Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Інфокомунікацій_____
(повна назва)

Кафедра _____Інформаційно-мережної інженерії_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____перший (бакалаврський)_____
Автоматизація розгортання серверної інфраструктури в інфокомунікаційних
_____мережах за допомогою Ansible_____
(тема)

Виконав:
здобувач __4__ року навчання,
групи _____ТРІМІ-21-2_____

_____Павло Кривонос_____
(власне ім'я, прізвище)

Спеціальність 172 Телекомунікації і радіотехніка_____
(код і повна назва спеціальності)
Тип програми ___освітньо-професійна_____
Освітня програма __Інформаційно-мережна інженерія_____
(повна назва освітньої програми)

Керівник ___ст. викл. Галина Ляшенко_____
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ _____ Валерій БЕЗРУК_____
(підпис) (власне ім'я, прізвище)

2025 р.

Не містить відомостей заборонених до відкритого публікування.

Студент / Павло Кривонос /

Керівник / Галина Ляшенко /

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій

Кафедра Інформаційно-мережної інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 172 Телекомунікації і радіотехніка
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Кривоносу Павлу Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Автоматизація розгортання серверної інфраструктури в інфокомунікаційних мережах за допомогою Ansible

затверджена наказом університету від 23 травня 2025 р. № 410Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 23 червня 2025 р.

3. Вихідні дані до роботи *Провести аналіз можливостей Ansible як засобу автоматизації, порівняти його з іншими подібними інструментами, дослідити модулі, що використовуються для налаштування систем.Дослідити процес автомантизації розгортання серверної інфраструктури в ініокомунікаційних мережах.*

4. Перелік питань, що потрібно опрацювати в роботі

1 Технології розгортання серверної інфраструктури,

2 Вибір технології автоматизації,

3 Розгортання серверної інфраструктури,

4 Написання Ansible playbook

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
Слайди презентації у Power Point _____
_____
_____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів	Примітка
1	Ознайомлення із завданням. Уточнення ТЗ.	26.05-27.05.2025	виконано
2	Підбір літератури за темою роботи.	26.05-28.05.2025	виконано
3	Виконання розділу 1	29.05-31.05.2025	виконано
4	Виконання розділу 2	01.06-05.06.2025	виконано
5	Виконання розділу 3	06.06-08.06.2025	виконано
6	Виконання розділу 4	09.06-13.06.2025	виконано
8	Оформлення пояснювальної записки	14.06-16.06.2025	виконано
9	Оформлення презентаційного матеріалу,	17.06-23.06.2025	виконано
	підготовка до захисту у ЕК		

Дата видачі завдання __26___ __травня_____2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _ст. викл. Галина Ляшенко_____
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 97 с., 38 рис., 10 табл., 18 джерел, 3 додатки.

Мета роботи – розробка скрипту Ansible для автоматичного налаштування програмної частини серверної інфраструктури.

У роботі було проведено аналіз Ansible як інструменту автоматизації, було порівняно його з іншими популярними аналогами.

Проведено аналіз модулів для виконання певних основних задач інструментом Ansible.

Розглянуто основні вимоги до скриптів та застосовано розповсюджені практики написання скрипту. Розроблено скрипт, що автоматично розгортає інструменти для моніторингу стану систем та розгортає GitHub Actions Runner на виділених для них хостах.

СЕРВЕРНА ІНФРАСТРУКТУРА, АВТОМАТИЗАЦІЯ, ANSIBLE, АДМІНІСТРУВАННЯ, НАЛАШТУВАННЯ

ABSTRACT

Explanatory note: 97 p., 38 fig, 10 tabl, 18 sources, 3 app.

The purpose of the work is to develop a script for automatic configuration of program part of server infrastructure.

The paper analyzed Ansible as automation tool, compared with popular analogues.

Analyzed modules for performing certain basic tasks via Ansible. Reviewed basic requirements for scripts and used best practices of script writing. Developed script that automatically deploys systems monitoring tools and deploys GitHub Actions Runner on dedicated hosts.

SERVER INFRASTRUCTURE, AUTOMATION, ANSIBLE, ADMINISTRATION, CONFIGURATION

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

VM – Virtual Machine

PVE – Proxmox Virtual Environment

AWS – Amazon Web Services

WSL – Windows Subsystem for Linux

SSH – Secure Shell

ОС – операційна система

BIOS – Basic Input/Output System

ПЗ – програмне забезпечення

LXC – Linux Container

KVM – Kernel-based Virtual Machine

DNS – Domain Name System

VLAN – Virtual Local Area Network

NAT – Network Address Translation

DHCP – Dynamic Host Configuration Protocol

UEFI – Unified Extensible Firmware Interface

CD-ROM – Compact Disc Read-Only Memory

ISO – International Organization for Standardization

ВСТУП

Темпи сьогоднішнього світу дуже високі в порівнянні з минулими десятиліттями, особливо стрімкими стали темпи розвитку сфери інформаційних технологій. В таких умовах час – майже найважливіший ресурс, який визначає успіх не тільки компаній, а країн і світу в цілому.

Саме тому велика кількість ресурсів та зусиль спрямовано на розробку та використання методів автоматизації розгортання та налаштування інфраструктури.

Автоматизація навіть незначної частини роботи, такої як встановлення кількох програм, економить величезну кількість часу та грошей, звільняючи від необхідності наймати все більше і більше фахівців при масштабуванні інфраструктури.

Теоретично, можливо настільки автоматизувати налаштування системи, що процес, який вручну займатиме тижні, буде виконано автоматично при введенні однієї команди за лічені хвилини.

І хоча компанії користуються різними операційними системами як для адміністрування, так і для роботи працівників, цілком можливо створити універсальний скрипт для налаштування кожної з них.

Однак при створенні подібного необхідно мати досвід з адміністрування цих операційних систем для врахування їх особливостей та обмежень, а також для запобігання виникнення помилок при роботі скрипту.

Як основу для створення скрипту для автоматизації налаштування систем різного призначення було обрано інструмент автоматизації Ansible.

1 ТЕХНОЛОГІЇ РОЗГОРТАННЯ СЕРВЕРНОЇ ІНФРАСТРУКТУРИ

Робота системного адміністратора і DevOps інженера вимагає розгортання, налаштування і підтримки роботи систем майже кожен день. Однак навіть виконання вищевказаних дій вимагає доволі багато часу і концентрації в окремих випадках, наприклад під час початкового налаштування інфраструктури компанії.

Окрім цього, на якість виконаних робіт має вплив людський фактор, який присутній в роботі навіть найкращих професіоналів. Саме тому люди, що працюють за даними професіями, вдаються до використання інструментів автоматизації, що звільняють від більшості рутинних задач і економлять велику частину робочого часу.

До появи Ansible основними інструментами автоматизації були Puppet та Chef, однак обидва інструмента мали свої недоліки, спільним і головним з яких була необхідність встановлення агенту на цільову машину, це значно сповільнювало процес автоматизації та підготовки до неї.

Саме тому у 2012 році було створено Ansible, що поєднує в собі простоту налаштування та відсутність необхідності у встановлені додаткового програмного забезпечення на цільові машини [1].

Як інструмент автоматизації Ansible може використовуватись для досить широкого спектру задач: від виконання команд у командному рядку цільової системи до налаштування хмарних сервісів.

Окремою перевагою даного інструмента є підтримка модулів, створених спільнотою або безпосередньо розробниками певного продукту, наприклад аналог інструменту для встановлення пакетів на системах сімейства Linux «apt», створений для ОС Windows, що має назву Chocolatey.

Розробники даного продукту створили свій модуль для Ansible для полегшення роботи безлічі користувачів. До модуля було додано функціонал з встановлення, видалення та налаштування програм за допомогою Chocolatey [2].

Подібно цьому існує безліч модулів, наприклад колекція Cisco, що містить у собі модулі для налаштування різних мережних пристроїв компанії Cisco, або модулі з колекції Community, що включають роботу з Docker та Windows (не зважаючи на те, що Ansible має офіційний модуль для роботи з даною операційною системою).

І навіть це не весь функціонал Ansible, оскільки даний інструмент написаний мовою Python, розробники надали можливість створювати свої модулі за потреби.

Це є ще однією перевагою Ansible порівняно з іншими інструментами.

Таким чином можна сказати що автоматизувати можливо більшість типових задач, що часто недооцінюється підприємствами, особливо невеликими. В майбутньому відсутність автоматизації навіть невеликої частини системи призводить до надмірних витрат.

В цілому автоматизація має бути створена якомога раніше і має передбачати хоча б частину ймовірних варіантів. Так, наприклад, має бути передбачений випадок, коли певна програма вже встановлена на цільовій машині, або коли цільова машина працює з іншою операційною системою.

Оптимально, коли скрипт автоматизації містить в собі окрім виконання основних завдань також перевірку оновлень як програм, так і системи в цілому, а також перевірку конфігурації системи на предмет вразливостей та помилок.

Отже, важливо впроваджувати автоматизацію на ранніх етапах, бо повторення рутинних задач людиною не гарантує відсутність помилок чи недоробок, особливо при налаштуванні великої кількості систем, цей недолік людської праці може бути усунений автоматизацією виконання цих рутинних завдань.

Окремим важливим пунктом в процесі автоматизації є документування певних дій. Деякі моменти на поточний момент автоматизувати неможливо, тому на допомогу приходить часткова автоматизація у вигляді документації з виконання подібних задач. Детальне пояснення необхідних дій знизить вплив людського фактору до мінімуму, хоча і не знизить його до нуля і вимагатиме робочого часу відповідальної за задачі людини.

Детальна документація має включати в себе як точну послідовність дій, бажано з ілюстраціями, так і пояснення, чому мають виконуватись саме такі дії та можливі наслідки їх невиконання.

1.1 Поняття інфокомунікаційної мережі

Інфокомунікаційна мережа – це система, що поєднує в собі технічні засоби, програмне забезпечення та пристрої, що призначені для передачі даних між кінцевими пристроями користувачів у будь-якій формі [3].

Така мережа складається з кінцевих пристроїв, між якими мають передаватися дані, цими пристроями можуть бути комп’ютери, смартфони, принтери, факси тощо. Головною вимогою до таких пристроїв є можливість підключення до мережі.

Наступним компонентом мережі є канали зв’язку, які можуть бути дротовими (Ethernet кабелі, оптоволоконні кабелі тощо), або бездротовими (Wi-Fi, мобільна мережа і т.д.). Канал зв’язку є середовищем, по якому дані передаються між пристроями.

Мережеве обладнання виконує функції маршрутизації, з’єднання і передачу даних між пристроями. Так, маршрутизатори при підключенні до кінцевої машини призначають їй певну IP-адресу, за якою інші кінцеві пристрої можуть звернутися до машини. Вони ж визначають, який з вузлів мережі має отримати відправлений пакет.

Програмне забезпечення, що включає в себе протоколи передачі даних, визначає, як дані мають передаватись і в якому вигляді. Ці ж протоколи визначають поведінку тих чи інших програм. Також до ПЗ відносяться будь-які програми, які так чи інакше взаємодіють з мережею, найбільш очевидний приклад – браузер.

В деяких випадках до цього списку компонентів додаються хмарні сервіси, що можуть зберігати та обробляти інформацію, до таких сервісів можна віднести AWS, Microsoft Azure, Google Cloud.

1.2 Методи розгортання інфраструктури

На сьогодні існує кілька основних підходів до розгортання серверної інфраструктури, кожен з яких має власні переваги та недоліки.

Традиційний метод розгортання інфраструктури локально, тобто встановлення серверів та іншого устаткування безпосередньо в приміщеннях компанії, такий метод гарантує доступність і безпеку, оскільки устаткування підключене напряму одне до одного. Однак цей метод вимагає значних витрат як на устаткування, так і на виділення приміщення для нього, також обслуговування повністю лягає на співробітників компанії.

Розгортання у хмарі більш вигідне, так як фізичне устаткування розташовується в дата-центрах провайдерів, однак це обмежує доступ до обладнання. Також плюсом такого методу є можливість масштабування в будь-який момент, так як більшість провайдерів надають таку можливість.

Інструменти віртуалізації забезпечують можливість розгортання інфраструктури в рамках одного або кількох серверів, хоча такий метод більше є підметодом двох вищевказаних. Такий метод дозволяє ізолювати певні ресурси інфраструктури і в разі виникнення проблем роботу віртуальних машин набагато швидше і легше відновити.

Схожим на віртуалізацію є метод використання контейнерів, такий метод значно швидший за розгортання віртуальних машин, однак вимагає більш детальної конфігурації, так як існує ймовірність виникнення вразливостей або втрати даних, тому такий метод краще всього використовувати в рамках віртуальних машин, а не цілого сервера.

Безсерверне розгортання також є доволі популярним методом, який працює за принципом запуску коду програми без доступу до сервера. Цей метод використовується саме в тих випадках, коли повноцінна інфраструктура не потрібна для роботи певного додатку.

Метод платформи як сервісу працює майже так само, як і безсерверний метод, однак на відміну від нього код програми виконується в рамках певної інфраструктури, до якої користувач має доступ, хоча і обмежено.

Інфраструктура як код – метод, в якому інфраструктура розгортається і підтримується певними інструментами автоматизації, такими, як Ansible, Puppet, Chef та Terraform. Цей метод більше являє собою саме метод автоматизації, який може використовувати будь-який з вищевказаних методів.

