до передньої частини підставки	50-130 мм
17	Кут нахилу підставки для ніг	Від горизонтальної площини	0-25
18	Глибина підставки для ніг	Від переднього краю підставки до її заднього краю	400 мм
19	Ширина підставки для ніг		300 мм
20	Пюпітр-підставка для документі в	Від горизонтальної площини	15 - 20°

В ході організації робочих місць на кожну ЕОМ виділяється площа, яка складає не менш, ніж 6 $м^2$, та об'єм, який становить не менш, ніж 20 $м^3$. Причому, зона, де розташовується робочий стіл, сервер або робоча станція, принтер, екран для графопроектора, займає відповідно 6-8 $м^2$. Висота приміщення не менша, ніж 4 м [13].

Робоче місце користувача ПК облаштоване одномісним столом та напівм'яким стільцем, висоту сидіння яких можна змінювати. Довжина стола користувача не менше 700 мм, ширина – забезпечує місце перед клавіатурою для розташування зошита або іншого приладдя. Поверхня стола має кут нахилу у межах $12\text{-}15^0$, лише іноді припустимою є її розташування у горизонтальній площині.

На робочому місці користувача ПК забезпечена відповідність висоти краю стола і стільця до росту та антропометричних особливостей організму користувачів.

Глибина простору для ніг під столом не менше 450 мм, а у випадку застосування високого стола та низького стільця і, отже, відсутності відповідності росту користувача конструктивним елементам робочого місця, використовується підставка для ніг, ширина якої становить – 350 мм, довжина – 400 мм, кут нахилу опорної поверхні – 15^o.

Столи з ЕОМ розміщено без розривів між ними, але при незначній кількості робочих столів з відеотерміналами перевагу варто віддавати розташуванню їх біля внутрішньої стіни.

Робота з комп'ютерною технікою вимагає обов'язкового дотримування правильної посадки. Користувач ЕОМ повинен сидіти прямо, з невеликим нахилом (до 5 – 7°) голови вперед, не сутулитися, спираючись нижніми краями лопаток на спинку стільця. Передпліччя повинні спиратися на поверхню стола, забезпечуючи зниження статичного напруження м'язів плечового поясу і рук, кути, що утворюються передпліччям і плечем, а також гомілкою і стегном, – складати не менш, ніж 90°.

Рівень очей припадає на центр екрана або на точку, яка розташована між верхньою та середньою третинами екрану, причому, лінія погляду є

перпендикулярною до площини екрана, а її відхилення у вертикальній площині – знаходитися у межах ±5–10^0. Оптимальний огляд у горизонтальній площині від центральної осі екрана у межах ±15–30°. Лише під час спостереження за інформацією, яка розміщена у найвіддаленіших ділянках екрану, кут огляду становить 40–45°.

Кут розглядання цифр та букв на екрані монітора не менше 20 кутових хвилин. Оптимальна відстань від очей до площини екрана монітора складає 600 –700 мм, допустима – не менше 500 мм. Розглядати інформацію на екрані з відстані менш, ніж 500 мм не рекомендується.

При проектуванні комп'ютерної мережі у дипломній роботі враховані вимоги до ергономічної організації робочого місця користувачів ПК, що дозволяє підвищити працездатність та зменшити негативний вплив на їх здоров'я згідно ДСанПін 3.3.2.007 – 98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин».

ВИСНОВКИ

Під час виконання кваліфікаційної роботи проведено аналіз завдань щодо організації kubernetes-кластера на основі Raspberry PI та досліджено особливості функціональних властивостей та структури відкритої платформи Kubernetus.

На основі результатів аналізу спроектовано архітектуру кластеру, до складу якого входить три мінікомп'ютери Raspberry PI 3 Model B. Один з цих вузлів налаштовано як master, що дає змогу забезпечити керування іншими хостами та виконанням тестових docker-контейнерів.

В процесі організації kubernetes-кластера використано додаткове комутаційне обладнання, зокрема Mikrotik RB760iGS (hEX S), що забезпечує доступ до мережі Інтернет та відповідає за безпеку доступу до елементів кластера, та некерований комутатор DGS-1005P, основна задача якого полягає у забезпеченні зв'язку між хостами кластера та маршрутизатором.

