Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“ Програмне забезпечення автоматизації налаштування середовища серверів”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Напалков С.А..
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О. М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *бакалавр*
Галузь знань *12 "Інформаційні технології"*
Спеціальність *123 Комп'ютерна інженерія*
Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« __ » _____ 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Напалкову Станіславу Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення автоматизації налаштування середовища серверів*

2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу *№46-02 від 17.01.2025 року*

3. Строк подання студентом роботи до захисту *24.05.2025 р.*

4. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення автоматизації налаштування середовища серверів*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » 01 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	24.05.2025 р.	

Студент _____ **Напалков С.А.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Дрєєв О.М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Напалков С.А. Програмне забезпечення автоматизації налаштування середовища серверів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній кваліфікаційній роботі розроблено програмне забезпечення, яке призначене для автоматизації налаштування серверного середовища.

Метою розробки є вдосконалення процесу управління серверним середовищем шляхом впровадження автоматизованих рішень, що забезпечують ефективну конфігурацію, інтеграцію сервісів, моніторинг та оптимізацію робочих процесів.

Результатом роботи є програмна реалізація системи, що дозволяє автоматизувати налаштування серверів, інтегрувати їх із існуючими сервісами та здійснювати аналіз продуктивності.

У процесі розробки проведено комплексний аналіз сучасних засобів автоматизації, описано ключові технології та компоненти системи.

Розроблено інтуїтивно зрозумілий користувацький інтерфейс і наведено інструкції щодо експлуатації.

Програма може використовуватись для серверних платформ під управлінням Linux, забезпечуючи високу надійність та масштабованість розробленого рішення.

Ключові слова: автоматизація, налаштування серверів, програмне забезпечення, оптимізація, серверне середовище.

ABSTRACT

Napalkov S.A. Software for automating the configuration of server environments. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025

In this bachelor's project, software intended for automating the configuration of server environments is developed.

The purpose of the development is to enhance the efficiency of managing server environments by implementing automated solutions that improve server configuration, service integration, monitoring, and optimization of operational processes.

As a result, a software system has been implemented which enables the automatic configuration of servers, their integration with existing services, and performance analysis.

During the development process, an in-depth analysis of current automation tools was conducted; key components and technologies of the system were described.

User-friendly interface was developed and usage instructions were provided.

The program can be applied on modern server platforms running Linux, ensuring high reliability and scalability of the developed solution.

Keywords: automation, server configuration, software, optimization, server environment.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

SSL/TLS - Secure Sockets Layer / Transport Layer Security

VLAN - Virtual Local Area Network

SSH - Secure Shell

HTTPS - HyperText Transfer Protocol Secure

VPN - Virtual Private Network

YAML - Yet Another Markup Language

ВСТУП

Актуальність теми

У сучасному інформаційному суспільстві роль автоматизації, безпеки та високої продуктивності серверної інфраструктури набуває вирішального значення. Розвиток цифрових технологій та зростання кількості веб-додатків змушує організації шукати більш ефективні способи управління ресурси, забезпечення надійного доступу до даних та захисту інформації від кібератак. Сучасні виклики в галузі кібербезпеки, зокрема атаки типу DoS, DDoS та інші способи несанкціонованого доступу, стимулюють впровадження новітніх технологій для побудови відмовостійких і масштабованих систем.

Разом з тим, технології віртуалізації та автоматизації розгортання (наприклад, рішення на базі Ansible із використанням декларативних YAML-playbooks) дозволяють знизити витрати на експлуатацію, скоротити людський фактор в адмініструванні і стандартизувати процеси. У зв'язку з цим, дослідження у даній сфері має важливе практичне значення, оскільки результати роботи можуть бути застосовані як для корпоративних рішень, так і для проектів середнього та малого бізнесу.

Мета й завдання дослідження

Основною метою даної випускної кваліфікаційної роботи є розробка інтегрованої серверної інфраструктури, яка забезпечує:

- Автоматизацію розгортання та централізоване управління різноманітними серверними компонентами.

- Високу відмовостійкість і масштабованість системи, що дозволяє ефективно обробляти великі обсяги трафіку.

- Забезпечення високого рівня безпеки через використання VPN, сучасних методів автентифікації та налаштувань захисту на рівні веб-серверів і баз даних.

- Централізований моніторинг стану компонентів інфраструктури для оперативного реагування на виникнення збоїв.

Щоб досягти поставленої мети, в ході дослідження планується вирішити такі завдання:

1. Провести аналіз існуючих технологій віртуалізації, автоматизації розгортання та систем моніторингу.

2. Розробити концептуальну модель інтегрованої інфраструктури, що об’єднує VPN, веб-сервіси, сервер бази даних, reverse proxy, систему моніторингу та платформу віртуалізації.

3. Створити автоматизовані сценарії налаштування (Ansible playbook-и з YAML-описом) для розгортання окремих компонентів системи.

4. Провести тестування працездатності, продуктивності та безпеки розробленої системи.

5. Обґрунтувати не тільки технічну доцільність, а й економічну ефективність впровадження даного рішення у реальних умовах.

Практична цінність отриманих результатів

Результати даної роботи мають високу практичну цінність, оскільки впровадження розробленої системи дозволяє:

- **Знизити операційні витрати:** Завдяки автоматизації налаштування серверів і централізованому управлінню, зменшується потреба у чисельному персоналі, відповідальному за адміністрування, що, у свою чергу, скорочує витрати на експлуатацію системи.

- **Підвищити рівень безпеки:** Використання VPN для забезпечення захищеного доступу, сучасних методів автентифікації, шифрування даних, а також впровадження reverse proxy із обмеженням доступу на основі IP-діапазонів сприяє значному посиленню захисних бар’єрів проти кіберзагроз.

- **Забезпечити високу продуктивність та відмовостійкість:** Інтегрована інфраструктура дозволяє ефективно розподіляти навантаження між кількома веб-

серверами, забезпечуючи безперебійний сервіс навіть при пікових навантаженнях або локальних збоях компонентів.

- **Оптимізувати процеси управління:** Використання декларативних YAML-playbook-ів для автоматизації розгортання гарантує стандартизованість і повторюваність налаштувань, що є надзвичайно важливим для великих корпоративних і хмарних рішень.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Основне призначення даної системи полягає у створенні єдиного інтегрованого рішення, яке охоплює наступні функції:

1. **Забезпечення захищеного доступу до внутрішньої мережі.** Для цього використовується VPN-сервер, який організовано з використанням рішення Pritunl. Завдяки підтримці двофакторної автентифікації, система гарантує, що до внутрішніх ресурсів можуть підключатися лише авторизовані користувачі, що суттєво підвищує рівень безпеки корпоративних мереж.

2. **Обслуговування веб-запитів та розподіл навантаження.** Розгорнуті два веб-сервіси – Web Service 1 та Web Service 2 – відповідальні за обробку запитів від користувачів. Ці сервери інтегровані з системою балансування навантаження, де кожен запит, що приходить ззовні (через Cloudflare), розподіляється автоматично між ними. Завдяки цьому досягається не тільки рівномірний розподіл обчислювального навантаження, але й забезпечення безперервності обслуговування навіть у разі збою одного з серверів.

3. **Надійне зберігання та обробка даних.** Сервер баз даних з використанням MariaDB забезпечує централізоване зберігання інформації веб-додатків. Висока продуктивність СУБД гарантує швидкий доступ до даних, а ретельне налаштування безпеки дозволяє уникнути несанкціонованого доступу, забезпечуючи цілісність та актуальність інформації.

4. **Моніторинг системи та автоматичне сповіщення про збої.** Впровадження сервера моніторингу на базі Zabbix дозволяє у режимі реального часу контролювати стан усіх компонентів інфраструктури. Система збирає детальні показники працездатності, допомагає виявити потенційні проблеми і автоматично сповіщає адміністраторів про виникнення критичних ситуацій, що

дає змогу миттєво реагувати та проводити необхідне модифікаційне обслуговування.

5. **Захист HTTP-запитів та забезпечення безпеки застосунку.** Використання NGINX reverse proxy дозволяє не лише ефективно балансувати навантаження між веб-серверами, але й здійснювати додатковий захист від зовнішніх загроз. Завдяки специфічним налаштуванням (зокрема, обмеженню доступу до системи лише з IP-діапазонів Cloudflare) система мінімізує ризики атак, забезпечуючи безперебійне функціонування веб-сервісів.

6. **Централізоване управління та масштабування інфраструктури.** Платформа віртуалізації Proxmox дозволяє об’єднати всі серверні компоненти в єдину систему, забезпечуючи гнучке управління ресурсами, швидкий розгорт сервісів і легке масштабування в разі збільшення навантаження або розширення функціоналу.

У підсумку, розроблена система спрямована на оптимізацію бізнес-процесів в галузі IT, зменшення операційних витрат, підвищення рівня безпеки та стабільності роботи інформаційних ресурсів. Інтеграція сучасних технологій та автоматизація розгортання забезпечує високу ефективність і адаптивність системи до мінливих умов ринку.

1.2 Область застосування

Розроблена система може бути впроваджена в різних сферах і використовуватися в умовах, коли необхідно поєднання високої продуктивності, безпеки та масштабованості. До основних областей застосування належать:

- **Корпоративні мережі та організації.** Для підприємств системи автоматизованого розгортання дозволяють забезпечити захищений доступ до внутрішніх ресурсів через VPN, контролювати стан IT-інфраструктури і своєчасно реагувати на виникнення збоїв. Це особливо актуально для банків,

телекомунікаційних компаній та великих корпорацій, де безпека даних та безперервність роботи мають першорядне значення.

- **Хостинг та розробка веб-додатків.** Веб-сервіси, що використовуються для обслуговування інтернет-ресурсів з високою відмовостійкістю, ідеально підходять для хостингових компаній, стартапів і навіть державних проектів, де важлива здатність системи обробляти велику кількість запитів одразу. Розподіл навантаження за допомогою NGINX і Cloudflare забезпечує ефективне управління піковим трафіком.

- **Інформаційні та науково-дослідні установи.** У таких організаціях часто виникає потреба в централізованому зберіганні великих обсягів даних і участі багатьох користувачів у спільній роботі інфраструктури. Система, що включає сервер баз даних, моніторинговий сервер і платформу віртуалізації, сприяє ефективному адмініструванню таких ресурсів.

- **Застосування DevOps-практик та хмарні рішення.** Завдяки використанню технологій автоматизації, таких як Ansible, розроблена система легко інтегрується в CI/CD процеси. Це дозволяє оперативно розгортати оновлення, впроваджувати нові функціональні можливості та проводити тестування навіть у великих, розподілених інфраструктурах.

- **Розгортання сучасних інформаційних систем.** Об’єднання VPN-сервера, веб-сервісів, бази даних і систем моніторингу в єдине рішення робить систему універсальним інструментом для створення гнучких інформаційних портфелів. Це дозволяє адаптувати систему до вимог ринку, інтегрувати нові технології та забезпечувати безперебійність роботи критично важливих процесів.

Розроблена система є комплексною платформою, яка інтегрує різні технологічні рішення для побудови високопродуктивної, захищеної та масштабованої інфраструктури. Основні технічні параметри системи включають:

- **Автоматизоване розгортання:** Система використовує Ansible для централізованого керування та розгортання конфігурацій усіх серверів, що дозволяє стандартизувати налаштування та зменшити людський фактор.

- **Балансування навантаження та відмовостійкість:** Використання NGINX reverse proxy та Cloudflare гарантує, що запити розподіляються між кількома веб-серверами, а у разі відмови одного компоненту система продовжує функціонувати без перебоїв.

- **Захищений доступ та централізоване управління:** VPN-сервер із підтримкою Pritunl і двофакторної автентифікації забезпечує безпечний доступ до внутрішніх ресурсів, що особливо актуально в умовах сучасних кіберзагроз.

- **Ефективне зберігання та обробка даних:** Використання MariaDB дозволяє забезпечити швидкий і стабільний доступ до інформації, а налаштування безпеки бази даних мінімізує ризики несанкціонованого доступу.

- **Система моніторингу і своєчасного реагування:** Zabbix, як засіб моніторингу, дозволяє відстежувати показники працездатності всієї інфраструктури в режимі реального часу, що сприяє оперативному виявленню та усуненню проблем.

У підсумку, розроблена система відповідає найсучаснішим вимогам у сфері ІТ. Її використання дозволить знизити операційні витрати, підвищити ефективність керування інфраструктурою та гарантувати високу ступінь безпеки та стабільності роботи як корпоративних мереж, так і хмарних платформ.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Ansible – це відкритий інструмент для автоматизації, який дозволяє керувати конфігураціями, розгортати додатки та оркеструвати IT-інфраструктуру за допомогою простих YAML-playbook-ів. Завдяки агентless-архітектурі Ansible легко інтегрується у різні середовища, дозволяючи досягати високої стандартизації налаштувань і швидкого масштабування без необхідності встановлення додаткових клієнтів на кожному вузлі.

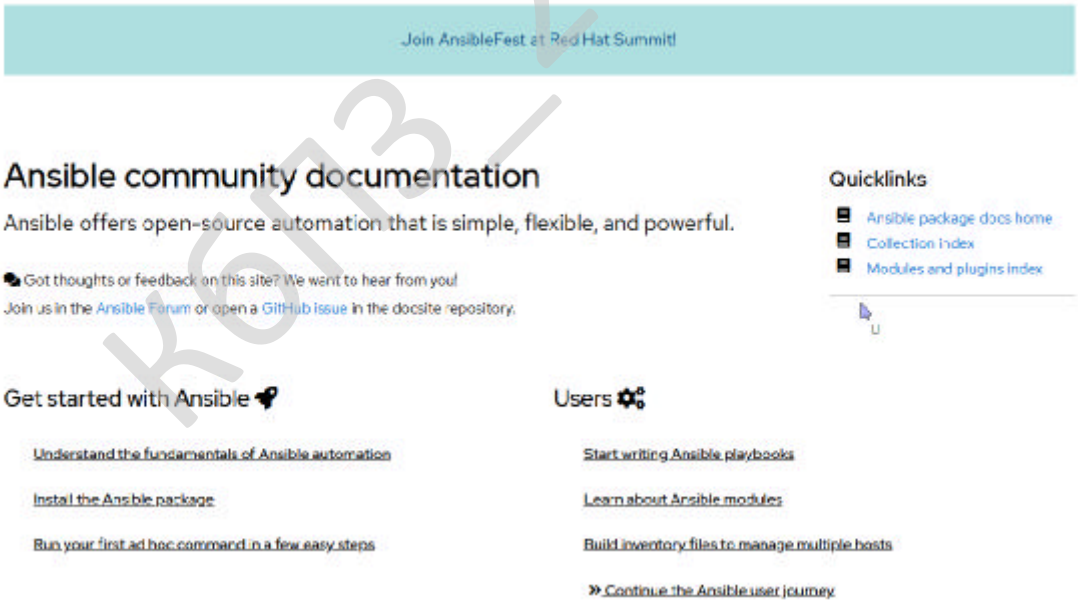

Рисунок 2.1 – Головна сторінка документації Ansible

Основні функції:

- Автоматизація налаштування конфігурації серверів
- Розгортання додатків і оновлень
- Оркестрація складних процесів між компонентами інфраструктури
- Виконання ad-hoc команд для швидких операцій
- Управління інвентаризацією хостів і груповим розгортанням

Puppet – це система управління конфігураціями, що використовує декларативний підхід через власну мову (Puppet DSL) для опису бажаного стану серверів. Завдяки постійному контролю за відповідністю фактичної конфігурації визначеним правилам Puppet дозволяє автоматично підтримувати системи в заданому стані, що особливо корисно у великих корпоративних середовищах.

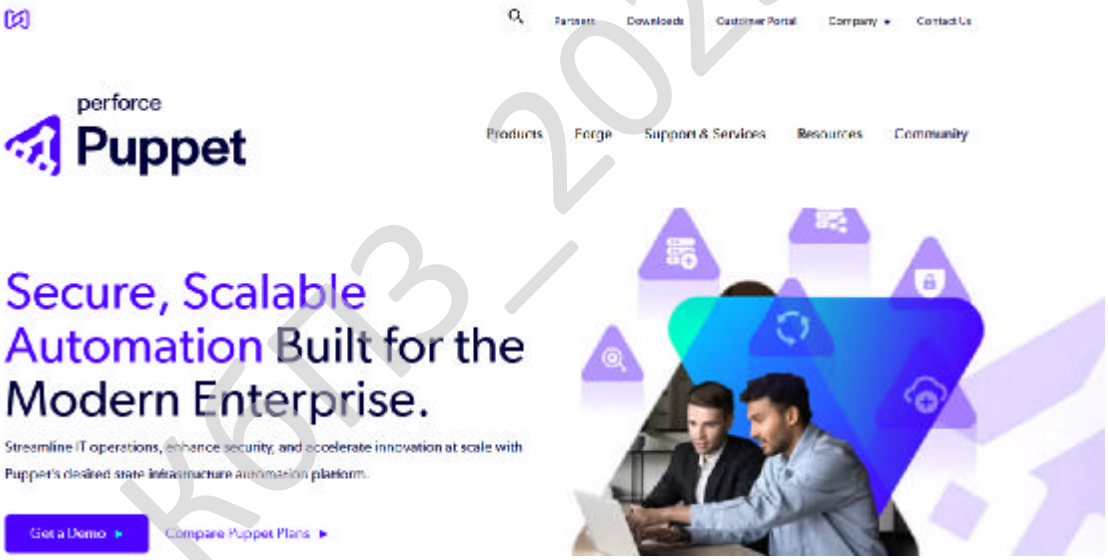

Рисунок 2.2 – Головна сторінка Puppet

Основні функції:

- Декларативне описання конфігурації інфраструктури
- Автоматичне забезпечення відповідності фактичного стану заданому
- Централізоване управління великою кількістю серверів
- Модульна організація конфігурацій для повторного використання
- Інтеграція з системами моніторингу та звітності

Chef – це інструмент для автоматизації, який використовує Ruby DSL для опису "рецептів" конфігурації, що дозволяють програмно керувати інфраструктурою. Chef орієнтований на підприємства, де потрібне детальне налаштування серверних середовищ та гнучка інтеграція процесів розгортання в CI/CD процеси.

