Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра інформаційних технологій та комп'ютерної інженерії

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня *магістра*

Студента Галушка Олександра Геннадійовича

академічної групи *123м – 22 – 1*

спеціальності *123 «Комп'ютерна інженерія»*

на тему: *Розробка серверного кластеру на основі програмного забезпечення з відкритим кодом*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		Рейтинговою	**Інституційною**	
кваліфікаційної роботи	*к.т.н., доц. Соколова Н.О.*			
розділів:				
Консультант	*Доц. Бешта Д.О.* *Ас.Панферова Я.В.* *Ас. Обиденний Є.О.* *Доц. Шедловська Я.І.*			
Рецензент	*Клименко А.В.*			
Нормоконтролер				

Дніпро
2023

ЗАТВЕРДЖЕНО:

завідувач кафедри

Інформаційних технологій та комп'ютерної інженерії

(повна наз ва)

_____ *д.т.н., проф. Гнатушенко В.В.*

(підпис) (прізвище, ініціали)

«_____» _____2023року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня *магістра*

студенту Галушку О.Г. **академічної групи** *123м-22-1*

спеціальності: *123 «Комп'ютерна інженерія»*

на тему «*Розробка серверного кластеру на основі програмного забезпечення з відкритим кодом»*

затверджену наказом ректора НТУ «Дніпровська політехніка»від *09.10.20223 р. №1227-с*

Розділ	Зміст	Терміни виконання
Розділ 1	Проаналізувати стан області рішення задач	9.10.2023 – 24.10.2023
Розділ 2	Провести огляд програмного забезпечення з відкритим кодом	24.10.2023 – 11.11.2023
Розділ 3	На основі відповідних матеріалів проаналізувати варіанти оптимального рішення для розробки власного проекту	12.11.2023 – 1.12.2023
Розділ 4	Реалізація розроблених компонентів комп'ютерної системи	1.12.2023- 08.12.2023

Завдання видано _____ *доц. Соколова Н.О.*

(підпис) (прізвище, ініціали)

Дата видачі: *09.10.2023 р.*

Дата подання до екзаменаційної комісії:_____

Прийнято до виконання _____ _____

(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 83 с., 10 рис., 2 табл., 2 додатки, 15 джерел.

Об’єкт дослідження: Системи кластеризації серверів.

Предмет дослідження: створення та аналіз підходів до реалізації серверного кластеру для підвищення продуктивності та відмовостійкості системи.

Мета дослідження: проаналізувати сучасні підходи реалізації серверного кластеру, розробити серверний кластер на основі ПЗ з відкритим кодом, перевірити його відмовостійкість.

Методи дослідження: теоретичний метод - моделювання, аналіз, порівняння та абстрагування, а також експериментальне моделювання та налаштування серверів.

У вступі до роботи обґрунтовано актуальність створення серверних кластерів як інструменту для забезпечення стійкості роботи веб-систем. Сформульовано мету роботи та визначено завдання, які потрібно вирішити.

У першому розділі проаналізовано основні принципи створення серверних кластерів. Розглянуто розподіл навантаження та високу доступність ресурсів як ключові аспекти структури кластера.

У другому розділі були розглянуті необхідні інструменти для розробки інформаційної системи тестування веб додатків. Були розглянути підходи до побудови архітектури системи.

У третьому розділі було описано складові серверного кластеру: балансувальник, основний та резервний сервери, сервер бази даних.

У четвертому розділі були розроблені тести, що відповідають технічним вимогам, що дозволяє перевірити різні аспекти функціональності та надійності веб-додатку.

СЕРВЕРНИЙ КЛАСТЕР, БАЛАНСУВАЛЬНИК, NGINX, APACHE, REVERSE-PROXY, РЕЗЕРВНЕ КОПІЮВАННЯ, VPS, СУБД.

ABSTRACT

Explanatory note: 83 p., 10 illustrations, 2 tables, 2 appendix, 15 sources.

Object of research: Server clustering systems.

Subject of research: creation and analysis of approaches to the implementation of a server cluster to improve system performance and fault tolerance.

Research methods: theoretical method - modelling, analysis, comparison and abstraction, as well as experimental modelling and server configuration.

In the introduction to the paper, the relevance of creating server clusters as a tool for ensuring the stability of web systems is substantiated. The purpose of the work is formulated and the tasks to be solved are defined.

The first section analyses the basic principles of creating server clusters. Load balancing and high availability of resources are considered as key aspects of the cluster structure.

The second section discusses the necessary tools for developing an information system for testing web applications. Approaches to building the system architecture were considered.

The third section describes the components of the server cluster: the balancer, the main and backup servers, and the database server.

In the fourth section, we developed tests that meet the technical requirements, which allows us to check various aspects of the functionality and reliability of the web application.

SERVER CLUSTER, BALANCER, NGINX, APACHE, REVERSE-PROXY, BACKUP, VPS, DBMS.

ЗМІСТ

ВСТУП

Актуальність роботи. Розробка сучасних проектів стала необхідністю через зростання вимог користувачів до технологій та послуг. Швидкі темпи технологічного розвитку та змін у суспільстві вимагають надійних, інноваційних рішень. Реагування на ці вимоги забезпечує конкурентоспроможність проектів та їх придатність для відповіді на сучасні потреби користувачів.

Створення серверних кластерів стає насущною потребою в контексті сучасних інформаційних систем. Ця необхідність зростає внаслідок розширення обсягів даних, підвищення вимог до безпеки та стійкості інфраструктури. Перш за все, кластери гарантують високу доступність систем. Коли один сервер або компонент недоступний, інші частини кластера автоматично беруть на себе роботу, запобігаючи відмовам у обслуговуванні та забезпечуючи безперервну роботу системи. Також кластери дозволяють легко масштабувати систему в залежності від зростання навантаження. Додавання нових серверів для оптимізації навантаження дозволяє підтримувати оптимальну продуктивність.

Використання кластерів також забезпечує оптимізацію ресурсів. Завдяки розподілу завдань між серверами, можна ефективно використовувати обчислювальні потужності та мінімізувати простір пам'яті.

Окрім цього, створення кластерів дозволяє покращити безпеку та здатність відновлення. Резервне копіювання даних та налаштування дублюючих систем дозволяють запобігти втраті даних та забезпечити можливість оперативного відновлення в разі виникнення проблем.

Кластеризація систем є елементом для забезпечення продуктивності й надійності в умовах постійно зростаючих потреб у обчислювальних ресурсах. Вона дозволяє досягти балансу між швидкістю, доступністю та відмовостійкістю, необхідними в сучасному IT-середовищі.

Серверні кластери відіграють важливу роль у сучасних IT-системах, забезпечуючи високу доступність, ефективне використання ресурсів та стійкість до відмов. Вони дозволяють масштабувати систему, оптимізувати навантаження та підвищувати безпеку даних. Важливість кластерів полягає в їхній здатності забезпечити стабільну та продуктивну роботу IT-інфраструктури в умовах постійно зростаючих вимог до обчислювальних ресурсів та надійності систем.

Одним із ключових аспектів створення серверного кластеру є ефективне розподілення завдань та навантаження між серверами, що дозволяє оптимізувати роботу кластера та забезпечувати високу доступність послуг користувачам.

Для досягнення поставленої мети необхідно виконати наступний комплекс задач:

1. Визначення серверного кластера.
2. Огляд існуючих рішень.
3. Огляд технологічних рішень.
4. Розробка інформаційної системи серверного кластеру

Наукова новизна полягає у реалізації інформаційної системи серверного кластеру на основі віртуальних серверів та використання програмного забезпечення з відкритим кодом. Даний проект спрямовано на створення актуальної системи для розміщення додатків та веб-сайтів з підвищеною відмовостійкістю.

Практичне значення. Висвітлення досліджень у цій роботі дозволяє існуючим та майбутнім системним адміністраторам розуміти, як реалізувати процес конфігурування серверів об’єднаних у кластер, з метою підвищення відмовостійкості системи.

Отримані знання можуть бути використані в освітніх програмах для навчання принципам обробки, зберігання та децентралізації інформації в системі. Результати даного проекту можуть бути корисними для комерційних компаній, так як система пропонує доступне рішення для бізнесів з метою зменшення затрат на готові рішення.

1 РОЗДІЛ

СТАН ПИТАННЯ І ПОСТАНОВКА ЗАДАЧІ

1.1 Призначення кластерних серверів

Збільшення кількості користувачів в Інтернеті збільшує навантаження на сервери та інфраструктуру. Оскільки велика кількість користувачів одночасно виконує багато завдань, таких як переглядання контенту, покупки, спілкування тощо, необхідні нові обчислювальні можливості.

Використання систем, таких як серверні кластери, необхідно для задоволення цих вимог до обчислювальних ресурсів і надійності. За допомогою кластеризації можна забезпечити безперервну доступність, оптимізувати роботу мережі та розподілити навантаження між кількома серверами. Для збільшення кількості користувачів серверні кластери можуть автоматично масштабуватися, додаючи додаткові сервери для оптимізації навантаження.

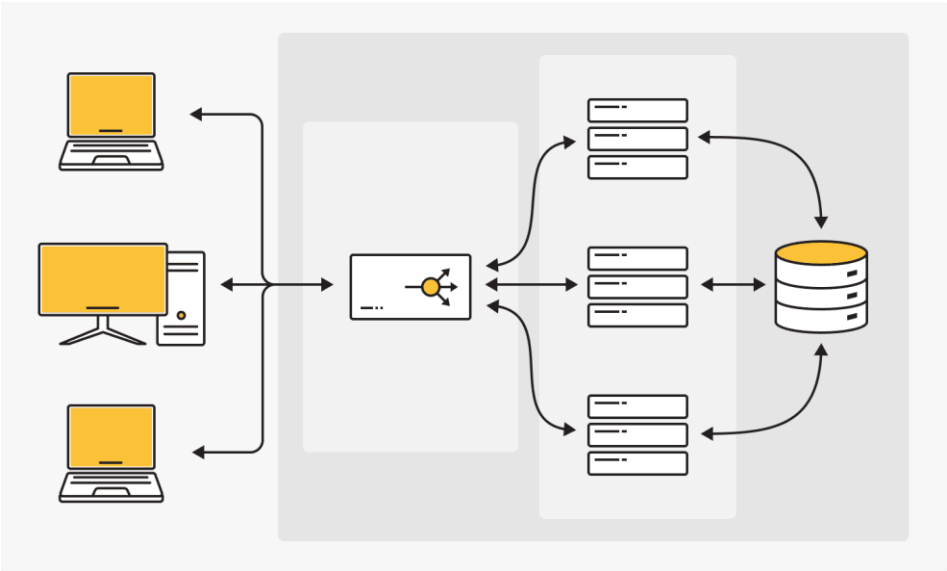

Рисунок 1.1 – Схема серверного кластеру

Застосування кластерів підвищує надійність системи. У випадку відмови одного сервера чи компонента, інші сервери або компоненти можуть продовжувати працювати. Це запобігає відмовам у обслуговуванні та гарантує безперервну доступність для користувачів.

Через їх здатність забезпечувати масштабування, надійність і ефективність у роботі з великою кількістю користувачів, враховуючи постійну потребу в обчислювальних ресурсах, серверні кластери стають важливими для мережі Інтернет.

1.2 Масштабованість

Масштабування системи - це процес розширення її функціональних можливостей для забезпечення оптимальної продуктивності при зростанні обсягів даних, користувачів або навантаження на систему. У сучасному світі, де величезні обсяги даних постійно збільшуються та вимоги користувачів швидко зростають, масштабування є важливим аспектом.

Кластеризація серверів або систем серверів є ключовим механізмом для масштабування. Вона полягає в об’єднанні кількох серверів у єдину систему, яка працює як єдиний обчислювальний ресурс. Це дозволяє розподіляти завдання між різними серверами, збільшуючи загальну обчислювальну потужність та надійність.

При зростанні користувачів чи обсягу даних, одного окремого серверу може бути недостатньо, для ефективної обробки всіх запитів. Тут важливою стає можливість додавати нові сервери до кластера для розподілу завдань та навантаження. Це забезпечує не лише більшу потужність, а й більшу стабільність, бо якщо один сервер відмовить, інші можуть взяти на себе частину роботи[1].

Масштабування системи та розподіл навантаження через серверний кластер забезпечують підвищення ефективності, швидкості обробки запитів та робочої стійкості, відповідаючи зростаючим потребам користувачів та обсягам даних.

1.3 Стійкість до відмови системи

Відмовостійкість серверного кластеру - це ключовий аспект в його роботі, оскільки жодна система не є повністю невразливою до можливих відмов чи випадкових проблем. Однак, застосування серверних кластерів може значно підвищити стійкість та надійність.

Кластеризація серверів дозволяє розділити завдання між декількома серверами. Якщо один сервер відмовить чи вийде з ладу, інші сервери можуть продовжувати працювати, взявши на себе частину навантаження. Це дозволяє системі продовжувати роботу навіть при виникненні проблем на окремих серверах.

Важливим аспектом відмовостійкості є також резервне зберігання даних. Кластери, зазвичай, використовують методи резервного копіювання та реплікації даних. Це означає, що якщо одне сховище даних вийде з ладу, інші резервні копії можуть бути використані для продовження роботи без втрати інформації. Використання серверних кластерів дозволяє працювати в мережі при будь-яких обставинах. Вони забезпечують надійний захист від відключень, якщо відбулися ті чи інші проблеми з енергозабезпеченням [2].

Крім цього, відмовостійкість передбачає автоматизоване виявлення відмов та швидке відновлення роботи. У випадку відмови окремого сервера, система керування кластером може перенаправити трафік на інші робочі сервери, щоб уникнути перерви у роботі.

Отже, відмовостійкість серверних кластерів забезпечує стабільну та надійну роботу системи, навіть у випадках можливих відмов чи проблем на окремих серверах.

1.4 DDoS

Кластери серверів, як системи з об’єднаними ресурсами, грають ключову роль у відповіді на атаки, зокрема DDoS. Забезпечуючи розподіл навантаження, вони можуть реагувати на великий об’єм запитів, розподіляючи їх між вузлами кластера. Це дозволяє зберігати доступність сервісу для користувачів, навіть у випадку спроб перенапруження.

Багато кластерних систем мають вбудовані засоби захисту, які дозволяють виявляти та фільтрувати шкідливий трафік, такий як пакети DDoS, перед тим як вони досягнуть серверів. Вони також можуть блокувати IP-адреси, що здійснюють атаку.

Резервні сервери, які часто входять до складу кластерів, грають важливу роль у стійкості до атак. Вони можуть перехоплювати трафік у випадку відмови основних серверів, що забезпечує доступність системи навіть під час атак.

Моніторинг та аналіз аномалій також важливі. Вони дозволяють виявляти незвичайну активність та сприяють вчасній реакції на атаки, що допомагає уникнути або пом’якшити їх наслідки.

1.5 Безпека даних, резервне копіювання

Забезпечення безпеки даних у серверних кластерах - це надзвичайно важлива складова. Вона включає наступні аспекти:

- Аутентифікація та авторизація: Гарантування тільки обмеженого доступу до серверів і баз даних тим, хто має на це право. Це зазвичай виконується через паролі, ідентифікаційні токени, ключі або біометричні дані.
- Шифрування: Застосування методів шифрування для захисту передачі та зберігання даних. Шифрування забезпечує захист даних від несанкціонованого доступу навіть у випадку проникнення.
- Захист від вторгнень: Встановлення систем виявлення та запобігання вторгнень для негайного реагування на потенційні загрози безпеки.
- Резервне копіювання: Регулярні та систематичні резервні копії даних, щоб в разі втрати інформації або витоку забезпечити можливість відновлення інформації.
- Оновлення та патчі: Регулярне оновлення програмного забезпечення, включаючи операційні системи та програми, для усунення вразливостей і вдосконалення захисту.
- Моніторинг і аудит: Систематичне спостереження за даними, доступом та подіями для виявлення аномалій або підозрілих дій, а також аудит дій, щоб виявити порушення безпеки.
- Цілісність даних: Захист цілісності даних для запобігання їх незаконним змінам, видаленню або порушенню.

Це досягається за допомогою різних методів, включаючи захист мережі, використання механізмів аутентифікації та авторизації, а також шифрування даних.