На кожному хості кластера встановлено операційну систему Ubuntu 18.04, а процес автоматизованого розгортання інфраструктури забезпечує інструмент Ansible. Окрім цього, при організації kubernetes-кластера використано Docker Engine в якості середовища виконання контейнера, kubelet («агент вузла»), що запущений на кожному вузлі кластера і безпосередньо інструмент створення кластера kubeadm nf kubectl, як інтерфейс командного рядка для запуску команд у кластерах.

У роботі проведено налаштування інтернет-шлюза та динамічного DNS для доступу до кластера з мережі Інтернет, а також забезпечено підвищення рівня безпеки за допомогою Cloudflare, що приховує сервер за своїми загальнодоступними IP-адресами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Domingus J. Cloud Native DevOps with Kubernetes. 2nd Ed. O'Reilly Media.2022. 456 p.
2. Domingus J., Arundel J. Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud 2nd. O'Reilly Media. 2022. 383 p.
3. Big Data What it is and why it matters URL: https://www.sas.com/en_us/insights/big-data/what-is-big-data.html (дата звернення 10.04.2023 р.)
4. Min Chen, Shiwen Mao, Yin Zhang, Victor C.M. Leung. Big Data. Related Technologies, Challenges, and Future Prospects. Spinger, 2014. 100 с.
5. Amazon EMR Documentation URL: https://docs.aws.amazon.com/emr (дата звернення 11.04.2023 р.)
6. Google Dataproc documentation URL: https://cloud.google.com/dataproc/docs (дата звернення 15.04.2022 р.)
7. Create a cluster. URL: https://cloud.google.com/dataproc/docs/guides/create-cluster#dataproc-create-cluster-gcloud (дата звернення 23.04.2023 р.)
8. Schults C. What Is Infrastructure as Code? How It Works, Best Practices, Tutorials URL: https://stackify.com/what-is-infrastructure-as-code-how-it-works-best-practices-tutorials/ (дата звернення 18.05.2023 р.)
9. Grady B. Object Oriented Design: With Applications / Booch Grady. Boston, MA: Pearson Education, 2007. 551 с.
10. Brikman Y. Terraform: Up & Running: Writing Infrastructure as Code. Sebastopol, CA: O'Reilly Media, 2019. 368 с.
11. Abd-Allah A. Extending reliability block diagrams to software architectures / Center for Software Engineering. Computer Science Department. University of Southern California. Los Angeles. Technical Report: USC-CSE-97-501. URL:http://sunset.usc.edu/publications/ TECHRPTS/1997/usccse97-501/usccse97-501.ps (дата звернення: 23.05.2023 р.)

12. Deploy on Kubernetes. URL: https://docs.docker.com/desktop/kubernetes/ (дата звернення 18.05.2023 р)

13. What is a Kubernetes cluster? URL: https://www.vmware.com/topics/glossary/content/kubernetes-cluster.html (дата звернення 25.05.2023 р)

14. What is Kubernetes infrastructure? URL: https://www.vmware.com/topics/glossary/content/kubernetes-infrastructure.html (дата звернення 25.05.2023 р)

15. Kubernetes Clusters: Everything You Need To Know. URL: https://www.containiq.com/post/kubernetes-cluster (дата звернення 25.05.2023 р)

16. Vault Documentation. URL: https://developer.hashicorp.com/vault/docs?host=www.vaultproject.io (дата звернення 25.05.2023 р)

17. How Ansible works. URL: https://www.ansible.com/overview/how-ansible-works (дата звернення 01.06.2023 р)

18. Red Hat Ansible Automation Platform. URL: https://www.redhat.com/en/technologies/management/ansible (дата звернення 01.06.2023 р)/

19. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп'ютеризованих вимірювальних систем і комплексів. Навчальний посібник. Тернопіль: ТНТУ. 2019. 150 с.

20. Осухівська Г.М., Тиш Є.В., Луцик Н.С., Паламар А.М. Методичні вказівки до виконання кваліфікаційних робіт здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль, ТНТУ. 2022. 28 с.

21. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Київ. 2018.

22. Катренко Л.А., Катренко А.В. Охорона праці в галузі комп'ютинґу. Львів: Магнолія-2006. 2012. 544 с.

23. Бедрій Я. Основи охорони праці користувачів персональних комп'ютерів: навчальний посібник для студентів ВНЗ та інженерів-практиків. Навчальна книга-Богдан. 2014. 144 с.