2 ВИБІР ТЕХНОЛОГІЇ ДЛЯ АВТОМАТИЗАЦІЇ

2.1 Огляд Ansible

Ansible являє собою інструмент для автоматизації, керування файлами конфігурації та встановлення програмного забезпечення. Даний інструмент є безагентним, тобто не потребує встановлення агентів на цільових машинах (наприклад на відміну від Chef), і здійснює підключення за протоколами SSH та WinRM (якщо цільова машина працює на ОС Windows). Однак використовувати Ansible можливо лише на операційних системах сімейства Unix та MacOS, тобто офіційної підтримки Windows не передбачено на момент написання даної роботи [4].

Обмеження на використання Ansible на ОС Windows можливо обійти шляхом встановлення Windows Subsystem for Linux (WSL).

На даний момент Ansible є найпопулярнішим інструментом автоматизації, випереджаючи Puppet та Chef. На відміну від конкурентів, Ansible використовує зрозумілу для читання людиною мову YAML (або YML) для налаштування команд, що виконуватимуться при роботі [4].

Однак ця мова вимагає дотримання певних правил, одним з головних є дотримання чіткої табуляції у файлах, що буде продемонстровано пізніше.

Основою Ansible є так званий «playbook» – набір інструкцій, що включає в себе визначення цільової машини, групи машин або всіх наявних машин, визначення шляху до файлу зі змінними (у разі відсутності розрізнення машин за групами) та визначення кроків, що мають бути виконані на цільовій машині (рис. 2.1) [4].

Останній пункт може бути подано як у виді прямих інструкцій, так і у вигляді так званих «ролей», що містять у собі набори задач, файли, значення змінних за замовчуванням. Підхід з «ролями» використовується значно частіше,

особливо у випадках, коли необхідно автоматизувати велику кількість речей, використання даного методу показано на рисунку 2.1.

Цільові машини або групи машин вказуються в файлі «inventory.ini», де окрім необхідних параметрів, таких як IP адреса машини, ім'я користувача та тип підключення (за замовчуванням використовується SSH) також можна вказати певні додаткові параметри, наприклад пароль для доступу, шлях до приватного ключа SSH, порт, за яким здійснити підключення тощо (рис. 2.2) [4].

Незважаючи на те, що Ansible не має чіткої файлової структури, тобто є можливість налаштування за своїм бажанням, в більшості випадків файлова структура матиме вигляд, як показано на рисунку 2.3.

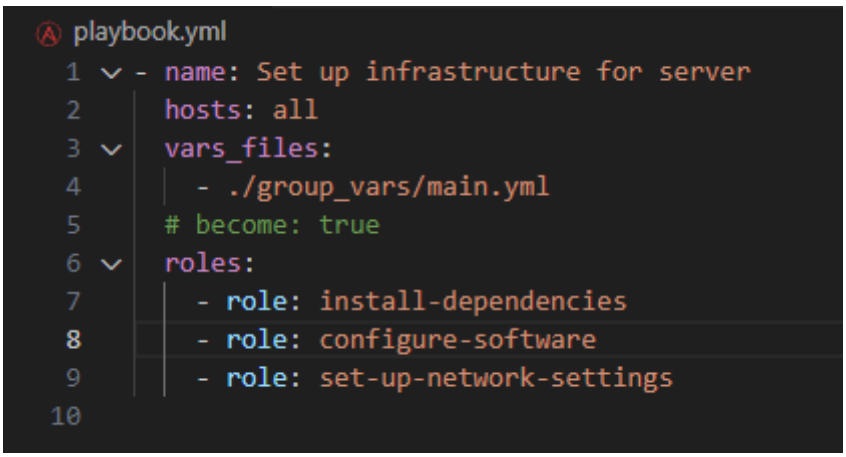

Рисунок 2.1 – Загальний вигляд файлу «playbook.yml»

Рисунок 2.2 – Загальний вигляд файлу «inventory.ini»

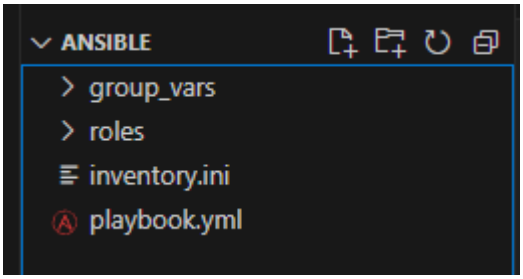

Рисунок 2.3 – Загальний вигляд файлової структури Ansible

Як було зазначено вище, Ansible було створено у 2012 році з метою створення потужного інструменту автоматизації, яким зможе користуватись будь-хто, а не спеціально навчений персонал. Окрім цього, ціллю створення цього інструменту була створення його простим у використанні і без високих вимог до залежностей і системи.

Так як конкуренти мали занадто складні інструменти автоматизації, поява Ansible призвела до ажіотажу серед спільноти, бо окрім простоти мови YAML, відсутності необхідності у використанні агентів, інструмент працював за принципом відправки даних на цільові машини, тобто цільова машина була лише приймальною стороною, не виконуючи ніяких дій окрім надання доступу до системи головній машині з встановленим Ansible. Також як окремий плюс виділялася простота у встановленні і малі вимоги до об’ємів пам’яті [1].

Окрім цього, Ansible керується таким поняттям, як «ідемпотентність» (idempotency), що означає, що незалежно від кількості запусків одного й того самого скрипту результат буде незмінним [4].

Однак Ansible, яким він є зараз, він став лише через рік після своєї появи, коли було додано вищеназвані «playbooks», ролі і модулі.

Окремим фактором популярності цього інструменту є відкритість коду, що дозволило людям зі всього світу шукати помилки та пропонувати свої рішення та шляхи покращення певних речей.

У 2015 році Ansible було куплено компанією RedHat. В цей час компанії почали активно шукати прості та ефективні інструменти автоматизації, можна

сказати, що це був переворотний момент в індустрії автоматизації, так як компанії почали розуміти весь потенціал цього напрямку [1].

В цілому рішення про купівлю Ansible несло в собі кілька ключових моментів, серед яких:

- Збереження відкритості коду та підтримки цього коду;
- Інтеграція з продуктами RedHat (дистрибутив RedHat Linux має вбудований Ansible);
- Збереження структури спільноти.

Всі ці кроки призвели до того, що Ansible став корпоративним інструментом, що призвело до певних побоювань у суспільстві.

Спільнота виражала свої думки щодо можливості втрати відкритості коду, втрати безкоштовних функцій та незалежності в майбутньому, однак всі ці сумніви були розвіяні самою RedHat, що продовжувала виконувати свої зобов'язання, наведені вище.

З часом Ansible став більшим ніж інструментом для налаштування, перетворившись на повноцінну платформу автоматизації і додавши до списку свого програмного забезпечення певні продукти, такі як AWX, що є повноцінним програмним забезпеченням для автоматизації зі своїм веб-інтерфейсом, Ansible Galaxy, що став свого роду бібліотекою модулів, як офіційних, модулів спільноти, так і самописних модулів ентузіастів.

Таким чином Ansible став одним з найпопулярніших інструментів в арсеналі DevOps інженерів та інших фахівців, що задіяні в роботі з IT інфраструктурою.

2.2 Огляд Chef

Для кращого розуміння переваг Ansible над іншими інструментами автоматизації варто провести аналіз інших популярних інструментів.

Chef, як і Ansible, являє собою інструмент керування конфігураціями, однак мова, якою написаний даний інструмент, а саме його клієнтська частина – Ruby, а серверна – мовою Erlang.

Основою даного інструменту є так звані «рецепти» (cookbook), у яких користувачем описуються стани певних програм, файлів, служб і так далі, в яких вони повинні бути у вказаний момент часу. У разі невідповідності записаному в «рецепті» стану, наприклад, програми, Chef намагатиметься змінити цей стан.

Даний інструмент може працювати за двома режимами: клієнт-сервер і автономному. У режимі клієнт-сервер клієнтом періодично відправляються дані про стан системи на сервер, де властивості індексуються і таким чином «рецепти» можуть отримувати доступ до цих даних для прийняття рішень про необхідність налаштування.

Такий підхід, хоча і вимагає встановлення клієнту на машину, що треба налаштовувати, надає низку переваг, серед яких:

Незалежність від користувача після налаштування клієнту та «рецептів», тобто система стає автоматизованою в плані постійного налаштування станів певних елементів системи. Користувач тільки вносить зміни в «рецепти», система сама їх використовує та запускає в роботу.

Відсутність необхідності налаштування моніторингу та додаткових перевірок. Всі необхідні елементи вже перевіряються сервером та налаштовуються автоматично у разі невідповідності отриманих даних до «рецепту».

Для повторення автоматичного використання в Ansible необхідне використання сторонніх інструментів, таких як GitHub Actions, Cron або Jenkins. І при цьому повністю повторити процес з Chef доволі складно.

Так само і для перевірок, вони повинні бути передбачені в playbook, в іншому випадку Ansible повторюватиме певну дію, навіть якщо вона вже була виконана, що в деяких випадках може призвести до втрати даних.

Однак завантаження та встановлення Chef потребує певних маніпуляцій, на відміну від Ansible, що встановлюється стандартним пакетним менеджером. Для встановлення Chef спочатку необхідно завантажити пакет з офіційного сайту спільноти, що відповідає цільовій ОС, після чого запустити цей пакет і дочекатись встановлення. Такі дії проводяться як при встановлені сервера, так і клієнта Chef.

Для підключення клієнта до сервера використовується зашифрований за алгоритмом RSA ключ. Таке підключення забезпечує безпеку як сервера, так і клієнта.

Зазвичай, сервер Chef автоматично генерує пару ключів, однак у разі відсутності її можна створити командою “ssh-keygen -t rsa”, при виконанні якої необхідно буде вказати ім’я для ключів (або пропустити цей крок для використання стандартного імені id_rsa), шлях для файлів (або пропустити для використання стандартного шляху ~/.ssh/), а також фразу-пароль (необов’язково).

2.3 Огляд Puppet

Фактично, Puppet працює за тими самими принципами, що й Chef – система складається з сервера та клієнта, клієнт періодично порівнює поточний стан системи з описаним у файлі конфігурації на сервері і у разі необхідності змінює систему у відповідності з конфігурацією.

Однак і між цими двома інструментами є кілька важливих відмінностей, зокрема мова, якою описуються необхідні дії, у Chef це мова Ruby, а Puppet має свою окрему декларативну мову Puppet DSL, яка є більш простою для розуміння навіть новачку.

Іншою важливою відмінністю між ними є те, як вони працюють, а саме що описується у файлах конфігурації. Chef у файлах конфігурації містить інструкції, що ведуть до певного стану системи, тобто користувач може сам визначати,

якими методами буде досягнуто тих чи інших результатів. При використанні Puppet користувач може лише вказати кінцевий стан системи без визначення, яким саме чином його буде досягнуто, здебільшого таке обмеження існує саме через використання окремої мови для налаштування файлів конфігурації, в той час як Chef фактично використовує мову програмування Ruby для тих самих цілей.

Однак в цьому плані Puppet є більш автоматизованим за рахунок вбудованих інструментів, таких як Facter, що збирає дані про цільову систему, що в подальшій роботі дозволить системі автоматично визначати, які саме команди треба виконувати в тому чи іншому випадку, наприклад при встановленні пакетів на кілька цільових машин, що працюють на різних дистрибутивах Linux, Puppet автоматично визначить, які команди необхідно використати, щоб система відповідала описаній в файлі конфігурації.

Таким чином при роботі Puppet сервер отримує дані про поточний стан цільової системи і на основі цих даних створює для кожної цільової системи окремий список інструкцій, що необхідно виконати для досягнення бажаного стану, це також включає всі залежності, наприклад треба виконати певну команду, яка присутня в певному пакеті, Puppet додатково встановить цей пакет.

Після виконання всіх інструкцій, отриманих від сервера, цільова машина надсилає детальний звіт про всі зміни, що було внесено, і про всі помилки, що виникли в процесі.

Таким чином можна сказати, що Puppet має не менше і не більше відмінностей з Ansible, ніж Ansible з Chef. Окрім цього, Ansible має певні спільні або частково спільні риси з цими двома інструментами.

Також цілком можливо досягти певної синергії між Ansible і цими інструментами, надавши Ansible роль первинного налаштування, наприклад, встановлення агентів на цільові машини, а іншому інструменту надати роль постійної підтримки стану системи.

Однак, як було сказано вище, Ansible цілком може підтримувати стан системи самостійно за використання додаткових інструментів.

2.4 Вибір інструменту автоматизації

На основі цих даних можливо порівняти технології методом аналізу ієрархій між собою для порівняння їх можливостей, переваг та недоліків один над одним.

Метод аналізу ієрархій полягає в структуруванні задачі прийняття рішень на базі багатокритеріальної ієрархії [5].

Даний метод є універсальним, тобто його можна застосовувати практично для будь-яких задач, включаючи дану [5].

Кожному критерію надається пріоритет від 1 до 9, де 1 – низький пріоритет, тобто критерій не надає великої переваги одному варіанту над іншим, 3 – низька перевага, 5 – доволі велика перевага, 7 – значна перевага 9 – найвищий пріоритет, що надає велику перевагу над іншими варіантами, всі парні числа в цьому діапазоні являються проміжними між рівнями пріоритету.