Іншим важливим аспектом є попередження втрати даних. Тут на перший план виходять регулярні резервні копії, які забезпечують можливість відновлення інформації у випадку непередбачуваних ситуацій. Крім цього, використання механізмів моніторингу і виявлення відмов допомагає ранньому виявленню проблем та швидкому їх вирішенню.

Також важливою є стійкість до атак. Системи кластеризації можуть розподіляти навантаження, що допомагає попереджати атаки типу DDoS та забезпечувати нормальну роботу в умовах високих навантажень.

Ще одним аспектом є реагування на загрози. Інтегровані системи безпеки допомагають виявляти та реагувати на потенційні загрози в реальному часі, що є критичним для забезпечення безпеки.

Отже, безпека серверних кластерів - це комплексний підхід, що об’єднує в собі заходи забезпечення доступності, конфіденційності та цілісності даних, а також захист від потенційних загроз і відмов.

Таким чином, наразі розробка та впровадження системи кластеризації серверів є важливим завданням, що значно поліпшить інфраструктуру, підвищить її відмовостійкість та ефективність у сучасному інформаційному світі.

1.6 Переваги використання кластерів

- Підвищення продуктивності. Кластери серверів можуть виконувати завдання швидше, ніж один сервер, оскільки завдання можуть розподілятися між декількома серверами. Це пов’язано з тим, що сервери можуть працювати паралельно, а не послідовно.
- Відмовостійкість. Кластери серверів можуть продовжувати працювати навіть у разі збою одного або декількох серверів. Це досягається за рахунок використання різних технологій, таких як балансування навантаження, резервне копіювання даних та реплікація даних.
- Доступність. Кластери серверів можуть забезпечувати безперервну роботу систем, навіть у разі запланованих або позапланових відключень. Це досягається за рахунок використання резервних

серверів, які можуть взяти на себе навантаження в разі збою основного сервера.

1.7 Сфери використання серверних кластерів

Приклади використання серверних кластерів:

- Банки використовують серверні кластери для обробки платежів та управління ризиками. Кластери серверів дозволяють банкам обробляти великі обсяги платежів за короткий час, а також забезпечувати безперервність роботи системи навіть у разі збою одного або декількох серверів.
- Роздрібні торговці використовують серверні кластери для обробки замовлень та управління запасами. Кластери серверів дозволяють роздрібним торговцям обробляти велику кількість замовлень та забезпечувати доступність веб-сайтів навіть у пікові періоди.
- Енергетичні компанії використовують серверні кластери для управління мережами та забезпечення безперервності електропостачання. Кластери серверів дозволяють енергетичним компаніям контролювати роботу електромереж та забезпечувати безперервне електропостачання навіть у разі аварій.
- Медичні установи використовують серверні кластери для зберігання медичних даних та забезпечення доступу до них. Кластери серверів дозволяють медичним установам зберігати великі обсяги медичних даних та забезпечувати доступ до них медичним працівникам в будь-який час.
- Збройні сили використовують серверні кластери для управління системами озброєння та забезпечення безпеки. Кластери серверів дозволяють збройним силам управляти складними системами озброєння та забезпечувати безпеку своїх мереж [5].

1.8 Існуючі рішення та пропозиції

Під час дослідження існуючих рішень, було виділено кілька пропозицій, котрі доступні у мережі для придбання.

Supermicro Hadoop

Сервіс solutions.asbis.ua представляє кластери Supermicro Hadoop - серію оптимізованих рішень для великих даних, які забезпечують високу продуктивність, високу надійність і високу масштабованість. Рішення Supermicro Hadoop є повністю інтегрованими, повністю оптимізованими і повністю протестованими кластерами «під ключ» з гнучкими пакетами підтримки, доступними для задоволення специфічних вимог замовника.

Кластери Supermicro Hadoop є перевірені в галузі сервери обчислень і зберігання даних високої щільності, заповнені кращими в своєму класі компонентами, обраними завдяки великому технічного проектуванню, валідації та тестування. Сертифіковані конфігурації дозволяють зробити припущення про розробку і розгортання по-справжньому масштабується інфраструктури обчислень і зберігання великих даних, яка відповідає самим вимогливим корпоративним IT-середам і центрам обробки даних [3].

Сервіс Hostpro.ua раніше представляв послуги кластеру, яка передбачає використання перехресних кабельних з’єднань. У цьому випадку сервери утворюють кластер за допомогою спеціальних з’єднань між самими серверами. Для цього потрібен спеціальний софт і не потрібні додаткові апаратні витрати або зовнішні пристрої.

Для всіх зовнішніх користувачів кластер виглядає як один сервер зі статичною IP- адресою. Запити обробляються основним сервером, при цьому всі операції відображаються (mirroring) на іншому сервері за допомогою спеціального блокового пристрою DRBD. Так, додатковий сервер має всю ту ж інформацію, що і основний, і в будь-який час може

«підставити плече», якщо основний сервер впаде. Для моніторингу стану сервера і перемикання виконання завдань на додатковий сервер використовується програмне забезпечення HeartBeat.

Таке рішення набагато надійніше звичайного хостингу, для якого падіння сервера може спричинити кілька годин даунтайму у зв'язку з відновленням усіх даних. Розміщені ж на кластері сайти постійно в мережі, оскільки додатковий сервер бере на себе обслуговування всіх запитів.

В даний час дане рішення більше не надається до придбання [4].

1.9 Висновки до першого розділу

Серверні кластери є потужним інструментом, який може використовуватися для підвищення продуктивності, відмовостійкості та доступності систем. Кластери серверів широко використовуються в різних галузях і можуть принести значні вигоди компаніям і організаціям.

Актуальність реалізації систем серверного кластеру полягає у тому, розробка серверного кластеру є актуальним рішенням для підвищення доступності, масштабованості та безпеки веб-сервісів чи додатків. Це також дозволяє ефективно використовувати ресурси, забезпечуючи високу продуктивність та гнучкість в управлінні навантаженням. Завдяки можливості кластеризації серверів розміщених у дата-центрах, розгортка кластеру може оптимізувати роботу для користувачів у різних частинах світу.

2 РОЗДІЛ

ТЕОРЕТИЧНА ЧАСТИНА

2.1 Основні напрямки вирішення задачі

Сучасні веб-сервіси та додатки потребують надійної та гнучкої інфраструктури, щоб забезпечити стабільну роботу та ефективну обробку даних. Створення систем, здатних обробляти високі навантаження, забезпечувати відмовостійкість і масштабованість, є важливим завданням у цій галузі.

Наразі інфраструктура серверів стикається з безліччю проблем, включно з ефективним розподілом навантаження між серверами, забезпеченням безперервної роботи в разі відмови основного сервера, централізованим управлінням даними та їхньою безпекою. Конвенціональні системи не завжди дають найкраще розв’язання цих проблем, тому що вони вимагають додаткових рішень або мають обмежену масштабованість.

У цій ситуації створення системи кластеризації серверів, запропонованої в цьому проекті, є рішенням. Ця система допомагає веб-сервісам і додаткам працювати краще, розподіляючи навантаження між серверами, забезпечуючи відмовостійкість і полегшуючи централізоване управління даними.

З урахуванням тенденцій, що існують у сфері розв’язання подібних проблем, можна виокремити кілька основних напрямів:

- Хмарні технології: Популярність кластеризації серверів у хмарі зростає. Хмарні провайдери виділяються гнучкістю, масштабованістю та керованістю. З огляду на гнучкість

масштабування і простоту використання, багато підприємств і підприємців вибирають хмарні рішення для своєї інфраструктури.

- Організація та оркестрація: Використання систем оркестрації та контейнерів (наприклад, Docker) забезпечує високу ізоляцію, масштабованість і можливість управління безліччю контейнерів. Це полегшує розгортання і масштабування додатків і дає змогу оптимізувати використання ресурсів.
- Використання штучного інтелекту: використання AI і ML для оптимізації роботи кластерів серверів. Сучасні інфраструктури залежать від аналізу даних для прогнозування навантажень і оптимізації ресурсів.
- Безпека: Збільшення загроз кібербезпеки вимагає створення захищених кластерів серверів. Шифрування, контроль доступу та моніторинг безпеки стають важливими компонентами сучасних рішень.
- Гібридні рішення: Компаніям дедалі більше подобається використовувати гібридні моделі, які об’єднують локальну та хмарну інфраструктуру. Це дає змогу балансувати між гнучкістю хмари та контролем над локальними системами й оптимізувати використання ресурсів.
- Постійний прогрес у розробці архітектурних рішень: Впровадження мікросервісної архітектури, наприклад, дає змогу створювати більш масштабовані, гнучкі та стійкі системи.

Ці тенденції відображають напрямки розвитку технологій серверної інфраструктури та надають великий огляд сучасних методів і рішень у цій галузі [6].

Вирішення наявних проблем серверної інфраструктури, щоб задовольнити сучасні вимоги до ефективності, відмовостійкості та масштабованості веб-сервісів і додатків, визначає важливість цієї роботи.

2.2 Визначення серверного кластера

Кластер серверів - це група серверів, що працюють разом в одній системі, щоб забезпечити користувачам більш високу доступність. Ці кластери використовуються для скорочення часу простою, дозволяючи іншого сервера продовжувати роботу в разі збою. Група серверів підключена до однієї системи. У той момент, коли один з цих серверів ставати недоступним, робоче навантаження перерозподіляється на інший сервер до того, як клієнт відчуває будь-який час простою [2].

Типовий серверний кластер складається з низки взаємопов’язаних серверів, які обробляють та розподіляють завдання між собою. Кластер може використовувати різні технології для забезпечення рівномірного розподілу завдань, включаючи балансування навантаження, резервне зберігання даних та механізми автоматичного відновлення після відмов.

Основна мета кластера - забезпечити високу доступність, швидкодію та масштабованість. Це досягається завдяки можливості роботи кількох серверів як єдиного механізму, що спільно обробляють дані та завдання.

Кластери можуть бути налаштовані для автоматичне виявлення та усунення відмов, що дозволяє забезпечити безперервну роботу системи навіть при випадкових помилках або відмовах окремих серверів.

Ця модель є важливою для багатьох сфер, особливо в онлайн-сервісах, електронній комерції та інших галузях, де важливо забезпечити стабільну та надійну роботу системи, навіть при великому навантаженні або випадкових відмовах.

Є кілька основних компонентів класичного кластера серверів:

- Взаємодія та спільна робота: Кластер серверів дозволяє декільком серверам працювати разом, об’єднуючи їхні ресурси для забезпечення вищої доступності, швидкодії або робочого ресурсу.

- Балансування навантаження: Балансувальники навантаження дозволяють розподіляти трафік між різними серверами в кластері, що покращує продуктивність і оптимізує використання ресурсів.
- Реплікація даних: Реплікація даних дозволяє копіювати дані на кілька серверів одночасно, що збільшує резервне збереження та надійність системи.
- Спільне зберігання ресурсів: Кластери серверів можуть мати спільний доступ до ресурсів за допомогою спільного зберігання, такого як розподілена файлова система або централізоване сховище даних.
- Резервне копіювання та відновлення: Кластери серверів можуть мати механізми автоматичного резервного копіювання та відновлення даних, що дозволяє зберігати дані та дозволяє швидко відновити їх після виникнення проблем.
- Висока доступність: кластери серверів зменшують ймовірність відмови сервера завдяки наявності резервних пристроїв, які відповідають за обробку запитів у випадку неполадок.

Формування кластерів серверів може відрізнятися залежно від вимог проекту та технологій, які використовуються.

Серверні кластери можуть бути побудовані на основі різних серверів, з різними конфігураціями та апаратним забезпеченням. Вони можуть бути розташовані в одному місці або в різних географічних регіонах (локаціях) [1].

2.3 Види кластерів

Основні види кластерів серверів:

- Високопродуктивні кластери (HPC) призначені для виконання ресурсоємних завдань, таких як обробка великих даних, машинне навчання та штучний інтелект. Кластери HPC можуть складатися з

сотень або навіть тисяч серверів, які об’єднані високошвидкісними мережами.

- Відмовостійкі кластери (HA) призначені для забезпечення безперервної роботи систем навіть у разі збою одного або декількох серверів. Кластери HA використовують різні технології, такі як балансування навантаження, резервне копіювання даних та реплікація даних, для забезпечення безперервності роботи.
- Балансувальники навантаження (LB) призначені для розподілу навантаження між декількома серверами. Кластери LB використовуються для підвищення продуктивності систем, які отримують велику кількість запитів від користувачів [2].

2.4 Визначення веб-серверу

Обробка запитів, надсилання веб-сторінок, файлів та інших ресурсів користувачам через Інтернет називається веб-сервером. Ця система дозволяє користувачам взаємодіяти з веб-застосунками, дозволяючи їм переглядати веб-сторінки, отримувати доступ до інформації, виконувати операції та передавати через Інтернет інформацію.

Такі як: веб-браузери, які запитують доступ до веб-сторінок та інших ресурсів, веб-сервери відіграють важливу роль у виконанні запитів користувачів. Після отримання запиту веб-сервер обробляє запит і відправляє відповідний контент або ресурс клієнту.

Цей модуль може працювати з іншими частинами, такими як бази даних, програми та різноманітні серверні програми для обробки та передачі даних. Веб-сервери працюють за допомогою протоколів зв’язку, таких як HTTP (Hypertext Transfer Protocol) і HTTPS (HTTP Secure), які дозволяють клієнтам спілкуватися з сервером.

Основною метою веб-сервера є надання користувачам надійного та ефективного доступу до веб-ресурсів, керування запитами користувачів, обробка цих запитів і надання відповідей. Ці функції включають доступ до веб-сторінок, мультимедійних файлів, даних або будь-яких інших ресурсів, які можна знайти в Інтернеті [6].

Apache, Nginx, Microsoft IIS, LiteSpeed та багато інших є типовими веб-серверами. Кожен має свої характеристики, переваги та недоліки.

Одним із найпопулярніших веб-серверів є Apache, який відомий своєю гнучкістю та великою кількістю функцій. Він підтримує багато модулів і працює на багатьох операційних системах.

Nginx — це простий і ефективний веб-сервер, розроблений для високого рівня продуктивності при великих одночасних підключеннях. Він потребує менше ресурсів і зазвичай працює краще для статичного контенту.

Веб-сервер Microsoft IIS інтегрований у Windows і доступний на платформах Windows. Він є частиною Windows Server і підтримує кілька мов програмування.

Порівняльна характеристика веб-серверів Apache та Nginx.

Apache та Nginx є двома найпопулярнішими веб-серверами в світі. Вони обидва безкоштовні та відкритого коду, і їх можна використовувати для обслуговування веб-сайтів, веб-додатків та інших веб-сервісів.

Apache, також відомий як HTTP-сервер Apache, - це стандартне програмне забезпечення веб-сервера з відкритим вихідним кодом, розроблене і підтримуване Apache Software Foundation. Це один з найпопулярніших варіантів програмного забезпечення для веб-серверів, на який припадає 31,9 відсотка ринку веб-сайтів з відомими веб-серверами.

Веб-сервер Apache використовує архітектуру, керовану процесами, яка створює новий потік для кожного нового запиту. Вперше він був випущений в 1995 році під ліцензією Apache і перетворився на домінуючий веб-сервер в індустрії. Apache часто зустрічається як частина технологічного стеку LAMP (Linux, Apache, MySQL і PHP), що дозволяє створювати швидкі та гнучкі веб-сайти і додатки. Завдяки своїм унікальним можливостям, Apache є популярним серверним програмним забезпеченням з 1996 року, та завдяки багаторічній розробці - документації та інтегрованій підтримці з боку багатьох взаємодоповнюючих проектів. Apache, відомий своєю багаторічною історією, є потужним і надійним веб-сервером. Він може похвалитися тривалою історією розробки та підтримки, яка забезпечує його стабільність завдяки діяльній міжнародній спільноті розробників і користувачів, які відіграли значну роль у його розвитку.

Крім того, це ідеальний варіант для веб-серверів. Наприклад, він підтримує багато мов скриптів, таких як PHP, Python і Perl, і працює на основних операційних системах, таких як Windows і macOS. Крім того, Apache вже включений у всі основні дистрибутиви Linux, що полегшує розгортання веб-додатків. Переваги та недоліки веб-серверу Apache представлено у таблиці 2.1.