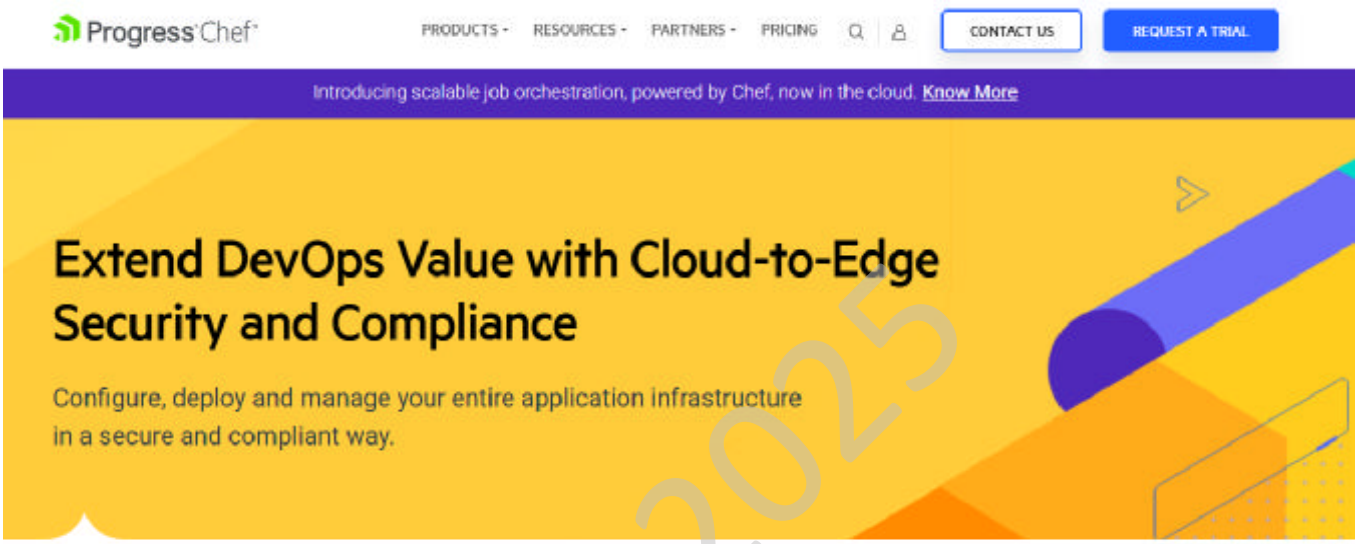

Рисунок 2.3 – Головна сторінка Chef

Основні функції:

- Опис конфігурацій у вигляді рецептів на базі Ruby
- Автоматичне застосування змін до конфігурації серверів
- Управління залежностями та середовищами розгортання
- Інтеграція з системами керування версіями і CI/CD
- Можливість глибокої кастомізації процесів розгортання

SaltStack – це система для оркестрації та управління конфігураціями, яка відзначається високою швидкістю виконання завдань завдяки асинхронній моделі управління. SaltStack підтримує як push-, так і pull-підходи, що робить його ефективним для масштабного керування сотнями або тисячами вузлів у динамічних середовищах.

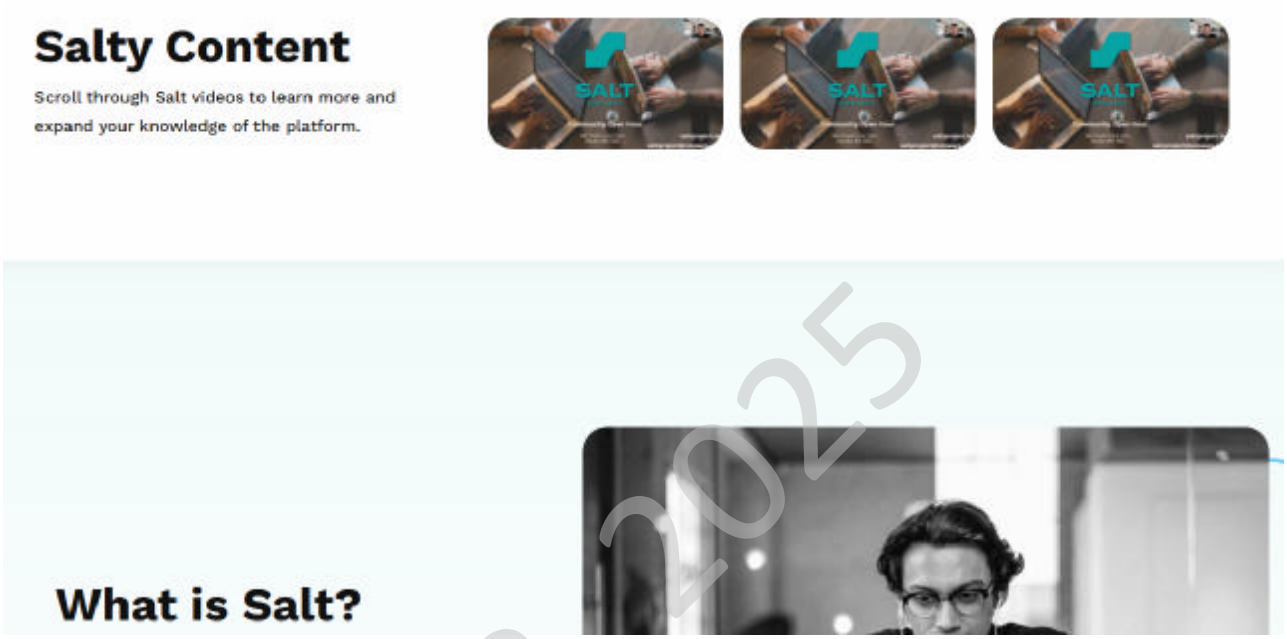

Рисунок 2.4 – Головна сторінка Salt

Основні функції:

- Швидке виконання команд у режимі реального часу
- Підтримка push- і pull-моделей керування
- Масштабоване управління великою кількістю серверів
- Автоматизація конфігурації та оркестрації процесів
- Збір та агрегація даних у режимі реального часу для аналізу та моніторингу

Terraform – це інструмент від HashiCorp, який спеціалізується на описі інфраструктури як коду, зосереджуючись переважно на розгортанні хмарних ресурсів. Він дозволяє створювати відтворювані та масштабовані описання інфраструктури, ефективно інтегруючи різні хмарні сервіси та забезпечуючи автоматизоване управління ресурсами.

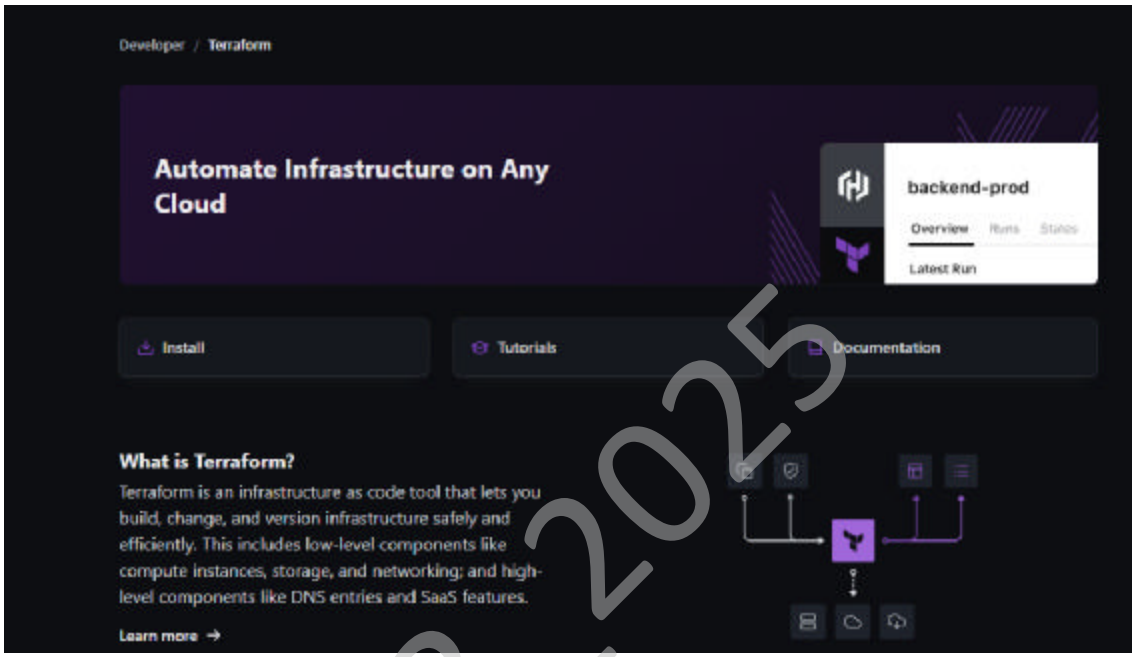

Рисунок 2.5 – Головна сторінка Terraform

Основні функції:

- Опис інфраструктури в декларативних конфігураційних файлах
- Автоматизоване створення, оновлення та видалення хмарних ресурсів
- Підтримка багатьох хмарних провайдерів та власних сервісів
- Забезпечення повторюваного розгортання інфраструктури
- Взаємодія з CI/CD процесами для автоматизації оновлень

CFEngine – це система управління конфігураціями з потужним двигуном для підтримки постійної відповідності інфраструктури заданому стану. Вона орієнтована на вирішення завдань високої ефективності та безперебійності, забезпечуючи автоматичне самовідновлення системи навіть у великих масштабах.

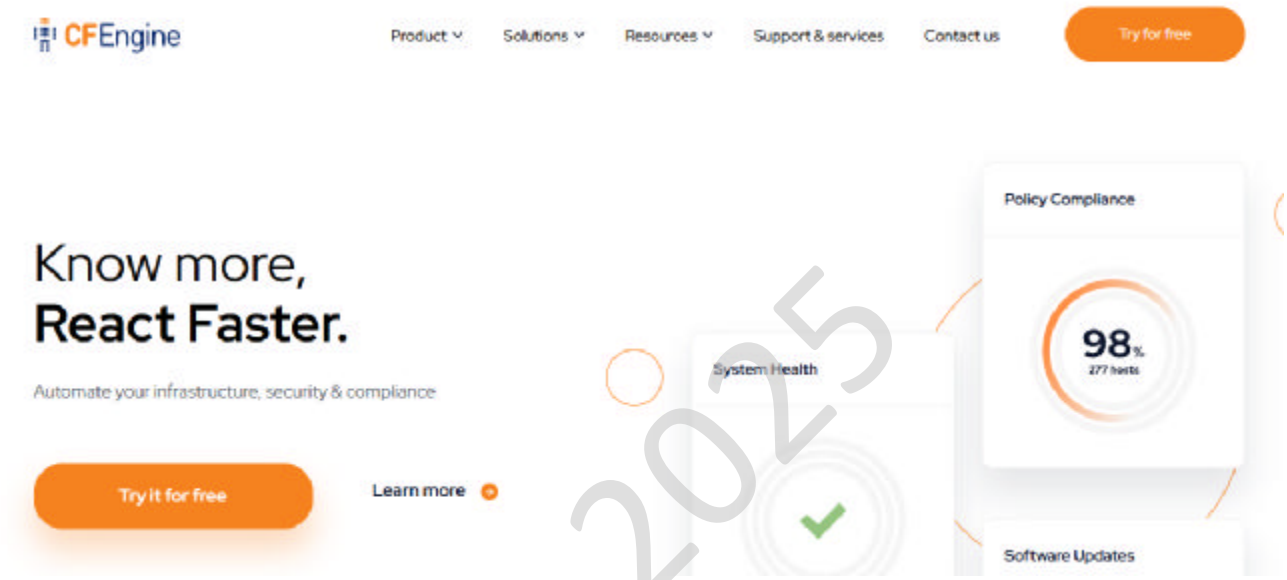

Рисунок 2.6 – Головна сторінка CFEngine

Основні функції:

- Забезпечення постійної відповідності системи заданому стану
- Висока швидкість виконання політик конфігурації
- Автоматичне виявлення та виправлення невідповідностей
- Централізоване управління великою кількістю вузлів
- Орієнтація на високі вимоги до продуктивності та безпеки

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки програмного забезпечення автоматизації налаштування середовища серверів було обрано середовище розробки PyCharm та мову програмування YAML.

YAML (YAML Ain't Markup Language) – це декларативна мова опису даних, яка вирізняється простим, інтуїтивно зрозумілим синтаксисом і високу читабельністю. Завдяки своїй легкості у використанні YAML дозволяє розробникам та адміністраторам швидко створювати, редагувати та підтримувати конфігураційні файли, що описують інфраструктуру як код. Це робить YAML оптимальним вибором для автоматизації процесів розгортання за допомогою таких інструментів, як Ansible, де важлива стандартизація налаштувань і повторюваність операцій. YAML забезпечує можливість зберігати складні структури даних із використанням вкладеності та списків, що підвищує зручність інтеграції з системами контролю версій і полегшує масштабування інформаційних систем.

YAML використовує інтуїтивно зрозумілий синтаксис, заснований на відступах (з використанням пробілів) для визначення ієрархії та структурованості даних. Ключі й значення розділяються символом двокрапки, після якої розміщується пробіл, що створює зрозумілий формат читання. Списки елементів позначаються за допомогою дефісів, розташованих на початку рядка з відповідним відступом, що вказує на їх перебування в одному рівні. Коментарі записуються за допомогою символу # і ігноруються при обробці. За рахунок можна використовувати як неформатований текст, так і рядкові значення у лапках для точного збереження форматування. Цей лаконічний синтаксис сприяє легкому розгляду, модифікації та інтеграції YAML у різні проєкти, особливо в контексті автоматизації процесів через інструменти як Ansible.

Завдяки потужним інструментам PyCharm для створення, редактування та тестування коду, розробка конфігураційних файлів у форматі YAML стає більш ефективною. Інтегровані можливості автозаповнення, перевірки синтаксису та дебагінгу значно скорочують час на виявлення та виправлення помилок, що є вкрай важливим при роботі з декларативними playbook-ами. Крім того, інтуїтивно зрозумілий інтерфейс PyCharm забезпечує зручність для як новачків, так і досвідчених розробників, що дозволяє швидко адаптуватися до роботи в одному єдиному середовищі, знижуючи витрати часу на перемикання між різними інструментами.

PyCharm — це потужне інтегроване середовище розробки (IDE) для мови програмування Python, створене компанією JetBrains. Воно поєднує в собі широкий набір інструментів, які допомагають розробникам писати, налагоджувати, тестувати та розгортати код ефективно та зручно. PyCharm підтримує всі сучасні версії Python і має інтеграцію з популярними фреймворками, такими як Django та Flask, що робить його ідеальним вибором для веброзробників

Крім того, PyCharm відзначається ефективною інтеграцією з системами контролю версій та іншими корисними інструментами, що сприяє організованій командній роботі та централізованому управлінню змінами у проектах. Це забезпечує безперебійну співпрацю між розробниками при роботі над великими проектами, де важлива повторюваність та стандартизація процесів. Завдяки підтримці різноманітних форматів файлів, PyCharm дозволяє зручно працювати не лише з програмним кодом, а й із конфігураційними файлами, що робить його оптимальним рішенням для впровадження та підтримки сучасних автоматизованих інфраструктурних рішень.

Проєкт у PyCharm організовується у вигляді структури каталогів, де верхній рівень називається коренем вмісту (content root). У межах цього кореня можна позначати папки як папки з вихідним кодом, тестові папки, ресурси або виключені з індексації. Така організація допомагає IDE ефективно індексувати

файли, розпізнавати імпорти та розділяти код для тестування і продуктивного використання.