Для порівняння необхідно визначити, які саме критерії впливатимуть на вибір, тому створено таблицю 2.1, до якої занесено всі критерії порівняння.

На основі цієї таблиці стало можливо обрати пріоритети для кожного з критеріїв. З цією таблицею проведено опитування серед освічених DevOps інженерів для утворення порівняльної таблиці критеріїв між собою. Також створено декомпозицію задачі вибору (рис. 2.4).

Після проведення опитування було розраховано ступінь узгодженості в показаннях експертів, отримано значення менше 15%, що свідчить про високий рівень узгодженості.

Оскільки в методі аналізу ієрархій проводиться попарне порівняння, створили таблицю таких порівнянь (табл. 2.2). Діагональ таблиці заповнили

значеннями «1», а елементи, що знаходяться під діагоналлю, встановили оберненими до елементів, що знаходяться над діагоналлю, значеннями.

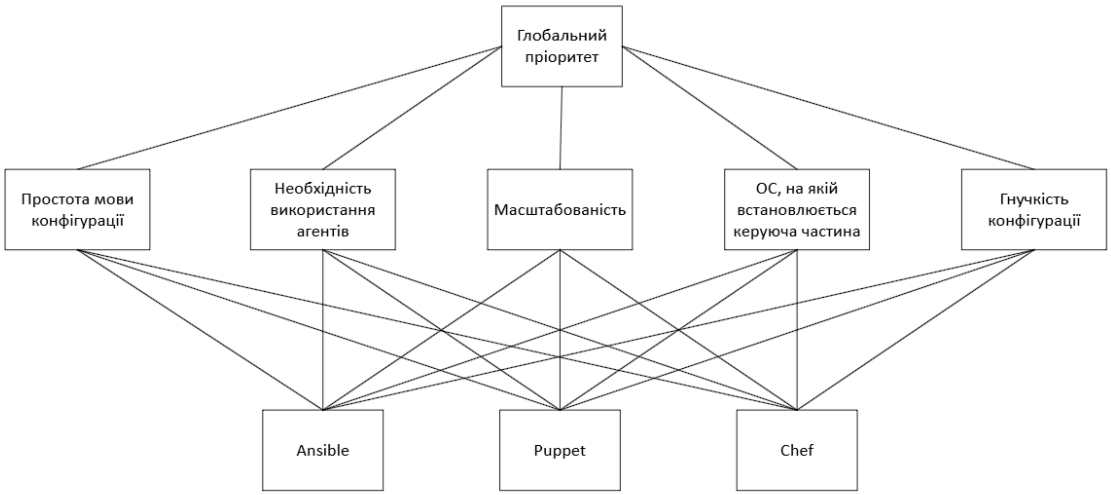

Рисунок 2.4 – Декомпозиція задачі вибору інструменту автоматизації

Таблиця 2.1 – Критерії для порівняння інструментів автоматизації

Характеристика	Ansible	Chef	Puppet
Мова конфігурації	YAML/YML	Ruby DSL	Puppet DSL
Використання агентів	Ні	Так	Так
Масштабованість	Всі масштаби інфраструктури	Великі масштаби	Великі масштаби
ОС для керуючої частини	Linux-based	Linux-based, MacOS, Microsoft Windows	Linux-based, Microsoft Windows
Гнучкість конфігурації	Легко додати або змінити код	Важко оновлювати код	Важко оновлювати код

Таблиця 2.2 – Порівняння важливості критеріїв між собою

	Мова конфігурації	Використання агентів	Масштабованість	ОС для керуючої частини	Гнучкість конфігурації
Мова конфігурації	1	3	1/5	1	1/5
Використання агентів	1/3	1	1/7	3	1/5
Масштабованість	5	7	1	9	5
ОС для керуючої частини	1	1/3	1/9	1	1/7
Гнучкість конфігурації	5	5	1/5	7	1

Для нормування цих значень використано формулу головного власного вектору матриці попарних порівнянь показників якості.

$$V_j = \sqrt[n]{\prod_{i=1}^{n} a_{ij}}, \qquad j = \overline{1,n}$$

Де n – кількість показників якості.

Отримавши це значення було розраховано вектор пріоритетів за формулою.

$$P_j = \frac{V_j}{S}, \qquad j = \overline{1,n}$$

Таблиця 2.3 – Значення векторів пріоритетів

Мова конфігурації	Використання агентів	Масштабованість	ОС для керуючої частини	Гнучкість конфігурації
0,108	0,059	0,248	0,038	0,547

Таким чином було виявлено, що більшість віддає перевагу саме гнучкості конфігурації, на другому місці – масштабованість.

Аналогічним чином було створено порівняльні таблиці інструментів для кожного критерія.

Таблиця 2.4 – Порівняння інструментів автоматизації за мовою конфігурації

	Ansible	Chef	Puppet
Ansible	1	7	4
Chef	1/7	1	1/3
Puppet	1/4	3	1

Таблиця 2.5 – Порівняння інструментів автоматизації за використанням агентів

	Ansible	Chef	Puppet
Ansible	1	7	7
Chef	1/7	1	1
Puppet	1/7	1	1

Таблиця 2.6 – Порівняння інструментів автоматизації за масштабованістю

	Ansible	Chef	Puppet
Ansible	1	5	5
Chef	1/5	1	1
Puppet	1/5	1	1

Таблиця 2.7 – Порівняння інструментів автоматизації за ОС для керуючої частини

	Ansible	Chef	Puppet
Ansible	1	1/5	1/5
Chef	5	1	1
Puppet	5	1	1

Таблиця 2.8 – Порівняння інструментів автоматизації за гнучкістю конфігурації

	Ansible	Chef	Puppet
Ansible	1	7	5
Chef	1/7	1	1/5
Puppet	1/5	5	1

Результати нормалізації цих даних та остаточний вибір технології занесено до таблиці 2.9.

Таблиця 2.9 – Результати обчислень значень глобального вектору пріоритетів

	k_1	k_2	k_3	k_4	k_5	Результат
Ansible	0,705	0,778	0,714	0,091	0,715	0,6006
Chef	0,084	0,111	0,143	0,455	0,067	0,172
Puppet	0,211	0,111	0,143	0,455	0,218	0,2276

Таким чином математично було доведено, що Ansible є найкращим вибором для поставленої задачі за п’ятьма критеріями, а саме за мовою конфігурації, необхідністю встановлення агентів, ОС для встановлення керуючої частини інструменту, масштабованістю та гнучкістю конфігурації.

Єдиний критерій, за яким Ansible програє аналогам – це ОС, на якій встановлюється керуюча частина. В цьому плані Ansible обмежується лише Linux системами.

2.5 Встановлення Ansible на ОС Windows

Як було зазначено вище, Ansible не має офіційної підтримки ОС Microsoft Windows, але це обмеження можливо обійти, встановивши WSL, для цього в першу чергу необхідно ввімкнути віртуалізацію на машині, для чого необхідно відкрити BIOS (Basic Input/Output System), місцезнаходження параметру віртуалізації і його назва відрізняється у різних виробників.

Після ввімкнення даної функції і перезавантаження машини є два шляхи встановлення WSL: за допомогою командного рядка або за допомогою пакетного менеджера Chocolatey.

У першому випадку необхідно відкрити оболонку командного рядка Windows PowerShell або сам командний рядок і ввести команду «wsl.exe -install». Якщо дана команда видає помилку, варто виконати команду «dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart», після чого виконати ще одну команду «dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart» і повторити першу команду [6].

Тобто не у всіх виданнях Windows додатковий компонент WSL ввімкнено, а в деяких виданнях цей компонент відсутній, наприклад у Home Edition.

У другому випадку треба встановити пакетний менеджер Chocolatey та виконати команду «choco install wsl» та погодитись зі встановленням. Після чого вмикаються всі необхідні компоненти і параметри і встановлюється програмне забезпечення.

В обох випадках після встановлення необхідно перезапустити комп’ютер для початку роботи. При виконанні команди «wsl» користувач перейде до встановленого за замовчуванням дистрибутиву (при першому запуску WSL завантажує останню можливу версію дистрибутиву Ubuntu Linux).

При першому запуску користувач отримає запит на введення ім’я користувача і паролю для входу, пропустити цей крок неможливо. Також варто

зазначити, що при введенні пароля не показуються ані введені символи, ані їх кількість.

Для встановлення додаткових дистрибутивів необхідно ввести у PowerShell або командному рядку команду «wsl --install -d <назва дистрибутива>», для виведення списку всіх доступних дистрибутивів вводиться команда «wsl --list --online».

Після встановлення для відкриття терміналу завантаженого дистрибутива вводиться команда «wsl -d <назва дистрибутива>».

Після виконання даних дій користувач фактично отримує віртуальну машину на ОС Ubuntu Linux, що використовує ресурси основної системи Windows без додаткових програм, таких як Docker, Hyper-V та інші.

3 РОЗГОРТАННЯ СЕРВЕРНОЇ ІНФРАСТРУКТУРИ

Як було сказано у попередньому розділі, інфокомунікаційна мережа складається з кінцевих пристроїв, пристроїв маршрутизації, програмного забезпечення, середовища передачі та іноді з хмарних сервісів.

Зазвичай безпосередньо серверна інфраструктура складається з головного додатку, для якого розгортається вся інфраструктура, програмного забезпечення для проксі запитів до додатку або інших компонентів, ПЗ для моніторингу стану тих чи інших компонентів та/або ресурсів машин, що входять до інфраструктури.

Однак така структура не є стандартною, кожна компанія в залежності від потреб та вимог може змінювати її. Так, можливо, що сервер містить в собі не додаток, а ПЗ для його розгортання, наприклад рушії GitHub, що обробляють інформацію з репозиторія і виконують дії, зазначені у файлах робочих процесів (workflow).

Тобто спрощено схему інфраструктури можна подати як систему, що складається з головного функціоналу, підтримуючого функціоналу і мережі.

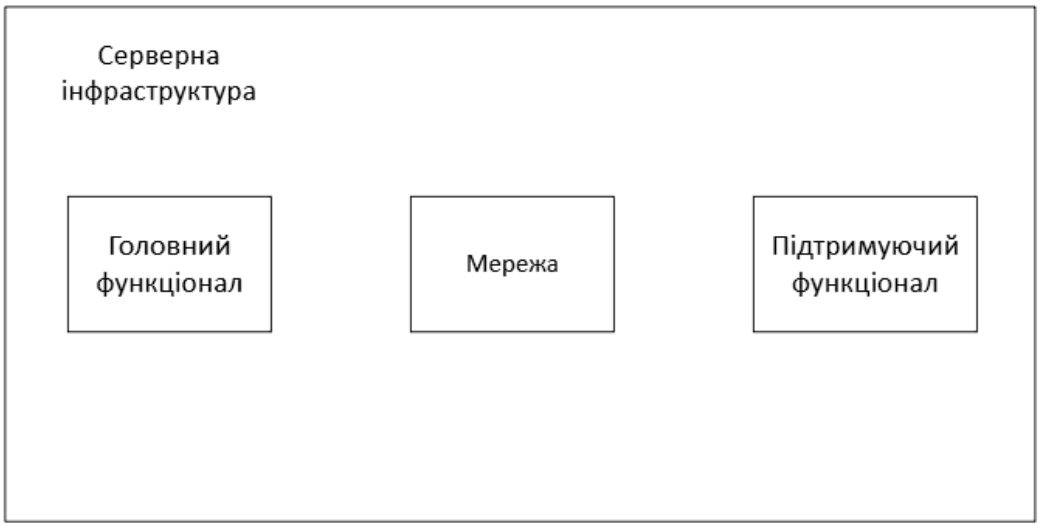

Рисунок 3.1 – Спрощена схема серверної інфраструктури

3.1 Сучасна серверна інфраструктура

На сьогоднішній день в компаніях спостерігаються тенденції з переходу з фізичних серверів у хмару, що обумовлено меншою вартістю хмарних серверів в порівнянні з фізичними, зменшенням об’ємів роботи з налаштування і підтримки їх роботи.

Так, хмарні провайдери, як Microsoft Azure, AWS, Google Cloud Platform фактично звільняють від роботи з мережним обладнанням, надаючи готові рішення з можливістю їх модифікації в певних рамках.

Таким чином Microsoft Azure дає можливість налаштувати, з яких IP адрес може здійснюватися підключення і за якими портами. Окрім цього є можливість налаштування розподільника навантаження (Load Balancer), що фактично є хмарним замінником проксі, таких як Nginx, Caddy та інші.

Для більшого контролю над всією інфраструктурою деякі компанії використовують інструменти керування контейнерами, таких як Kubernetes, однак подібні інструменти вимагають глибоких знань і більше інших методів вимагає автоматизації.

Ще одним варіантом для розгортання серверної інфраструктури є оренда або використання особистого сервера з встановленою на ньому платформою віртуалізації, наприклад Proxmox. Такий підхід дозволяє контролювати кожний аспект інфраструктури.

Саме такий підхід буде використано в даній роботі, так як він надає більшу свободу у налаштуванні за хмарні платформи. Однак для порівняння буде наведено і спосіб розгортання у хмарній платформі.