Таблиця 2.1. Переваги та недоліки веб-серверу Apache .

Переваги	Недоліки
Рішення з відкритим вихідним кодом, доступне для приватних осіб та організацій безкоштовно.	Ресурсоємне програмне забезпечення, яке може споживати більше процесора та пам’яті, ніж альтернативи.
Підтримує сторонні модулі, плагіни та розширення, які розширюють його функціональність.	Може бути не ідеальним для доставки статичного контенту через свою ресурсоємність.

Гнучке та надійне серверне програмне забезпечення з потужним досвідом тестування та розробки.	Складна конфігурація через велику кількість функцій.
Підтримка спільноти активних користувачів.	Відсутність підтримки асинхронної обробки.
Надає root-доступ до конфігурацій сервера; непривілейовані користувачі можуть редагувати дозволи за допомогою файлу .htaccess.	
Ідеально підходить для середовищ віртуального хостингу.	

NGINX - це веб-сервер, який використовує асинхронну архітектуру, керовану подіями, для доставки контенту клієнтам. Вперше його було створено у 2002 році, а згодом випущено для широкого загалу у 2004 році.

Як і Apache, NGINX є проектом з відкритим вихідним кодом, який можна вільно використовувати, модифікувати та поширювати. Неблокуюча архітектура NGINX, керована подіями, робить його ідеальним рішенням для обробки декількох запитів. NGINX працює шляхом створення одного процесу-контролера і декількох робочих процесів. Контролер забезпечує роботу системи, але кожен робочий процес виконує окремі завдання.

Оскільки кожен процес є асинхронним, робочі процеси можуть виконувати декілька запитів, не блокуючи інші запити. Незважаючи на велику кількість процесів, NGINX використовує мало пам’яті, що підвищує ефективність. Це робить NGINX ідеальним для веб-сайтів з високим

трафіком, надаючи йому перевагу над іншими веб-серверами. Проте, NGINX все ще має деякі спільні риси з Apache. Наприклад, NGINX може працювати з файловими системами HTTP, HTTPS, HTTP/2 і WebSocket, а також з термінацією SSL/TLS. Ви також можете використовувати NGINX, як зворотний проксі-сервер, який розподіляє клієнтські запити між декількома серверами.

Отже NGINX ідеально підходить для балансування навантаження, кешування, WebSockets і перезапису URL-адрес, що дозволяє адміністраторам змінювати вхідні запити та дозволяти або обмежувати доступ.

З великою кількістю модулів розширення NGINX забезпечує додаткову функціональність сайту. Крім того, він працює з усіма основними операційними системами Linux і Unix. Хоча його можна інсталювати на Microsoft Windows, він може бути менш ефективним, особливо при масштабуванні або роботі з UDP-аутентифікацією. Переваги та недоліки веб-серверу NGINX представлено у таблиці 2.2

Таблиця 2.2 - Переваги та недоліки веб-серверу NGINX

Плюси	Мінуси
Асинхронна та керована подіями архітектура робить NGINX	Обмежена підтримка операційної систем.
ідеальною для виконання декількох паралельних запитів.	
Ідеально підходить для серверів з високим трафіком.	Не має власної підтримки динамічного контенту і використовує проксі-запити для всього динамічного контенту на

	внутрішній сервер перед тим, як відправити його клієнту.
Може використовуватися як зворотний проксі-сервер.	Значною мірою покладається на використання зовнішніх модулів сторонніх розробників.
Малий обсяг пам’яті та ефективне використання процесора і пам’яті.	
Простий у налаштуванні та використанні, навіть для початківців.	
Активна підтримка спільноти та документація. [6]	

2.5 Система управління базами даних

Система управління базами даних (СУБД) — це програма, яка дозволяє створювати та керувати базами даних. Вона надає можливість створення, зміни та вилучення даних із структурованих баз даних.

СУБД дозволяє користувачам отримати доступ до даних за допомогою певного набору інструментів і запитів. Вона охороняє та організовує дані, контролює доступ до них і гарантує, що дані є безпечними та надійними завдяки механізмам резервного копіювання та відновлення.

Ця система використовується в багатьох сферах, від бізнесу до наукових досліджень, де організація, зберігання та доступ до великих обсягів даних є важливими. Основними завданнями бази даних є управління даними, забезпечення безпеки та збереження їх цілісності .

Крім реляційних СУБД, розрізняють такі типи, залежно від моделі БД:

- Ієрархічні – у яких інформація у базі даних зберігається як об’єкти, що об’єднані в деревоподібну структуру. У порівнянні з іншими типами

СУБД, ієрархічна СУБД має відносно невеликий арсенал операцій з маніпулювання даними (втім, цього достатньо для вирішення більшості завдань).

- Мережеві – СУБД, які як і ієрархічні мають деревоподібну структуру, але відрізняються від ієрархічних СУБД тим, що будь-який запис-нащадок може мати відразу кілька предків. Складність побудови такої БД компенсується гарними показниками швидкістю доступу та низькою витратою оперативної пам’яті.
- Об’єктно-орієнтовані (об’єктні) СУБД - бази даних, де кожен запис розглядають як окремий об’єкт із деяким набором властивостей, що взаємодіє з іншими об’єктами БД. У мові запитів реалізовано парадигму ООП - об’єкти, класи, успадкування.
- Об’єктно-реляційні СУБД - у яких об’єднано властивості реляційних та об’єктних СУБД.

ПЗ для роботи з базами даних може бути локальним або розподіленим. Локальні СУБД розміщують усі свої компоненти на одному комп’ютері, розподілені СУБД можуть розташовуватися на кількох робочих станціях [7]

.

2.5.1 Вибір СУБД для проекту

MySQL – це одна з найпопулярніших реляційних баз даних. Цю СУБД часто включають до складу серверів. MySQL працює на всіх основних платформах, включаючи Red Hat, Fedora, Ubuntu, Debian, Solaris, Microsoft Windows та Apple MacOS. Реалізована підтримка кількох процесорів та багатопоточність. Через API з базами даних можна працювати за допомогою популярних мов програмування (C, C++, C#, PHP, Java, Ruby, Python та Perl). Бібліотека функцій SQL реалізується через високо-оптимізовану бібліотеку класів, завдяки чому досягається висока швидкість роботи і відсутні витоки пам’яті. У MySQL реалізовано підтримку Novell Cluster.

Oracle RDBMS (Oracle Database) -це СУБД, яка ідеально підходить для великих компаній, які обробляють великі кількості даних. Вона доступна в редакціях від Enterprise Edition до Lite для мобільних пристроїв. Безкоштовна версія Express Edition (XE) має обмеження щодо оперативної пам'яті (1 ГБ), одного процесора та максимального об'єму БД для редакції 11 ГБ. У СУБД Oracle є інструменти управління базами даних «на всі випадки життя», що дозволяє автономне адміністрування. Досягається високий рівень безпеки, оскільки кожна транзакція проводиться окремо [8].

СУБД PostgreSQL. незважаючи на те, що СУБД є безкоштовним програмним забезпеченням, за впровадження, модифікацію та підтримку все ще потрібно платити.

PostgreSQL може працювати з базами даних та таблицями з будь-якою кількістю записів і індексів. Розширюваність СУБД полягає в тому, що вони можуть включати власні перетворення типів, доменів, індексів, операторів, процедурних мов і типів. Завантаження С-сумісних модулів підтримується. Складні та надійні механізми транзакцій і реплікації доступні для PostgreSQL. Масиви даних, включаючи багатовимірні, композитні типи та інші складні структури, а також бітові рядки та мережеві адреси доступні для PostgreSQL. СУБД підтримують бази NoSQL, XML і JSON [7]

Microsoft SQL Server

Досить популярна СУБД від Microsoft, яка орієнтована на платформу Windows. Управління БД здійснюється за допомогою процедурного розширення мови SQL Transact-SQL. Воно розширює можливості роботи з рядками, датами, математикою та іншими функціями, а також додає керуючі оператори до основного функціоналу мови запитів. Оскільки у проекті використовується програмне забезпечення з відкритим кодом, а саме Ubuntu Server 18.04, то дана СУБД не підходить [7].

MariaDB

MariaDB - це система управління базами даних, що походить від MySQL та розробляється спільнотою розробників. Вона використовується для зберігання та організації даних у веб-серверах. MariaDB має відкрите джерело, що означає, що код програми вільно доступний для використання та модифікації спільнотою користувачів [8]

Основною метою MariaDB є надання швидкого, надійного та ефективного зберігання даних у веб-серверних додатках. Вона підтримує багато різних операційних систем і мов програмування, що робить її популярним вибором для великої кількості розробників програмного забезпечення та веб-сайтів.

MariaDB також відома своєю відмінною швидкістю та можливостями оптимізації. Вона надає різні можливості для вирішення проблем скалабельності та безпеки даних.

Існуючі можливості серверних СУБД відбивають сучасні тенденції розвитку інформаційних систем, такі, як використання багатопроцесорних систем і розподіленої обробки даних, створення розподілених систем, зокрема з використанням технологій Internet, застосування засобів швидкої розробки додатків, створення систем підтримки прийняття рішень із використанням аналітичної обробки даних, а також усе більше підвищуються вимоги до надійності інформаційних систем [8].

2.5.2 Веб-інтерфейс phpMyAdmin

phpMyAdmin - Це основний інструмент для роботи з динамічними сайтами. Щоб сформувати контент, необхідно створити HTML-каркас та відобразити на сторінці за допомогою PHP скриптів інформацію з бази даних. Першою генерується користувацька база даних з привілеями адміністратора: можна створювати, видаляти, редагувати таблиці, додавати

нові рядки. Інтуїтивно зрозумілий інтерфейс не вимагає знання мови, достатньо розуміти синтаксис: Type, Table, Alter, Create. Потрібний рядок можна знайти за допомогою швидкого пошуку [9].

MySQL у поєднанні з phpMyAdmin - оптимальний вибір для проекту з кластером серверів. MySQL, як сучасна система управління базами даних, має відмінну продуктивність та надійність, дозволяючи забезпечити швидкий доступ до даних та ефективне їх управління. phpMyAdmin, з іншого боку, забезпечує зручний інтерфейс для адміністрування баз даних

MySQL через веб-браузер. Використання MySQL в поєднанні з phpMyAdmin дозволить ефективно керувати та моніторити дані в кластері серверів, забезпечуючи простий та доступний інтерфейс для роботи з базами даних [9].

2.6 PHP

PHP — це скриптова мова програмування, створена для генерації HTML-сторінок на стороні вебсервера. Спочатку PHP розшифровувалася як Personal Home Page, але нині її офіційна назва PHP: Hypertext Preprocessor.

PHP є однією з найпоширеніших мов, яку використовують у сфері веброзробки, її підтримує більшість хостинг-провайдерів. PHP інтерпретує вебсервер у HTML-код, який передається на сторону клієнта. На відміну від JavaScript, користувач не бачить PHP-коду, тому що браузер отримує готовий HTML-код. Це є перевагою з погляду безпеки, але погіршує інтерактивність сторінок [10].

PHP є скриптовою мовою програмування, що виконується на сервері, вона вбудована безпосередньо у веб-сторінки. Коли веб-сторінка з PHP-кодом відправляється на сервер, сервер обробляє PHP-код та генерує HTML, який повертається браузеру користувача. Це дозволяє розробникам

створювати веб-сторінки, які можуть динамічно змінюватися в залежності від взаємодії з користувачем, стану бази даних та інших умов.

PHP дозволяє взаємодіяти з різними системними ресурсами, такими як бази даних, файлові системи та мережеві порти, що робить його корисним для розробки різноманітних веб-додатків, від блогів до електронної комерції.

Оскільки PHP є відкритим і безкоштовним програмним забезпеченням, воно широко використовується у веб-розробці. Багато веб-серверів, таких як Apache, Nginx та IIS, підтримують PHP, дозволяючи розгортання веб-додатків, що використовують цю мову. Однією з його особливостей є легкість використання та широкий вибір фреймворків, що робить його популярним серед розробників.

2.7 Firewall

Firewall - це програмне забезпечення або апаратний пристрій, який контролює вхідний та вихідний мережевий трафік, фільтрує його і застосовує правила для блокування чи дозволу певних типів з’єднань на основі, заздалегідь визначених, правил безпеки. Фаєрволи використовуються для захисту серверів від шкідливих атак, таких як DDoS, SQL-ін’єкції, кросс-сайт скриптінг тощо. Вони визначають правила, які контролюють трафік, і можуть блокувати чи дозволяти з’єднання на основі таких параметрів, як IP-адреса, порти, протоколи тощо.

Є два типи фаєрволів: програмні та апаратні. Програмні фаєрволи запускаються як програмне забезпечення на сервері і контролюють трафік через операційну систему. Апаратні фаєрволи - це фізичні пристрої, які розміщені між сервером та мережею, і вони фільтрують трафік перед його досягненням сервера.

Фаєрволи встановлюють правила на основі безпеки та потреб системи. Вони можуть блокувати доступ до певних служб, портів чи IP-адрес, що зменшує ризик вразливості серверів. Основна мета фаєрволів - забезпечити безпеку мереж та серверів, запобігаючи несанкціонованому доступу та захистивши їх від потенційних загроз.

Iptables відомі для більшості системних адміністраторів. Він існує вже давно і включений за замовчуванням у ядрі Linux. Ми можемо використовувати iptables для блокування однієї IP-адреси, кількох IP-адрес або цілих мереж. Це може бути корисно, якщо ви отримуєте сканування портів, що повторюються, або є свідком невдалого несанкціонованого доступу до ваших файлів журналу. Блокування IP — ефективніший захід безпеки [11].

Використання iptables може бути наступним:

1. **Фільтрація пакетів**: Iptables може блокувати або дозволяти певні типи пакетів на основі IP-адреси, портів, протоколів тощо. Це дає можливість створювати правила для обмеження доступу до сервера зовнішніми користувачами.
2. **Network Address Translation (NAT)**: Ця функція дозволяє змінювати або переписувати IP-адреси та порти пакетів у мережі, що дозволяє, наприклад, приховати захищений сервер за публічною IP-адресою.
3. **Маскірування (Masquerading)**: Використовується для приховання справжніх IP-адрес клієнтів у мережі, дозволяючи їм отримувати доступ до ресурсів за межами локальної мережі.
4. **Створення тунелів та VPN**: Іноді iptables використовують для налаштування віртуальних приватних мереж (VPN) або тунелів між серверами.

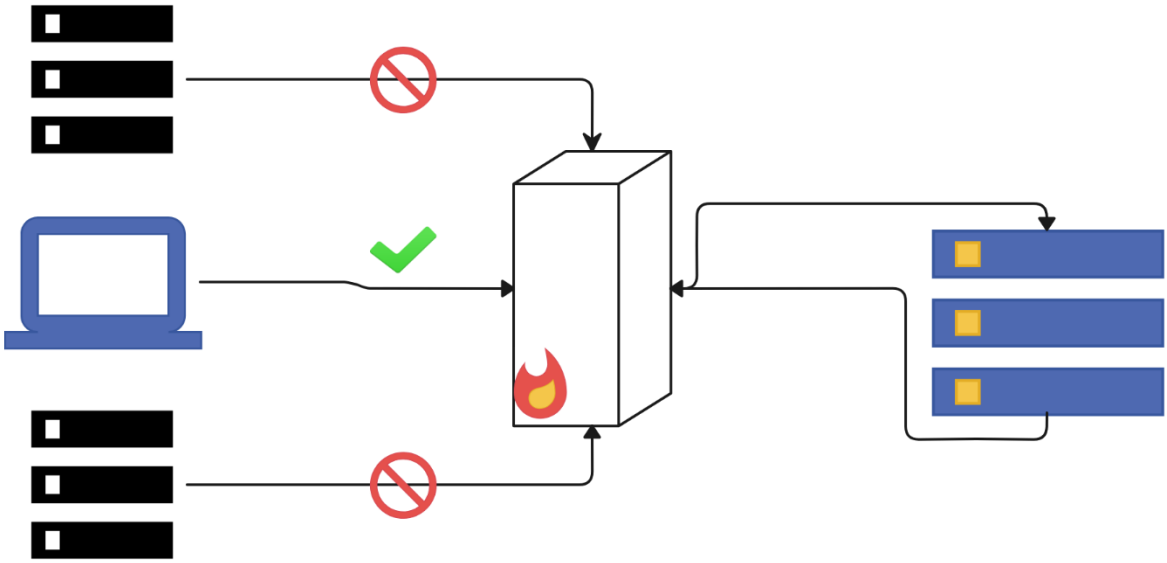

Рисунок 2.1 – Схема роботи фаєрволу

Iptables – це потужний інструмент, він може використовуватися для налаштування складних систем безпеки, але вимагає глибокого розуміння мережевих протоколів і принципів роботи мережі. Неправильне конфігурування iptables може призвести до блокування легітимного трафіку або створення вразливостей у мережевій безпеці.