Інтерфейс PyCharm складається з кількох логічних зон: панелі проекту, редактора коду, панелі навігації, лівої та правої колонок (gutter) для відображення брейкпоінтів, номерів рядків, помилок і попереджень. Ця структура дозволяє швидко орієнтуватися у коді, запускати додатки, налагоджувати їх та контролювати якість коду в реальному часі.

Архітектура PyCharm базується на модульному підході, що забезпечує гнучкість і масштабованість середовища розробки. Кожен функціональний блок IDE реалізований як окремий модуль, який відповідає за певний набір задач, наприклад, редактор коду, система налагодження, підтримка систем контролю версій або інтеграція з базами даних. Така модульність дозволяє розробникам JetBrains легко додавати нові функції через плагіни та налаштовувати середовище під індивідуальні потреби користувачів.

Модулі в PyCharm мають чітко визначені межі та взаємодіють між собою через інтерфейси, що забезпечує слабку зв’язність і високу інкапсуляцію. Це дозволяє мінімізувати залежності між компонентами, що полегшує підтримку та розвиток IDE. Наприклад, модуль автодоповнення коду працює незалежно від модуля налагодження, але може отримувати необхідні дані через стандартизовані API.

Архітектура також передбачає використання загального ядра, яке містить базові сервіси, необхідні для роботи всіх модулів. Сюди входять такі функції, як логування, обробка помилок, управління конфігурацією та індексація проектів. Завдяки цьому кожен модуль може зосередитися на своїй основній задачі, не дублюючи загальний функціонал.

Інтерфейс користувача PyCharm організований у вигляді окремих панелей і вікон, які також реалізовані як модулі. Це дозволяє користувачам налаштовувати робочий простір, додавати або приховувати потрібні інструменти, а розробникам — легко розширювати UI за допомогою плагінів. Така архітектура забезпечує зручність і ефективність роботи з кодом.

2.3 Розгорнута постановка завдання

В рамках випускної кваліфікаційної роботи на тему “Програмне забезпечення автоматизації налаштування середовища серверів” розроблено наступний перелік завдань:

Провести критичний аналіз існуючих рішень. Необхідно здійснити детальний огляд сучасних систем автоматизації та управління конфігураціями (Ansible, Puppet, Chef, SaltStack, Terraform) та рішень для віртуалізації (Proxmox VE, VMware vSphere, Microsoft Hyper-V, Citrix Hypervisor). В аналізі слід виявити сильні та слабкі сторони кожного підходу з акцентом на аспекти безпеки, універсальності, продуктивності та масштабованості. Результати аналізу використати для формування технічних вимог до розроблюваної системи, щоб визначити, які характеристики та можливості відповідають умовам реальної експлуатації та допомагають знизити експлуатаційні витрати.

Обґрунтувати вибір архітектури системи. На основі проведеного аналізу необхідно сформувати концептуальну модель інтегрованої серверної інфраструктури, що включає VPN-сервер з підтримкою двофакторної автентифікації, два веб-сервіси, налаштовані через NGINX reverse proxy для балансування навантаження, сервер бази даних на базі MariaDB та систему моніторингу з використанням Zabbix. Вибір архітектури має бути обґрунтований з точки зору здатності забезпечити високий рівень відмовостійкості, безпеки та масштабованості. Необхідно побудувати структурну та функціональну схеми системи, які ілюструють взаємодію всіх компонентів та логіку обробки запитів у

системі.

Сформувати висновки та перспективи подальшого розвитку. Після реалізації всіх етапів робіт необхідно провести комплексну оцінку виконаного обсягу робіт, проаналізувати діючість інтегрованої системи з точки зору її продуктивності, безпеки, відмовостійкості та зручності управління. Висновки мають містити оцінку придатності системи для впровадження в умовах реального використання, а також визначити перспективи подальшого розвитку – розширення функціональних можливостей, інтеграцію з іншими технологічними платформами та оптимізацію алгоритмів моніторингу та автоматизації.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Система побудована за принципом «інфраструктура як код» і складається з кількох взаємопов’язаних компонентів, що забезпечують автоматизацію розгортання, управління та моніторинг серверних ресурсів. Основні елементи включають VPN-сервер для захищеного доступу, веб-сервіси для обробки запитів, сервер баз даних для зберігання інформації, NGINX reverse proxy для балансування навантаження, систему моніторингу на базі Zabbix, а також платформу віртуалізації Proxmox для централізованого управління ресурсами.

VPN-сервер виступає першим рівнем захисту, забезпечуючи безпечний вхід до внутрішньої мережі організації. Система використовує рішення Pritunl, яке дозволяє встановити захищене з’єднання завдяки двофакторній автентифікації. Такий підхід не лише гарантує ідентифікацію користувачів, але й формує зашифрований тунель для передачі даних, що мінімізує ризик несанкціонованого доступу або перехоплення інформації. Завдяки цьому компоненту, кожен користувач, що прагне отримати доступ до системи, забезпечується високим рівнем безпеки ще до початку взаємодії з іншими елементами інфраструктури.

Забезпечення безпеки роботи системи є одним із ключових аспектів. Впровадження VPN-сервера з підтримкою двофакторної автентифікації гарантує, що доступ до внутрішньої мережі матимуть лише уповноважені користувачі, а шифрування даних під час передачі мінімізує ризики несанкціонованого доступу. Крім того, використання NGINX reverse proxy з фільтрацією доступу за IP-діапазонами забезпечує додатковий рівень захисту від зовнішніх кібератак.

NGINX reverse proxy виступає ключовим елементом для балансування навантаження і захисту HTTP-запитів, виконуючи роль фільтрації запитів за IP-діапазонами (наприклад, через Cloudflare) та централізованого логування. Це дозволяє відстежувати активність користувачів і вчасно ідентифікувати потенційні загрози. Водночас, платформа віртуалізації Proxmox забезпечує розгортання всіх серверних компонентів у віртуалізованому середовищі, що дає змогу централізовано управляти ресурсами. Proxmox дозволяє гнучко масштабувати інфраструктуру, реагувати на зміну навантаження і оперативно забезпечувати технічну підтримку, що сприяє високій адаптивності та економічності системи.

Веб-сервіси є центральним елементом системи і відповідають за обробку запитів від користувачів. За рахунок використання NGINX reverse proxy запити розподіляються між декількома вузлами, що працюють паралельно, що забезпечує балансування навантаження. Цей підхід дозволяє системі підтримувати високу продуктивність та ефективно обробляти навіть пікові навантаження, гарантуючи безперебійну роботу сервісів.

Основна обробка запитів користувачів здійснюється за допомогою двох незалежних веб-сервісів. Ці сервери розміщуються паралельно, що дозволяє рівномірно розподіляти навантаження та збільшувати загальну продуктивність системи. За участю NGINX reverse proxy, що виконує роль балансувальника навантаження, система коректно розподіляє вхідні запити між обома веб-серверами, що гарантує стабільну роботу навіть у разі недоступності одного з серверів. Така архітектурна модель дозволяє забезпечити безперебійну роботу веб-сервісів, незважаючи на пікові навантаження або технічні збої на окремих вузлах.

Сервер баз даних, що реалізовано на основі MariaDB, забезпечує зберігання та оперативний доступ до інформації, необхідної для роботи веб-сервісів. Швидка обробка запитів до бази даних і підтримка масштабованості дозволяють ефективно працювати з великими обсягами даних, оскільки

інформація, оброблена веб-сервісами, миттєво записується або оновлюється в базі за допомогою оптимізованих алгоритмів управління даними.

Для зберігання і оперативного доступу до інформації веб-додатків було обрано MariaDB. Цей сервер баз даних забезпечує надійне зберігання даних, підтримуючи їх цілісність і високошвидкісний доступ. Оптимізовані налаштування СУБД дозволяють ефективно виконувати операції читання і запису, що є критично важливим при обробці великих обсягів даних. Надійні механізми резервного копіювання та відновлення, інтегровані в MariaDB, знижують ризик втрати даних при виникненні збоїв, що позитивно впливає на безперервність роботи системи.

Завдяки інтегрованій системі моніторингу на базі Zabbix всі компоненти інфраструктури постійно контролюються в режимі реального часу. Це дозволяє оперативно виявляти збої і сповіщати адміністраторів про відхилення від нормальної роботи, що сприяє автоматичному відновленню системи або швидкому втручанню. Центральне управління через платформу Proxmox забезпечує гнучкість і масштабованість системи, що є критично важливим для впровадження рішення в умовах сучасного IT-середовища з високою продуктивністю і вимогами до безпеки.

Підключення користувачів до інтернет-додатку здійснюється традиційним шляхом через NGINX reverse proxy, який приймає запити від звичайних користувачів через Cloudflare. Це забезпечує доступ до веб-застосунку, який обслуговується паралельно розгорнутими Web Service 1 та Web Service 2, таким чином забезпечуючи балансування навантаження і підвищення відмовостійкості системи. Окрім того, адміністративний персонал використовує VPN-підключення для доступу до інфраструктури системи з метою її обслуговування, моніторингу та внесення технічних змін, що гарантує безпечний захищений доступ до внутрішніх ресурсів. Коли запит надходить через Cloudflare до NGINX, він перевіряється на відповідність налаштованим IP-діапазонам і розподіляється між веб-сервісами для подальшої обробки. У разі потреби обробка запиту взаємодіє із

сервером баз даних для отримання або збереження даних. Протягом усього процесу система моніторингу (на базі Zabbix) відстежує продуктивність і функціонування всіх компонентів, що дозволяє оперативно виявляти та усувати недоліки. Така організація взаємодії забезпечує високий рівень доступності, безпеки і продуктивності як для кінцевих користувачів, так і для адміністраторів, що обслуговують систему через VPN.

3.2 Розробка структурної схеми

На рисунку 3.1 зображено структуру схему взаємодії клієнтів з системою, яка складається з наступних блоків:

- **Клієнти (Клієнт 1, Клієнт …, Клієнт N).** Цей блок представляє групу кінцевих користувачів або пристроїв, які надсилають запити до системи. Кожен з користувачів може бути представленим як окрема сутність, що ініціює взаємодію з веб-додатком.

Запити можуть формуватися з використанням різних пристроїв (комп'ютерів, смартфонів, планшетів) і надходити з різних географічних розташувань, що створює вимогу до високої гнучкості і масштабованості системи.

- **Інтернет.** Блок «Інтернет» відображає глобальну мережу, яка є каналом передачі даних між кінцевими користувачами та серверною інфраструктурою. Він забезпечує з'єднання між клієнтами та зовнішніми сервісами, зокрема Cloudflare.

Цей блок є фундаментальним, оскільки забезпечує маршрутизацію даних через публічні канали, а також впливає на затримки і загальну продуктивність системи. У контексті схеми, Інтернет дозволяє клієнтам надсилати запити, які потім проходять через ряд проміжних блоків для фільтрації та обробки.

– **Cloudflare.** Cloudflare виступає як проміжний шар між Інтернетом і внутрішніми серверними компонентами. Він забезпечує підвищену безпеку, оптимізацію трафіку та поліпшення продуктивності за рахунок кешування і розподілу навантаження. Cloudflare приймає запити від клієнтів, фільтрує їх за допомогою встановлених політик і передає далі в внутрішню мережу. Сервіс також залучає додаткові алгоритми для запобігання DDoS-атакам і може виконувати різного роду аудити вхідного трафіку, що суттєво підвищує стійкість системи до зовнішніх загроз.

– **Блокування за геолокацією.** Цей блок відповідає за фільтрацію запитів на основі географічного розташування клієнтів. За допомогою визначення IP-адрес і порівняння їх з дозволеними або заблокованими діапазонами, система може вирішувати, з яких регіонів надходять запити, та відсіювати ті, що не відповідають встановленим критеріям. Такий підхід використовується для запобігання спробам доступу з потенційно небезпечних регіонів або для забезпечення додаткової відповідності політикам безпеки організації. Блокування за геолокацією допомагає знизити ризики несанкціонованого доступу та забезпечує додатковий рівень контролю над трафіком.

– **Перевірка валідності запиту.** Після первинного фільтрування за геолокацією наступним етапом є перевірка валідності самого запиту. Цей блок аналізує вхідні дані — наприклад, заголовки HTTP, сертифікати безпеки або інші параметри — для визначення того, чи є запит достовірним та відповідає встановленим стандартам безпеки. Перевірка може включати додаткові методи аутентифікації або валідації вмісту запиту та гарантує, що некоректні або потенційно шкідливі запити не потрапляють далі у внутрішню інфраструктуру. Це критично важливо для забезпечення захищеності системи від зловмисних атак.

– **Передача запиту на Web-server.** Після проходження попереднього етапу фільтрації та перевірки запит передається до веб-серверів. Цей блок відповідає за маршрутизацію запиту до одного з доступних серверів, що обслуговують веб-додаток. В процесі передачі використовуються алгоритми балансування навантаження, які визначають оптимальний сервер для обробки конкретного запиту з урахуванням поточної завантаженості і продуктивності. Цей етап є ключовим для забезпечення стабільності роботи веб-додатків, адже він дозволяє мінімізувати затримки і рівномірно розподілити обчислювальні навантаження.

– **Обробка запиту Web-server.** Веб-сервер, отримавши запит, виконує обчислювальну і логічну обробку, що включає виконання бізнес-логіки, взаємодію з базою даних і підготовку відповіді. Обробка може включати перевірку вхідних даних, генерацію динамічного контенту та інтеграцію різних підсистем для формування остаточного результату. Завдяки цьому блоку користувачі отримують релевантну інформацію чи виконані операції, залежно від характеру запиту. Веб-сервери зазвичай працюють паралельно для забезпечення високої продуктивності і відмовостійкості.

– **Формування і відправка відповіді.** Після обробки запиту веб-сервер формує відповідь, яка містить результати операцій або інформацію, що була запрошена користувачем. Цей блок відповідає за упаковку даних, їх форматування відповідно до встановлених стандартів і вихідну передачу через систему. Відповідь надходить назад через NGINX reverse proxy, де може бути проведено додаткову перевірку або оптимізацію, перш ніж потрапити до кінцевих клієнтів. Така організація дозволяє системі забезпечити швидкий і надійний зворотній зв’язок, що є важливим для задоволення вимог кінцевих користувачів і підтримання високої якості обслуговування.

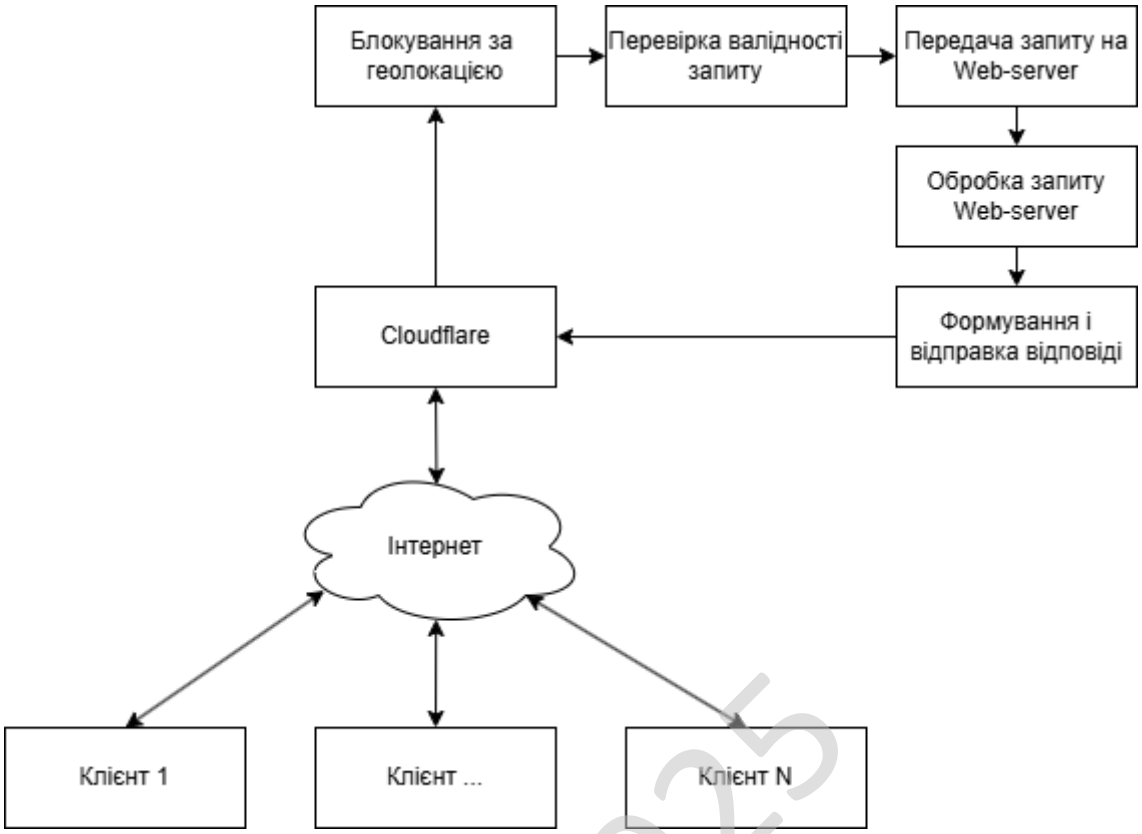

Рисунок 3.1 – Структурна схема взаємодії клієнтів з системою

На рисунку 3.2 зображено структуру схему взаємодії адміністратора з системою, яка складається з наступних блоків:

– **Адміністратор.** Цей блок позначає користувача з рівнем доступу адміністратора, який ініціює процес керування системою та надсилання інструкцій.