3.2 Proxmox Virtual Environment

Proxmox (Proxmox Virtual Environment) – платформа віртуалізації, створена на базі дистрибутиву Debian Linux. Дана платформа пропонує два варіанти віртуалізації: Linux Container (LXC) та KVM (Kernel-based Virtual Machine).

В даному випадку можливо використовувати одразу обидва варіанти, хоча найбільш прийнятним буде саме LXC, так як система, що розглядається, не містить елементів, які обов’язково мають працювати окремо від інших.

Відмінність цих двох варіантів полягає в тому, що фактично KVM є окремою віртуальною машиною із окремим ядром, тоді як LXC є контейнером.

І хоча окрема віртуальна машина може бути корисною в деяких випадках, вона також використовує більше серверних ресурсів, зокрема дискового простору.

Дана платформа може бути встановлена виключно на серверах на базі ОС Linux.

3.3 Встановлення Proxmox Virtual Environment

Для встановлення PVE необхідно підключитись до сервера за допомогою протоколу SSH або за можливості виконувати команди безпосередньо в терміналі сервера.

Для початку необхідно перевірити і за необхідності встановити пакети ovmf (Open Virtual Machine Firmware) та wget. Перший пакет є необхідним для роботи самого PVE, а другий необхідний для завантаження файлів за посиланнями.

Для перевірки і встановлення пакетів виконується команда «apt -y install ovmf wget», де «apt» являється пакетним менеджером у деяких дистрибутивах Linux, таких як Ubuntu, аргумент «-y» автоматично приймає умови встановлення пакетів, так як іноді встановлення вимагає підтвердження з боку користувача,

аргумент «install» вказує на те, що має бути виконана операція встановлення пакету або його оновлення, якщо він вже присутній на машині, останні два аргументи вказують безпосередньо пакети, що необхідно встановити. Інструмент «apt» працює таким чином, що при встановленні нового пакету проводиться перевірка залежностей цього пакету і їх встановлення.

Після встановлення цих двох пакетів пакетом wget необхідно завантажити ISO образ PVE. Для цього виконується команда «wget -O pve.iso https://enterprise.proxmox.com/iso/proxmox-ve_8.4-1.iso», де аргумент «-O» вказує на те, в який локальний файл завантажити файл з серверу, після чого вказується посилання на файл, що треба завантажити, однак варто звертати увагу на цифри, що присутні в посиланні, так як вони являються версією PVE, що буде завантажено, для перевірки версії варто перейти на офіційний сайт Proxmox за посиланням https://www.proxmox.com/en/downloads/proxmox-virtual-environment і знайти номер останньої версії.

Після цього необхідно перевірити, чи підтримує сервер UEFI чи тільки BIOS. Для цього варто виконати команду «efibootmgr», дана команда має вивести повідомлення виду «EFI variables are not supported on this system.» у разі відсутності підтримки UEFI на сервері.

В залежності від наявності чи відсутності підтримки UEFI необхідно виконати одну з двох команд. У разі наявності підтримки: «printf "change vnc password\n%s\n" "abcd_123456" | qemu-system-x86_64 -enable-kvm -bios /usr/share/ovmf/OVMF.fd -cpu host -smp 4 -m 4096 -boot d -cdrom ./pve.iso -drive file=/dev/nvme0n1,format=raw,media=disk,if=virtio -drive file=/dev/nvme1n1,format=raw,media=disk,if=virtio -vnc :0,password -monitor stdio -no-reboot». У разі її відсутності: «printf "change vnc password\n%s\n" "abcd_123456" | qemu-system-x86_64 -enable-kvm -cpu host -smp 4 -m 4096 -boot d -cdrom ./pve.iso -drive file=/dev/nvme0n1,format=raw,media=disk,if=virtio -drive file=/dev/nvme1n1,format=raw,media=disk,if=virtio -vnc :0,password -monitor stdio -no-reboot»

Відмінність цих двох команд полягає тільки в аргументі «-bios /usr/share/ovmf/OVMF.fd», що при використанні перемикає режим роботи віртуальної машини з BIOS на UEFI. Перший аргумент обох команд «printf» посилає команду на зміну пароля для VNC сервера. Аргумент «qemu-system-x86_64» проводить запуск віртуальної машини, «-enable-kvm» вмикає апаратне прискорення для віртуальної машини, «-cpu» host вказує на необхідність використання особливостей процесора, аналогічних до процесора самого сервера. Наступні два аргументи «-smp» і «-m» вказують на кількість ядер і об'єму оперативної пам'яті, що виділяються віртуальній машині. Аргументи «-boot d» та «-cdrom» вказуються для визначення порядку запуску носіїв даних, в даному випадку першим запускається CD-ROM, наступний аргумент містить в собі образ системи, що треба запустити, тобто завантажений раніше файл ISO. Аргумент «-drive file=» визначає які локальні директорії будуть прив'язані до віртуальної машини. Нарешті, аргумент «-vnc :0,password» вказує на пристрій, що виводить зображення з віртуальної машини в консоль VNC і вмикає вимогу введення пароля для отримання доступу, це зроблено в цілях безпеки, а слабкий пароль використовується більше для зручності, так як термінал VNC не надає можливості копіювання тексту з однієї машини на іншу. Останні два аргументи замінюють виведення консолі сервера на консоль віртуальної машини і вказує віртуальній машині на зупинку після вимкнення замість перезавантаження.

Вище було згадано VNC, що розшифровується як Virtual Network Computing, даний інструмент дає змогу керувати машинами віддалено. Схожий за функціоналом інструмент є вбудованим в ОС Windows і має назву RDP (Remote Desktop Protocol), однак він прив'язаний до однієї ОС, коли VNC може працювати на будь-якій системі.

Даний інструмент працює за схемою «клієнт-сервер», де сервером являється машина, яку користувач хоче контролювати. Як клієнтська частина існує безліч програм, таких як RealVNC Viewer.

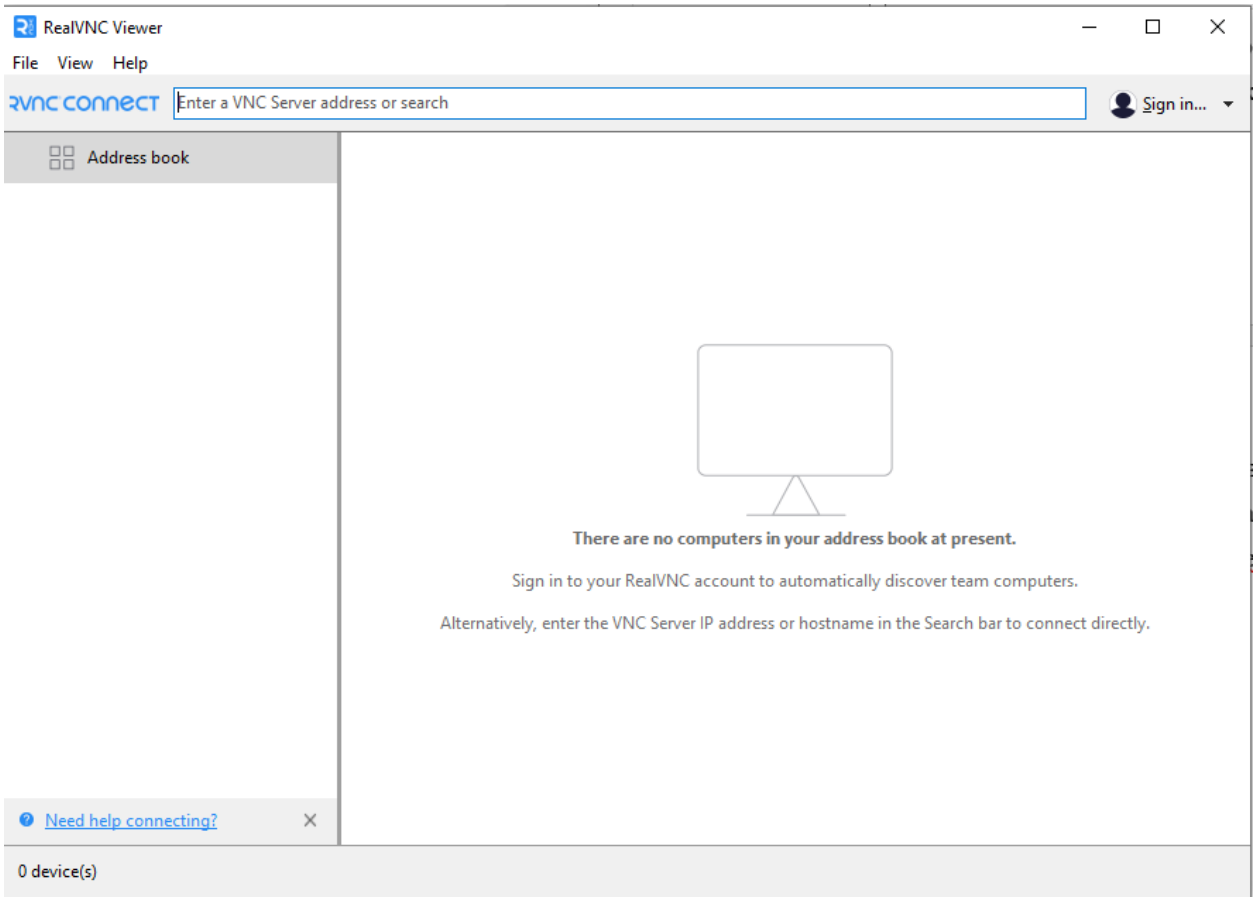

Рисунок 3.2 – Інтерфейс RealVNC Viewer

У верхньому полі необхідно вказати IP адресу серверу та пароль, що було вказано в команді вище. Якщо попередні кроки було виконано вірно, користувач побачить менеджер встановлення PVE.

Рисунок 3.3 – Головна сторінка менеджера встановлення PVE

Для встановлення необхідно натиснути підсвічений варіант – «Install Proxmox VE (Graphical)», так як цей тип встановлення має в собі графічний інтерфейс, без якого навіть досвідченому користувачу буде доволі важко користуватись цією платформою.

В першому ж відкритому розділі необхідно натиснути «Options» і обрати тип файлової системи zfs (RAID1), так як в даному випадку це буде найкращий варіант за всіма параметрами. Також у вкладці «Advanced Options» варто обрати тип стискання lz4.

Рисунок 3.4 – Налаштування файлової системи PVE

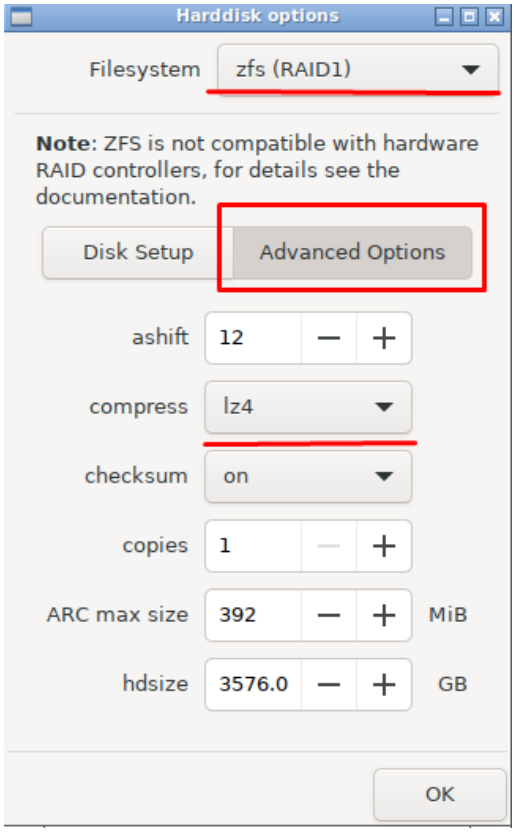

Рисунок 3.5 – Додаткові налаштування файлової системи PVE

Після введення цих даних і натискання кнопки «Next» користувач переходить до налаштування локації і часового поясу, як і вказано в самому менеджері встановлення, вказання країни необхідне для пошуку найближчих серверів з дзеркалами для завантаження оновлень. Варто зазначити, що при виборі країни часовий пояс і розкладка клавіатури буде обрано автоматично з можливістю заміни вручну. Тому в першу чергу вказується саме країна розташування.