2.8 Cron-задачі

Cron – це програма планування завдань у Unix-подібних операційних системах, яка дозволяє запускати автоматизовані завдання у визначені часи або періодично. Це потужний інструмент, який використовується для автоматизації виконання різних завдань на сервері з певною регулярністю.

Можливості Cron значно розширюються через можливість налаштування часових параметрів для запуску завдань. Завдяки різним командам і параметрам можна виконувати різноманітні завдання, такі як виконання скриптів, запуск програм, відправлення email-повідомлень та багато іншого.

Типові можливості Cron включають:

- Запуск команд і скриптів: Це найпоширеніший спосіб використання Cron. За допомогою нього можна запускати скрипти або команди в певні моменти часу або періодично.
- Періодичність: Cron дає можливість виконувати завдання на основі різних періодичних шаблонів - щоденно, щотижня, щомісяця тощо.
- Робота зі списками: можна використовувати списки значень для складніших графіків виконання завдань.
- Робота з файлами конфігурації: дає можливість налаштувати файл конфігурації Cron для кожного користувача окремо, дозволяючи різним користувачам запускати різні завдання.
- Логування і повідомлення про помилки: Cron може логувати виконання завдань та повідомляти про будь-які помилки, що виникають під час їх виконання.

Загалом, Cron відмінно підходить для автоматизації різноманітних процесів на серверах, що дозволяє виконувати завдання у встановлені часи або періодично, забезпечуючи стабільну та ефективну роботу системи.

2.9 Протоколи управління серверами

SSH (Secure Shell) та FTP (File Transfer Protocol) - це два різних протоколи, які використовуються для керування серверами та передачі файлів відповідно.

SSH - це криптографічний протокол, який дозволяє безпечно з’єднатися з віддаленим сервером та виконувати команди на ньому. Він забезпечує шифрування даних, аутентифікацію та захист від перехоплення інформації під час передачі між комп’ютерами. За допомогою SSH можна виконувати різноманітні операції на сервері, від встановлення програм до керування файловою системою.

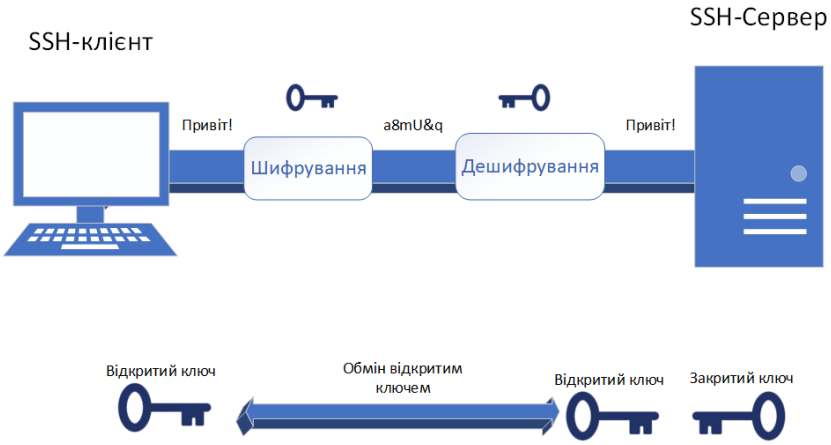

Рисунок 2.2 – Схема роботи протоколу SSH

FTP - це протокол передачі файлів, який використовується для завантаження та скачування файлів між локальним комп'ютером та віддаленим сервером. FTP не забезпечує шифрування за замовчуванням, тому він менш безпечний у порівнянні з SSH. Проте FTP є простим у використанні та підтримується багатьма програмами для передачі файлів.

Основна відмінність між цими двома протоколами полягає у безпеці та функціональності. SSH забезпечує безпеку під час з'єднання та передачі даних, тоді як FTP, якщо використовується без захисту, може бути менш захищеним. У випадку, якщо потрібно лише передавати файли, FTP може бути зручним варіантом. Однак, якщо потрібно виконати операції на сервері та забезпечити безпеку, то SSH є кращим варіантом.

FTPS та SFTP - це два протоколи, які використовуються для безпечної передачі файлів, але вони відрізняються за принципом роботи та рівнем безпеки.

FTPS (FTP over SSL/TLS) використовує SSL або TLS для шифрування з'єднання між клієнтом та сервером FTP. Це розширення протоколу FTP, яке надає захищений канал зв'язку для передачі файлів. Використання SSL/TLS

забезпечує шифрування даних під час передачі і вимагає встановлення сертифікатів для аутентифікації сервера.

SFTP (SSH File Transfer Protocol) використовує SSH для безпечної передачі файлів між клієнтом та сервером. Це побудовано на базі SSH, тому використовує той самий захищений канал зв'язку. SFTP відрізняється від

FTPS тим, що використовує одне з'єднання для передачі даних та управління, що робить його більш ефективним у використанні та безпеці.

Головна відмінність між FTPS та SFTP полягає у протоколі, на якому вони базуються. FTPS використовує захищений канал через SSL/TLS на базі протоколу FTP, тоді як SFTP використовує SSH для передачі файлів. Ці протоколи забезпечують безпечність передачі даних, але SFTP часто вважається більш простим у використанні та управлінні.

2.10 Load Balancer

У серверних кластерах, Load Balancer - це компонент, який відповідає за розподіл навантаження між різними серверами у кластері. Основна мета Load Balancer у контексті кластера полягає в забезпеченні рівномірного розподілу трафіку та завдань між усіма серверами у кластері.

Це допомагає уникнути перенавантаження будь-якого окремого сервера та підвищує ефективність виконання запитів користувачів. Load Balancer може використовувати різні алгоритми для розподілу навантаження, такі як Round Robin (циклічний розподіл), Least Connections (розподіл за найменшою кількістю активних з'єднань), IP Hashing (розподіл за IP-адресою), шляхом аналізу змісту запиту та інші.

Цей елемент забезпечує надійність та стійкість системи, оскільки навантаження рівномірно розподіляється між серверами, навіть якщо один або кілька серверів відмовлять у роботі. Такий підхід забезпечує неперервну

доступність сервісу або веб-сайту, що важливо у великих мережевих середовищах.

2.11 Системи моніторингу

Системи моніторингу - це програмне забезпечення, що використовується для нагляду та відстеження працездатності, ефективності та безпеки комп'ютерних систем, мереж та програм. Основна мета таких систем - збір, аналіз та візуалізація даних про стан системи, компонентів, ресурсів та мережі.

Системи моніторингу включають в себе інструменти для:

- Збору даних: Вони збирають інформацію про різні параметри, такі як навантаження CPU, використання пам'яті, обсяг мережевого трафіку, стан дискових просторів та інші метрики з різних пристроїв та сервісів.
- Моніторингу та відстеження: Системи виявляють аномалії, проблеми та попереджають про можливі недоліки або відмови в системі шляхом моніторингу надходження даних.
- Аналізу та візуалізації: Вони допомагають у відображенні даних у зручному для сприйняття вигляді: графіки, діаграми, таблиці, що дозволяє швидше виявляти проблеми або аномалії.
- Сповіщення та автоматизації: Вони надсилають сповіщення адміністраторам про виявлені проблеми та можуть виконувати автоматичні дії для відновлення працездатності.

Ці системи використовуються для підтримки оптимальної працездатності серверів, мереж та програм, а також для попередження можливих проблем, що дозволяє оперативно реагувати та забезпечувати безперебійну роботу інфраструктури.

2.12 Резервне копіювання

Резервне копіювання файлів системи та дампів баз даних - це критичний аспект управління даними для будь-якої інформаційної системи. Важливість цього процесу полягає в забезпеченні безпеки та можливості відновлення даних в разі втрати чи пошкодження оригіналів. Для цього використовують різні методи та інструменти, включаючи скрипти, які автоматизують процес резервного копіювання.

Скрипти для резервного копіювання можуть виконувати різні завдання, такі як створення архіву з файлами системи, копіювання цього архіву на віддалені сервери, збереження дампів баз даних, регулярне створення резервних копій з можливістю налаштування часу виконання.

Важливість резервного копіювання полягає в тому, що воно забезпечує:

- Безпеку даних: Якщо відбудеться втрата чи пошкодження даних, резервні копії, дозволяють відновити інформацію до стану на момент останнього резервного копіювання.
- Надійність системи: Відновлення даних із резервних копій дозволяє швидко відновити роботу системи після випадків втрати чи пошкодження даних.
- Забезпечення бізнес-процесів: Резервне копіювання дозволяє підтримувати безперебійну роботу бізнес-процесів у випадку виникнення проблем.

Користуючись скриптами для резервного копіювання, адміністратори можуть автоматизувати процес створення резервних копій та забезпечити безпеку даних та стабільну роботу системи.

2.13 Заходи безпеки серверу

Щоб захистити Linux-сервери від потенційних загроз, система безпеки має низку важливих елементів. Користувачі, які не є адміністраторами, є першим рівнем захисту. Кожен користувач має обліковий запис з обмеженими привілеями, що допомагає уникнути небезпечних дій у випадку компрометації даних.

Застосування фаєрволів, таких як iptables, UWF (Uncomplicated Firewall) або firewalld, є важливим для управління трафіком у мережі. Це дозволяє встановити правила фільтрації трафіку та блокування підозрілих з’єднань, що зменшує ймовірність небезпеки.

Зміна звичайного порту SSH (22-й) на інший може підвищити безпеку, оскільки це може зробити більш складним для зловмисників знайти службу SSH на сервері. Використання SSH-ключів для аутентифікації, а не паролів, також забезпечує більшу безпеку.

Крім того, важливо регулярно оновити програмне забезпечення та операційну систему. Моніторинг системних журналів допомагає виявляти підозрілу активність і потенційні загрози, щоб швидко реагувати та вирішувати проблеми.

2.14 Визначення віртуального виділеного сервера (VPS)

Віртуальні сервери — це програмні області, які працюють як окремі фізичні сервери, але використовують ресурси спільного обладнання (Фізичного виділеного сервера). Однією фізичною машиною можна керувати кількома віртуальними серверами, кожен з яких відокремлений від інших і має свою операційну систему, програми та налаштування.

Віртуалізація — це технологія, яка ділить фізичний сервер на окремі віртуальні копії. Кожен віртуальний сервер може мати власні

програми, файли, конфігурації та системні ресурси, такі як процесор, оперативна пам'ять і дисковий простір. Розподіл ресурсів фізичного сервера між віртуальними серверами дозволяє максимізувати їх використання.

Віртуальні сервери мають багато переваг, включаючи ефективне використання обладнання, легкість масштабування, можливість резервування та відновлення, а також ізоляцію ресурсів для безпеки та стабільності. Вони добре сприймаються у великих підприємствах, веб-хостингах та інших сферах, де необхідно ефективно використовувати ресурси сервера та гарантувати надійну роботу окремих сервісів чи додатків.

2.15 Типи віртуалізації

Технологія, відома як віртуалізація, яка дозволяє створювати віртуальні варіанти операційних систем на фізичній машині. Паравіртуалізація та повна віртуалізація є двома основними підходами до віртуалізації.

Паравіртуалізація – це метод, який дозволяє гостям операційних систем працювати під керуванням гіпервізора, надаючи їм доступ до обладнання через інтерфейс, адаптований до віртуальних машин. Система паравіртуалізації OpenVZ дозволяє створювати віртуальні простори для користувачів на одній фізичній машині. Хоча кожен контейнер має свою власну операційну систему, вони використовують спільне ядро операційної системи господаря.

Повна віртуалізація надає гостьовій операційній системі оточення, подібне до того, яке є в реальних фізичних машинах. KVM — це технологія повної віртуалізації, вбудована в ядро Linux і призначена для створення віртуальних машин на базі Linux. KVM підтримує апаратну віртуалізацію за допомогою спеціального модуля ядра. Кожен образ використовує власне

ядро операційної системи, незалежно від гіпервізора та інших гостьових систем.

У порівнянні з OpenVZ, KVM кращий через можливість використовувати різне програмне забезпечення, підтримку різних операційних систем, ізоляцію ресурсів між віртуальними машинами та більшу надійність завдяки реальній ізоляції ядра операційної системи

2.16 Вибір операційної системи

Для встановлення операційної системи на віртуальний виділений сервер, хостинг провайдер зазвичай надає наступні популярні ОП операційні системи: Almalinux, CentOS, Debian, Ubuntu.

CentOS

CentOS 7 та CentOS 8 - це операційні системи з відкритим вихідним кодом, що базуються на вихідних кодах Red Hat Enterprise Linux (RHEL). CentOS (Community ENTerprise Operating System) спроектована для надання безкоштовної, стабільної та надійної платформи для серверів та робочих станцій.

CentOS 8, замінивши CentOS 7, приніс оновлення та нові функції. Однією з головних особливостей була нова система управління пакунками - dnf. Крім того, CentOS 8 був орієнтований на впровадження контейнеризації, включаючи підтримку Docker та Kubernetes.

Обидві версії CentOS були відомі своєю стабільністю, довгостроковою підтримкою та безпекою, що робило їх популярними серед адміністраторів систем і розробників програмного забезпечення. Однак, після анонсу змін у політиці підтримки CentOS у 2020 році, корпорація Red Hat припинила розробку CentOS Linux, що призвело до появи проектів спільноти, таких як AlmaLinux, Rocky Linux, які постають як альтернативи після цієї зміни [12].

AlmaLinux

AlmaLinux 8 і 9 – це операційні системи, що базуються на відкритих вихідних кодах CentOS 8 і 9 після переходу корпорації Red Hat від підтримки CentOS. Вони створені з метою надання стабільної та надійної альтернативи користувачам, які раніше використовували CentOS і були звиклими до його функціоналу та сумісності з Red Hat Enterprise Linux (RHEL).

AlmaLinux спрямований на комерційний сектор та бізнес-сегмент, надаючи довгострокову підтримку, що робить його відмінним вибором для серверних рішень. Одним із ключових елементів є гарантована сумісність зі стандартами RHEL, що робить міграцію з CentOS на AlmaLinux максимально безболісною [13].

Основні риси AlmaLinux полягають у надійності, безпеці та стабільності. Вона має високу сумісність з RHEL у термінах бінарних пакунків, тому практично всі програми, які працюють на RHEL, також працюватимуть на AlmaLinux без будь-яких модифікацій або адаптацій. Також AlmaLinux надає довгострокову підтримку пакунків, що дає можливість розраховувати на стабільну і надійну роботу системи протягом тривалого часу.

Debian

Debian -- масштабний проект, метою якого є створення універсальної вільної операційної системи та програмної інфраструктури для неї. Офіційна назва -- Debian GNU/Linux -- визначає Debian, як систему, що працює на ядрі Linux та в більшості використовує програмне забезпечення, розроблене в рамках проекту GNU Фундацією Вільного Програмного Забезпечення (FSF, Free Software Foundation). На сьогодні, разом з ядром та базовою системою, що формують власне операційну систему, в Debian входять понад 18,000 пакунків різноманітного програмного забезпечення.

Пошук розробниками загальноприйнятих та універсальних рішень найрізноманітніших проблем дозволило Debian, на відміну від інших дистрибутивів, працювати не лише на ядрі Linux: існують успішні проекти перенесення Debian на ядра Hurd, FreeBSD, NetBSD, тощо.

Найкраща система керування пакунками в світі дозволяє за лічені хвилини встановити потрібну програму без розбирання складних залежностей. Крім того, є витончені інструменти, які дозволяють користувачам змінювати або складати програми відповідно до їхніх потреб. Будь-яка програма легко може бути оптимізована для роботи на певному обладнанні.