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем і програмної інженерії

Кафедра комп’ютерних систем та мереж

“Затверджую”

Завідувач кафедри КС

_____ Осухівська Г.М.

“___” _____2023 р

KUBERNETES-КЛАСТЕР НА ОСНОВІ RASPBERRY PI

ТЕХНІЧНЕ ЗАВДАННЯ

на _*10*_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп’ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІ-41
_____ к.т.н., доц. Луцків А.М.	_____ Крайник О.В.
«____» _____ 2023 р.	«____» _____ 2023 р.

Тернопіль 2023

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Kubernetes-кластер на основі Raspberry PI».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.042.00.00

1.2 Виконавець

Студент групи СІ-41, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Крайник Олексій Володимирович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№ 4.7-238 від 28.02.2023 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 28.02.2023 р.

Плановий термін завершення виконання кваліфікаційної роботи – 24.06.2023 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ISO, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи.

Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Kubernetes-кластер на основі Raspberry PI призначений для забезпечення швидкого масштабування, розгортання та управлінням програмними додатками.

Основна перевага використання Kubernetes у середовищі користувача, особливо у випадку оптимізації розробки програм для «наземної» інфраструктури, полягає в тому, що це забезпечує організацію платформи для планування та запуску контейнерів у кластерах фізичних або віртуальних машин (ВМ) в ролі яких виступають 3 мінікомп’ютери Raspberry PI.

У даному випадку, kubernetes планується використовувати як засіб автоматизованого розгортання та оркестрації контейнерів, що забезпечує функції масштабування, балансування навантаження, організації безпеки та інші.

Якщо говорити ширше, то організація такого кластеру допоможе повністю реалізувати і покладатися на інфраструктуру на основі контейнерів у виробничих середовищах. Враховуючи те, що kubernetes — це засіб автоматизації операційних

завдань, то за допомогою нього можна робити багато того ж, що й інші платформи додатків або системи керування, але для користувацьких контейнерів.

На kubernetes- кластер покладаються функції по типу тих, які забезпечує хмарна інфраструктура, тобто програмні додатки можна запускати наче у хмароподібному середовищі, отримувати доступ до них ззовні, а програмному забезпеченню, що зберігає стани надається можливість використовувати рівень зберігання даних.

2.2 Мета створення системи

Мета побудови та налаштування kubernetes-кластера на основі Raspberry PI полягає у забезпеченні можливості реалізації дешевого інструменту масштабування інфраструктури та управління навантаженням на вузли кластера, а також організації безпечного доступу з мережі Інтернет для управління як самим кластером, так і його вузлами.

Для досягнення мети щодо організації kubernetes-кластера у кваліфікаційній роботі визначено наступні задачі::

– обґрунтувати вибір апаратного та програмного забезпечення для побудови kubernetes-кластера;

– дослідити технічні характеристики апаратних компонентів кластера;

– провести налаштування комунікаційної інфраструктури для доступу до мережі Інтернет;

– зареєструвати доменне ім’я, як точку входу для безпечного доступу до ресурсів кластера ;

– організувати налаштування головного (master) вузла та інших вузлів (slave) кластера;

– забезпечити налаштування безпеки доступу до кластера як з мережі Інтерент так і з мережі кластера;

– провести тестування kubernetes-кластера шляхом запуску ngnix сервера.

2.3 Характеристика об'єкту

2.3.1 Основні задачі та функції об’єкту

Основна задача kubernetes-кластера полягає в автоматизації процесів розгортання інфраструктури при проведенні обчислень, а також автоматизації управління виконанням контейнерів. Окрім цього, розробники також можуть створювати хмарні додатки і використовувати Kubernetes як платформу для виконання з використанням шаблонів.

Шаблони – це інструменти, необхідні розробнику Kubernetes для створення програм і служб на основі контейнерів. Основні функції Kubernetes:

– оркестрація контейнерів на кількох хостах;

– оптимізоване використання апаратного забезпечення з метою максимізації ресурсів необхідних для запуску корпоративних програм користувача;

– контроль і автоматизація розгортання та оновлення програм;

– підключення і додавання сховищ для запуску програм із збереженням стану;

– масштабування контейнерних програм та їхніх ресурсів в режимі реального часу;

– декларативне керування службами, що гарантує, що розгорнуті програми завжди працюють так, як заплановано;

– перевірка працездатності і самовідновлення користувацьких програм за допомогою авторозміщення, автоперезапуску, автореплікації та автомасштабування.