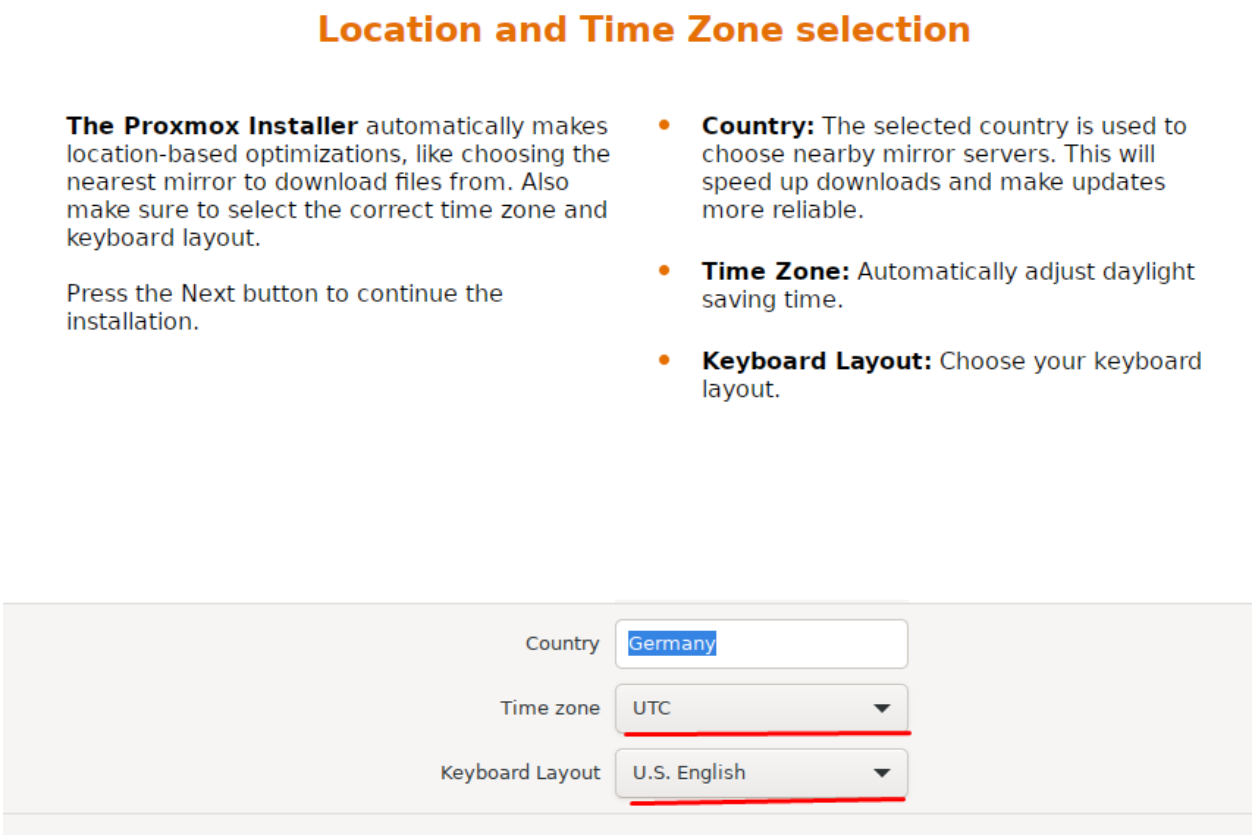

Рисунок 3.6 – Налаштування локації та часового поясу PVE

Наступним кроком необхідно вказати дані для користувача, а саме ім’я, адрес електронної пошти і пароль. Після чого необхідно вказати ім’я хоста, це ні на що не впливає, однак варто вказати зрозумілу назву, наприклад «runners-

server», інші параметри треба залишити без змін, інакше можуть виникнути проблеми з доступом до PVE.

Після цього запускається встановлення PVE, по завершенню якого сеанс у RealVNC Viewer завершиться і знову необхідно перейти до консолі сервера за допомогою SSH або прямим доступом до неї і виконати одну з команд в залежності від наявності підтримки UEFI.

За наявності підтримки: «qemu-system-x86_64 -enable-kvm -bios /usr/share/ovmf/OVMF.fd -cpu host -device e1000,netdev=net0 -netdev user,id=net0,hostfwd=tcp::5555-:22 -smp 4 -m 4096 -drive file=/dev/nvme0n1,format=raw,media=disk,if=virtio -drive file=/dev/nvme1n1,format=raw,media=disk,if=virtio -vnc :0».

За відсутності: «qemu-system-x86_64 -enable-kvm -cpu host -device e1000,netdev=net0 -netdev user,id=net0,hostfwd=tcp::5555-:22 -smp 4 -m 4096 -drive file=/dev/nvme0n1,format=raw,media=disk,if=virtio -drive file=/dev/nvme1n1,format=raw,media=disk,if=virtio -vnc :0»

Дані команди, на відміну від попередніх схожих, не запускають менеджер встановлення PVE, а запускають безпосередньо саму платформу, однак її налаштування ще не завершено. На даному етапі не налаштовано мережу в рамках PVE.

Так як остання введена команда запустила PVE під портом 5555, стало можливим підключення безпосередньо до платформи, для цього до команди для підключення за протоколом SSH до сервера треба додати аргумент «-p 5555».

Після підключення необхідно відкрити файл за шляхом «/etc/network/interfaces» будь-яким зручним редактором тексту, даний файл містить в собі опис мережних інтерфейсів, якими користуватиметься Proxmox. Необхідно налаштувати його так, щоб він мав вигляд, як показано на рисунку 3.7.

```
# network interface settings; autogenerated
# Please do NOT modify this file directly, unless you know what
# you're doing.
#
# If you want to manage parts of the network configuration manually,
# please utilize the 'source' or 'source-directory' directives to do
# so.
# PVE will preserve these directives, but will NOT read its network
# configuration from sourced files, so do not attempt to move any of
# the PVE managed interfaces into external files!
source /etc/network/interfaces.d/*
auto lo
iface lo inet loopback
iface lo inet6 loopback
auto enp7s0
iface enp7s0 inet static
        address <ip_address>/<subnet>
        gateway <gateway_address>
        up route add -net <network_address> netmask <network_mask> gw <gateway> dev enp7s0
iface enp7s0 inet6 static
        address <ipv6_address>/<ipv6_subnet>
        gateway <ipv6_gateway>
iface eth0 inet manual
auto vmbr1
iface vmbr1 inet static
        address <router_ip_address>/<router_subnet>
        bridge-ports none
        bridge-stp off
        bridge-fd 0
# OPNsense WAN
```

Рисунок 3.7 – Вміст файлу налаштування мережних інтерфейсів

В даному випадку на сервері використовується інтерфейс enp7s0, хоча вони можуть відрізнятись від сервера до сервера, єдине, що змінюється в такому випадку – саме назва інтерфейсу, інші налаштування аналогічні. В лінії «address» (на рисунку 3.7 лінія номер 17) вказується IP адреса, за якою можна буде відкрити веб-інтерфейс Proxmox VE, після адреси через символ «/» вказується підмережа цієї адреси. У наступній лінії «gateway» вказується шлюз адреси, тобто адреса з закінченням «.1» (наприклад 168.24.61.1). Наступна лінія додає інтерфейсу статичну маршрутизацію до мережі, що вказується як адреса з закінченням «.0» (наприклад 168.24.61.0), також тут вказується шлюз і пристрій, для якого налаштовується маршрутизація. Наступні три лінії – це налаштування для IPv6, аналогічні налаштуванням IPv4.

Починаючи з лінії 25 проводиться налаштування майбутньої внутрішньої мережі, однак на відміну від налаштування першого інтерфейсу, тут не налаштовується шлюз, маршрутизація і поведінка для IPv6. Однак налаштовуються «мости», а саме вказується, що «міст» не буде прив'язано до фізичного інтерфейсу і що «міст» буде використовуватись для внутрішньої мережі директивою «bridge-ports none». Наступна директива «bridge-stp off» вимикає використання STP, що використовується для попередження появи зациклень в Ethernet мережах. Однак мережа, що створюється, не буде побудована на Ethernet і є локальною, тобто використовуватиме розділення по VLAN, тому даний параметр нам непотрібен. Директива «bridge-fd 0» встановлює затримку в передачі пакетів через «міст» до нуля, що зроблено для підвищення швидкості.

Після введення всіх необхідних даних необхідно зберегти файл і перезапустити PVE командою «reboot now». Варто зазначити, що у разі некоректної конфігурації файлу «interfaces» існує висока ймовірність втрати доступу до PVE як через веб інтерфейс, так і протоколом SSH, що призводить до необхідності провести всі дії цього пункту роботи повторно з самого початку. Іноді це вимагає використання так званої «rescue-системи» сервера.

В разі коректного налаштування мережних інтерфейсів веб-інтерфейс PVE буде доступним за адресою «IP-адреса:8006» (варто перевірити, чи доступний порт 8006 для підключення на сторінці провайдера сервера), після чого PVE запитає дані для входу, що було вказано при встановленні.

Після входу варто відкрити консоль і змінити пароль у разі використання слабкого паролю при встановленні. Для цього треба виконати команду «passwd» і двічі ввести новий пароль. Також варто закрити порт 111 для підключення, так як цей порт використовується Portmapper (Open Network Computing Remote Procedure Call), також відомий як сервіс «rpcbind», що потенційно надає можливість несанкціонованого отримання доступу до списку працюючих на сервері сервісів, що може призвести до отримання злочинцями доступу до

серверу або його даних. PVE відкриває цей порт за замовчуванням, для закриття доступу до цього порту необхідно виконати такі команди: «iptables -A INPUT -p tcp -m tcp --dport 111 -j DROP» для закриття порту за протоколом TCP, «iptables -A INPUT -p udp -m udp --dport 111 -j DROP» для закриття доступу за протоколом UDP, «sudo /sbin/iptables-save» для збереження змін. Також можна повністю вимкнути даний сервіс, якщо користувач впевнений, що він не потрібен, для цього необхідно виконати команди: «sudo systemctl disable --now rpcbind rpcbind.socket» для вимкнення сервісу і сокету для нього, а також для виключення сервісу з автозапуску, «sudo systemctl mask rpcbind» для виключення можливості запуску сервісу в цілому, тобто його неможливо буде запустити ані вручну, ані залежностями.

Також для налаштування внутрішньої мережі необхідно ввімкнути перенаправлення IP, для цього треба скористатись командою «sysctl -w net.ipv4.ip_forward=1», ця команда фактично змінює конфігурацію «sysctl», але тільки до перезапуску машини. Тому варто вручну додати цей параметр до файлу конфігурації «/etc/sysctl.conf», тобто додати «net.ipv4.ip_forward=1» або знайти цей параметр у файлі і розкоментувати його, прибравши символ «#» на початку строки. Якщо просто ввімкнути цей параметр без зміни файлу конфігурації, після перезапуску PVE або сервера в цілому віртуальні машини втратять зв’язок з мережею.

3.4 OPNsense

OPNsense – програмне забезпечення для маршрутизації та налаштування firewall, що було випущено у 2015 році. Фактично дане ПЗ є відгалуженням PFsense. Однак на відміну від попередника OPNsense має низку переваг, такі як відкритість коду, частіші оновлення і зрозуміліший інтерфейс.

В цілому, OPNsense можна назвати маршрутизатором, що може бути встановлено як віртуальну машину на сервері. До функцій входять налаштування

«firewall», налаштування маршрутизації, створення VPN шлюзів, аналіз пакетів, що проходять через OPNsense, налаштування проксі, керування трафіком, налаштування динамічного виділення IP адрес та інші. Окремим функціоналом слугують плагіни, наприклад «Let's Encrypt ACME client» для створення та використання SSL сертифікатів. Як було вказано вище, OPNsense може бути встановлено як віртуальну машину, все через те, що дана програма постачається як окрема система.

3.5 Встановлення OPNsense

Перед встановленням необхідно перевірити, чи встановлено у PVE DNS сервери і які саме, для цього необхідно відкрити веб інтерфейс PVE, відкрити інтерфейс вузла і перевірити його налаштування DNS. Треба перевірити, чи встановлені саме сервери «1.1.1.1» і «8.8.8.8». Перший відноситься до провайдера Cloudflare, другий до Google. Такий вибір обумовлено швидкістю та надійністю в порівнянні з іншими серверами.

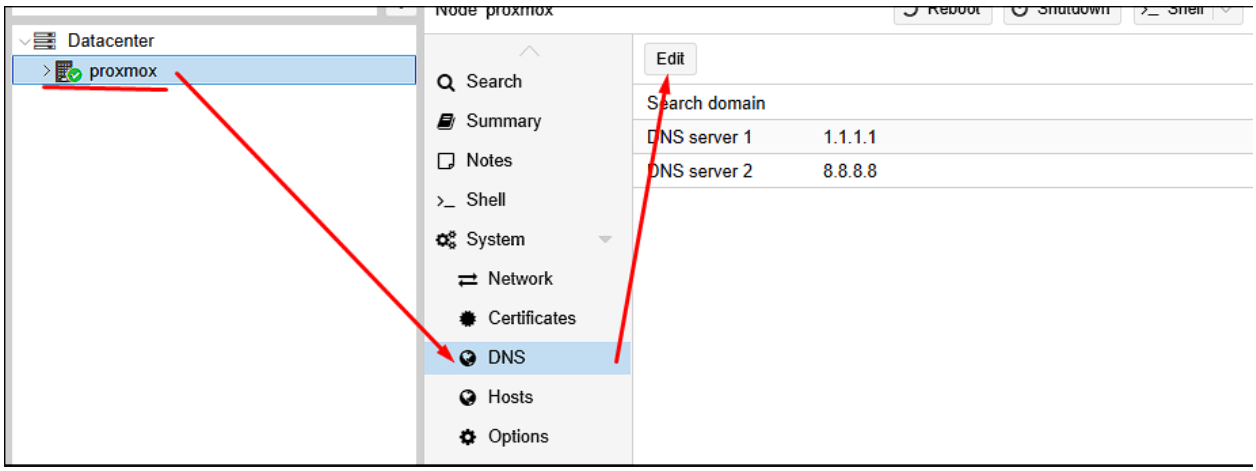

Рисунок 3.8 – DNS сервери, додані до PVE

Після перевірки або додавання DNS серверів необхідно завантажити ISO образ OPNsense, це можна зробити двома шляхами: завантажити образ спочатку

на локальний ПК і завантажити його в PVE, або завантажити його напряму до PVE. Другий спосіб є набагато легшим і швидшим, але не завжди є можливість його використовувати.