Існує велика кількість захоплюючих і різноманітних проектів на основі Debian, що свідчить про те, наскільки популярний він як серед розробників, так і серед користувачів. З них найвідомішими є Knoppix, найпоширеніший Live-CD-дистрибутив, і Ubuntu, який на сьогодні вважається одним із найзручніших Linux-дистрибутивів.

Кількість користувачів Debian постійно зростає. Завдяки цьому ви завжди зможете знайти підтримку у своєму прагненні ознайомитись з цією гнучкою та технологічно довершеною операційною системою [14].

Кожна нова версія Debian намагається вдосконалити функціонал, забезпечити більшу стабільність та безпеку для користувачів. Debian використовується в різних сферах, включаючи сервери, настільні комп'ютери та вбудовані системи, завдяки своїй надійності та універсальності.

Ubuntu server

Дистрибутив Ubuntu був створений з метою стати зручною та зрозумілою ОС (на основі Linux) для рядового користувача. В Мережі існує велика кількість інструкцій та порад щодо користування системою для новачків. За офіційними даними дистрибутивом Ubuntu користуються понад

20 млн користувачів. Нові релізи постійно випускаються як з довгостроковою Ubuntu LTS, так і з короткостроковою підтримкою. Частина застарілих версій не підтримуються розробниками.

Для більш легкої роботи з операційними системами на сервері використовуються панелі керування, тому при роботі з Ubuntu на сервері ви також можна встановити додатково панель керування, яка спростить роботу [15].

Серед наведених дистрибутивів Linux, вибір операційної системи було зроблено саме на Ubuntu 18.04 64bit Minimal. Операційна система Ubuntu має зручні інструменти для експлуатації, широкий спектр документацій для використання утиліт та програм, а також відповідає мінімально рекомендованим технічним характеристикам серверів, котрі використовувались під час виконання кваліфікаційної роботи. Досвід роботи з даною операційною системою надав змогу швидко опанувати та вдосконалити знання базовим інструментарієм, що надало змогу використовувати сервери без додаткового встановлення платної панелі управління, тим самим зменшивши собівартість проекту.

2.17 Висновки до другого розділу

Впровадження серверного кластеру та застосування сучасних веб-серверів, фаєрволу iptables та автоматизованого резервного копіювання є важливими стратегіями для забезпечення надійності, безпеки та ефективності функціонування серверної інфраструктури. Враховуючи основні аспекти захисту, оптимізації та автоматизації, серверний кластер буде спрямовано на підвищення відмовостійкості. Використання сучасного програмного забезпечення з відкритим кодом, надає змогу розміщувати проекти та веб застосунки, що потребують безперебійної роботи серверу, який підтримує використання сучасних технологій.

3 РОЗДІЛ

МОДЕЛІ ТА МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ

3.1 Структура серверного кластеру

Розподіл навантаження та висока доступність ресурсів є основними принципами структури серверного кластера. У цьому випадку на кластері є п'ять серверів. Це містить балансувальника, який відповідає за розподіл запитів на основний або резервний сервер, залежно від того, наскільки вони доступні.

На основному сервері розташований сайт, який обробляє та надсилає дані користувачам. Резервний сервер працює в іншому місці і забезпечує резервну копію сайту в разі відмови основного сервера. На кластері також є сервер з базою даних, який підключений до основного та резервного серверів, що дозволяє користувачам завжди мати доступ до даних. Завдяки цій структурі можна забезпечити швидку та безперебійну роботу на веб-сайті, а також підвищити його продуктивність.

Можна розширити даний кластер додаванням додаткових серверів, які можуть обробляти дані, забезпечувати резервні копії або балансувати навантаження. Крім того, використання резервного сервера в іншому місці захищає основний сервер від можливих аварій або виключень. Щоб забезпечити стійку та ефективну роботу сайту, усі сервери кластера працюють разом.

Загалом, ця структура серверного кластеру забезпечує високу доступність сайту та відмовостійкість. Це означає, що сайт продовжує працювати навіть у разі відмови окремих серверів завдяки роботі резервних серверів і балансувальника. Такі системи можна назвати відмовостійкими кластерами (HA), оскільки вони гарантують, що сайт працює ефективно та стабільно навіть у разі виникнення непередбачуваних ситуацій.

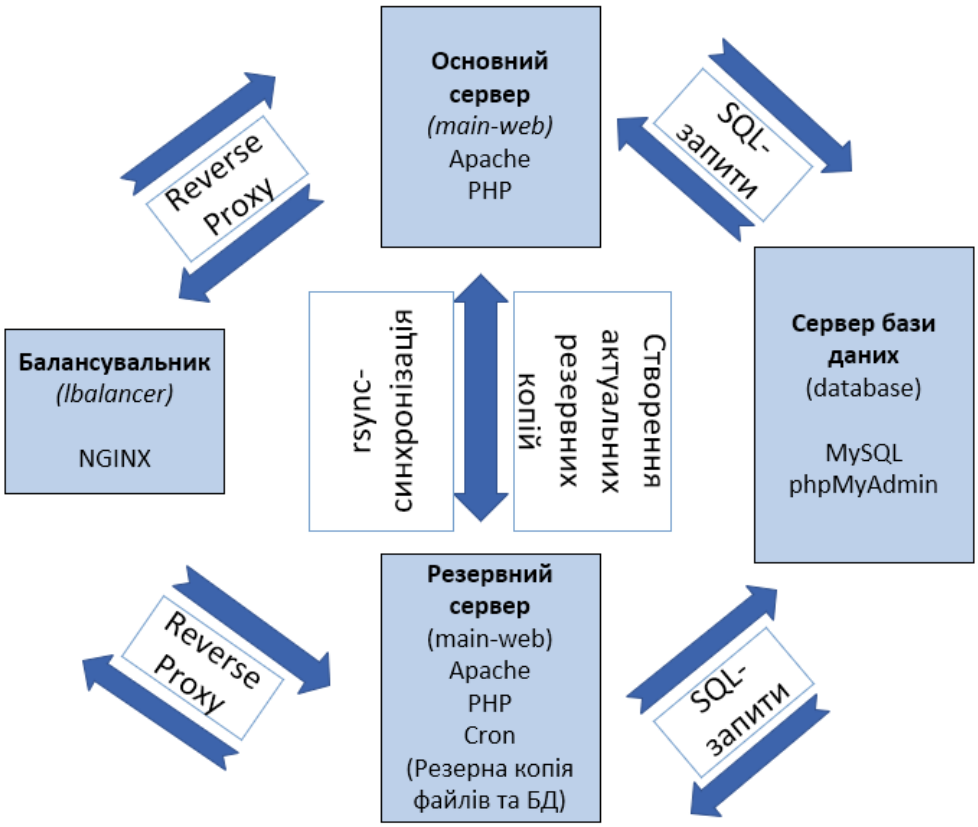

Рисунок 3.1 – Схема роботи системи серверного кластеру

3.2 Сервер балансувальник

В структурі серверного кластера балансувальник є важливою частиною. Він відповідає за розподіл навантаження між основним і резервним серверами, враховуючи їх завантаженість і доступність. Для цього він використовує алгоритми балансування, які допомагають оптимізувати розподіл запитів і зменшити навантаження на окремі сервери. Балансувальник також може перенаправляти запити на резервний сервер у разі відмови основного сервера. Це забезпечує стійку та безперебійну роботу сайту для користувачів.

Крім того, балансувальник відповідає за моніторинг доступності серверів і виявлення проблем з їх роботою. У випадку відмови одного з серверів він може автоматично перенаправляти запити на інший сервер, що

гарантує стабільний доступ до сайту. Балансувальник також може регулювати пріоритети для різних запитів і гарантувати рівномірне розподілення навантаження між серверами. В цілому, робота балансувальника є критично важливою для забезпечення високої ефективності та доступності серверного кластера.

Для того, щоб балансувальник працював ефективно, необхідно правильно налаштувати його алгоритми та функції моніторингу. Тестування та поновлення програмного забезпечення балансувальника повинні проводитися регулярно, щоб запобігти потенційним проблемам. Це визначає ефективність і стабільність роботи кластера.

Балансувальник також збільшує продуктивність і швидкість роботи сайту для користувачів, оскільки запити розподіляються між різними серверами. Це дозволяє підтримувати більше одночасних з’єднань і забезпечує більш швидку обробку даних. У загальному, балансувальник є важливою частиною структури серверного кластера, який допомагає гарантувати високу доступність і ефективність роботи сайту.

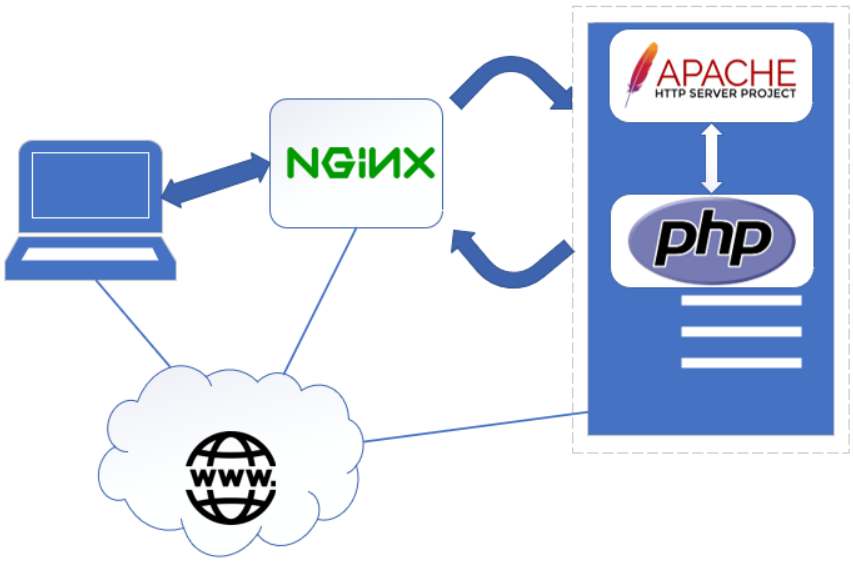

Рисунок 3.2 – Схема роботи зворотного проксування

Балансувальник є важливою частиною серверного кластеру, який допомагає забезпечити високу доступність і ефективність роботи сайту. Він відповідає за розподіл навантаження між серверами, нагляд за їх доступністю та виявлення проблем з роботою. Крім того, він може перенаправляти запити на резервний сервер у разі відмови основного сервера, що дозволяє підтримувати безперервну роботу сайту. Що є важливим для успіху бізнесу та задоволення потреб користувачів, балансувальник збільшує продуктивність і швидкість роботи сайту для користувачів.

3.3 Основний сервер

Основним компонентом структури серверного кластера є основний сервер. Він відповідає за основні завдання сайту, такі як обробка запитів користувачів і збереження даних. Крім того, основний сервер має резервні копії даних, які забезпечують безпеку даних і дозволяють їх відновити в разі необхідності. Щоб забезпечити стійку та безперебійну роботу сайту, функції основного сервера можуть бути автоматично перенаправленні на резервний сервер у разі відмови цього сервера.

Визначення та вирішення проблем з роботою сайту також залежить від основного сервера. Він стежить за своєю роботою та співпрацює з балансувальником, щоб забезпечити ідеальний розподіл навантаження. Крім того, зазвичай на основному сервері знаходиться база даних, яка містить всю інформацію про сайт. Це дозволяє отримувати актуальну інформацію та швидко реагувати на запити користувачів. У цілому, основний сервер є важливою частиною структури серверного кластеру, і він допомагає забезпечити ефективну та стабільну роботу сайту.

Крім того, основний сервер відповідає за визначення та вирішення проблем, пов’язаних із роботою сайту. Він контролює свою роботу та

працює разом з балансувальником, щоб забезпечити ідеальний розподіл навантаження. Крім того, зазвичай основний сервер містить базу даних, яка зберігає всю інформацію про сайт. Це дозволяє отримувати поточну інформацію та відповідати на запити користувачів швидко. У структурі серверного кластеру основний сервер забезпечує ефективну та стабільну роботу сайту.

Крім того, на основному сервері зазвичай знаходяться програми, які відповідають за роботу сайту, такі як веб-сервери, бази даних та інші. Щоб забезпечити ефективну та безперебійну роботу сайту, ці програми повинні бути налаштовані та підтримуватися в актуальному стані. Крім того, основний сервер може бути розширений і покращений, щоб забезпечити більшу масштабованість і продуктивність. У підсумку, основний сервер є центральним елементом серверного кластеру, який грає важливу роль у забезпеченні ефективності та доступності сайту для користувачів.

3.4 Резервний сервер

Резервний сервер є важливою частиною серверного кластера, оскільки він дозволяє швидко відновити роботу сайту у разі відмови основного сервера. Він має точну копію даних з основного сервера, що запобігає втраті важливих даних і гарантує безперебійну роботу веб-сайту. Якщо він відмовив, резервний сервер може виконувати завдання основного сервера. У цілому, використання резервного сервера є надзвичайно важливим для забезпечення безперебійної роботи користувачів на сайті.

Окрім того, резервний сервер може бути використаний для розширення потужності серверного кластеру в разі збільшення навантаження на сайт. Це дозволяє забезпечити більшу продуктивність та швидкість роботи сайту у разі необхідності. Крім того, на резервному сервері також можуть бути встановлені та налаштовані додаткові програмні

засоби для покращення роботи сайту, такі як кешування, оптимізація швидкості завантаження та інші. У підсумку, резервний сервер є важливим елементом серверного кластеру, який забезпечує стабільну та ефективну роботу сайту.

Резервний сервер також може бути використаний для проведення тестування та впровадження нових функцій чи оновлень сайту без впливу на його основну роботу. Це дозволяє забезпечити безперебійну роботу сайту та впровадження змін без негативного впливу на користувачів. Крім того, резервний сервер може бути використаний для збереження бекапів даних та налаштування автоматичного відновлення у разі необхідності. У підсумку, резервний сервер є важливим елементом інфраструктури сайту, який забезпечує стійку та ефективну роботу.

Резервний сервер - це важлива складова структури серверного кластеру, яка забезпечує можливість швидкого відновлення роботи сайту у разі відмови основного сервера. Він має точну копію даних з основного сервера, а також може виконувати його функції та бути використаним для розширення потужності та проведення тестування чи збереження бекапів даних. Резервний сервер допомагає забезпечити стійку та безперебійну роботу сайту для користувачів.

3.5 Сервер бази даних

Сервер бази даних - це окремий сервер, який забезпечує збереження та обробку даних для сайту. Він виконує основну функцію збереження та організації інформації, що використовується на сайті, таких як статті, користувачі, замовлення та інші. Сервер бази даних також відповідає за забезпечення доступу до цих даних та їх безпеку. Він може бути інтегрований з основним сервером або бути окремим сервером у складі серверного кластеру. У підсумку, сервер бази даних є важливим елементом

інфраструктури сайту, який забезпечує ефективну та безперебійну роботу з даними.

На сервері бази даних зазвичай використовуються спеціальні програмні засоби для збереження та обробки інформації, такі як MySQL, PostgreSQL, MongoDB та інші. Вони дозволяють ефективно зберігати та організовувати дані, а також забезпечують швидкий доступ до них. Крім того, на сервері бази даних можуть бути налаштовані різноманітні параметри для оптимізації його роботи та підвищення продуктивності. У цілому, сервер бази даних є невід'ємною частиною інфраструктури сайту та відіграє важливу роль у забезпеченні ефективності та безпеки обробки інформації.

Одним з основних завдань сервера бази даних є забезпечення цілісності та безпеки даних. Він має вбудовані механізми для захисту від несанкціонованого доступу та втрати інформації. Також, на сервері бази даних можуть бути встановлені резервні копії даних для запобігання втраті в разі аварійної ситуації. Крім того, сервер бази даних може бути налаштований для реплікації даних на інших серверах, що забезпечує більшу надійність та доступність даних. У підсумку, сервер бази даних є важливим елементом інфраструктури сайту, який забезпечує безпеку та доступність даних для користувачів.

У системі буде використовуватись MySQL як база даних та phpMyAdmin як інтерфейс для управління цією базою даних. MySQL є одним з найпоширеніших та надійних реляційних баз даних, який використовується для збереження та обробки інформації на сайті. phpMyAdmin, у свою чергу, є зручним інструментом для управління базою даних, який дозволяє виконувати різноманітні операції, такі як створення таблиць, додавання та редагування даних, запити та інші. Разом вони становлять важливу частину інфраструктури сайту та допомагають забезпечити ефективне та безперебійне збереження та обробку даних.