– **Інтернет.** Блок відповідає за мережеве з'єднання, через яке адміністратор отримує доступ до системи, забезпечуючи передачу даних через глобальну мережу.

– **VPN.** Використання VPN забезпечує створення захищеного каналу зв'язку, який ізолює трафік та гарантує безпеку даних під час передавання між адміністратором і серверною інфраструктурою.

– **Ansible.** Автоматизаційний інструмент Ansible відповідає за формування сценаріїв (playbook'ів), що містять команди, та їх відправку до серверів для управління та налаштування ресурсів.

– **SSH сесія.** Цей блок позначає встановлення захищеного з’єднання з серверами через протокол SSH, який забезпечує безпечну передачу команд від Ansible до віддалених серверів.

– **Передача інструкцій на сервер.** На цьому етапі Ansible надсилає сформовані інструкції до сервера через встановлене SSH-з’єднання, забезпечуючи правильну комунікацію між керуючим та керованим компонентами.

– **Виконання інструкцій.** Сервер приймає отримані команди та виконує їх, реалізуючи необхідні операції та налаштування згідно з переданими інструкціями.

– **Формування і відправка відповіді.** Після виконання команд сервер формує відповідь, яка відображає стан виконання операцій, і відправляє її назад адміністратору через SSH сесію, завершуючи цикл управління.

Рисунок 3.2 – Структурна схема взаємодії адміністратора з системою

3.3 Розробка функціональної схеми

Функціональна схема включає механізми перевірки валідності запитів, алгоритми взаємодії веб-сервісів із базою даних, а також логіку моніторингу продуктивності через Zabbix. Завдяки цьому схема надає точне уявлення про послідовність обробки кожного запиту, визначає правила маршрутизації трафіку та описує механізми автоматичного реагування на збої.

На рисунку 3.3 зображено функціональну схему взаємодії клієнтів з системою, яка складається з наступних трьох основних блоків – “Клієнт”, “Cloudflare” та “Сервер”.

Блок “Клієнт” складається з наступних пунктів:

– Інтернет – глобальна мережа забезпечує передачу запитів від клієнта до веб-сервера, дозволяючи користувачам надсилати HTTP/HTTPS-запити до веб-додатків. Вона слугує середовищем для комунікації між користувачем та інфраструктурою сервера через різні маршрути.

– Модуль формування HTTPS запиту – генерує правильний формат HTTP/HTTPS-запиту з необхідними заголовками, параметрами та методами (GET, POST, PUT тощо). Він структурує запит відповідно до специфікації сервера, включаючи автентифікаційні дані та кодовану інформацію.

– Модуль відправки HTTPS запиту – відповідає за безпечну передачу сформованого запиту до призначеного веб-сервера через зашифрований канал HTTPS. Цей модуль ініціює запит і контролює його надсилання, забезпечуючи надійну передачу даних відповідно до стандартів безпеки.

– Модуль прийому відповіді – отримує відповідь від сервера, обробляє отримані заголовки та вміст для подальшого відображення користувачу. Він також може аналізувати статус-код відповіді для коректної обробки (наприклад, 200 – успіх, 404 – ресурс не знайдено, 500 – внутрішня помилка сервера).

Блок “Cloudflare” складається з наступних пунктів:

– Модуль блокування за геолокацією – виконує перевірку IP-адреси

користувача та визначає його географічне розташування, після чого порівнює його з налаштованими політиками доступу. Якщо запит надходить із забороненого регіону, він блокується або обмежується відповідно до правил безпеки.

– Модуль перевірки валідності запитую – аналізує структуру HTTP-запиту, перевіряє заголовки, сертифікати безпеки та автентифікаційні дані для виявлення підозрілих запитів. Некоректні або потенційно шкідливі запити можуть бути заблоковані або оброблені з додатковими рівнями перевірки.

– Модуль передачі запиту на сервер – після успішного проходження всіх перевірок цей модуль переспрямовує запит до веб-сервера для подальшої обробки. Він також може виконувати оптимізацію трафіку, забезпечуючи швидку та ефективну маршрутизацію запитів.

Блок “Сервер” складається з наступних пунктів:

– Модуль прийому запиту – веб-сервер отримує запит, перевіряє його на коректність та автентичність, а також фіксує необхідні параметри для подальшої обробки. Якщо запит відповідає критеріям безпеки та структурі протоколу, він передається у наступний модуль для виконання необхідних операцій.

– Модуль обробки запиту – виконує основні логічні та обчислювальні операції відповідно до вмісту запиту, зокрема взаємодію з базою даних, обробку введених даних або виконання внутрішніх алгоритмів. Це ключовий етап, де визначається, які дії слід виконати для підготовки коректної відповіді користувачу.

– Модуль формування відповіді – на основі обробленого запиту створює відповідь у форматі, що відповідає встановленим стандартам (наприклад, JSON або HTML), забезпечуючи її коректне відображення для користувача. Після цього відповідь відправляється назад через мережу до клієнта, гарантує швидку і безперебійну комунікацію.

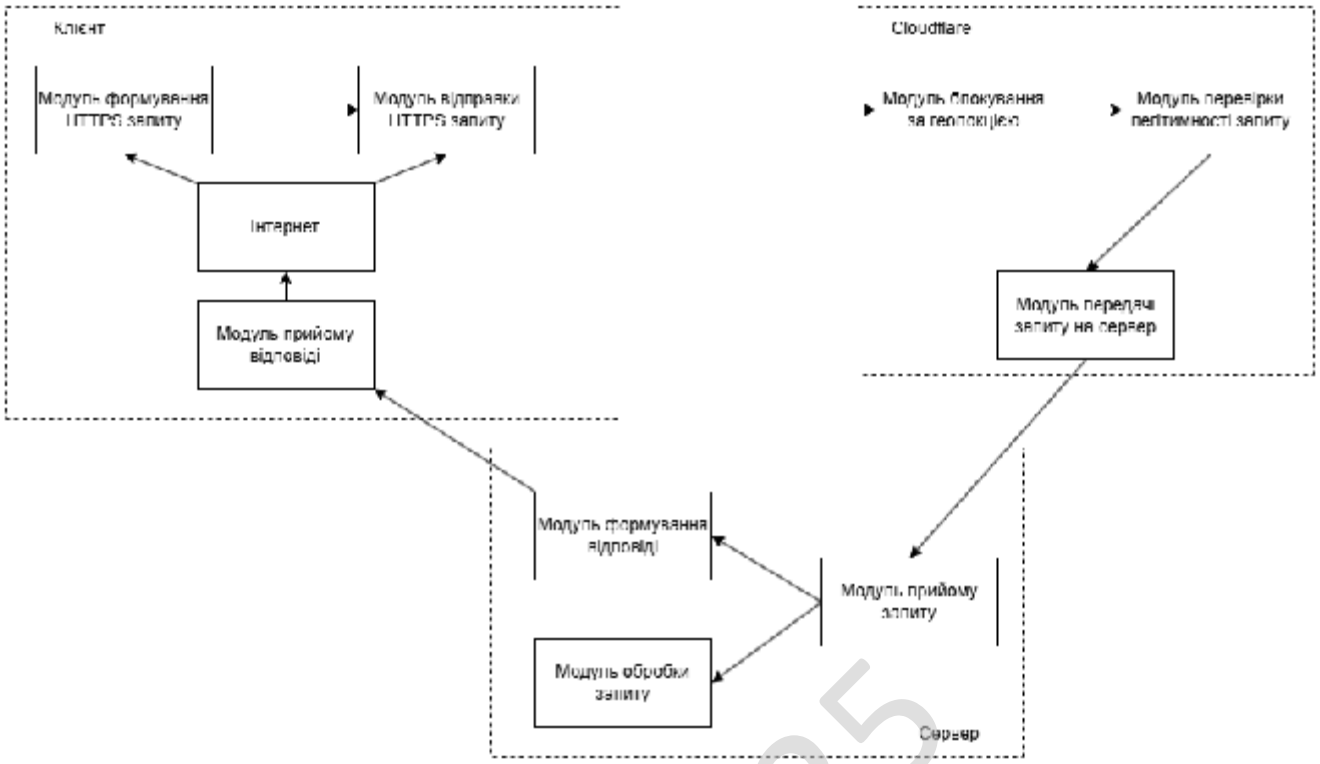

Рисунок 3.3 – Функціональна схема взаємодії користувачів з системою

На рисунку 3.4 зображено функціональну схему взаємодії клієнтів з системою, яка складається з наступних двох основних блоків – “Адміністратор” та “Сервер”.

Блок “Адміністратор” складається з наступних модулів:

– **Модуль підключення VPN.** Відповідає за встановлення захищеного каналу зв’язку між адміністратором та сервером через Інтернет. Завдяки VPN забезпечується шифрування та ізоляція переданих даних, що гарантує високий рівень безпеки при обміні інформацією.

– **Модуль відправки команд для налаштування.** Відповідає за формування та передачу команд конфігурації від адміністратора до серверу. Він інтегрує всі необхідні інструкції, які мають бути виконані на стороні серверу, та відправляє їх через встановлений захищений канал зв’язку.

– **Модуль отримання результату налаштування.** Після виконання команд сервером, приймає результати налаштування. Він перевіряє і збирає інформацію про статус виконання завдань, що дозволяє адміністратору отримувати зворотний зв’язок з детальним логуванням виконаних операцій.

Блок “Сервер” складається з наступних пунктів:

– **Модуль з’єднання по SSH.** На стороні серверу цей блок забезпечує встановлення безпечного з’єднання через SSH протокол. Він гарантує, що команди, отримані від адміністратора, передаються з належним рівнем шифрування та аутентифікації, що захищає систему від несанкціонованого доступу.

– **Модуль формування завдань налаштування.** Цей блок обробляє отримані від адміністратора інструкції та перетворює їх у конкретні завдання для виконання. Він аналізує команди, визначає необхідні дії та готує їх до подальшого виконання на сервері.

– **Модуль виконання налаштування.** Призначення цього блоку – безпосереднє виконання сформованих завдань. Сервер виконує необхідні налаштування, змінює конфігурації чи встановлює потрібні параметри згідно з отриманими інструкціями, забезпечуючи коректну роботу системи.

– **Модуль відправки результату налаштування.** Після завершення виконання завдань цей блок відповідає за формування звіту про виконання і передачу результатів назад до адміністратора. Це завершує цикл взаємодії, забезпечуючи повний зворотний зв’язок із зазначенням успішних операцій або помилок, які потребують уваги.

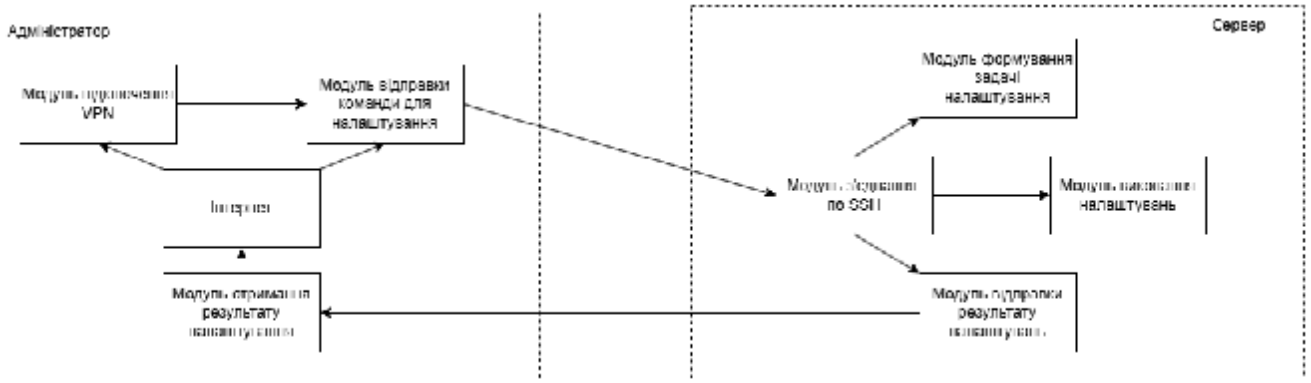

Рисунок 3.4 – Функціональна схема взаємодії адміністратора з системою

3.4 Розробка діаграми процесів

На рисунку 3.5 зображено діаграму процесів програмного забезпечення, яка демонструє логіку системи а також як в ній реалізовано потік даних.

Спочатку клієнт формує HTTPS-запит, додаючи необхідні параметри та заголовки, після чого передає його через інтернет у напрямку серверної інфраструктури. Запит спочатку проходить через Cloudflare, який виступає як перший рівень захисту та продуктивності, здійснюючи блокування за геолокацією та перевірку валідності запиту. Це дозволяє забезпечити захист від несанкціонованих спроб доступу та фільтрувати запити відповідно до налаштованих правил безпеки, перш ніж вони потраплять до основних серверів.

Після проходження первинної перевірки запит передається до веб-серверів, які відповідають за його обробку. На цьому етапі сервер аналізує отриману інформацію, виконує необхідні обчислення або запити до бази даних, а також застосовує внутрішні алгоритми для формування необхідного контенту. Якщо запит вимагає доступу до даних, сервер звертається до бази даних, де здійснюється пошук, запис або модифікація інформації.

Після цього веб-сервер формує відповідь у вигляді HTML-коду, JSON-даних або іншого формату, який передається назад у систему маршрутизації.

Сформована відповідь повертається через Cloudflare, де вона може

проходити додаткову оптимізацію, кешування або шифрування перед фінальним відправленням клієнту. На цьому етапі також можуть застосовуватися алгоритми безпеки, які перевіряють, чи відповідь коректно сформована і не містить небезпечних елементів.

Після цього відповідь надходить до клієнта, де модуль прийому відповіді обробляє отримані дані та відображає їх у веб-інтерфейсі користувача. Завдяки такій структурі потоку даних забезпечується ефективне управління трафіком, висока продуктивність системи, а також захист від загроз, що виникають при роботі з відкритими мережами.

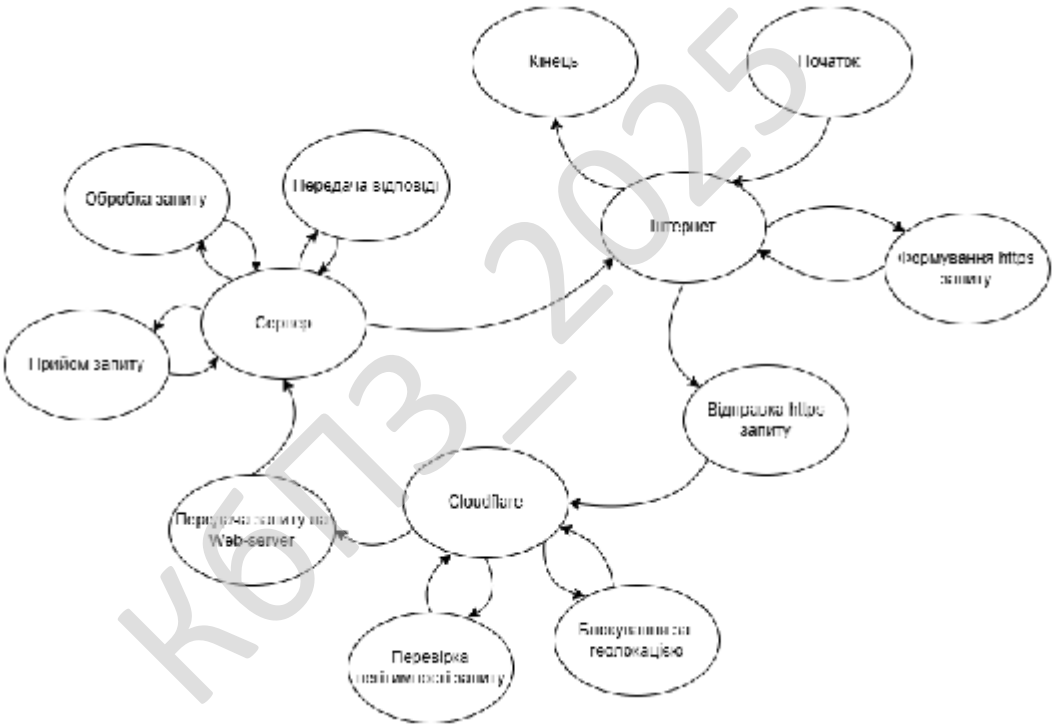

Рисунок 3.5 – Діаграма процесів системи з позиції клієнта

На рисунку 3.6 зображено діаграму процесів програмного забезпечення, яка демонструє логіку системи а також як в ній реалізовано потік даних.