ISO образ OPNsense можна знайти на офіційному сайті за посиланням: https://opnsense.org/download/ . Однак необхідно врахувати те, що дане посилання відправляє користувача на сторінку завантаження, а не містить необхідний файл, спочатку на цій сторінці необхідно обрати архітектуру системи, в даному випадку це «amd64» (варто зазначити, що інших варіантів немає на даний момент), після цього необхідно вибрати тип образу, для встановлення на віртуальній машині потрібен саме «dvd» тип. Після вибору цих двох параметрів треба натиснути на кнопку «Download» лівою кнопкою миші і дочекатись завантаження. Для завантаження образу одразу в PVE необхідно натиснути правою кнопкою миші по тій самій кнопці і натиснути «Скопіювати посилання», після чого в веб інтерфейсі PVE перейти до вкладки сховища даних, зазвичай воно має назву «local» (також має бути сховище з назвою «local-<тип файлової системи>», однак воно використовується для віртуальних машин та контейнерів), далі перейти у вкладку «ISO Images» та натиснути кнопку «Download from URL» для прямого завантаження образу в PVE або «Upload» для завантаження з локального комп'ютера.

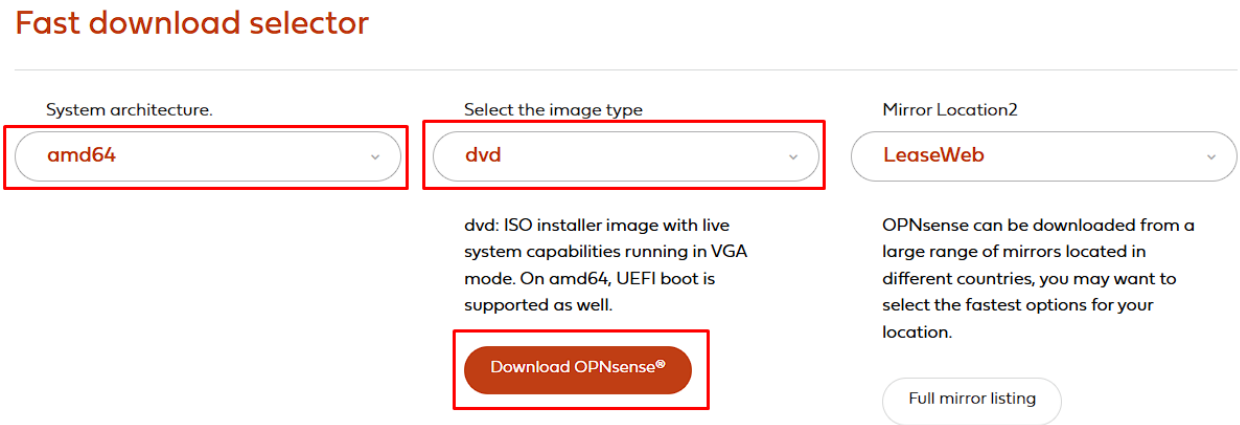

Рисунок 3.9 – Завантаження ISO образу OPNsense

Після завершення завантаження необхідно створити віртуальну машину, на якій буде встановлено OPNsense, для цього в веб інтерфейсі PVE у верхньому правому кутку необхідно натиснути «Create VM». Створення віртуальної машини включає в себе налаштування назви та ідентифікаційного номера, вибору ISO образу, з якого запускати машину, налаштування фізичних параметрів, таких як об’єм диску, оперативної пам’яті, кількості процесорів, а також налаштування мережі.

Ідентифікаційний номер встановлюється автоматично з можливістю його зміни, можна залишати стандартний номер, так як в даний момент PVE не містить інших машин і розділення груп машин по номерам нам не потрібне.

Назву варто обирати зрозумілу і таку, яка відображає функціонал машини, в даному випадку варто назвати машину «OPNsense».

Як ISO образ треба обрати завантажений образ, у вкладці «System» треба відмітити використання «Qemu Agent», дана функція надає користувачу доступ до багатьох функцій віртуальних машин, таких як створення знімків (snapshotting), призначення машинам IP-адрес і так далі.

Для OPNsense оптимальним об’ємом диску є 32 гігабайти, однак при довшому зберіганні логів системи або великих об’ємах пакетів, що проходитимуть через OPNsense, це значення має бути збільшено. Можливість зміни цих параметрів передбачена в налаштуваннях кожної віртуальної машини PVE.

Для даного випадку достатньо буде виділити 2 ядра і 2 гігабайти оперативної пам’яті, однак за умов, вказаних вище, цей об’єм може бути необхідно збільшити.

У налаштуваннях мережі треба обрати міст «vmbr1», вимкнути «firewall», встановлення саме цього мосту треба для можливості OPNsense спілкуватись з PVE, а вимикаємо «firewall» тому, що вбудований firewall PVE не налаштовано,

що нам і не треба, так як всі правила маршрутизації буде налаштовано саме в OPNsense.

Після створення віртуальної машини необхідно перейти до налаштувань її обладнання, це вкладка «Hardware», і в ній необхідно додати ще один мережний пристрій, який також не використовуватиме «firewall», але буде використовувати міст «vmbr2», цей міст в даному випадку є внутрішньою мережею, в якій працюватимуть всі віртуальні машини. Після цього необхідно запустити віртуальну машину і дочекатись запиту на вхід.

Для запуску інсталятора необхідно ввести логін «installer» і пароль «opnsense», після чого користувача відправить в менеджер встановлення. Першим кроком буде вибір розкладки, зазвичай користувач має залишити стандартну англійську, після чого обрати «Install (UFS)», обрати диск, який було додано при створенні, тобто диск з об’ємом 32 гігабайти, після чого буде запропоновано очистити диск, необхідно погодитись, після чого треба обрати опцію «Complete install» і дочекатись встановлення.

Після перезапуску машини необхідно ввійти в систему як інший користувач, логін «root», пароль «opnsense», тепер необхідно налаштувати мережні інтерфейси, для цього необхідно ввести цифру 1, так як ця опція відповідає за призначення інтерфейсів.

Далі необхідно пропустити перші два запити (двічі натиснути Enter на клавіатурі), після чого ввести «vtnet0», ввести «vtnet1», пропустити наступний запит і ввести «y», після цього треба дочекатись кінця налаштування інтерфейсів і обрати опцію 6 для перезапуску машини.

3.6 Налаштування OPNsense

Після налаштування інтерфейсів через консоль необхідно відкрити веб інтерфейс OPNsense для продовження налаштування. Так як машина прив’язана до внутрішньої мережі і поки не отримала зовнішню адресу, необхідно створити

віртуальну машину з графічним інтерфейсом для підключення до веб інтерфейсу OPNsense.

Задля збереження ресурсів сервера краще обрати невимогливу і малу за об’ємом систему, наприклад дистрибутив Linux Mint. Його вибір обґрунтовано малою вагою, мінімальними вимогами до фізичних компонентів і можливості запуску системи без її встановлення. Даний дистрибутив можна завантажити з офіційного сайту: https://linuxmint.com/edition.php?id=319.

З цим дистрибутивом необхідно створити віртуальну машину з параметрами, ідентичними до машини OPNsense. Єдина відмінність полягає в прив’язаних мережних інтерфейсах: Mint та будь-які інші віртуальні машини, окрім OPNsense повинні мати лише один мережний інтерфейс «vmbr2», так як він створений саме для локальної мережі. Однак у випадках, коли необхідно провести розділення інфраструктури на кілька елементів, можна створити декілька мережних інтерфейсів, аналогічних за налаштуваннями до «vmbr2» і створити на їх основі окремі віртуальні машини або частини, що керуються особистими OPNsense.

Після створення віртуальної машини необхідно її запустити і відкрити браузер, в пошуковій стрічці вписати локальну адресу машини з OPNsense, для цього треба повернутись в її консоль, авторизуватись, якщо це не було виконано і знайти строку «LAN», в якій і буде вказано локальну IP адресу.

При першому запуску веб інтерфейсу відкривається майстер встановлення, першою сторінкою буде «General Information», тут вказується ім’я хоста, яке можна залишити стандартним, доменне ім’я, якщо планується додавання SSL і підтримки повноцінного HTTPS, обирається мова інтерфейсу, DNS сервери, які варто встановити відповідно до встановлених в PVE, а також налаштування DNS (рис. 3.10).

«Override DNS» дає дозвіл на використання DNS сервера провайдера інтернет послуг замість вказаних в налаштуваннях OPNsense. Це налаштування дає змогу підключатись до мережі Інтернет без обмежень. В даному випадку це

налаштування варто ввімкнути, так як немає підстав сумніватись в DNS сервері провайдера і немає необхідності в повному контролі серверів, що використовуються в мережі.

«Enable Resolver» відповідає за ввімкнення вбудованого резолвера OPNsense, його варто вмикати у разі необхідності збільшення швидкості запитів і відповідей всередині мережі, а також для підвищення приватності, так як запити проходять лише всередині мережі, виключаючи посередництво провайдера. В даному випадку дана опція є корисною, тому її варто ввімкнути.

«Enable DNSSEC Support» відповідає за ввімкнення захисту DNS, що включає в себе перевірку криптографічних підписів DNS серверів, а також попередження атак типу «спуфінг» і «людина посередині». Тобто ця опція має бути ввімкнена, якщо користувач хоче підвищити безпеку мережі.

«Harden DNSSEC data» являється налаштуванням додаткового рівня захисту попередньої опції. Це включає відхилення будь-яких запитів з некоректними підписами і протидії атакам, спрямованих на послаблення DNSSEC. Дане налаштування треба ввімкнути якщо є необхідність забезпечення максимального захисту мережі.

Після введення всіх необхідних даних та опцій необхідно натиснути «Next» для продовження.

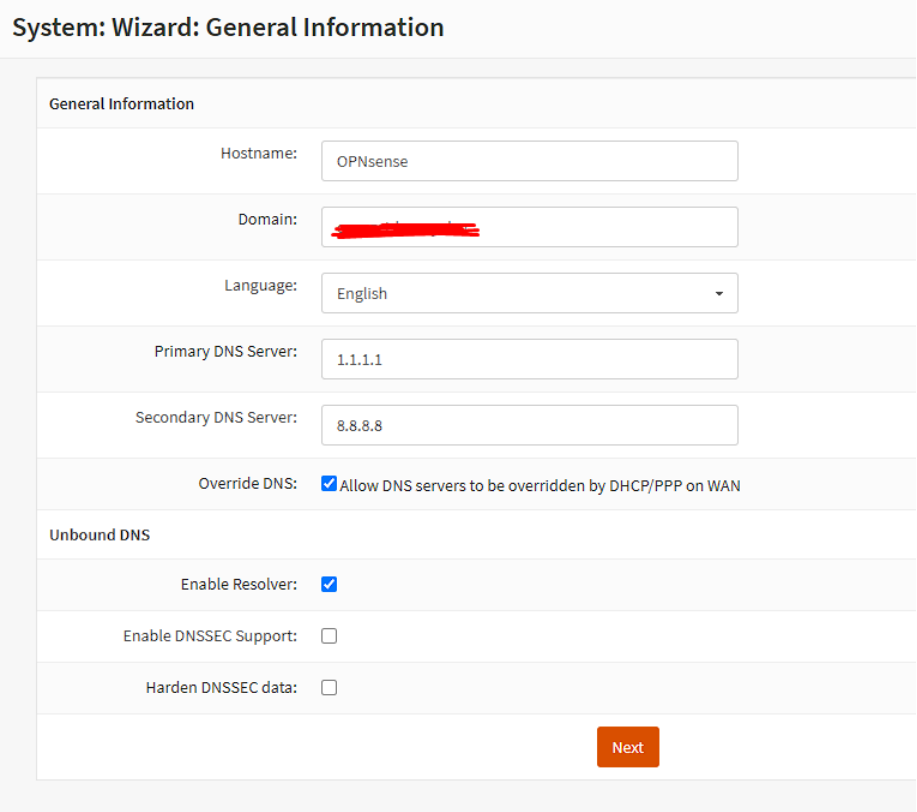

Рисунок 3.10 – Загальні налаштування OPNsense

Наступною в налаштуванні є сторінка «Time Server Information», на якій вказується часовий сервер та часовий пояс, ці опції можна залишити без змін і перейти до наступної сторінки – «Configure WAN interface».

Першою опцією є тип конфігурації IPv4, ця опція визначає, яким чином буде призначено зовнішню IP адресу для OPNsense, серед варіантів є «Static», «DHCP», «PPPoE» і «PPTP». В даному випадку, як і в більшості інших, найкращим варіантом буде саме «Static» через необхідність вказати адресу вручну і не повертатись до її налаштування.

Наступною опцією є налаштування MAC адреси, що є необов'язковим. Однак у деяких випадках вказання цієї адреси є вимогою провайдера.

«MTU» контролює максимально допустимий розмір пакету без його фрагментації, як вказано в описі налаштування, за замовчуванням (якщо

залишити строку пустою), для PPPoE це 1492 байти, для інших підключень – 1500. В більшості випадків можна залишити значення за замовчуванням.