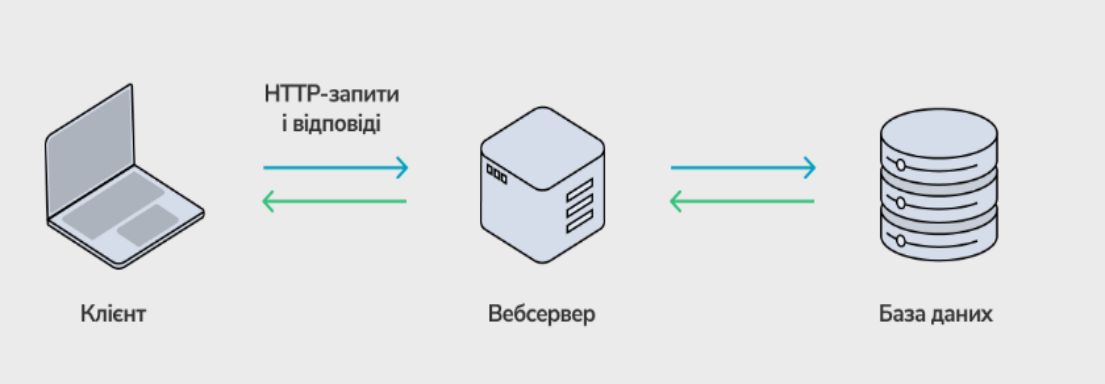

Рисунок 3.3 - Схема роботи веб-серверу та бази даних

3.6 Висновки до третього розділу

Система серверного кластера основана на чотирох серверах з різними функціями, які спільно працюють для забезпечення ефективності та стійкості веб-сайту. Балансувальник розподіляє навантаження між основним та резервним серверами, забезпечуючи стійку та безперебійну роботу та використовує зворотнє проксування. Основний сервер відповідає за обробку запитів користувачів та збереження даних, здійснює роботу разом із балансувальником для оптимального розподілу навантаження. Резервний сервер грає критичну роль у відновленні роботи сайту у разі відмови основного сервера, забезпечуючи точну копію даних та можливість розширення ресурсів системи, а також автоматизоване резервне копіювання даних. Особливо важливою є роль сервера бази даних, який зберігає та обробляє інформацію сайту, забезпечуючи її безпеку та доступність. Кожен з цих серверів відіграє ключову роль у забезпеченні ефективності, стабільності та безперебійності роботи веб-сайту.

4 РОЗДІЛ

РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ КОМПОНЕНТІВ КОМП'ЮТЕРНОЇ СИСТЕМИ

4.1. Базове налаштування серверів.

Початок кожного конфігурування серверів було розпочато з оновлення локальної бази даних пакетів до останніх доступних версій з репозиторіїв командою apt update. Тобто оновлюється інформація про доступність оновлень пакетів, а не встановлює оновлення. Командою apt upgrade було отримано та інстальовано оновлення до вже встановлених пакетів до їх останніх доступних версій з репозиторіїв.

Згідно з основними заходами безпеки, було встановлено не стандартний порт для підключення по SSH. Для встановлення власного номеру порту для підключення, було скориговано конфігураційний файл /etc/ssh/sshd_config, в якому було встановлено значення Port 2002.

Кожне підключення до серверів після зміни порту, повинно відбуватись командами у SSH клієнтах зі вказанням спеціального порта. У нашому випадку команда для підключень має вид:

```
ssh $user@server_IP_adress -p 2003
```

Завдяки цим діям знижується ризик сканування серверів на доступність до підключення та подальшого підбору паролів ботами. На кожному сервері було встановлено non-root користувача з правами sudo. Кожен сервер має власного користувача з іменами ролей серверів, а саме:

- lbalancer – користувач серверу балансувальника.
- main-web – користувач основного серверу з веб-контентом.

- database – користувач серверу бази даних.
- secondary-web – користувач резервного серверу.

Для створення користувачів та їх додавання до додаткової групи sudo, було використано наступні команди:

```
adduser $username
usermod -aG sudo $username
```

На кожен сервер кластеру було встановлено утиліту iptables, яка виконує роль фаєрволу та дає можливість управління правилами брандмауера.

Для різних серверів було встановлено різні правила щодо вхідних та вихідних з'єднань, які надають змогу контролювати тільки безпечні підключення для утиліт, котрі використовуються для роботи всієї системи. Таким чином для дозволу підключення до серверів по 2002 порту (SSH) було встановлене наступне правило:

```
sudo iptables -A INPUT -p tcp --dport 2002 -j ACCEPT
```

Перелічені налаштування допоможуть знизити ризик взлому системи та несанкціонованих підключень, що дозволить уникнути компрометації даних або виводу системи з ладу.

4.2 Налаштування балансувальника

Основна роль балансувальника – це розподіл навантаження запитів від клієнтів до системи. Основа серверного кластеру закладається в налаштуванні сервера балансувальника.

На основі проведеного аналізу, було використано веб-сервер NGINX для ролі балансування запитів та використання зворотного проксування на основний та резервні сервери.

Метою коректної роботи балансувальника (load balancer) було поставлено перенаправлення запитів клієнтів на основний сервер з сайтом (main-web). Якщо main-web сервер не відповідає, то всі запити клієнтів повинні перенаправлятись на резервний сервер (secondary-web), який буде підміняти основний на час його недоступності. Для реалізації було обрано використання , а саме зворотного проксі (reverse proxy).

Зворотний проксі-сервер – це тип проксі-сервера, який встановлюється між клієнтами та внутрішніми/внутрішніми серверами, наприклад, HTTP-сервером, таким як NGINX, Apache тощо, або серверами застосунків, написаними на Nodejs, Python, Java, Ruby, PHP та багатьох інших мовах програмування.

Це шлюз або сервер-посередник, який приймає запит клієнта, передає його одному або декільком внутрішнім серверам, а потім отримує від них відповідь і повертає її клієнту, створюючи враження, що контент отримано від самого зворотного проксі-сервера.

На рисунку 4.1 показана схематична робота балансувальника у серверному кластері.

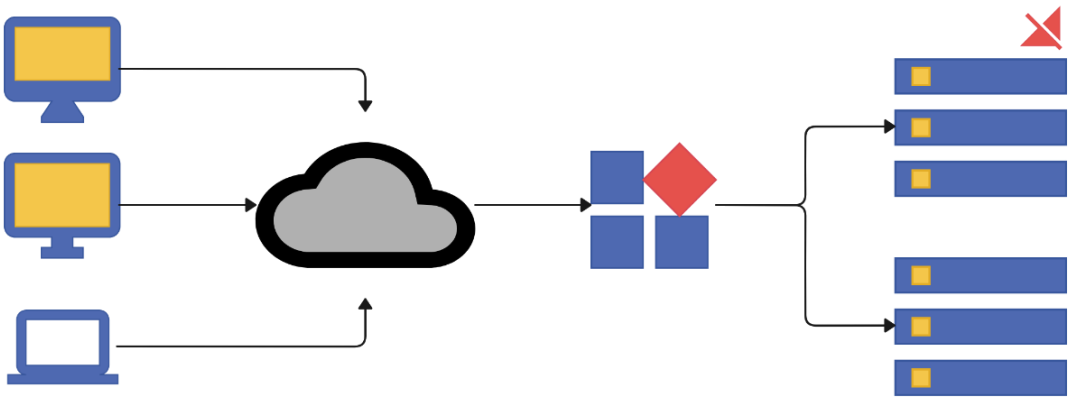

Рисунок 4.1 – Схема роботи балансувальника у серверному кластері

Зазвичай зворотний проксі-сервер – це внутрішній проксі-сервер, який використовується як “зовнішній” для контролю і захисту доступу до внутрішніх серверів у приватній мережі: зазвичай він розміщується за мережевим брандмауером.

Він допомагає внутрішнім серверам досягти анонімності для підвищення їхньої безпеки. В IT-інфраструктурі зворотний проксі може також виконувати функції міжмережевого екрана застосунків, балансувальника навантаження, термінатора TLS, веб-прискорювача (завдяки кешуванню статичного і динамічного вмісту) і багато іншого.

Основні налаштування зворотного проксування було можливим за допомогою конфігураційного файла веб-сервера NGINX, який у нашому випадку було створено за шляхом: /etc/nginx/sites-available/lbalance.

Створення конфігураційного файла було розпочато з директиви upstream backend, в якому визначалась назва пула серверів та було вказано IP серверів, які будуть прослуховуватись для подальшої взаємодії. Для списку серверів у пулі було примінено параметр weight, що означає вагу сервера, тобто його пріорітетність і відношення кількості запитів користувачів. У цьому випадку недоступності основного серверу, всі запити користувачів будуть перенаправлятись на резервний сервер.

Для кожного серверу було вказано директиву server, що NGINX слухатиме вхідні HTTP-запити на порти 80 та 443 (SSL шифрування), і проксирувати їх на пул серверів backend. Функція proxy_pass визначає адресу для проксіювання запитів, а proxy_set_header встановлює заголовки запиту, щоб зберегти інформацію під час проксіювання запитів на балансувальнику. На кінцевому етапі, працююча конгфігурація має вид:

```
upstream backend {
    server 208.69.117.98 weight=9;  #main-web_server
    server 206.54.190.166 backup weight=1;  #secondary-
web_server
}

server {
    listen 80;

location / {
    proxy_pass http://backend/;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
}
}
server {
    listen 443;

location / {
    proxy_pass http://backend/;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
}
}
```

4.3 Налаштування основного сервера

Налаштування основного серверу було розпочато з основних заходів безпеки та встановлення необхідного для роботи програмного забезпечення такі як: PHP та його розширення, веб-сервер Apache, текстовий редактор nano та інші.

Для коректної роботи сайту було створено конфігураційний файл веб-серверу Apache, в якому вказуються основні правила взаємодії з запитами та портами сервера. Також для зручності використання серверу, було підключено можливість використання файлів .htaccess для директорії /var/www/, що надають змогу прописувати правила взаємодії сайту з веб-сервером такі як: редагування лімітів ресурсів PHP, встановлення перенаправлень, захист від пошукових ботів, що створюють не бажане навантаження на сервер та інші. Приклад конфігураційного файла веб-серверу Apache:

```
<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
</Directory>

<VirtualHost *:80>

             ServerAdmin webmaster@localhost
        DocumentRoot /var/www/wordpress

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Для перевірки працездатності проксування з сервера балансування, було використано запит за допомогою cURL з локального комп’ютера.

```
PS C:\Users> curl -d http://78.140.162.163
```

За результатами виконання запиту, ми отримали наступний результат:

```
StatusCode        : 200
StatusDescription : OK
Content           :
```

```
                    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
                    <html
xmlns="http://www.w3.org/1999/xhtml">
                      <!--
                        Modified from the Debi...
RawContent        : HTTP/1.1 200 OK
                    Vary: Accept-Encoding
                    Connection: close
                    Accept-Ranges: bytes
                    Content-Length: 10918
                    Content-Type: text/html
                    Date: Sun, 17 Dec 2023 12:20:34 GMT
                    ETag: "2aa6-60cb36924dd83"
                    Last-M...
Forms             : {}
Headers           : {[Vary, Accept-Encoding], [Connection,
close], [Accept-Ranges, bytes], [Content-Length, 10918]...}
Images            : {@{innerHTML=; innerText=; outerHTML=<IMG
class=floating_element alt="Ubuntu Logo" src="/icons/ubuntu-
logo.png">; outerText=; tagName=IMG; class=floating_element;
alt=Ubuntu Logo; src=/icons/ubuntu-logo.png}}
InputFields       : {}
Links             : {@{innerHTML=manual; innerText=manual;
outerHTML=<A href="/manual">manual</A>; outerText=manual;
tagName=A; href=/manual}, @{innerHTML=public_html;
innerText=public_html; outerHTML=<A
href="http://httpd.apache.org/docs/2.4/mod/mod_userdir.html"
rel=n
                    ofollow>public_html</A>;
outerText=public_html; tagName=A;
href=http://httpd.apache.org/docs/2.4/mod/mod_userdir.html;
rel=nofollow}, @{innerHTML=existing bug reports;
innerText=existing bug reports; outerHTML=<A
href="https://bugs.launchpad.net/ubun
                    tu/+source/apache2" rel=nofollow>existing
bug reports</A>; outerText=existing bug reports; tagName=A;
href=https://bugs.launchpad.net/ubuntu/+source/apache2;
rel=nofollow}}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 10918
```

Згідно з отриманими даними ми бачимо, що запит, який поступає на сервер балансувальник успішно перенаправляється на основний сервер (`StatusCode:200`), на якому встановлено Apache. У результатах ми можемо спостерігати дані про стандартну сторінку Apache, які доступні при переході на IP-адресу основного серверу.

На наступному етапі налаштування серверу було встановлено систему керування вмістом сайту (CMS) WordPress, яку чисто називають «двигун сайту».

Інсталяцію було проведено з офіційного джерела, на якому надається архів з необхідними даними для початку розгортання системи на віртуальному хостингу. Після розпакування файлів, їх було перенесено до домашньої директорії з коректними правами на файли і папки, щоб уникнути некоректної роботи сайту у майбутньому.

Для коректної роботи сайту також зроблене налаштування Apache для пріоритетності виконання файлу PHP у конфігурації /etc/apache2/mods-enabled/dir.conf:

```
DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
```

Директива DirectoryIndex відповідає за пріоритетність виконання файлу index.php, щодо інших наявних файлів index. Це необхідно, щоб уникнути можливих розбіжностей у відображенні головної сторінки сайту, якщо у кореневу папку будуть добавлені подібні файли, наприклад index.php, яка часто використовується як стандартна сторінка системи керування змістом або хостинг провайдера, у випадку користування віртуальним веб-хостингом.

4.4 Налаштування сервера бази даних

Однією з базових компонентів роботи сайтів основаних на CMS WordPress є база даних. Особливістю розробленого серверного кластеру є віддаленого використання бази даних, для роботи сайту на основному та резервному сервері. Унікальність такого підходу полягає в тому, що для обох серверів буде використовуватись єдина база даних, та питання актуальності даних у ній буде вирішене. Завдяки подальшому

налаштуванню резервного сервера, база даних буде використовуватись навіть у період недоступності основного сервера.

Для реалізації такого підходу, на сервері було встановлено mysql сервер та подальше його конфігурування для віддаленого використання на інших серверах кластера.

Для можливості підключення до серверу бази даних було змінено параметр bind-address, який за замовчуванням використовує локальну IP адресу 127.0.0.1. Для цього параметру було вказано IP 0.0.0.0, що означає можливість підключення з будь якого IP до бази даних.

Подібне використання конфігурації могло б зменшити рівень безпеки бази даних, але для наших серверів у кластері працюють правила фаєрволу iptables, які контролюють можливість підключення для конкретних портів. Для використання бази даних використовується порт 3306, тому використання цього порту було дозволено правилами фаєрволу тільки для IP адрес серверів кластеру.

В наступних кроках було створено базу даних WordPress та її користувача wpuser, для якого було встановлено всі дозволи до управління з IP основного та резервного серверів, як це представлено у наступних командах:

```
GRANT ALL PRIVILEGES ON wordpress.* TO 'wpuser'@'206.54.190.166'
IDENTIFIED BY 'Admin123';
GRANT ALL PRIVILEGES ON wordpress.* TO
'wpuser'@'208.69.117.98' IDENTIFIED BY 'Admin123';
```

Після внесення всіх необхідних налаштувань на сервері бази даних, можна використовувати параметри для підключення до бази даних у системі керування змістом WordPress, які вказуються у файлі wp-config.php.

wp-config.php буде мати наступні параметри:

```
// ** Database settings - You can get this info from y$/** The
name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );
```

```
/** Database username */
define( 'DB_USER', 'wpuser' );

/** Database password */
define( 'DB_PASSWORD', 'Admin123' );

/** Database hostname */
define( 'DB_HOST', '78.140.162.220:3306' );

/** Database charset to use in creating database table$define(
'DB_CHARSET', 'utf8' );