Потік даних починається з ініціації процесу адміністратором, який встановлює захищене з’єднання через VPN. Це з’єднання забезпечує шифрування інформації та ізоляцію внутрішнього трафіку від зовнішніх загроз, що дозволяє

гарантувати цілісність і конфіденційність переданих запитів. VPN створює безпечний тунель, завдяки якому дані не можуть бути легко перехоплені або змінені під час їх передачі через Інтернет.

Після встановлення VPN-з’єднання запити та команди для налаштування системи передаються через безпечний канал SSH до серверної частини. Сервер приймає ці дані і обробляє їх, перетворюючи отримані інструкції у вигляд конкретних завдань, придатних для виконання.

Після обробки даних сервер виконує налаштування, формуючи результат виконання, який включає інформацію про успішність операцій, статус виконання або повідомлення про можливі помилки. Отриманий результат повертається назад до адміністратора через той же захищений канал, що забезпечує повний цикл зворотного зв’язку.

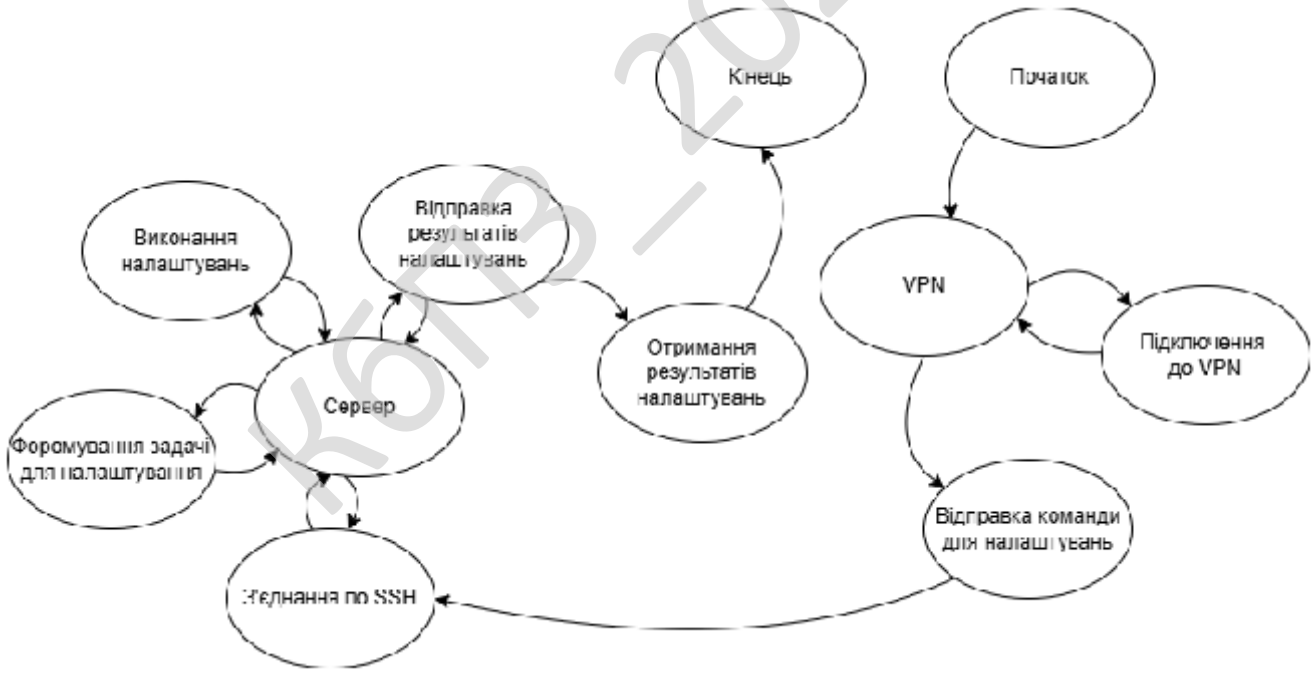

Рисунок 3.6 – Діаграма процесів системи з позиції адміністратора

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему взаємодії клієнтів з програмним забезпеченням. Вона складається з наступних пунктів:

– Введення запиту. Користувач вводить запит з необхідними параметрами, які будуть передані для аналізу системою. Введена інформація фіксується та направляється на перевірку.

– Перевірка за геолокацією. Система перевіряє, чи входить геолокація користувача в список заборонених геолокацій. Якщо геолокація входить до заборонених – користувач блокується.

– Перевірка валідності запиту. Перевіряється структура запиту, відповідність формату та правильність наданих параметрів. Якщо запит не проходить валідацію, процес завершується або запит повертається користувачу для виправлення.

– Отримання відповіді. Система формує відповідь на валідний запит, збираючи необхідні дані з відповідних джерел. Отримана інформація передається користувачеві у визначеному форматі.

Рисунок 4.1 – Блок-схема взаємодії клієнтів з програмним забезпеченням

На рисунку 4.2 наведено блок-схему взаємодії адміністратора з програмним забезпеченням. Вона складається з наступних пунктів:

– **Підключення до VPN.** Адміністратор встановлює захищене з’єднання через VPN, що забезпечує безпечний доступ до серверної інфраструктури. Це гарантує шифрування трафіку та захист даних від стороннього втручання.

– **Перевірка прав користувача.** Система перевіряє, чи має користувач необхідні привілеї для виконання адміністративних команд. Якщо доступ обмежений, запит може бути відхилений або потребувати додаткової авторизації.

– **Введення команди для налаштування сервера.** Адміністратор вводить необхідні команди для зміни конфігурації або запуску сервісів на сервері. Ці команди передаються через SSH-з’єднання для подальшого виконання.

– **Виконання зазначених налаштувань.** Сервер обробляє отримані команди та змінює конфігурацію згідно з заданими параметрами. Виконання здійснюється відповідно до визначених правил безпеки та системних політик.

– **Отримання результату виконання налагштувань.** Після завершення процесу сервер надсилає адміністратору відповідь із деталями виконаних операцій. Це дозволяє перевірити успішність змін або виявити можливі помилки в налаштуванні.

Рисунок 4.2 – Блок-схема взаємодії адміністратора з програмним забезпеченням

На рисунку 4.3 наведено блок-схему процесу виконання налаштувань. Вона складається з наступних пунктів:

– **Отримання запиту.** Система приймає запит від адміністратора, який містить необхідні параметри для виконання операцій.

– **Перевірка прав користувача.** Перевіряється рівень прав доступу користувача для виконання адміністративних команд. Якщо користувач не має необхідних привілеїв, запит може бути відхилений або потребувати додаткової авторизації.

– **Визначення заданих параметрів з запиту.** Система аналізує отримані дані, виділяючи критичні параметри для виконання налаштувань.

– **Формування задачі налаштувань.** На основі заданих параметрів створюється завдання, яке містить послідовність команд для виконання.

– **Відправка сформованої задачі на сервер.** Сформоване завдання передається серверу через захищений канал зв’язку.

– **Виконання сформованої задачі налаштування.** Сервер виконує отримане завдання, змінюючи конфігурацію відповідно до заданих параметрів. Під час виконання перевіряється правильність усіх операцій та можливі помилки.

– **Відправка відповіді з результатами налаштування.** Після завершення операцій сервер формує звіт із результатами виконання. Відповідь надсилається адміністратору, щоб підтвердити успішність або повідомити про необхідність внесення змін.

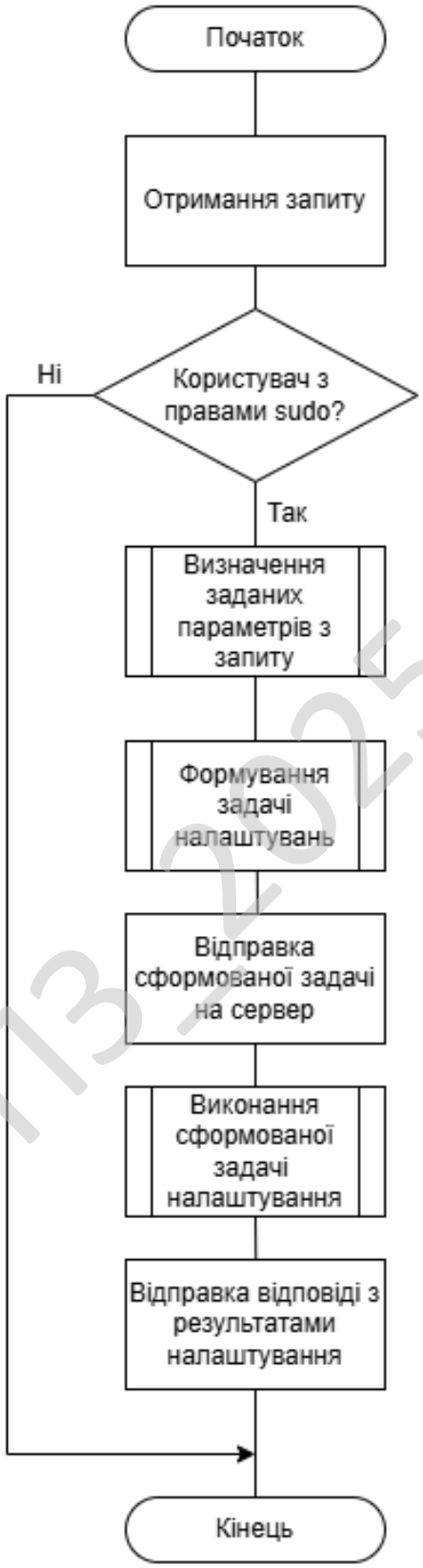

Рисунок 4.3 – Блок-схема процесу виконання налаштувань

4.2 Захист розробленого програмного забезпечення

Ubuntu 24.04 LTS з точки зору безпеки

В якості основної ОС була вибрана Ubuntu 24.04 LTS. Ubuntu 24.04 LTS надає високий рівень безпеки завдяки впровадженим вбудованим засобам захисту, зокрема механізму AppArmor, який обмежує дії додатків відповідно до заданих політик. Цей підхід забезпечує помітне зменшення ризику несанкціонованого доступу до критичних системних ресурсів, адже навіть у разі компрометації деякого процесу, його можливості впливати на інші частини системи залишаються обмеженими. Додатково, система використовує сучасні криптографічні стандарти для перевірки підписів пакетів, що гарантує встановлення лише автентичного та незмінного програмного забезпечення. Також, в Ubuntu інтегровані засоби для моніторингу подій безпеки, які у поєднанні з ефективними журналами подій дозволяють швидко ідентифікувати потенційні загрози.

Що стосується оновлень безпеки, Ubuntu 24.04 LTS відома своєю стабільною та тривалою підтримкою. Як правило, даний LTS-реліз отримує регулярні патчі та оновлення безпеки протягом п’яти років із моменту випуску, що дозволяє виробничим середовищам залишатися захищеними проти нових і виявлених вразливостей. Крім стандартного циклу оновлень, доступні послуги Extended Security Maintenance (ESM) дозволяють продовжити отримання критичних патчів навіть після завершення базового терміну підтримки. Це забезпечує безперервний захист системи, особливо в умовах, коли перенесення критичних сервісів на новішу версію може бути ускладненим або потребувати значної реорганізації.

З точки зору безпеки, Ubuntu 24.04 LTS також забезпечує інтегровані фаєрволи, такі як UFW (Uncomplicated Firewall), які дозволяють легко налаштовувати правила доступу до системи. Усі пакунки, що поширюються через офіційні репозиторії, проходять ретельну перевірку на наявність вразливостей,

що мінімізує ризик встановлення шкідливого або модифікованого програмного забезпечення. Система також підтримує концепцію «безпечного завантаження» (Secure Boot), яка додатково перевіряє цілісність завантажувального процесу і унеможливлює запуск неавторизованих операційних систем або модулів драйверів. Таким чином, Ubuntu 24.04 LTS забезпечує комплексний підхід до безпеки, який охоплює як захист на рівні ядра системи, так і безпечну доставку та оновлення програмного забезпечення.

Pritunl з 2FA для організації VPN з точки зору безпеки

Pritunl використовується як альтернативне рішення традиційному OpenVPN завдяки зосередженості на спрощенні процесу налаштування захищених з'єднань, що мінімізує людський фактор і знижує ризик виникнення конфігураційних помилок. З точки зору безпеки, простота налаштування значить меншу ймовірність помилок, які могли б стати вектором для атак, оскільки система дозволяє швидко впровадити стандартизовані та перевірені параметри з'єднання. Крім того, інтерфейс Pritunl дозволяє централізовано управляти користувачами і регулювати політики доступу, що дає змогу оперативно реагувати на інциденти та забезпечує послідовне застосування заходів безпеки. Це забезпечує надійне та безперебійне функціонування VPN, що є критично важливим для захищеного віддаленого доступу до внутрішньої інфраструктури.

Інтеграція двофакторної автентифікації (2FA) у Pritunl є одним із ключових механізмів посилення безпеки. Навіть якщо основний пароль компрометовано, додатковий фактор авторизації (наприклад, мобільний додаток для генерації OTP-кодів або апаратний токен) унеможливлює несанкціонований доступ, оскільки зловмиснику необхідно володіти цим другим елементом для успішного аутентифікаційного процесу. Такий підхід значно ускладнює атаки типу brute-force або соціальної інженерії, оскільки зловмиснику доведеться отримати доступ до двох незалежних засобів автентифікації. Завдяки цьому, навіть компрометація одного з чинників не призводить до повного доступу, що

робить систему більш стійкою до атак із використанням викрадених облікових даних.

Безпека віддалених з'єднань підкріплюється додатковими шарами захисту, що впроваджуються в Pritunl. Сучасні алгоритми шифрування, зокрема використання TLS/SSL для встановлення тунелів, забезпечують цілісність і конфіденційність передаваних даних. Крім того, централізоване управління і миттєва можливість моніторингу активності користувачів дозволяють оперативно виявляти підозрілі дії, порушення політик доступу та інші аномалії. Таким чином, використання Pritunl разом із двофакторною автентифікацією створює багаторівневий захисний механізм, який відповідає сучасним вимогам до безпечної організації віддаленого доступу, знижує ризик несанкціонованого проникнення та забезпечує стабільне та надійне керування мережевими з'єднаннями.

Налаштування SSH з точки зору безпеки

Налаштування SSH-з'єднань є критично важливим елементом забезпечення безпеки серверної інфраструктури. Виключення автентифікації за паролем на користь використання лише криптографічних ключів значно знижує ризик компрометації через атаки методом brute-force та словникові атаки, оскільки ключі, які зазвичай мають велику ентропію та є захищеними криптографічними алгоритмами, важко піддібрати. Використання ключової пари (відкритий і закритий ключ) забезпечує автентичність зв'язку, адже закритий ключ зберігається виключно у користувача, а відкритий ключ розподіляється між серверами. Такий підхід дозволяє мінімізувати вплив людського фактора та зменшити можливість втручання зловмисників у процес аутентифікації.

Заборона віддаленого з'єднання для користувача root – ще один важливий аспект безпеки, який запобігає прямому доступу до критичних системних ресурсів. Замість прямого входу як root, користувачі повинні використовувати звичайні акаунти з обмеженими правами, а потім підвищувати свої привілеї через sudo або інший механізм ескалації прав, що централізовано контролюється та

веде журнал дій. Це дозволяє ізолювати доступ до системних компонентів та знижує ризик масштабного компрометації сервера, оскільки навіть у разі викрадення прав неавтентифікованого користувача, зловмиснику буде значно складніше отримати повний доступ до системи. Така практична політика також спрощує аудит використання привілеїв, завдяки чому адміністраторам легше відслідковувати будь-які спроби несанкціонованого доступу.

Відмова від автентифікації за паролем гарантує, що навіть зловмисники, отримавши доступ до потенційно слабких паролів, не зможуть увійти до системи. Адже вся аутентифікація базується на криптографічних ключах, які набагато важче підробити або підібрати, особливо якщо вони захищені додатковою фразою-паролем. Цей підхід усуває можливість використання широкої палітри атак, орієнтованих на паролі, і дозволяє системі зосередитися на надійному управлінні ключовими парами. Крім того, вимкнення логіну по паролю створює чіткий механізм автентифікації, що застосовується на всіх рівнях безпеки SSH-з'єднання.