«MSS» відповідає за максимально допустимий розмір пакетів, що передаються протоколом TCP/IP, як вказано в описі, в більшості випадків значення має бути таким самим, як і в попередній опції. Це значення треба збільшувати у випадках, коли сторінки не завантажуються повністю або взагалі не завантажуються і у випадках несподіваних розривів підключень VPN.

В полі «IP Address» вказується безпосередньо адреса, яку користувач бажає призначити OPNsense, однак також варто звернути увагу на поле «Subnet mask», при неправильному вказанні якого існує можливість відмови в підключенні. Зазвичай провайдери пишуть номер маски підмережі поруч з адресою в кабінеті користувача.

Поле «Upstream Gateway» визначає шлюз мережі, в даному випадку це IP адреса сервера та, відповідно, адреса PVE без порта.

Окремими налаштуваннями є налаштування блокування певних мереж. Так, параметр «Block RFC1918 Private Networks» блокує підключення з усіх приватних мереж, включаючи мережі «broadcast», «loopback», локальні мережі 192.168.x.x. В більшості випадків ця опція має бути ввімкнута.

«Block logon networks» відповідає за блокування підключень з IP адрес, які занесено до списку адрес, що використовувались для кібератак або адрес, які не призначено адміністрацією адресного простору Інтернет (IANA). Як і попередня опція, має бути ввімкнута.

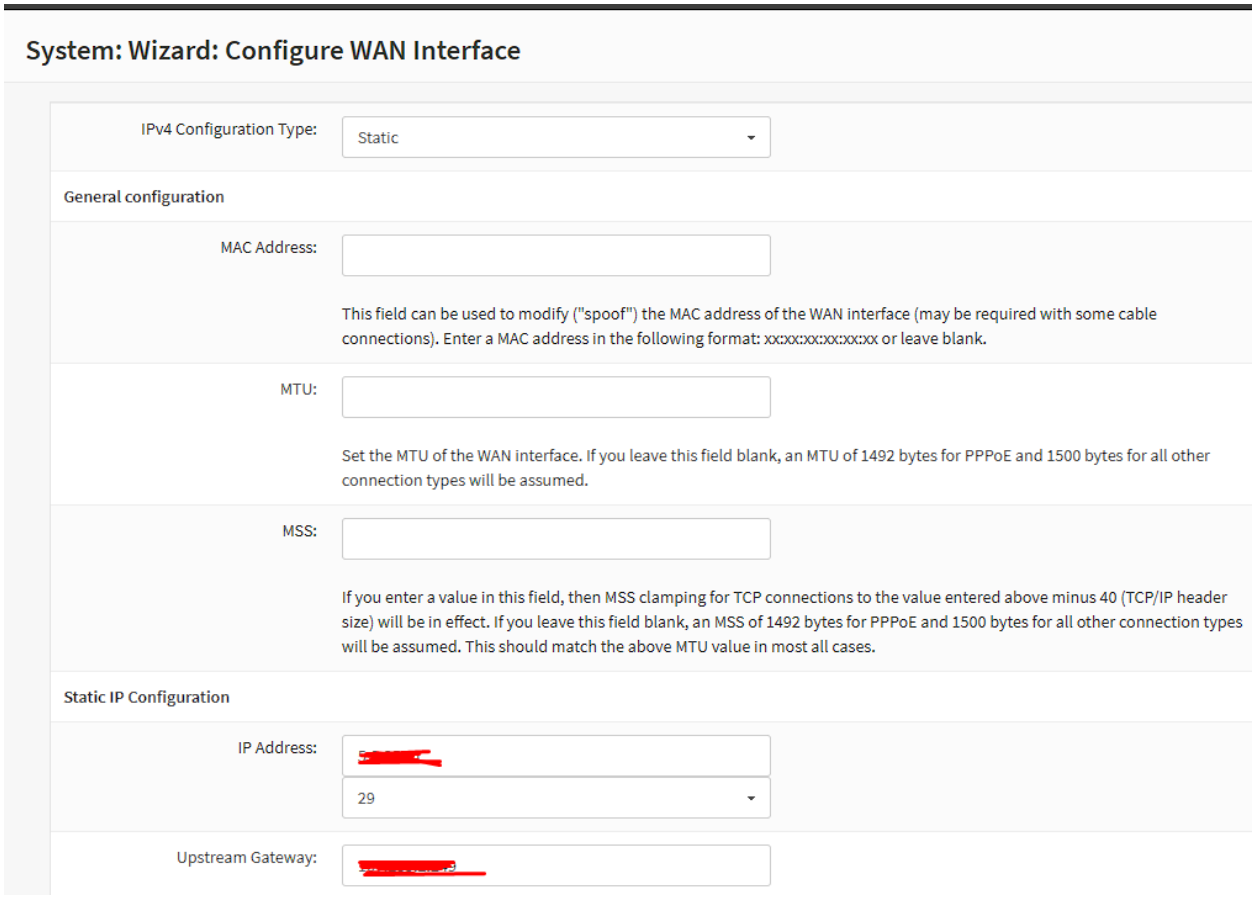

Рисунок 3.11– Налаштування WAN інтерфейсу

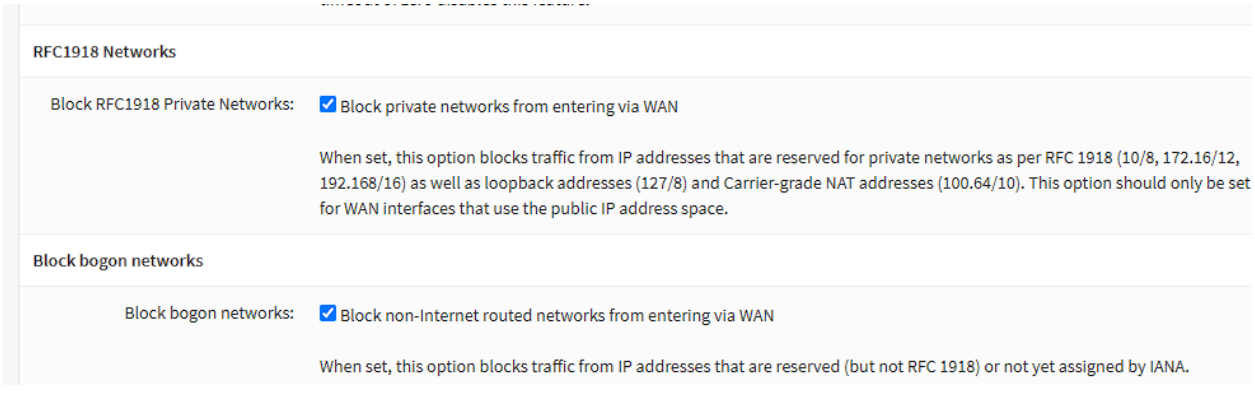

Рисунок 3.12 – Налаштування WAN інтерфейсу

Наступним кроком є налаштування локального інтерфейсу на сторінці «Configure LAN Interface», де вказується лише внутрішня IP адреса OPNsense і її маска підмережі, можна залишити значення за замовчуванням.

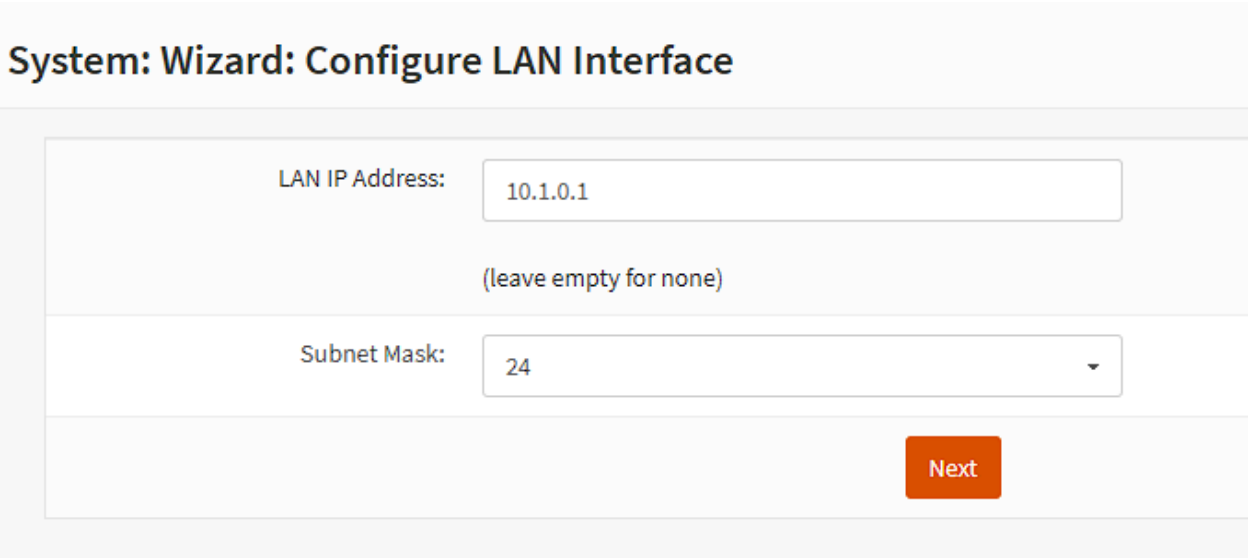

Рисунок 3.13 – Налаштування LAN інтерфейсу

Наступним кроком є зміна паролю для авторизації в веб інтерфейсі, так як стандартний пароль занадто слабкий, цей крок необхідний.

Після цього буде запропоновано перезапустити OPNsense, з чим треба погодитись і дочекатись перезавантаження. Наступним етапом налаштування OPNsense буде створення та конфігурування віртуальної локальної мережі (VLAN).

Для початку в веб інтерфейсі необхідно натиснути «Firewall», потім «Rules», потім «WAN», це відкриє сторінку налаштування правил фаєрволу для глобальної мережі. Тут необхідно натиснути на кнопку створення нового правила (символ «+» на помаранчевому фоні).

У полі «Action» необхідно обрати «Pass», так як інші варіанти («Block» та «Reject») блокують можливість відправки пакетів до вказаного в наступних полях інтерфейсу. Різниця між «Block» і «Reject» полягає в тому, як програма

поводитиметься з пакетами, в першому випадку відправлений пакет не повертається до відправника.

Поле «Disabled» відповідає за те, чи буде правило виконуватись, чи ні, це корисно у випадках, коли певний функціонал тестується і немає необхідності видаляти правило.

Поле «Quick» визначає поведінку програми щодо пакетів відносно правил, що підходять цьому пакету. Наприклад, якщо є кілька правил, що стосуються інтерфейсу глобальної мережі, то при ввімкнені даної опції виконається лише перше з них, при вимкненій опції навпаки, виконається останнє правило.

Дана опція корисна, коли є специфічні умови для певних дій з пакетами, наприклад відмова в підключенні до певних портів або адрес.

Поле «Interface» визначає мережний інтерфейс, якого стосується правило, в даному випадку треба обрати інтерфейс «WAN».

В полі «Direction» обирається напрямлення пакетів, якого стосується правило, є лише два варіанти: всередину («in») і назовні («out»). В даному випадку обрали перший варіант.

В полі «TCP/IP Version» обирається версія IP, якої стосується правило, серед варіантів лише версії 4 та 6 або обидві, в даному випадку обирається перший варіант, так як використання шостої версії не передбачено і не є необхідним.

«Protocol» визначає, якого протоколу стосується правило, в переліку варіантів представлено велику кількість протоколів, але в даному випадку обирається «TCP», так як немає необхідності у виборі будь-якого іншого протоколу.

Опції «Source / Invert» і «Destination / Invert» виконують функцію інвертування відповідників, тобто замість виконання правила для вказаного джерела або призначення пакета, правило виконуватиметься для всіх, окрім вказаних. В даному випадку це не потрібно.

В полях «Source» і «Destination» вказуються джерело та місце призначення пакета відповідно, до яких застосовуватиметься правило, в даному випадку як джерело вказується «any», тобто будь-яке, а точка призначення – адреса глобальної мережі («WAN address»), це необхідно для того, щоб була можливість підключення до цих адрес ззовні.

У цих полях також є опція «Advanced», при натисканні на яку розгортається меню «Source/Destination port range» в залежності від того, де саме було натиснуто цю кнопку (в полі «Source» або «Destination»). В цих меню можна вказати спектр портів, до яких застосовуватиметься правило, передбачено як конкретні значення, такі як «HTTP», «HTTPS», «SSH», так і варіант «other», що дає можливість вказувати свої значення. В даному випадку вказуються порти 80 (HTTP) та 443 (HTTPS) для точки призначення пакетів.

Наступною опцією є «Log», що відповідає за ввімкнення логування пакетів, що відповідають цьому правилу. Наступні опції, такі як «Category» та «Description», використовуються суто для розрізнення правил між собою за категорією або описом.