/** The database collate type. Don't change this if in$define(
'DB_COLLATE', '' );
```

Після підключення бази даних до CMS WordPress наш сайт вже доступний за IP адресою балансувальника. Для перевірки було виконано cURL запит, результат якого наведено нижче:

```
StatusCode        : 200

StatusDescription : OK

Content           : <!DOCTYPE html>
                    <html lang="uk">
                    <head>
                        <meta charset="UTF-8" />
                        <meta name="viewport"
content="width=device-width, initial-scale=1" />
                    <meta name='robots' content='noindex,
nofollow' />
                    <title>Main...
RawContent        : HTTP/1.1 200 OK
                    Transfer-Encoding: chunked
                    Connection: keep-alive
                    Link:
<http://78.140.162.163/index.php?rest_route=/>;
rel="https://api.w.org/"
                    Vary: Accept-Encoding
                    Content-Type: text/html; cha...
Forms             : {}

Headers           : {[Transfer-Encoding, chunked],
[Connection, keep-alive], [Link,
<http://78.140.162.163/index.php?re
                    st_route=/>; rel="https://api.w.org/"],
[Vary, Accept-Encoding]...}
Images            : {@{innerHTML=; innerText=; outerHTML=<IMG
alt="Building exterior in Toronto, Canada" src="http://78
```

```
                    .140.162.163/wp-
content/themes/twentytwentyfour/assets/images/building-
exterior.webp">; outerText=;
                     tagName=IMG; alt=Building exterior in
Toronto, Canada; src=http://78.140.162.163/wp-content/themes
                    /twentytwentyfour/assets/images/building-
exterior.webp}, @{innerHTML=; innerText=; outerHTML=<IMG a
                    lt="Tourist taking photo of a building"
src="http://78.140.162.163/wp-content/themes/twentytwentyfo
                    ur/assets/images/tourist-and-
building.webp">; outerText=; tagName=IMG; alt=Tourist taking
photo of
                    a building; src=http://78.140.162.163/wp-
content/themes/twentytwentyfour/assets/images/tourist-and-
                    building.webp}, @{innerHTML=; innerText=;
outerHTML=<IMG alt="Windows of a building in Nuremberg, G
                    ermany" src="http://78.140.162.163/wp-
content/themes/twentytwentyfour/assets/images/windows.webp">;
                     outerText=; tagName=IMG; alt=Windows of a
building in Nuremberg, Germany; src=http://78.140.162.16
                    3/wp-
content/themes/twentytwentyfour/assets/images/windows.webp}}

InputFields       : {}


Links             : {@{innerHTML=Main-web; innerText=Main-web;
outerHTML=<A href="http://78.140.162.163" rel=home targe
                    t=_self aria-current="page">Main-web</A>;
outerText=Main-web; tagName=A; href=http://78.140.162.163
                    ; rel=home; target=_self; aria-
current=page}, @{innerHTML=Зразок сторінки; innerText=Зразок
сторінк
                    и; outerHTML=<A class="wp-block-pages-
list__item__link wp-block-navigation-item__content" href="htt
                    p://78.140.162.163/?page_id=2">Зразок
сторінки</A>; outerText=Зразок сторінки; tagName=A; class=wp-
                    block-pages-list__item__link wp-block-
navigation-item__content; href=http://78.140.162.163/?page_id
                    =2}, @{innerHTML=Привіт, світ!;
innerText=Привіт, світ!; outerHTML=<A
href="http://78.140.162.163/?
                    p=1" target=_self>Привіт, світ!</A>;
outerText=Привіт, світ!; tagName=A; href=http://78.140.162.163
                    /?p=1; target=_self}, @{innerHTML=Гру 17,
2023; innerText=Гру 17, 2023; outerHTML=<A href="http://7
                    8.140.162.163/?p=1">Гру 17, 2023</A>;
outerText=Гру 17, 2023; tagName=A; href=http://78.140.162.163
                    /?p=1}...}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 83972
```

Значення параметру StatusCode з кодом 200, повідомляє нам про успішне виконання запиту до сайту. Оскільки результат виведення заголовків, посилань та зображень змінився на значення, які віддає WordPress, це означає успішне встановлення системи керування змістом сайту, коректну конфігурацію проксування на сервері балансувальнику та коректне віддалене підключення до бази даних. Успішне відображення працюючого сайту можна спостерігати за адресою https://78.140.162.163/, результат якої зображено у рисунку 4.1.

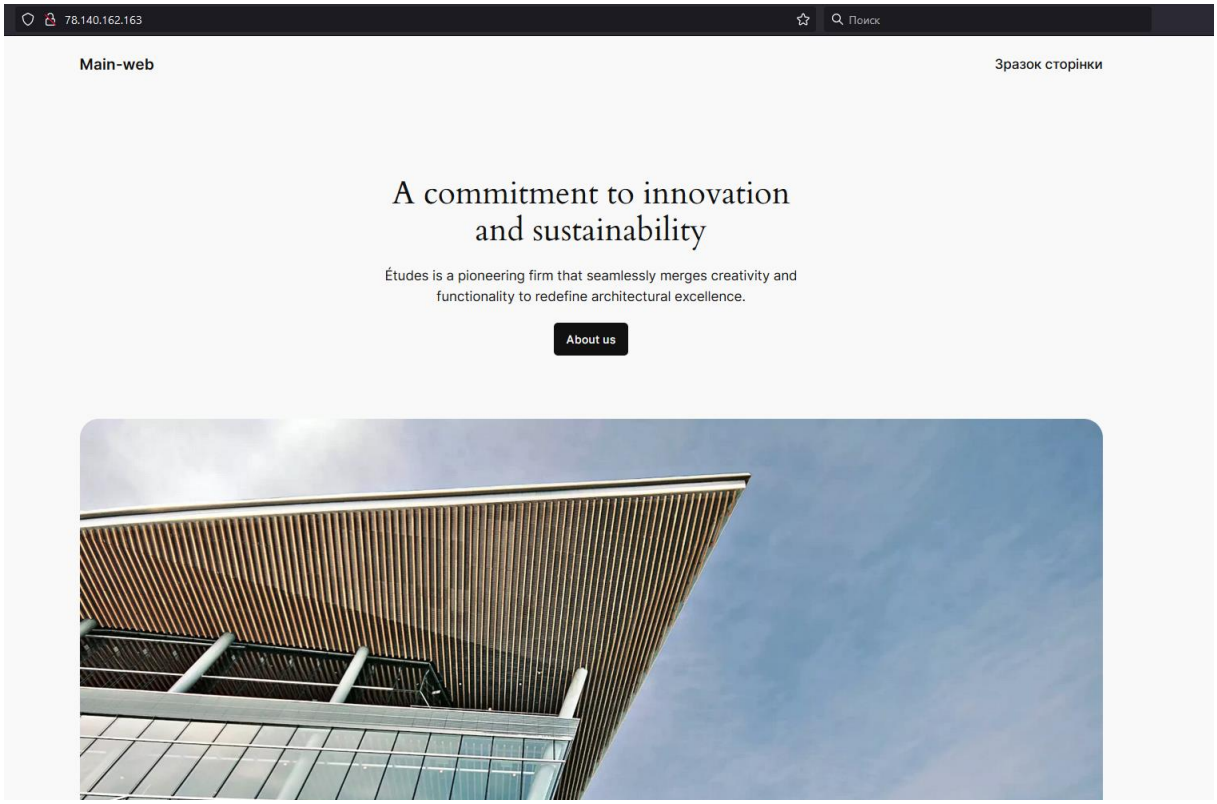

Рисунок 4.2 – Відображення коректної роботи сайта

Для зручності управління базою даних, було встановлено веб-додаток phpMyAdmin, який доступний для адміністрування за посиланням http://78.140.162.220/phpmyadmin.

Успішний вхід користувача wpuser для управління базою даних wordpress можна спостерігати на рисунку 4.3.

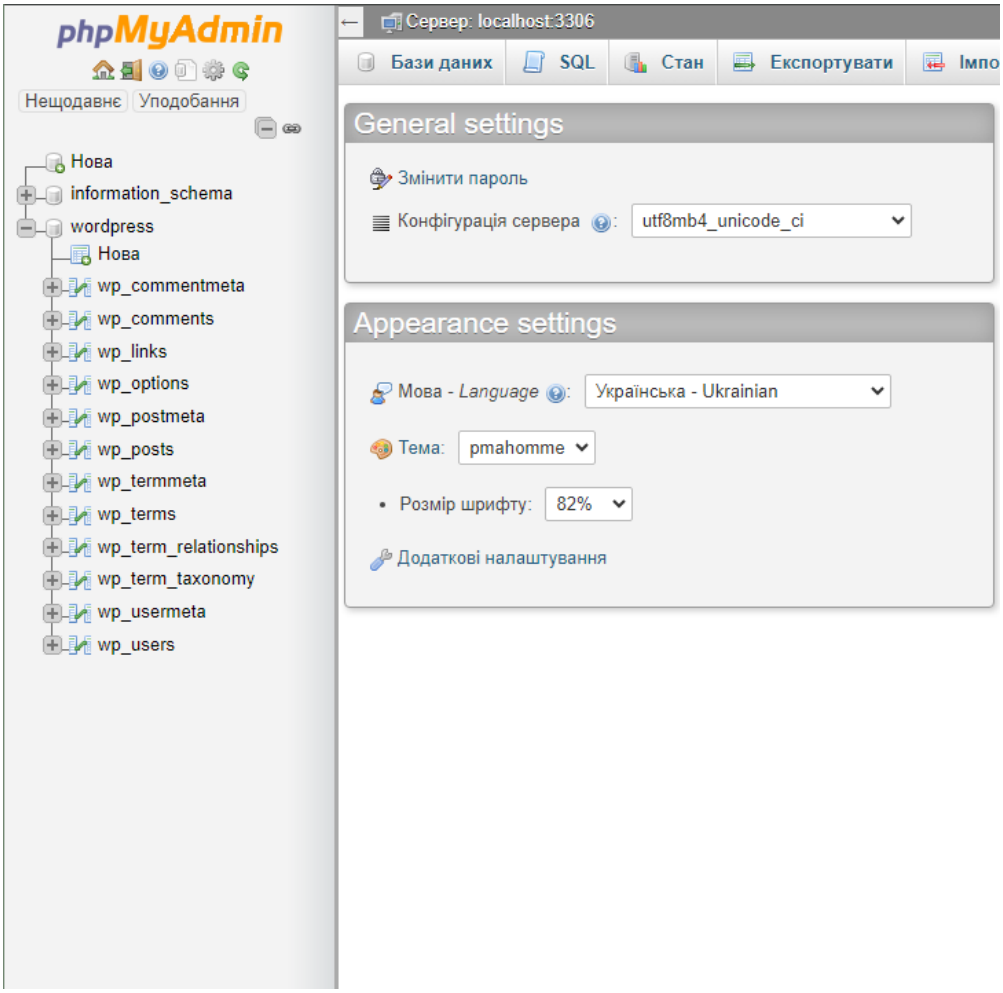

Рисунок 4.3 – Демонстрація графічного інтерфейсу phpMyAdmin

4.5.Налаштування резервного сервера

Головною задачею резервного серверу (secondary-web) було визначене створення серверу, котрий буде приймати на себе навантаження з серверу балансувальника, у разі недоступності основного сервера. Так як конфігурація балансувальника вже реалізована та використовує зворотне проксування на IP адресу резервного серверу, то головною метою сервера є відображення актуальних даних для користувачів та оновлення даних на сервері бази даних.

Для реалізації поставленої задачі було створено ssh-ключ на резервному сервері, який був скопійований на основний сервер за допомогою команди:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub root@208.69.117.98
```

Завдяки авторизації по ключу, ми маємо можливість підключення з резервного серверу до основного без введення паролю, що надає можливість розробки та виконання скриптів для автоматизації синхронізації файлів між серверами. Для цього було використано cron-планувальник, для якого задане виконання наступної команди кожні 2 години:

```
0 */2 * * * /usr/bin/rsync -avz -e "ssh -p 2002"
/var/www/wordpress/
root@208.69.117.98:/var/www/wordpress/
```

Тепер сервер може повністю брати на себе навантаження з основного сервера, маючи актуальні дані та використовує актуальну базу даних з серверу MySQL.

4.6. Автоматизація резервного копіювання на резервному сервері

Для автоматизації створення резервних копій було розроблено bash-скрипт, який містить в собі команди створення архіву з файлами сайту, та видалення архівів старіших ніж 7 днів. Завдяки цьому дисковий простір резервного серверу буде оптимізовано, а можливість відновлення сайту в ручному режимі для адміністратора буде завжди доступна. Оскільки резервний сервер розташовано в іншому дата-центрі (США) - це зменшує шанс повної втрати даних сайту.

У випадку критичної ситуації, є можливість створення нової моди, на якій можна буде провести повторне інсталювання системи та завантажити архів сайту вже на новий сервер кластеру. Виконуваний скрипт backup_script_and_remove_old.sh містить в собі наступні команди:

```
#!/bin/bash

# перемінна для вказання директорії зберігання резервних копій
backup_dir="/var/backups"


# перемінна для вказання директорії, яка буде заархівована у
папку зберігання резервних копій
source_dir="/var/www/wordpress"


# Створення ім'я архіву, який буде відображати дату свторення
архіву
backup_file="$backup_dir/wordpress_backup_$(date +'%Y-%m-
%d').tar.gz"


# Створення архіву
tar -czf "$backup_file" "$source_dir"


# Видалення архівів, які зберігаються більше ніж 7 днів
find "$backup_dir" -name 'wordpress_backup_*' -type f -mtime
+7 -exec rm {} \;
```

Для автоматизації його виконання, було створено cron-завдання:

```
0 5 * * * /var/backups/backup_script.sh
```

Запланована cron-задача автоматизує виконання скрипта backup_script_and_remove_old.sh щодня о 05:00 годині.

Запропоновані скрипти та завдання Cron дозволяють автоматизувати створення резервних копій вашого веб-сайту без необхідності використання зовнішніх сервісів чи інструментів. Крон-завдання дозволяють запускати скрипти в певний час без вашої участі. Це автоматизує процес створення резервних копій і звільняє вас від необхідності ручного керування процесом.

Автоматизація та оптимізація процесу резервного копіювання дозволяє ефективно використовувати ресурси сервера та зменшити ризики втрати даних у випадку аварії чи нештатної ситуації.

Такі скрипти та завдання Cron є ефективним способом забезпечити регулярне створення резервних копій вашого веб-сайту, зменшуючи ризики втрати даних та забезпечуючи простий та надійний спосіб відновлення у разі потреби.

4.7 Перевірка стабільності системи

Для перевірки стабільності системи було штучно спровоковано нештатну ситуацію, для цього було виключено від мережі основний сервер (208.69.117.98). Щоб запевнитись у його недоступності, було виконано команду ping з локального ПК:

```
ping 208.69.117.98
PING 208.69.117.98 (208.69.117.98) 56(84) bytes of data.
From 78.140.189.193 icmp_seq=1 Time to live exceeded
From 78.140.189.193 icmp_seq=2 Time to live exceeded
From 78.140.189.193 icmp_seq=3 Time to live exceeded
From 78.140.189.193 icmp_seq=4 Time to live exceeded
From 78.140.189.193 icmp_seq=5 Time to live exceeded
--- 208.69.117.98 ping statistics ---
5 packets transmitted, 0 received, +5 errors, 100% packet loss,
time 4006ms
```

Після того як було отримано негативний результат перевірки основного сервера, перевірили коректність роботи балансувальника. Для цього виконали curl запит до IP балансувальника (78.140.162.163).