Однак Kubernetes покладається на інші проекти, щоб повністю надавати ці організовані послуги. Додавши інші проекти з відкритим вихідним кодом, можна повністю використати потужність Kubernetes. Ці необхідні частини включають (серед іншого):

– реєстр через такі проекти, як Docker Registry;

– мережу за допомогою таких проектів, як OpenvSwitch та інтелектуальну граничну маршрутизацію;

– телеметрію через такі проекти, як Kibana, Hawkular і Elastic;

– безпеку за допомогою таких проектів, як LDAP, SELinux, RBAC і OAUTH з багаторівневими рівнями;

– автоматизацію з додаванням playbook Ansible для встановлення та керування життєвим циклом кластера;

– сервіси через багатий каталог популярних шаблонів програм.

При проектуванні Kubernetes-кластера доцільно використовувати Ansible для автоматизованого розгортання програмного забезпечення та доступу до Microtik-маршрутизатора. Безпека доступу ззовні забезпечується шляхом використання CloudFlare, а всередині кластера – через налаштування маршрутизатора.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

Кластер kubernetes повинен працювати як хмарна інфраструктура, де можна запускати програми у хмароподібному середовищі, отримувати до них доступ з мережі Інтернет, програми зі збереженням стану матимуть можливість використовувати рівень зберігання даних.

Важливими про побудові кластера є:

– вимоги до пристрою Raspberry Pi;

– підготовка вузла kubernetes;

– налаштування кластера.

В якості комутаційного повинен використовуватися маршрутизатор Microtik hEX S(RB760iGS). За $69,00 він пропонує широкий вибір функцій, які зазвичай доступні лише на маршрутизаторах високого класу від таких постачальників, як Cisco, Juniper та інших. Один порт маршрутизатора повинен використовуватися для підключення до Інтернет-провайдера. Інший порт має з'єднувати маршрутизатор з комутатором.

Для об’єднання Raspberry PI повинен використовуватися некерований комутатор, наприклад, D-Link DGS-1005 P, який коштує менше 50 доларів, він має п'ять портів Gigabit Ethernet. Чотири порти підтримують живлення через Ethernet (PoE). Порт 5 використовується для підключення до маршрутизатора. Такий тип зв’язку повинен забезпечувати висхідний зв’язок. Інші чотири порти можна використовувати для підключення до пристроїв Raspberry Pi.

3.1.1 Вимоги до структури та функціонування системи

Структура кластера характеризується такими основними компонентами:

– Mikrotik RB760iGS — маршрутизатор, що забезпечує кластер Kubernetes підключенням до Інтернету, комутацією VLAN, сервером DHCP, NAT, динамічним DNS та іншими;

– D-Link DGS-1005P — некерований комутатор з 4 портами PoE, який забезпечує Raspberry Pi комутацією живлення та канального рівня з маршрутизатором і вузлами Kubernetes;

– Raspberry Pi 3 Model B+ — міні-комп’ютер із 64-розрядним чотирьохядерним процесором 1,4 ГГц, 1 ГБ оперативної пам’яті, Gigabit Ethernet Raspberry Pi PoE;

– HAT — розширення Raspberry Pi, яке дозволяє живити пристрій через Ethernet microSDHC MIREX (class10) 8GB — SD-карта з операційною системою для Raspberry Pi;

– microSDHC MIREX (class10) 8GB — SD-карта з операційною системою для Raspberry Pi.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Зв’язок між компонентами кластеру забезпечуються як на програмному, так і на апаратному рівні. Апаратним рівнем визначено проводове з’єднання за допомогою Ethernet кабеля категорії 5e.

На рівні програмного забезпечення забезпечується передача даних за допомогою транспортного рівня і протоколу TCP/IP. З'єднання з мережею Інтернет керованого маршрутизатора забезпечується на основі оптоволоконної лінії зв'язку.

3.1.3 Вимоги по діагностуванню системи

Діагностика kubernetes-кластера на основі Raspberry PI передбачає тестування коректності його функціонування за визначеним розкладом або у випадку виникнення нештатних ситуацій, які характеризуються виходом з ладу апаратного забезпечення або програмних збоїв.