Додатковим рівнем захисту впроваджується механізм Challenge-Response авторизації, який служить для двофакторної перевірки під час встановлення SSH-з'єднання. Цей механізм вимагає від користувача не лише презентації правильного приватного ключа, а й відповіді на динамічно згенерований запит, який може бути реалізований через одноразовий пароль (OTP) або інший засіб додаткової верифікації. Challenge-Response забезпечує багаторівневий захист, адже навіть при можливій компрометації приватного ключа зловмиснику буде важко пройти додаткову перевірку. Таке комплексне налаштування забезпечує максимально можливу стійкість до несанкціонованого доступу, що є критичним для управління чутливими та життєво важливими серверами.

```
- hosts: all
  become: yes
  tasks:
    - name: Оновлення списку пакетів
      apt:
        update_cache: yes

    - name: Оновлення системи
      apt:
```

```
    upgrade: dist

- name: Встановлення базових утиліт
  apt:
    name:
      - vim
      - curl
      - htop
    state: present

- name: Заборонити віддалений логін для root
  lineinfile:
    path: /etc/ssh/sshd_config
    regexp: '^PermitRootLogin'
    line: 'PermitRootLogin no'

- name: Заборонити авторизацію за паролем
  lineinfile:
    path: /etc/ssh/sshd_config
    regexp: '^PasswordAuthentication'
    line: 'PasswordAuthentication no'

- name: Дозволити тільки авторизацію за ключами
  lineinfile:
    path: /etc/ssh/sshd_config
    regexp: '^PubkeyAuthentication'
    line: 'PubkeyAuthentication yes'

- name: Заборонити Challenge-Response авторизацію
  lineinfile:
    path: /etc/ssh/sshd_config
    regexp: '^ChallengeResponseAuthentication'
    line: 'ChallengeResponseAuthentication no'

- name: Заборонити авторизацію за паролем у 50-cloud-init.conf
  lineinfile:
    path: /etc/ssh/sshd_config.d/50-cloud-init.conf
    regexp: '^PasswordAuthentication'
    line: 'PasswordAuthentication no'
    create: yes

- name: Перезапустити SSH для застосування змін
  systemd:
    name: ssh
    state: restarted
```


Використання Cloudflare для додаткового захисту

Cloudflare виступає у ролі першої лінії оборони, розташовуючись між користувачами та серверною інфраструктурою, що дозволяє приховати реальні IP-адреси серверів і мінімізувати прямий вплив зловмисників на внутрішні системи. Він аналізує вхідний трафік на мережевому рівні, автоматично визначає підозрілу активність і блокує шкідливі запити ще до того, як вони досягнуть основного серверу. Цей механізм є особливо ефективним у захисті від DDoS-атак, які можуть перевантажити сервер, оскільки Cloudflare розподіляє трафік

через свою мережу розподілених центрів обробки даних. Завдяки такій архітектурі сервер отримує лише попередньо відфільтрований та перевірений трафік, що значно покращує загальний рівень безпеки.

Крім захисту на мережевому рівні, Cloudflare забезпечує інтегрований веб-фаєрвол (WAF), який аналізує HTTP/HTTPS-запити, застосовуючи сигнатури відомих атак та інтелектуальні алгоритми поведінкового аналізу. WAF дозволяє адмініструвати точкові правила фільтрації, що дозволяють блокувати специфічні загрози, наприклад, SQL-ін'єкції, XSS-атаки та інші вектори атак на рівні веб-додатків. Правила фільтрації можуть бути адаптовані до специфіки конкретного проекту, що забезпечує гнучкість і високу точність у визначенні та нейтралізації загроз. Такий багаторівневий захист дозволяє оперативно реагувати на нові загрози та мінімізувати ризики експлуатації вразливостей.

Застосування механізмів кешування та географічного розподілу контенту, які пропонує Cloudflare, не лише прискорює доставку даних кінцевим користувачам, а й знижує навантаження на вихідні сервери. Завдяки кешування популярного контенту, сервер отримує менше запитів безпосередньо, що дозволяє уникнути перевантажень під час пікових навантажень або DDoS-атак. Розподіл трафіку через численні сервери CDN допомагає ізолювати вплив шкідливих запитів, забезпечуючи безперервну роботу додатку навіть у кризових ситуаціях. Такий підхід значно підвищує продуктивність сервісу, створюючи додатковий шар безпеки та стабільності.

Додатково, Cloudflare дозволяє налаштовувати детальні політики доступу, зокрема за допомогою блокування небажаних IP-адрес, обмеження кількості запитів (rate limiting) та застосування Challenge Pages для підозрілої активності. Інтегровані засоби аналітики надають змогу адміністраторам детально відслідковувати характер трафіку, ідентифікувати потенційні загрози та оперативно реагувати на кризи безпеки. Крім того, підтримка сучасних криптографічних протоколів, включаючи SSL/TLS шифрування, гарантує, що весь трафік між клієнтами та Cloudflare є захищеним від перехоплення та

модифікації. Завдяки совокупності цих механізмів, Cloudflare значно підвищує рівень захисту, забезпечуючи безпечну і стабільну роботу всієї інфраструктури, навіть в умовах високої загрози.

Також на сервері організовний доступ для https запитів тільки з мережі Cloudflare та з локальної мережі

```
    # Дозволити доступ лише з Cloudflare IP-діапазонів та з мережі VPN
allow 173.245.48.0/20;
allow 103.21.244.0/22;
allow 103.22.200.0/22;
allow 103.31.4.0/22;
allow 141.101.64.0/18;
allow 108.162.192.0/18;
allow 190.93.240.0/20;
allow 188.114.96.0/20;
allow 197.234.240.0/22;
allow 198.41.128.0/17;
allow 162.158.0.0/15;
allow 104.16.0.0/13;
allow 104.24.0.0/14;
allow 172.16.100.0/24;
deny all;
```

Використання Zabbix для моніторингу і посилення безпеки

Zabbix — це потужний інструмент моніторингу, який дозволяє здійснювати безперервний контроль за всіма критичними компонентами інфраструктури, включаючи сервери, мережеві пристрої, бази даних та додатки. Завдяки можливості збору метрик у режимі реального часу, Zabbix виявляє аномалії у поведінці системи, що можуть свідчити про потенційні загрози або несправності. Система працює на основі налаштованих порогових значень (trigger'ів), що миттєво генерують сповіщення при перевищенні встановлених лімітів, що дозволяє оперативно реагувати на інциденти. Інтеграція Zabbix з системою сповіщень (через email, SMS або месенджери) забезпечує швидке інформування адміністраторів про критичні події, зменшуючи час реагування та мінімізуючи вплив потенційних атак або несправностей.

Крім звичайного моніторингу продуктивності, Zabbix також виконує функції аналізу безпеки, відслідковуючи дивовижну активність та інциденти, які можуть свідчити про спроби злому або компрометації системи. За допомогою адаптованих правил і шаблонів моніторингу, можна налаштувати виявлення підозрілих спроб аутентифікації, несподівані зміни в конфігурації або

аномальний мережевий трафік. Це дає змогу проводити комплексний аудит безпеки, включаючи збір логів, кореляцію подій та аналіз історичних даних, що сприяє проактивному виявленню потенційних загроз до того, як вони перетворяться у критичні інциденти. Такий підхід дозволяє адміністраторам не лише реагувати на інциденти, але й проводити ретельне розслідування зі збору детальної інформації про кожен випадок, що є надзвичайно корисним для вдосконалення заходів безпеки.

Окрім цього, Zabbix сприяє підвищенню загального рівня безпеки за рахунок інтеграції з іншими системами управління та контролю, що дозволяє автоматизувати процеси сканування та аналізу вразливостей. Система може бути налаштована для автоматичного корегування конфігурацій, виконання скриптів для ізоляції вразливих вузлів або блокування підозрілих IP-адрес. Така автоматизація дозволяє не лише своєчасно виявляти загрози, але й оперативно реагувати на них, мінімізуючи ризики поширення атак. Отже, використання Zabbix є надзвичайно важливим елементом забезпечення інформаційної безпеки, який дозволяє не лише контролювати стабільність роботи системи, але й активно посилювати захист від зовнішніх і внутрішніх загроз.

Єдина система встановлення користувачів через Ansible

Єдина система встановлення користувачів, реалізована за допомогою Ansible, є потужним інструментом автоматизації, що централізує процес створення, управління та налаштування користувацьких акаунтів на всіх серверах інфраструктури. За допомогою Ansible playbooks визначаються необхідні параметри для кожного користувача — права доступу, домашні каталоги, набір груп, налаштування оболонки та безпечні параметри аутентифікації. Ідемпотентний підхід Ansible гарантує, що при повторному застосуванні конфігураційні зміни додаються лише в разі потреби, а система залишається в узгодженому стані без дублювання або конфліктів. Така автоматизація скорочує людський фактор, знижуючи ризик помилок, пов’язаних із ручним налаштуванням, та забезпечує послідовність політик доступу по всій організації.

Крім того, централізоване управління користувачами забезпечує високу прозорість і повноту аудиту всіх змін. Завдяки зберіганню YAML-файлів із описом користувачів у системах контролю версій, а також використанню журналів виконання завдань Ansible, адміністратори можуть відслідковувати, хто і коли вносив зміни в систему. Це дозволяє оперативно виявляти несанкціоновані зміни чи відхилення від затверджених налаштувань безпеки. Інтегроване централізоване управління сприяє високій відповідності стандартам безпеки, що є критично важливим для організацій, де розподілена інфраструктура та численні сервери потребують стабільного і контрольованого доступу.

З точки зору безпеки, використання Ansible для встановлення користувачів дозволяє легко впроваджувати завчасно визначені політики безпеки на всіх вузлах мережі. Це означає, що кожен акаунт буде створено з відповідними рівнями доступу, налаштованим обмеженням прав і навіть інтегрованими механізмами двофакторної автентифікації, якщо це передбачено політикою організації. Атоматизація процесу встановлення користувачів не тільки забезпечує одноманітність налаштувань, але і виступає потужним засобом захисту від потенційних загроз, оскільки централізоване адміністрування унеможливлює локальні відхилення від затверджених стандартів конфігурації. Таким чином, єдина система встановлення користувачів через Ansible є невід'ємною частиною сучасного підходу до безпечного, ефективного та масштабованого управління доступом до критичних ресурсів організації.

Таким чином, комплексна система захисту побудована за принципом многослойності, що дозволяє ефективно мінімізувати ризики несанкціонованого доступу і забезпечити стабільну роботу всієї інфраструктури. Використання Ubuntu 24.04 LTS гарантує надійну операційну платформу з регулярними оновленнями безпеки та впровадженими механізмами, наприклад, AppArmor, що обмежують дії додатків і забезпечують цілісність системи навіть у разі виникнення загроз. Захищений VPN через Pritunl з інтегрованою двофакторною автентифікацією розширює можливості безпечного віддаленого доступу,

знижуючи шанси компрометації облікових даних завдяки використанню потужних криптографічних стандартів і додаткових рівнів перевірки особистості.

Крім того, налаштовані SSH-з'єднання, що використовують виключно криптографічні ключі та впровадження Challenge-Response авторизації, забезпечують максимально захищене з'єднання, позбавлене вразливостей, притаманних використанню паролів, що суттєво мінімізує можливості для атак типу brute-force. Додаткові заходи, такі як заборона віддаленого доступу для користувача root і обмеження логіну, значно ускладнюють зловмисникам проникнення в систему, забезпечуючи високу контрольованість процесів аутентифікації.

Cloudflare виступає незамінним ланком у захисті від зовнішніх загроз, таких як DDoS-атаки та зловмисний трафік, завдяки механізмам фільтрації, кешування та веб-фаєрволу, що дозволяють зменшити навантаження на внутрішні сервери та приховати їхні реальні IP-адреси. Інтегроване моніторингове рішення Zabbix постійно відслідковує всі критичні показники інфраструктури, оперативно виявляє аномалії та негайно сповіщає адміністраторів про потенційні загрози, що дозволяє своєчасно реагувати на інциденти безпеки. Поєднання цих рішень формує багаторівневий оборонний бар'єр, який відповідає сучасним вимогам захищеності інформаційних систем і підвищує стійкість системи як до зовнішніх, так і до внутрішніх загроз.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програмне забезпечення автоматизації налаштування середовища серверів налічує наступні основні параметри:

– “--ask-vault-password” / “--ask-vault-pass” – запитує пароль для розшифрування даних, захищених за допомогою ansible-vault. Використовується для роботи із зашифрованими змінними та файлами;

– “-i” – вказує інвентарний файл (inventory.yml), що містить список хостів, до яких буде виконано підключення;

– “-u” – задає користувача, від імені якого буде здійснене підключення до серверів по SSH;

– “--private-key” – використовує конкретний приватний SSH-ключ для автентифікації без пароля;

– “-vvv” – увімкнення розширеного режиму виводу (verbose mode), що дозволяє детально аналізувати виконання команд і усувати можливі проблеми;

– “--tags” – використовується для запуску лише тих завдань, які містять задані теги. Це дозволяє виконувати окремі частини playbook'у без запуску всього сценарію;

– “--tags” (Теги Ansible) – дозволяють гнучко керувати виконанням завдань, вибірково запускаючи потрібні частини сценарію. Наприклад, у playbook’у можна позначити блоки тегами security, install, update, і потім запустити лише певний набір завдань за допомогою --tags security. Це економить час та спрощує адміністрування великих конфігурацій;

– “-K” / “--ask-become-pass” – вказує Ansible на необхідність запитати пароль для become (підвищення привілеїв), тобто виконання команд з правами sudo або іншого користувача із розширеними привілеями;

– “--skip-tags” (SKIP_TAGS) – використовується для пропуску завдань у Ansible playbook, які мають зазначені теги;

– “--syntax-check” – використовується для перевірки синтаксису Ansible playbook без його фактичного виконання.

Програмне забезпечення автоматизації налаштування середовища серверів налічує наступні основні команди:

– ansible-playbook create-user.yml -i inventory.yml -u diplom --ask-become --ask-pass --ask-vault-pass – виконується перший раз для розподілення користувачів по серверах;

– ansible-playbook secure-ssh.yml -i inventory.yml -u diplom1 --private-key ~/.ssh/diplom1 -K – використовується для налаштування безпечних з’єднань по SSH. Також в даному playbook проходять наступні налаштування: заборона віддаленого логіну для root, авторизація за паролем, дозвіл на авторизацію тільки по ключах, заборона Challenge-Response авторизації;

– ansible-playbook create-zabbix-server.yml -i inventory.yml -u diplom1 --private-key ~/.ssh/diplom1 -K --ask-vault-pass – створення Zabbix Server, налаштування бази даних для нього;

– ansible-playbook create-zabbix-agent.yml -i inventory.yml -u diplom1 --private-key ~/.ssh/diplom1 -K --ask-vault-pass – інсталює zabbix-agent2 на всі сервери. Перевіряє наявність вже встановлених агентів, видаляє їх, якщо вони є, видаляє старі конфігурації для zabbix-agent, встановлює zabbix-agent2 та налаштовує коректні конфігураційні файли для підключення до серверу моніторингу Zabbix;

– ansible-playbook create-reverse.yml -i inventory.yml -u diplom1 --private-key ~/.ssh/diplom1 -K --ask-vault-pass – інсталює nginx та створює необхідні конфігураційні файли для балансування та коректного проксювання на сервери Web-service-1 та Web-service-2

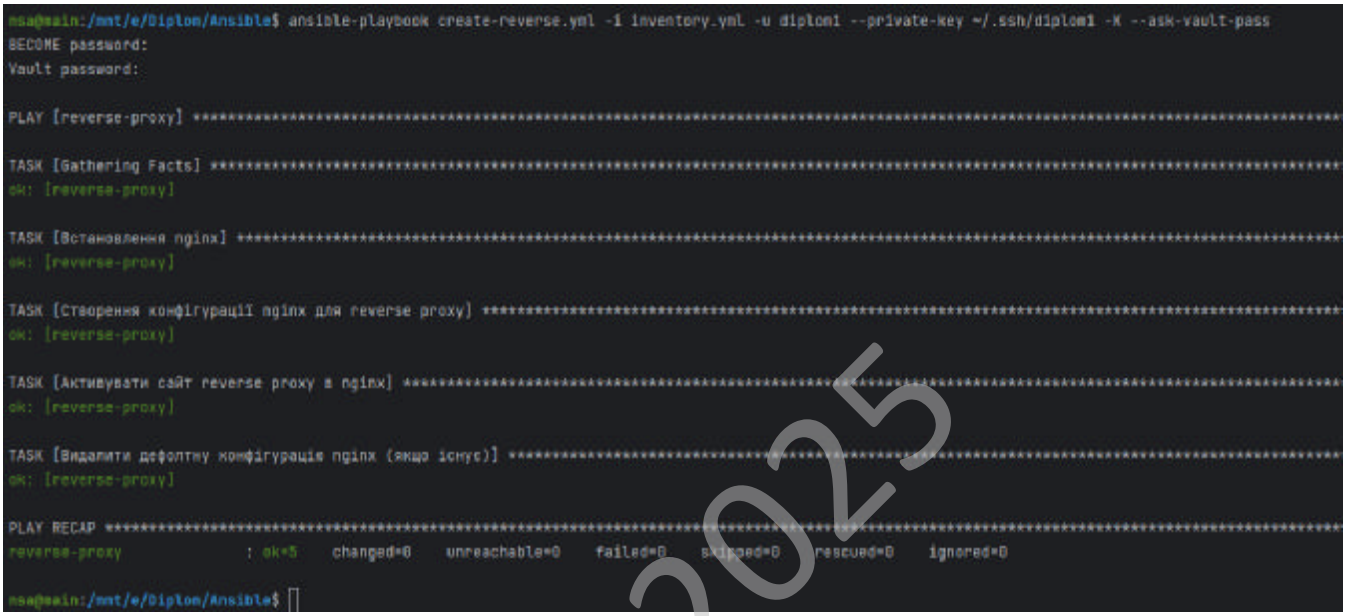

Рисунок 5.1 – Результат виконання команди ansible-playbook create-reverse.yml -i inventory.yml -u diplom1 --private-key ~/.ssh/diplom1 -K --ask-vault-pass

З рисунку 5.1 можна побачити, що було виконано завдання по встановленню конфігурування nginx, а саме:

– Встановлення nginx (якщо вже встановлений, то зміни не застосовуються).