Результат відпрацювання команди `curl http://78.140.162.163`:

```
StatusCode        : 200
StatusDescription : OK
Content           : <!DOCTYPE html>
                    <html lang="uk">
                    <head>
                        <meta charset="UTF-8" />
                        <meta name="viewport"
content="width=device-width, initial-scale=1" />
                    <meta name='robots' content='noindex,
nofollow' />
                    <title>Main...

RawContent        : HTTP/1.1 200 OK
                    Transfer-Encoding: chunked
                    Connection: keep-alive
                    Link:
<http://78.140.162.163/index.php?rest_route=/>;
rel="https://api.w.org/"
                    Vary: Accept-Encoding
                    Content-Type: text/html; cha...

Forms             : {}
Headers           : {[Transfer-Encoding, chunked],
[Connection, keep-alive], [Link,
<http://78.140.162.163/index.php?re
                    st_route=/>; rel="https://api.w.org/"],
[Vary, Accept-Encoding]...}
Images            : {@{innerHTML=; innerText=; outerHTML=<IMG
alt="Building exterior in Toronto, Canada" src="http://78
                    .140.162.163/wp-
content/themes/twentytwentyfour/assets/images/building-
exterior.webp">; outerText=;
                     tagName=IMG; alt=Building exterior in
Toronto, Canada; src=http://78.140.162.163/wp-content/themes
                    /twentytwentyfour/assets/images/building-
exterior.webp}, @{innerHTML=; innerText=; outerHTML=<IMG a
                    lt="Tourist taking photo of a building"
src="http://78.140.162.163/wp-content/themes/twentytwentyfo
                    ur/assets/images/tourist-and-
building.webp">; outerText=; tagName=IMG; alt=Tourist taking
photo of
```

```
                    a building; src=http://78.140.162.163/wp-
content/themes/twentytwentyfour/assets/images/tourist-and-
                    building.webp}, @{innerHTML=; innerText=;
outerHTML=<IMG alt="Windows of a building in Nuremberg, G
                    ermany" src="http://78.140.162.163/wp-
content/themes/twentytwentyfour/assets/images/windows.webp">;
                     outerText=; tagName=IMG; alt=Windows of a
building in Nuremberg, Germany; src=http://78.140.162.16
                    3/wp-
content/themes/twentytwentyfour/assets/images/windows.webp}}

InputFields         : {}

Links               : {@{innerHTML=Main-web; innerText=Main-web;
outerHTML=<A href="http://78.140.162.163" rel=home targe
                    t=_self aria-current="page">Main-web</A>;
outerText=Main-web; tagName=A; href=http://78.140.162.163
                    ; rel=home; target=_self; aria-
current=page}, @{innerHTML=Зразок сторінки; innerText=Зразок
сторінк
                    и; outerHTML=<A class="wp-block-pages-
list__item__link wp-block-navigation-item__content" href="htt
                    p://78.140.162.163/?page_id=2">Зразок
сторінки</A>; outerText=Зразок сторінки; tagName=A; class=wp-
                    block-pages-list__item__link wp-block-
navigation-item__content; href=http://78.140.162.163/?page_id
                    =2}, @{innerHTML=Привіт, світ!;
innerText=Привіт, світ!; outerHTML=<A
href="http://78.140.162.163/?
                    p=1" target=_self>Привіт, світ!</A>;
outerText=Привіт, світ!; tagName=A; href=http://78.140.162.163
                    /?p=1; target=_self}, @{innerHTML=Гру 17,
2023; innerText=Гру 17, 2023; outerHTML=<A href="http://7
                    8.140.162.163/?p=1">Гру 17, 2023</A>;
outerText=Гру 17, 2023; tagName=A; href=http://78.140.162.163
                    /?p=1}...}
ParsedHtml          : mshtml.HTMLDocumentClass
RawContentLength    : 83972
```

У порівнянні з результатом curl запиту під час працюючого основного серверу бачимо, що результат аналогічний первинному.

Для перевірки коректної роботи необхідних компонентів для роботи сайту, включаючи PHP модулі та з’єднання з сервером бази даних, було виконано з’єднання за адресою http://78.140.162.163/ вже безпосередньо у веб-браузері.

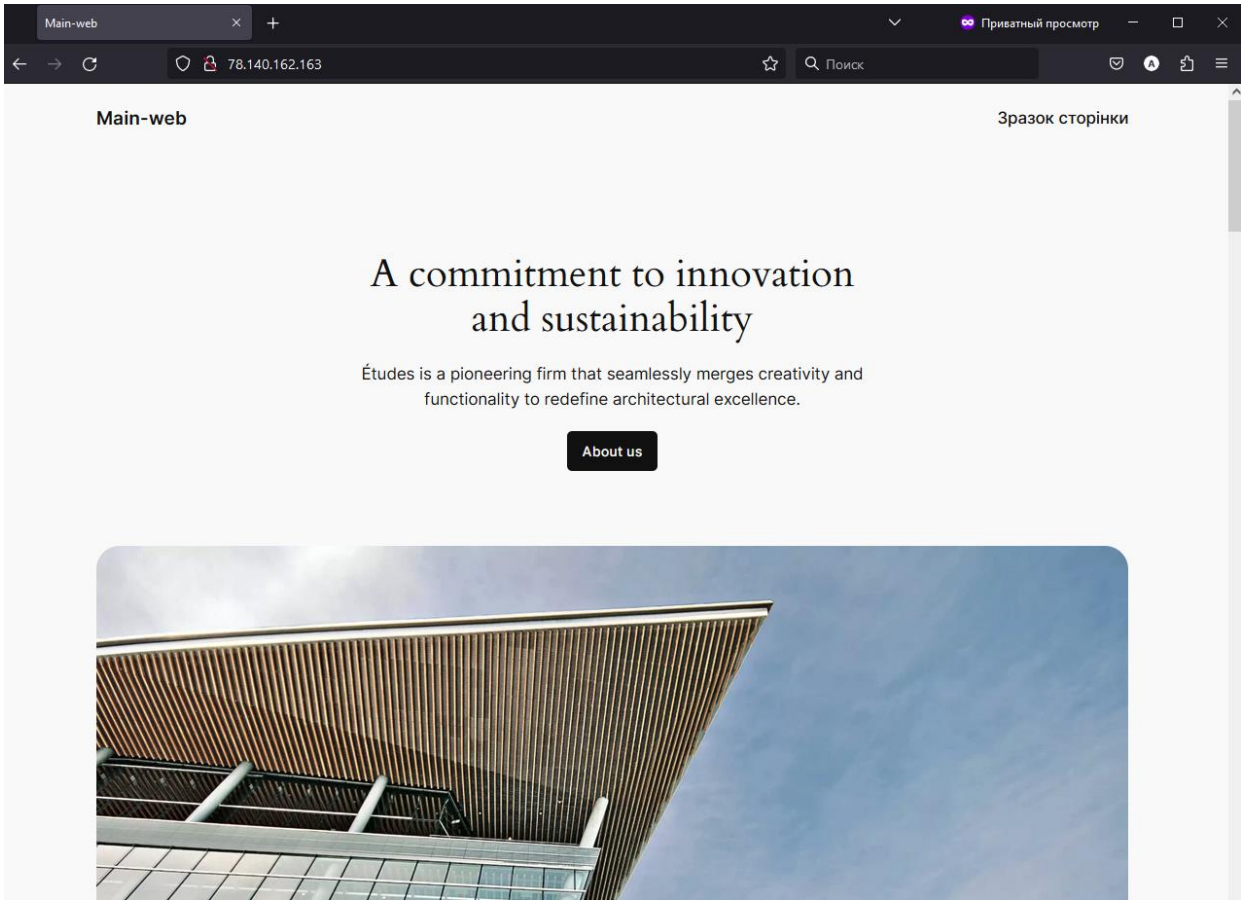

Рисунок 4.4 – Демонстрація працюючого веб-сайту

На головній сторінці сайта успішно відпрацьовують PHP скрипти, а помилки про невдале з’єднання з базою даних відсутні.

Тож можна зробити висновок, що система успішно пройшла перевірку на виведення зі стабільного стану основного сервера. Резервний сервер продовжує стабільну роботу сайту, під час відсутності основного серверу.

Завдяки цьому, розміщений веб-додаток або проект продовжує свою роботу, а адміністратори системи мають змогу відновити роботу основного серверу та провести діагностику його не стабільної роботи, щоб уникнути повторної ситуації у майбутньому.

4.8 Висновки до четвертого розділу

Основна мета налаштувань серверів полягала в забезпеченні їхньої безпеки, ефективності та підвищення відмовостійкості. Підвищення рівня захисту від несанкціонованих доступів було досягнуте завдяки оновленню бази даних пакетів, встановленню нестандартного SSH-порту, правил фаєрволу та іншим заходам, які зменшують ризики компрометації серверів.

Налаштування балансувальника полягало у налаштуванні реверс проксування за допомогою веб-сервера NGINX та його конфігураційних параметрів для ефективного розподілу навантаження між серверами та забезпечення безпеки та доступності для користувачів.

Розроблений кластер для CMS WordPress використовує віддалену базу даних, яка обслуговує як основний, так і резервний сервери. Це унікальне рішення дозволяє забезпечити єдиний доступ до бази даних для обох серверів і розв'язати проблему актуальності даних, навіть при недоступності основного сервера.

В ході розробки було використано планувальник cron-задач, для автоматичної архівації резервних копій сайту, та зберігання їх на резервному сервері, що дозволить вберегти данні від їх пошкодження або втрати.

ВИСНОВКИ

У ході даної кваліфікаційної роботи було реалізовано розробку серверного кластеру та проведено широкий спектр досліджень, що спрямовані на підвищення відмовостійкості, актуальності, ефективності та безпеки серверної інфраструктури. Основні кроки та визначені напрямки вирішення завдань проекту можна узагальнити наступним чином:

1. **Аналіз та вибір технологій**: в рамках проекту був проведений аналіз різних аспектів створення серверного кластеру. Були визначені основні параметри, такі як використання веб-серверів Apache та NGINX, застосування firewall iptables для забезпечення безпеки, а також впровадження автоматизованого резервного копіювання за допомогою скрипту та cron-планувальника.
2. **Структура серверного кластеру**: Розроблена структура серверного кластеру, що включає чотири сервери з різними функціональними обов'язками. У цій структурі ключові ролі відведено балансувальнику, основному та резервному серверам, а також серверу бази даних. Ця структура дозволяє забезпечити високу доступність та відмовостійкість веб-сайту.
3. **Балансування навантаження**: Реалізовано ефективне балансування навантаження за допомогою балансувальника, який враховує завантаженість та доступність основного та резервного серверів. Це забезпечує стійку та безперебійну роботу сайту, оптимізуючи розподіл запитів, та високу доступність у разі втрати зв’язку з основним сервером.
4. **Безпека та захист**: Налаштовано заходи безпеки, які включають правила фаєрволу iptables та зміну параметрів безпеки, таких як SSH-порт. Застосовано принципи відділення прав доступу для користувачів

та змінено параметр bind-address для забезпечення віддаленого підключення до бази даних.

5. **Резервне копіювання та автоматизація**: Використано скрипти та cron-задачі для автоматизованого резервного копіювання, що сприяє забезпеченню безпеки та надійності даних. Це дозволяє уникнути втрати чи пошкодження інформації.

Загальною метою налаштувань було досягнення високого рівня безпеки, ефективності та відмовостійкості серверного кластеру. Впровадження сучасних технологій та відкритих програмних рішень дозволяє максимально використовувати ресурси та підтримувати високий стандарт функціонування веб-сайту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основні чинники та тенденції в зростанні даних URL: https://onbiz.biz/storage-data/

2. Серверний кластер - навіщо він потрібен? URL: https://onehostplanet.ua/ua/news/cerverniy-klaster-navishcho-vin-potriben

3. Рішення Supermicro для Hadoop, URL: https://solutions.asbis.ua/solutions/software-defined-infrastructure/rishennya-supermicro-dlya-hadoop

4. Що таке серверний кластер? URL: https://server-shop.ua/ua/understanding-clustering-for-servers.html

5. Метод адаптивного балансування навантаження в кластерних системах військового призначення на основі рівноваги Неша (Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, Київ), URL: http://znp-cvsd.nuou.org.ua/article/view/275401

5. Кластери, URL: https://www.solti.ua/systemna-integraciya/serwerni-rischennja/systemy-bezpeky-ta-keruvan/

6. NGINX vs. Apache — Choosing the best web server in 2023, URL: https://www.nexcess.net/blog/nginx-vs-apache/

7. Аналіз найактуальніших серверних систем управління базами даних URL: https://science.lpnu.ua/sites/default/files/journal-paper/2017/dec/7245/14.pdf

8. СУБД: які бувають, як вибрати, URL: https://highload.today/uk/subd-yaki-buvayut-yak-vibrati/

9. Що таке phpmyadmin ?, URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-phpmyadmin/

10. Як стати PHP-розробником. План дій для початківців, URL: https://dou.ua/lenta/articles/how-to-learn-php/

11. Ефективне блокування IP за допомогою Iptables, URL: https://linuxthebest.net/uk/efektyvne-blokuvannya-ip-za-dopomogoyu-iptables/

12. OS Centos - опис ОС: актуальні версії, плюси і мінуси системи, URL: https://hyperhost.ua/info/uk/os-centos-opis-os-aktualni-versii-plyusi-i-minusi-sistemi

13. About AlmaLinux Wiki, URL: https://wiki.almalinux.org/

14. Про Debian, URL: https://www.debian.org/intro/about.uk.html

15. Масштабування за допомогою сервера Ubuntu, URL: https://ubuntu.com/server

ДОДАТОК А

Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Найменування	Кільк. аркушів	Примітки
1				
2		Документація		
3				
4	**ІТКІ.КР.22.02.ДА.ПЗ**	Пояснювальна записка	83	Формат А4
5				
6		Презентація		
7				
8		Диск CD-R з презентацією	1	Диск CD-R
9				

Зм.	Ар-куш	№ докум.	Підпис	Дата	**ІТКІ.КР.22.02.ДА.ПЗ.**			
Розроб.		Галушко О.Г.			**Матеріали кваліфікаційної роботи**	Літ.	Аркуш	Аркушів
Керівник		Соколова Н.О.				Н	1	1
Рецензент		Клименко А.В.				НТУ «ДП», 12; 123м-22-1		
Н.контр.		Шедловська Я.І.						
Зав. каф.		Гнатушенко В.В.						

ДОДАТОК Б

78.140.162.163:/etc/nginx/sites-enabled/lbalancer:

```
upstream backend {
    server 208.69.117.98 weight=9; #main_server
    server 206.54.190.166 backup weight=1; #secondary_server
}

server {
    listen 80;

location / {
    proxy_pass http://backend/;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
}
}
server {
    listen 443;

location / {
    proxy_pass http://backend/;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
}
}
```

206.54.190.166:/tmp/crontab.xXeOB6/crontab

```
0 */2 * * * /usr/bin/rsync -avz -e "ssh -p 2002"
/var/www/wordpress/ root@208.69.117.98:/$

0 5 * * * /var/backups/backup_script.sh
```

```
206.54.190.166: /var/backups/backup_script_and_remove_old.sh

#!/bin/bash
```

```
# перемінна для вказання директорії зберігання резервних копій
backup_dir="/var/backups"

# перемінна для вказання директорії, яка буде заархівована у
папку зберігання резервних к$
source_dir="/var/www/wordpress"

# Створення ім'я архіву, який буде відображати дату свторення
архіву
backup_file="$backup_dir/wordpress_backup_$(date +'%Y-%m-
%d').tar.gz"

# Створення архіву
tar -czf "$backup_file" "$source_dir"

# Видалення архівів, які зберігаються більше ніж 7 днів
find "$backup_dir" -name 'wordpress_backup_*' -type f -mtime
+7 -exec rm {} \;
```

206.54.190.166:/etc/apache2/sites-enabled/000-default.conf

```
<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
</Directory>

<VirtualHost *:80>

                ServerAdmin webmaster@localhost
        DocumentRoot /var/www/wordpress

        ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>


#<VirtualHost *:80>
# directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used
when creating
        # redirection URLs. In the context of virtual hosts,
the ServerName
        # specifies what hostname must appear in the request's
Host: header to
        # match this virtual host. For the default virtual
host (this file) this
        # value is not decisive as it is used as a last resort
host regardless.
        # However, you must set it for any further virtual
host explicitly.
        #ServerName www.example.com


#       ServerAdmin webmaster@localhost
#       DocumentRoot /var/www/html


        # Available loglevels: trace8, ..., trace1, debug,
info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for
particular
        # modules, e.g.
        #LogLevel info sFor most configuration files from
conf-available/, which are
        # enabled or disabled at a global level, it is
possible to
        # include a line for only one particular virtual host.
For example the
        # following line enables the CGI configuration for
this host only
```

```
        # after it has been globally disabled with
"a2disconf".

        #Include conf-available/serve-cgi-bin.conf

#</VirtualHost>


# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```