3.1.4 Перспективи розвитку, модернізація системи

Перспективи розвитку kubernetes-кластера передбачають масштабованість апаратних компонентів кластера, зокрема нарощування кількості Raspberry PI, як вузлів для виконання обчислень. Окрім цього, передбачається додавання функціональності шляхом додаткової оптимізації розподілу навантаження на вузли кластера та візуалізації навантаженості вузлів в реальному часі.

3.1.5 Вимоги до надійності системи

Вимоги надійності до kubernetes-кластера передбачають безперебійність його роботи як при виконанні обчислень, так і у випадку режиму очікування. Це означає готовність та налаштовуваність усіх компонент до визначеного режиму функціонування. Окрім цього, важливим показником надійності є захищеність кластеру з точки зору зовнішнього програмного доступу та з локальної комп'ютерної мережі.

У випадку виникнення збою у роботі компонентів, відновлення працездатності не повинно займати багато часу, тобто кластер в цілому та усі його компоненти повинні бути ремонтопридатними.

3.1.6 Вимоги до функцій та задач, які виконує система

Основна функція, яка покладається на kubernetes-кластер полягає у забезпеченні розгортання інфраструктури при виконанні обчислень або запуску користувацьких додатків на виконання з використанням Raspberry PI. Основне завдання, яке необхідно реалізувати у кваліфікаційній роботі – забезпечити можливість запуску сервера nginx, на якому може бути розгорнуте користувацьке програмне забезпечення з оптимальним розподілом задач на виконання окремими вузлами Raspberry PI та захищеним публічним доступом з мережі Інтерент. Серед найбільш важливих вимог, які висуваються до kubernetes-кластера можна виділити наступні:

– забезпечення оркестрації контейнерів на вузлах кластера;

– оптимізація ресурсів апаратних пристроїв для виконання поставлених перед кластером завдань;

– забезпечення автоматизації та контролю за розгортанням і оновленням програмного забезпечення;

– управління підключенням і додаванням сховищ для програм, що підтримують збереження стану;

– масштабування інфраструктури за вимогою та у режимі реального часу;

– декларативне управління службами для гарантування коректності функціонування програм;

– перевірка працездатності і самовідновлення користувацьких програм за допомогою авторозміщення, автоперезапуску, автореплікації та автомасштабування.

3.1.7 Вимоги до апаратного забезпечення

Вимоги до маршрутизатора:

– Mikrotik RB760iGS.

Вимоги до комутатора:

– D-Link DGS-1005P;

Вимоги до хостів кластера:

– Raspberry Pi 3 Model B+;

Додаткові вимоги:

– HAT розширення Raspberry Pi

Вимоги до дискового простору:

– microSDHC MIREX (class10) 8GB;

3.1.8 Вимоги до програмного забезпечення

Програмне забезпечення, яке використовується при побудові kubernetes-кластера включає.

– операційна система Ubuntu 18.04;

– середовище виконання контейнера Docker Engine;

– «агент вузла кластера» – kubelet;

– інструмент для створення кластерів – kubeadm;

– інтерфейс командного рядка для запуску команд у кластерах – kubernetes kubectl.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

1 Структура інструментів kubernetes.

2 Структура Docker-контейнера.

3 Архітектура kubernetes-кластера.

4 Характеристики Raspberry PI.

5 Сценарії взаємодії користувача з кластером.

*Примітка: У комплект документації можуть вноситися міни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ етапу	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка і затвердження технічного завдання	28.02-13.03.2023
2	Аналіз технічного завдання	15.03-02.04.2023
3	Визначення вимог до апаратного та програмного забезпечення kubernetes-кластера	03.04-18.04.2023
4	Проектування структури kubernetes-кластера	19.04-04.05.2023
5	Налаштування параметрів kubernetes-кластера та сценаріїв використання	04.05-12.05.2023
6	Розробка інструкцій із встановлення та налаштування параметрів безпеки kubernetes-кластера	12.05-29.05.2023
7	Безпека життєдіяльності, основи охорони праці	01.06-05.06.2023
8	Оформлення кваліфікаційної роботи	05.06-12.06.2023
9	Попередній захист кваліфікаційної роботи	12.06-17.06.2023
10	Захист кваліфікаційної роботи	19.06-24.06.2023

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.