– Створення конфігураційного файлу d1ploma.site.conf з шаблону .jinja2 templates /nginx_reverse.conf.j2.

– Видалення дефолтної конфігурації сайту nginx.

– Перезапуск сервісу nginx.

– ansible-playbook create-mysql.yml -i inventory.yml -u diplom1 --private-key ~/.ssh/diplom1 -K --ask-vault-pass – створює головну базу даних та конфігурує її;

– ansible-playbook create-lamp.yml -i inventory.yml -u diplom1 --private-key ~/.ssh/diplom1 -K --ask-vault-pass - встановлює apache2 для управління веб-додатками на серверах Web-service-1 та Web-service-2.

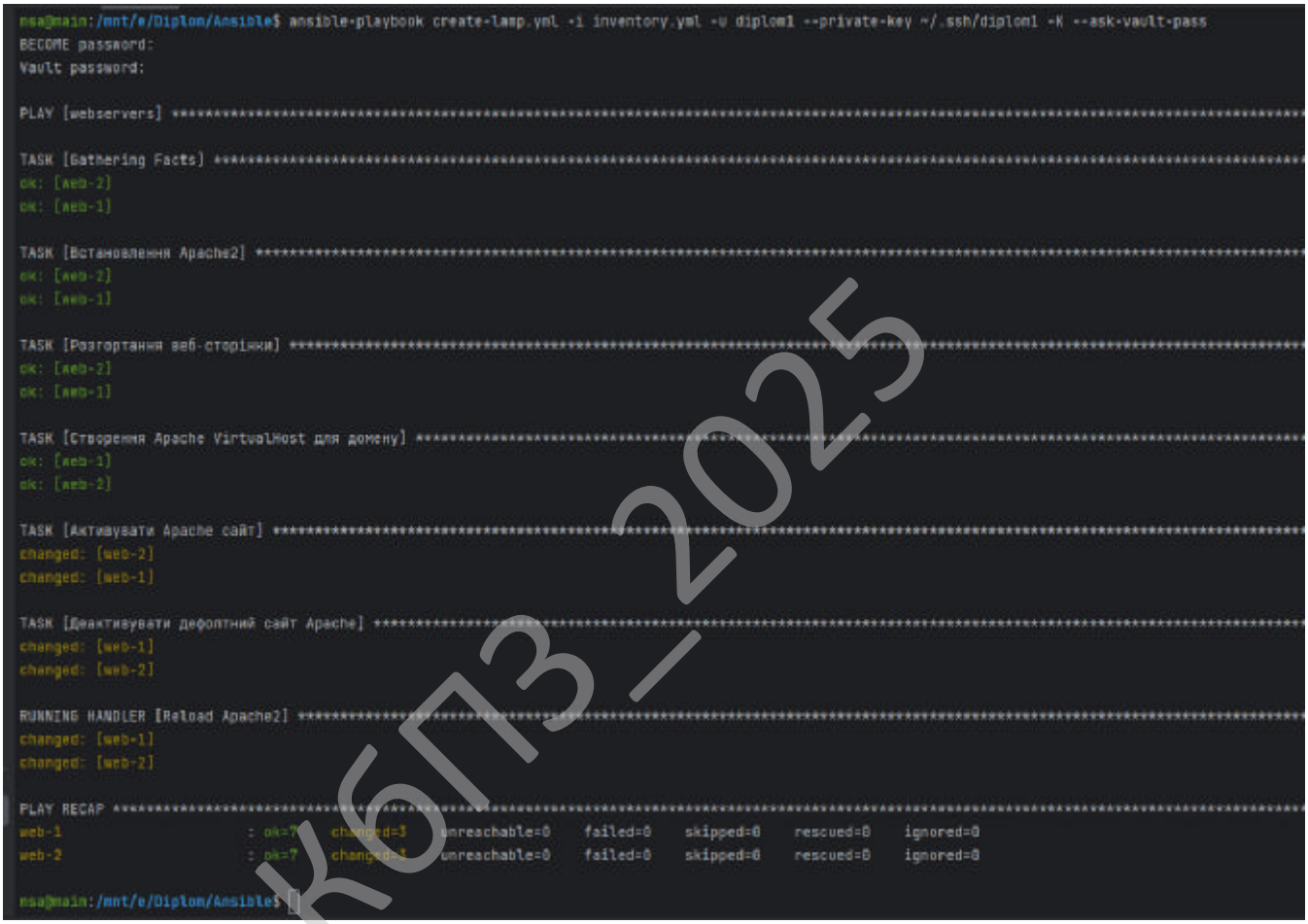

Рисунок 5.2 – Результат виконання команди ansible-playbook create-lamp.yml -i inventory.yml -u diplom1 --private-key ~/.ssh/diplom1 -K --ask-vault-pass

З рисунку 5.1 можна побачити, що виконалось конфігурування двох серверів, на які було встановлено веб сервер apache2, створені шаблони веб додатків та веб сервер було перезапущено.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної роботи, призначено для автоматизації налаштування середовища серверів.

Для вирішення поставленої мети було:

- Проведено Критичний аналіз існуючих рішень.
- Здійснено детальний огляд сучасних систем автоматизації та управління конфігураціями, рішень для віртуалізації
- Виявлено сильні та слабкі сторони кожного підходу з акцентом на аспекти безпеки, універсальності, продуктивності та масштабованості.
- Використані результати аналізу для формування технічних вимог до розроблюваної системи.
- Обґрунтовано вибір архітектури системи.
- На основі проведеного аналізу сформовано концептуальну модель інтегрованої серверної інфраструктури, що включає VPN-сервер з підтримкою двофакторної автентифікації, два веб-сервіси, налаштовані через NGINX reverse proxy для балансування навантаження, сервер бази даних на базі MariaDB та систему моніторингу з використанням Zabbix.
- Побудовано структурну та функціональну схеми системи, які ілюструють взаємодію всіх компонентів та логіку обробки запитів у системі.

В випускній кваліфікаційній роботі було розроблено комплексну систему захисту, що базується на багаторівневому підході та інтеграції сучасних технологій. Кожен рівень охоплює конкретну область захисту – від операційної системи та мережевої безпеки до централізованого моніторингу та автоматизації. Такий підхід дозволив мінімізувати ризики несанкціонованого доступу, забезпечуючи одночасно стабільну роботу та високу ефективність управління інфраструктурою.

Використання Pritunl VPN з інтегрованою двофакторною автентифікацією (2FA) дозволило забезпечити надзвичайно захищене віддалене підключення до внутрішньої інфраструктури. Такий підхід гарантує, що навіть у разі компрометації основного пароля зловмисник не зможе отримати доступ до системи, оскільки необхідна додаткова форма перевірки особистості. Цей рівень захисту особливо важливий для розподілених систем, де контроль доступу має бути бездоганно послідовним та централізованим.

Конфігурація SSH-з'єднань, що базується виключно на використанні криптографічних ключів, разом із забороною логіну за паролем та прямого доступу для користувача root, значно зменшує вектор можливих атак типу brute-force та інші спроби експлуатації слабких місць. Додатковий механізм Challenge-Response авторизації забезпечив ще один рівень перевірки, що ускладнює несанкціоноване проникнення навіть у разі компроментації одного з компонентів аутентифікації.

Інтеграція Cloudflare виступила додатковим бар'єром між зовнішнім трафіком і серверною інфраструктурою, приховуючи справжні IP-адреси серверів і забезпечуючи фільтрацію запитів. Використання веб-фаєрволу, механізмів кешування та розподілу навантаження дозволило ефективно зменшувати ризики DDoS-атак і блокувати підозрілий трафік.

Встановлення Zabbix як системи моніторингу дозволило забезпечити постійний контроль за всіма критичними компонентами інфраструктури. Реальний час збору метрик та сповіщення в режимі реального часу дозволяє оперативно реагувати на будь-які аномалії, що можуть свідчити про потенційну загрозу. Система сповіщень і автоматизоване логування подій сприяють виявленню та аналізу інцидентів ще на ранніх етапах їх виникнення.

Використання Ansible для централізованого управління користувачами та розгортання конфігурацій дозволило стандартизувати налаштування на всіх вузлах інфраструктури. Завдяки ідемпотентній природі Ansible, конфігурації застосовуються рівномірно, що переконує в однорідності політик безпеки та

значно знижує ризики, пов’язані з людським фактором. Використання тегів дозволило гнучко керувати виконанням окремих завдань, що сортується за пріоритетами безпеки.

Комплексна система захисту, яка включає використання Ubuntu 24.04 LTS, захищених VPN-з’єднань через Pritunl з 2FA, ретельно налаштованих SSH-з’єднань, додаткової мережевої фільтрації від Cloudflare, прозорого моніторингу через Zabbix та централізованого управління з Ansible, демонструє високий рівень стійкості до сучасних загроз інформаційної безпеки.

Результати дипломної роботи підтверджують, що впровадження інтегрованої системи безпеки може бути ефективним інструментом для захисту критичних ресурсів організації, а також служити прикладом масштабованого та адаптивного підходу до розв’язання завдань безпеки в сучасних IT-системах.

Практична цінність реалізованої системи полягає у забезпеченні багаторівневого захисту критичних даних та ресурсів організації, що сприяє зниженню ризиків несанкціонованого доступу та фінансових втрат. Завдяки використанню сучасних технологій, таких як Ubuntu 24.04 LTS, захищені VPN (Pritunl з 2FA), налаштовані SSH-з’єднання за ключами, додатковий мережевий захист через Cloudflare та централізована автоматизація управління за допомогою Ansible із супроводом інтегрованого моніторингу через Zabbix, система демонструє високу стабільність і ефективність роботи в умовах сучасних кіберзагроз.

У результаті виконання кваліфікаційної роботи було досягнуто основної мети – створення програмного забезпечення, здатного ефективно автоматизувати налаштування середовища серверів з можливістю масштабування, розширення функцій та адаптації до специфічних умов експлуатації. Отримані результати можуть слугувати основою для подальших досліджень та практичного застосування у суміжних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ubuntu Server Documentation [Електронний ресурс]. Режим доступу – https://ubuntu.com/server/docs

2. Ubuntu Release Notes [Електронний ресурс]. Режим доступу – https://wiki.ubuntu.com/UbuntuReleases

3. Ubuntu Security Notices [Електронний ресурс]. Режим доступу – https://ubuntu.com/security/notices

4. AppArmor Documentation [Електронний ресурс]. Режим доступу – https://wiki.ubuntu.com/AppArmor

5. Ansible Documentation [Електронний ресурс]. Режим доступу – https://docs.ansible.com/

6. Ansible Playbook Introduction [Електронний ресурс]. Режим доступу – https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html

7. Ansible Vault Documentation [Електронний ресурс]. Режим доступу – https://docs.ansible.com/ansible/latest/user_guide/vault.html

8. Ansible Inventory Documentation [Електронний ресурс]. Режим доступу – https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

9. Ansible Tags Documentation [Електронний ресурс]. Режим доступу – https://docs.ansible.com/ansible/latest/user_guide/playbooks_tags.html

10. Ansible Command Line Tools [Електронний ресурс]. Режим доступу – https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html

11. Pritunl Official Documentation [Електронний ресурс]. Режим доступу – https://docs.pritunl.com

12. Pritunl Installation Guide [Електронний ресурс]. Режим доступу – https://docs.pritunl.com/docs/latest/Installation

13. Pritunl Two-Factor Authentication (2FA) [Електронний ресурс]. Режим доступу – https://docs.pritunl.com/docs/latest/2fa

14. OpenVPN Community Resources [Електронний ресурс]. Режим доступу – https://openvpn.net/community-resources/

15. OpenVPN Documentation [Електронний ресурс]. Режим доступу – https://openvpn.net/vpn-server-resources/

16. Опис протоколу SSH (RFC 4251) [Електронний ресурс]. Режим доступу – https://tools.ietf.org/html/rfc4251

17. OpenSSH Manual Pages [Електронний ресурс]. Режим доступу – https://www.openssh.com/manual.html

18. SSH Key Authentication Overview [Електронний ресурс]. Режим доступу – https://www.ssh.com/academy/ssh/key/overview

19. SSH Hardening Best Practices [Електронний ресурс]. Режим доступу – https://www.ssh.com/academy/ssh/hardening

20. Challenge-Response Authentication (Overview) [Електронний ресурс]. Режим доступу – https://en.wikipedia.org/wiki/Challenge%E2%80%93response_authentication

21. Cloudflare Developers Portal [Електронний ресурс]. Режим доступу – https://developers.cloudflare.com/

22. Cloudflare Web Application Firewall (WAF) [Електронний ресурс]. Режим доступу – https://developers.cloudflare.com/waf/

23. Cloudflare DDoS Protection Overview [Електронний ресурс]. Режим доступу – https://www.cloudflare.com/learning/ddos/what-is-ddos/

24. Cloudflare Caching Documentation [Електронний ресурс]. Режим доступу – https://developers.cloudflare.com/cache/

25. Cloudflare SSL/TLS Documentation [Електронний ресурс]. Режим доступу – https://developers.cloudflare.com/ssl/

26. Cloudflare API Documentation [Електронний ресурс]. Режим доступу – https://api.cloudflare.com/

27. Zabbix Official Documentation [Електронний ресурс]. Режим доступу – https://www.zabbix.com/documentation/current/manual

28. Zabbix API Documentation [Електронний ресурс]. Режим доступу – https://www.zabbix.com/documentation/current/manual/api

29. Zabbix Notifications and Alerts [Електронний ресурс]. Режим доступу – https://www.zabbix.com/documentation/current/manual/config/notifications

30. NGINX Official Documentation [Електронний ресурс]. Режим доступу – https://nginx.org/en/docs/

31. NGINX Reverse Proxy Configuration [Електронний ресурс]. Режим доступу – https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

32. NGINX SSL/TLS Configuration [Електронний ресурс]. Режим доступу – https://docs.nginx.com/nginx/admin-guide/security-controls/securing-http-traffic-upstream/

33. Cisco DDoS Protection Best Practices [Електронний ресурс]. Режим доступу – https://www.cisco.com/c/en/us/products/security/ddos-defense/index.html

34. CIS Ubuntu Linux Benchmark [Електронний ресурс]. Режим доступу – https://www.cisecurity.org/cis-benchmarks/

35. Secure Boot Documentation on Ubuntu [Електронний ресурс]. Режим доступу – https://ubuntu.com/blog/how-to-use-secure-boot-on-ubuntu

36. Debian Administration Handbook [Електронний ресурс]. Режим доступу – https://debian-handbook.info/browse/stable/

37. SELinux vs. AppArmor Comparison [Електронний ресурс]. Режим доступу – https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/selinux_policy_guide/

38. OWASP SSH Security Cheat Sheet [Електронний ресурс]. Режим доступу – https://cheatsheetseries.owasp.org/cheatsheets/SSH_Security_Cheat_Sheet.html

39. Ansible Idempotency Concepts [Електронний ресурс]. Режим доступу – https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#idempotency

40. Ansible Best Practices [Електронний ресурс]. Режим доступу – https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html

41. YAML Specification 1.2 [Електронний ресурс]. Режим доступу – https://yaml.org/spec/1.2/spec.html

42. GitHub: Ansible Examples Repository [Електронний ресурс]. Режим доступу – https://github.com/ansible/ansible-examples

43. DigiCert Certificate Management [Електронний ресурс]. Режим доступу – https://www.digicert.com/

44. Mozilla Server Side TLS Guidelines [Електронний ресурс]. Режим доступу – https://wiki.mozilla.org/Security/Server_Side_TLS

45. Cloud Security Alliance Resources [Електронний ресурс]. Режим доступу – https://cloudsecurityalliance.org/

46. CIS Benchmarks [Електронний ресурс]. Режим доступу – https://www.cisecurity.org/cis-benchmarks/

47. OWASP Official Website [Електронний ресурс]. Режим доступу – https://owasp.org/

48. NIST Cybersecurity Framework [Електронний ресурс]. Режим доступу – https://www.nist.gov/cyberframework

49. ISO/IEC 27001 Information Security Standard [Електронний ресурс]. Режим доступу – https://www.iso.org/isoiec-27001-information-security.html

50. FIPS Publications (NIST) [Електронний ресурс]. Режим доступу – https://csrc.nist.gov/publications/fips

51. Centralized Log Management Overview [Електронний ресурс]. Режим доступу – https://www.loggly.com/ultimate-guide/centralized-log-management/

52. Sudoers Manual [Електронний ресурс]. Режим доступу – https://www.sudo.ws/man/1.8.13/sudoers.man.html

53. Linux User and Group Management [Електронний ресурс]. Режим доступу – https://www.cyberciti.biz/faq/linux-user-group-creation/

54. Ansible Vault Encryption Best Practices [Електронний ресурс]. Режим доступу – https://www.ansible.com/blog/ansible-vault-ultimate-guide

55. Cloudflare Rate Limiting Documentation [Електронний ресурс]. Режим доступу – https://developers.cloudflare.com/rate-limiting/

56. Prometheus Monitoring Overview [Електронний ресурс]. Режим доступу – https://prometheus.io/docs/introduction/overview/

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку автоматизації налаштування середовища серверів.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу №46-02 від 17.01.2025 року, видане на кафедрі кібербезпеки та програмного забезпечення.

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення автоматизації налаштування середовища серверів.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– автоматизацію налаштування середовища серверів;

– цілісність даних у процесі роботи;

– захищені канали зв’язку.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення повинно забезпечувати стабільну роботу інфраструктури.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Система повинна забезпечувати безперебійну роботу з доступністю не менше 99,99% часу, а також підтримувати автоматичне відновлення після будь-яких збоїв або аварійних ситуацій. Інтегровані механізми централізованого моніторингу та аварійного відновлення гарантують своєчасну діагностику та усунення потенційних проблем, що сприяє високій надійності інформаційної інфраструктури.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Система повинна включати сучасні сервери з підтримкою апаратного шифрування та захищених обчислень, а також мережеве обладнання, здатне забезпечувати надійний трафік та мінімізувати затримки. Всі технічні засоби мають відповідати вимогам продуктивності, резервування критичних компонентів, масштабованості та інтеграції засобів моніторингу для оперативного виявлення та усунення можливих збоїв. У адміністратора вже має бути предустановлений пакет ПЗ Ansible та наявність необхідних файлів конфігурацій .yml.

5.8 Вимоги до інформаційної і програмної сумісності

Система повинна забезпечувати повну сумісність з сучасними операційними системами, мережевими протоколами та криптографічними стандартами, що використовуються для захисту даних. Програмне забезпечення має підтримувати інтеграцію з засобами моніторингу, управління доступом та автоматизації, а також гарантувати коректну взаємодію між різними технологічними компонентами без втрати продуктивності або порушення безпеки.

5.8.1 Обладнання

Комп’ютер Intel® Core™ i5-13500 14 Core "Raptor Lake-S", 64 GB DDR4 RAM, 2 x 512 GB NVMe SSD (Gen 4, Software RAID 1), 1 Gbit/s bandwidth

5.8.2 Мова програмування

Програму розроблено на мові програмування YAML.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД.

7 Перелік документів, що розробляються

– Структурна схема системи.
– Функціональна схема системи.
– Діаграма процесів.
– Блок-схема алгоритму роботи програми.

– Пояснювальна записка.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 24.05.2025 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист __.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ О.М. Дрєєв

Програмне забезпечення автоматизації налаштування середовища серверів

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 15

Літера: РП

Кропивницький – 2025 року

// secure-ssh.yml

```
- hosts: all
  become: yes
  tasks:
    - name: Оновлення списку пакетів
      apt:
        update_cache: yes

    - name: Оновлення системи
      apt:
        upgrade: dist

    - name: Встановлення базових утиліт
      apt:
        name:
          - vim
          - curl
          - htop
        state: present

    - name: Заборонити віддалений логін для root
      lineinfile:
        path: /etc/ssh/sshd_config
        regexp: '^PermitRootLogin'
        line: 'PermitRootLogin no'

    - name: Заборонити авторизацію за паролем
      lineinfile:
        path: /etc/ssh/sshd_config
        regexp: '^PasswordAuthentication'
        line: 'PasswordAuthentication no'

    - name: Дозволити тільки авторизацію за ключами
      lineinfile:
        path: /etc/ssh/sshd_config
        regexp: '^PubkeyAuthentication'
        line: 'PubkeyAuthentication yes'

    - name: Заборонити Challenge-Response авторизацію
      lineinfile:
        path: /etc/ssh/sshd_config
        regexp: '^ChallengeResponseAuthentication'
        line: 'ChallengeResponseAuthentication no'

    - name: Заборонити авторизацію за паролем у 50-cloud-init.conf
      lineinfile:
        path: /etc/ssh/sshd_config.d/50-cloud-init.conf
        regexp: '^PasswordAuthentication'
        line: 'PasswordAuthentication no'
        create: yes

    - name: Перезапустити SSH для застосування змін
      systemd:
        name: ssh
        state: restarted
```

//inventory.yml

```
vpn:
  hosts:
    vpn-server:
      ansible_host: 172.16.100.2

webservers:
  hosts:
    web-1:
      ansible_host: 172.16.100.3
    web-2:
      ansible_host: 172.16.100.4

dbservers:
  hosts:
    db-server:
      ansible_host: 172.16.100.6

monitoring:
  hosts:
    zabbix:
      ansible_host: 172.16.100.5

proxy:
  hosts:
    reverse-proxy:
      ansible_host: 172.16.100.50
```

// create-zabbix-server.yml

```
- hosts: zabbix
  become: yes
  vars_files:
    - vault.yml
  vars:
    ansible_python_interpreter: /usr/bin/python3
  tasks:
    - name: Download Zabbix repository package
      get_url:
        url:
https://repo.zabbix.com/zabbix/7.2/release/ubuntu/pool/main/z/zabbix-
release/zabbix-release_latest_7.2+ubuntu24.04_all.deb
        dest: /tmp/zabbix-release.deb

    - name: Install Zabbix repository package
      command: dpkg -i /tmp/zabbix-release.deb

    - name: Update apt cache
      apt:
        update_cache: yes

    - name: download and install MySQL server
      apt:
        name: mysql-server
        state: present

    - name: install and enable MySQL
      systemd:
        name: mysql
        state: started
        enabled: yes

    - name: Install Zabbix packages
      apt:
        name:
          - zabbix-server-mysql
          - zabbix-frontend-php
          - zabbix-nginx-conf
          - zabbix-sql-scripts
          - zabbix-agent
          - python3-pymysql
        state: present

    - name: Ensure MySQL is running
      service:
        name: mysql
        state: started
        enabled: yes

    - name: Create Zabbix database
      mysql_db:
        name: zabbix
        state: present
        encoding: utf8mb4
        collation: utf8mb4_bin
      ignore_errors: yes

    - name: Create Zabbix database user
      mysql_user:
        name: zabbix
        password: "zabbixpass"
        #password: "{{ vault_zabbix_db_password }}"
        priv: "zabbix.*:ALL"
        host: localhost
```

```
        state: present
      ignore_errors: yes

    - name: Enable log_bin_trust_function_creators
      mysql_query:
        query: "SET GLOBAL log_bin_trust_function_creators = 1;"
        login_user: root
        login_password: "{{ vault_mysql_root_password }}"

    - name: Extract Zabbix schema file
      command: gzip -d /usr/share/zabbix/sql-scripts/mysql/server.sql.gz
      args:
        creates: /usr/share/zabbix/sql-scripts/mysql/server.sql

    - name: check if exists table users.ibd
      command: ls /var/lib/mysql/zabbix/users.ibd
      register: users_table_check
      ignore_errors: yes
      tags: import_schema

    - name: Import Zabbix schema into database (only if users.ibd doesn't
exist)
      raw: "cat /usr/share/zabbix/sql-scripts/mysql/server.sql | mysql --
default-character-set=utf8mb4 -uzabbix -pzabbixpass zabbix"
      when: users_table_check.rc != 0
      tags: import_schema

    - name: Disable log_bin_trust_function_creators
      mysql_query:
        query: "SET GLOBAL log_bin_trust_function_creators = 0;"
        login_user: root
        login_password: "{{ vault_mysql_root_password }}"

    - name: Configure Zabbix server database password
      lineinfile:
        path: /etc/zabbix/zabbix_server.conf
        regexp: '^DBPassword='
        line: "DBPassword=zabbixpass"

    - name: Configure Zabbix frontend (Nginx)
      lineinfile:
        path: /etc/zabbix/nginx.conf
        insertafter: '^server {'
        line: 'listen 8080;'
      tags: nginx

    - name: Configure Zabbix frontend (Server Name)
      lineinfile:
        path: /etc/zabbix/nginx.conf
        regexp: '^# server_name'
        insertafter: '^server {'
        line: 'server_name zabbix.d1ploma.site;'
      tags: nginx

    - name: Restart Zabbix services
      systemd:
        name: "{{ item }}"
        state: restarted
        enabled: yes
      loop:
        - zabbix-server
        - zabbix-agent
        - nginx
        - php8.3-fpm
```

//create-zabbix-agent.yml

```
- hosts: all
  become: yes
  vars:
    zabbix_repo_url:
"https://repo.zabbix.com/zabbix/7.2/release/ubuntu/pool/main/z/zabbix-
release/zabbix-release_latest_7.2+ubuntu24.04_all.deb"
    zabbix_repo_deb: "/tmp/zabbix-release_latest_7.2+ubuntu24.04_all.deb"

  tasks:
    - name: Видалення старих версій Zabbix Agent
      apt:
        name:
          - zabbix-agent2
          - zabbix-agent
        state: absent

    - name: Видалення залишкових конфігураційних файлів
      file:
        path: "{{ item }}"
        state: absent
      loop:
        - /etc/zabbix/zabbix_agentd.conf
        - /etc/zabbix/zabbix_agent2.conf
        - /var/log/zabbix/

    - name: Завантаження пакета репозиторію Zabbix
      get_url:
        url: "{{ zabbix_repo_url }}"
        dest: "{{ zabbix_repo_deb }}"
        mode: '0644'

    - name: Встановлення пакета репозиторію Zabbix
      command: dpkg -i {{ zabbix_repo_deb }}

    - name: Оновлення кешу пакетного менеджера
      apt:
        update_cache: yes
        cache_valid_time: 600

    - name: Встановлення Zabbix Agent2
      apt:
        name: zabbix-agent2
        state: present

    - name: Переконатися, що файл конфігурації існує
      file:
        path: /etc/zabbix/zabbix_agent2.conf
        state: touch
        mode: '0644'

    - name: Налаштування Zabbix Agent2 - встановлення Server
      lineinfile:
        path: /etc/zabbix/zabbix_agent2.conf
        regexp: '^Server='
        line: 'Server=172.16.100.5'
        state: present
      notify: Перезапустити Zabbix Agent2

    - name: Налаштування Zabbix Agent2 - встановлення ServerActive
      lineinfile:
        path: /etc/zabbix/zabbix_agent2.conf
        regexp: '^ServerActive='
        line: 'ServerActive=172.16.100.5'
        state: present
```

```
    notify: Перезапустити Zabbix Agent2

handlers:
  - name: Перезапустити Zabbix Agent2
    systemd:
      name: zabbix-agent2
      state: restarted
      enabled: yes
```

// create-user.yml

```
- hosts: all
  become: yes
  vars_files:
    - vault.yml
  tasks:

    - name: Створення користувача diplom1 з хешованим паролем
      user:
        name: diplom1
        password: "{{ vault_diplom_password }}"
        groups: sudo
        shell: /bin/bash
        state: present
        update_password: always

    - name: Додавання SSH-ключа для diplom1
      authorized_key:
        user: diplom1
        key: "{{ lookup('file', 'keys/diplom1.pub') }}"
        state: present
```

// create-reverse.yml

```
---
- hosts: reverse-proxy
  become: yes
  vars:
    domain: "d1ploma.site"
    # Список backend-серверів (Web service 1 та 2)
    web_servers:
      - "172.16.100.3"
      - "172.16.100.4"
  tasks:
    - name: Встановлення nginx
      apt:
        name: nginx
        state: present
        update_cache: yes

    - name: Створення конфігурації nginx для reverse proxy
      template:
        src: templates/nginx_reverse.conf.j2
        dest: /etc/nginx/sites-available/{{ domain }}.conf

    - name: Активувати сайт reverse proxy в nginx
      file:
        src: /etc/nginx/sites-available/{{ domain }}.conf
        dest: /etc/nginx/sites-enabled/{{ domain }}.conf
        state: link

    - name: Видалити дефолтну конфігурацію nginx (якщо існує)
      file:
        path: /etc/nginx/sites-enabled/default
        state: absent

  handlers:
    - name: Reload nginx
      service:
        name: nginx
        state: reloaded
```

// create-mysql.yml

```
- hosts: db-server
  become: yes
  vars:
    ansible_python_interpreter: /usr/bin/python3
    db_name: "webapp_db"
    db_user: "wpuser"
    db_password: "SuperSecretFFE333444"
    mysql_root_password: "openrootpass"   # openrootpass
  tasks:
    - name: Встановлення PyMySQL для роботи з MariaDB (Python 3)
      apt:
        name: python3-pymysql
        state: present

    - name: Створення файлу /root/.my.cnf для MySQL-клієнта
      copy:
        dest: /root/.my.cnf
        content: |
          [client]
          user=root
          password={{ mysql_root_password }}
        mode: '0600'

    - name: Встановлення MariaDB Server
      apt:
        name: mariadb-server
        state: present
        update_cache: yes

    - name: Запуск та увімкнення MariaDB
      service:
        name: mariadb
        state: started
        enabled: yes

    - name: Налаштування root користувача для використання
mysql_native_password
      community.mysql.mysql_user:
        name: root
        host: localhost
        password: "{{ mysql_root_password }}"
        plugin: mysql_native_password
        state: present
        login_unix_socket: /var/run/mysqld/mysqld.sock

    - name: Створення бази даних для веб-додатку
      community.mysql.mysql_db:
        name: "{{ db_name }}"
        state: present
        login_user: root
        login_password: "{{ mysql_root_password }}"
        login_unix_socket: /var/run/mysqld/mysqld.sock

    - name: Створення користувача бази даних та надання прав
      community.mysql.mysql_user:
        name: "{{ db_user }}"
        password: "{{ db_password }}"
        priv: "{{ db_name }}.*:ALL"
        state: present
        login_user: root
        login_password: "{{ mysql_root_password }}"
        login_unix_socket: /var/run/mysqld/mysqld.sock
```

// create-lamp.yml

```
---
- hosts: webservers
  become: yes
  vars:
    domain: "d1ploma.site"
  tasks:
    - name: Встановлення Apache2
      apt:
        name: apache2
        state: present
        update_cache: yes

    - name: Розгортання веб-сторінки
      copy:
        src: "files/{{ inventory_hostname }}_index.html"
        dest: /var/www/html/index.html
        mode: '0644'

    - name: Створення Apache VirtualHost для домену
      template:
        src: templates/apache_vhost.conf.j2
        dest: /etc/apache2/sites-available/{{ inventory_hostname }}.conf

    - name: Активувати Apache сайт
      command: a2ensite {{ inventory_hostname }}.conf
      notify: Reload Apache2

    - name: Деактивувати дефолтний сайт Apache
      command: a2dissite 000-default.conf
      notify: Reload Apache2

  handlers:
    - name: Reload Apache2
      service:
        name: apache2
        state: reloaded
```

/ templates / nginx_reverse.conf.j2

```
upstream web_backend {
{% for ip in web_servers %}
    server {{ ip }};
{% endfor %}
}

server {
    listen 80;
    server_name {{ domain }};

    access_log /var/log/nginx/d1ploma_access.log;
    error_log /var/log/nginx/d1ploma_errors.log;

    # Дозволити доступ лише з Cloudflare IP-діапазонів
    allow 173.245.48.0/20;
    allow 103.21.244.0/22;
    allow 103.22.200.0/22;
    allow 103.31.4.0/22;
    allow 141.101.64.0/18;
    allow 108.162.192.0/18;
    allow 190.93.240.0/20;
    allow 188.114.96.0/20;
    allow 197.234.240.0/22;
    allow 198.41.128.0/17;
    allow 162.158.0.0/15;
    allow 104.16.0.0/13;
    allow 104.24.0.0/14;
    allow 172.16.100.0/24; # Дозвіл з мережі VPN
    deny all;

    location / {
        proxy_pass http://web_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;
    }
}
}
```

/ templates /apache_vhost.conf.j2

```
<VirtualHost *:80>
    ServerName d1ploma.site
    DocumentRoot /var/www/html

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

    ErrorLog /var/log/apache2/{{ inventory_hostname }}-error.log
    CustomLog /var/log/apache2/{{ inventory_hostname }}-access.log combined
</VirtualHost>
```

/Files// web-2_index.html

```
<html>
  <head><title>Web Service 2</title></head>
  <body>
    <h1>Hello, this is server 2 answer</h1>
  </body>
</html>
```

/Files// web-1_index.html

```
<html>
  <head><title>Web Service 1</title></head>
  <body>
    <h1>Hello, this is server 1 answer</h1>
  </body>
</html>
```