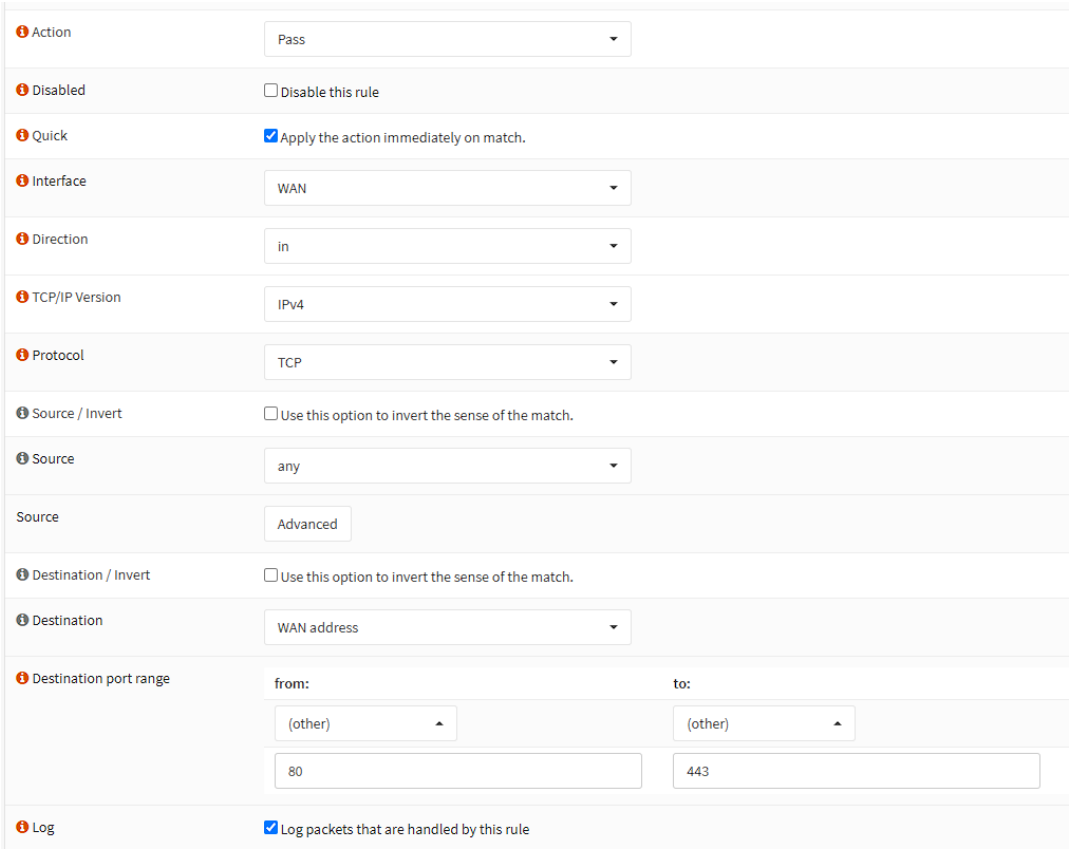

Рисунок 3.14– Налаштування правила для інтерфейсу глобальної мережі

Аналогічним чином треба налаштували правило для прийняття ICMP трафіку, тобто в полі вибору протоколу необхідно обрати саме «ICMP», після чого під цим полем з’явиться ще одне – «ICMP type», де треба обрати «Echo Request», це робиться для того, щоб можна було перевіряти доступність машин утилітами, такими як «ping».

Також у полі «Destination» замість «WAN address» треба обрати «This Firewall» і не обирати певний перелік портів, тобто залишити значення «any».

Після створення та перевірки правильності усіх введених параметрів, необхідно перейти до створення віртуальної локальної мережі, для цього в лівій панелі OPNsense розгортається вкладка «Interfaces», потім «Other Types», де обирається «VLAN». У відкритій сторінці необхідно створити нову мережу, натиснувши символ «+» у правому боці.

В меню налаштувань необхідно вказати назву пристрою у полі «Device», можна залишити пустим для генерації випадкової назви, а при присвоєнні своєї назви варто враховувати, що ім’я пристрою може мати лише цифри, букви і точки, наприклад «vlan0.101».

В полі «Parent» обирається «батьківський» пристрій, до якого буде прив’язано віртуально мережу, в даному випадку треба обрати пристрій з позначкою «LAN», тобто пристрій «vtnet1».

Поле «VLAN tag» вказує на номер віртуальної мережі, він має бути унікальним в межах OPNsense і повинен мати номер від 1 до 4094. Вказується номер 101. За цим номером буде призначатись мережа для віртуальних машин всередині PVE.

«VLAN priority» в даному випадку залишається 0, так як трафік в мережі не обмежуватиметься одним типом даних, що передається в мережі. Однак якщо в мережі переважатиме, наприклад, VoIP зв’язок, варто встановити цей параметр на 5 (Voice).

Після додавання за бажанням опису треба зберегти зміни і перевірити правильність введених даних, після чого необхідно присвоїти новий інтерфейс, для цього в лівій панелі OPNsense у вкладці «Interfaces» необхідно перейти в «Assignments», де в нижній частині екрану знаходяться всі непризначені інтерфейси, для його присвоєння достатньо натиснути на нього, додати опис і натиснути «Save».

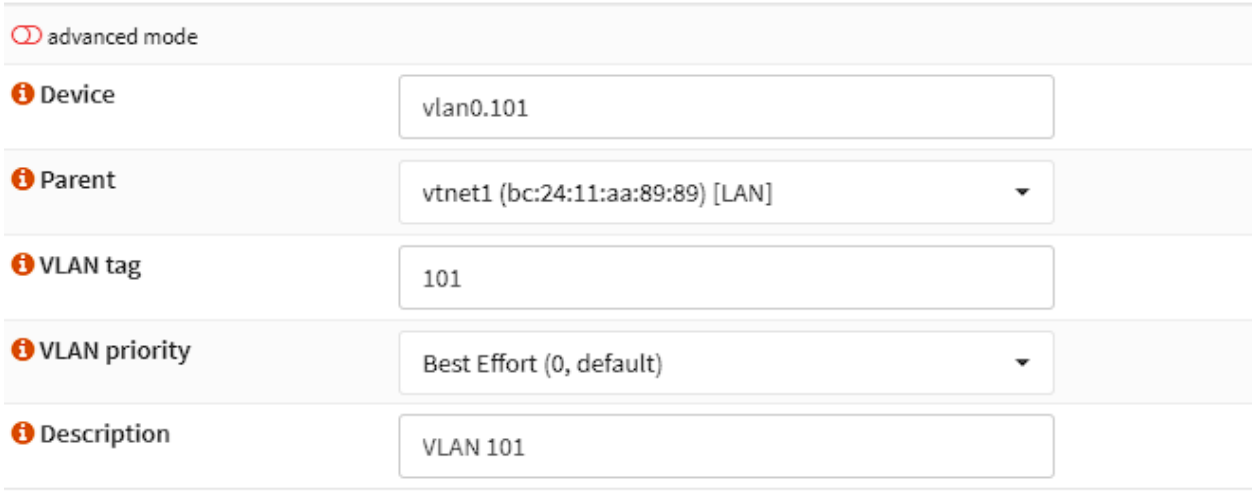

Рисунок 3.15 – Параметри VLAN

Наступним кроком є налаштування діапазону IP адрес, що можуть бути призначені для віртуальних машин. Для цього у вкладці «Interfaces» необхідно натиснути на тільки що створений інтерфейс, це відкриє меню налаштування цієї VLAN, перше, на що треба звернути увагу – чи відмічений параметр «Enable», так як він визначає, чи працюватиме інтерфейс, опціонально можна захистити інтерфейс від видалення, ввімкнувши параметр «Lock».

Основна частина налаштування знаходиться в блоці «Generic configuration», налаштування ідентичні налаштуванням інтерфейсу WAN, однак в даному випадку не треба вмикати такі параметри, як «Block private networks» і «Block logon networks», так як фактично ми створюємо саме приватну мережу в діапазоні «broadcast» адрес (10.x.x.x). Параметри «IPv4/v6 Configuration Type» обираються «Static IPv4» та «None» відповідно. Такий вибір обумовлено тим, що шоста версія в даній мережі непотрібна через малі масштаби інфраструктури, тому діапазону адрес четвертої версії буде достатньо. Наступні параметри цього блоку пропускаються.

В блоці «Static IPv4 configuration» необхідно вказати діапазон адрес і маску підмережі, в даному випадку обрано діапазон 10.1.101.1 і маску 24, що дозволяє мати 254 унікальні адреси всередині мережі, такий вибір дозволить не

перейматись щодо малої кількості доступних адрес для віртуальних машин, хоча в даному випадку буде достатньо і маски 26, що має 60 доступних адрес.

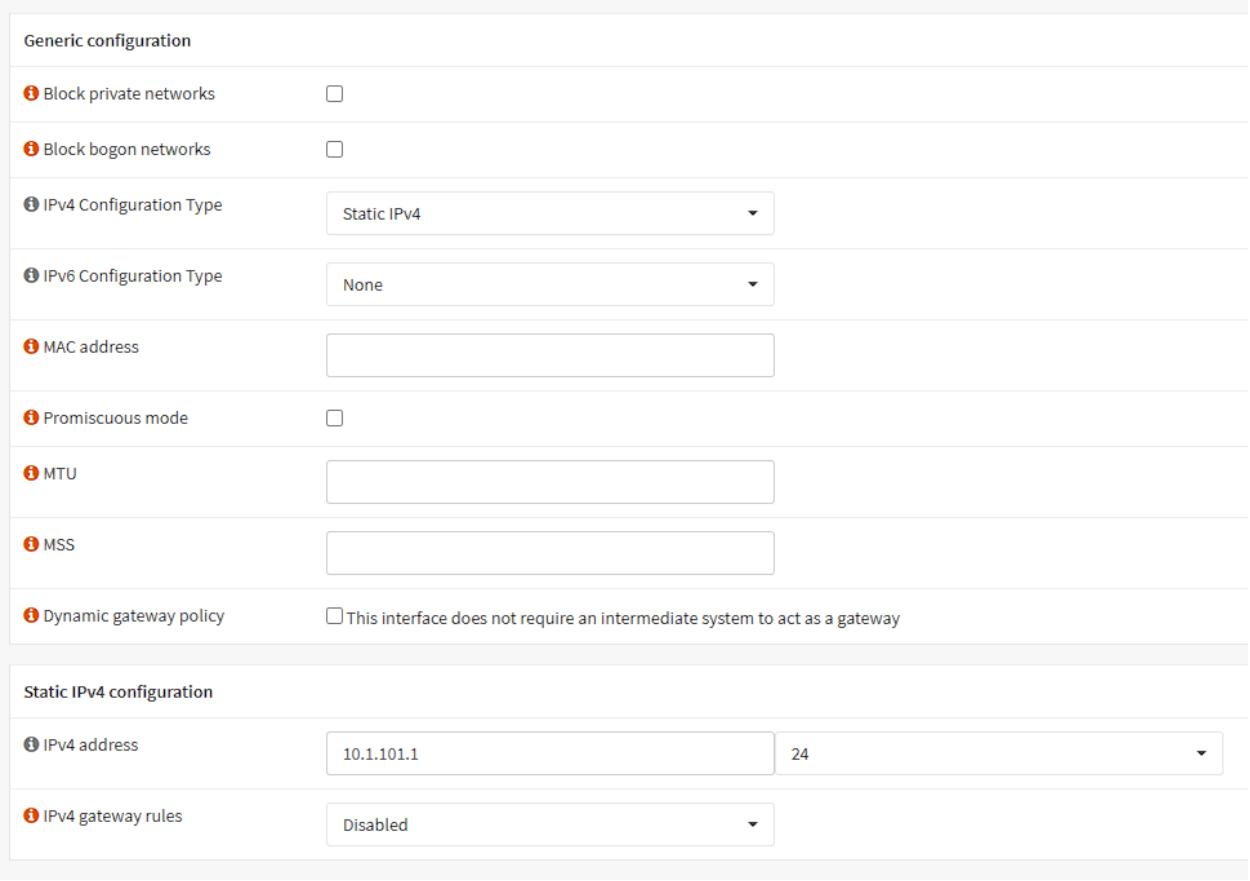

Рисунок 3.16 – Налаштування діапазону IP адрес для внутрішньої мережі

Наступним кроком є налаштування DHCP, для цього необхідно у вкладці «Services» натиснути «ISC DHCPv4», де обрати створений інтерфейс. Це налаштування виконується для автоматичного присвоєння адрес новим віртуальним машинам в мережі.

В першу чергу треба ввімкнути DHCP для цього інтерфейсу, для цього необхідно відмітити параметр «Enable», параметр «Deny unknown clients» залишається вимкненим, так як не вказуються MAC адреси для машин. Параметр «Ignore Client UIDs» визначає, чи буде використовуватись UID машини для присвоєння адреси чи присвоєння проводитиметься лише за MAC адресами. В

деяких випадках, включаючи даний, це необов'язково, так як відомо, що діапазон адрес буде важко вичерпати.

Поля «Subnet», «Subnet mask» і «Available range» містять в собі інформацію про мережу, а саме підмережу, її маску і доступний діапазон IP адрес.

У полі «Range» треба ввести діапазон адрес, з якого адреси будуть надаватись автоматично, вказується діапазон 10.1.101.100 – 10.1.101.254. Таким чином мережа матиме 98 резервних адрес на випадок, коли машині треба надати статичну адресу.

Поле «DNS servers» обирається відповідно до такого ж параметру в OPNsense.

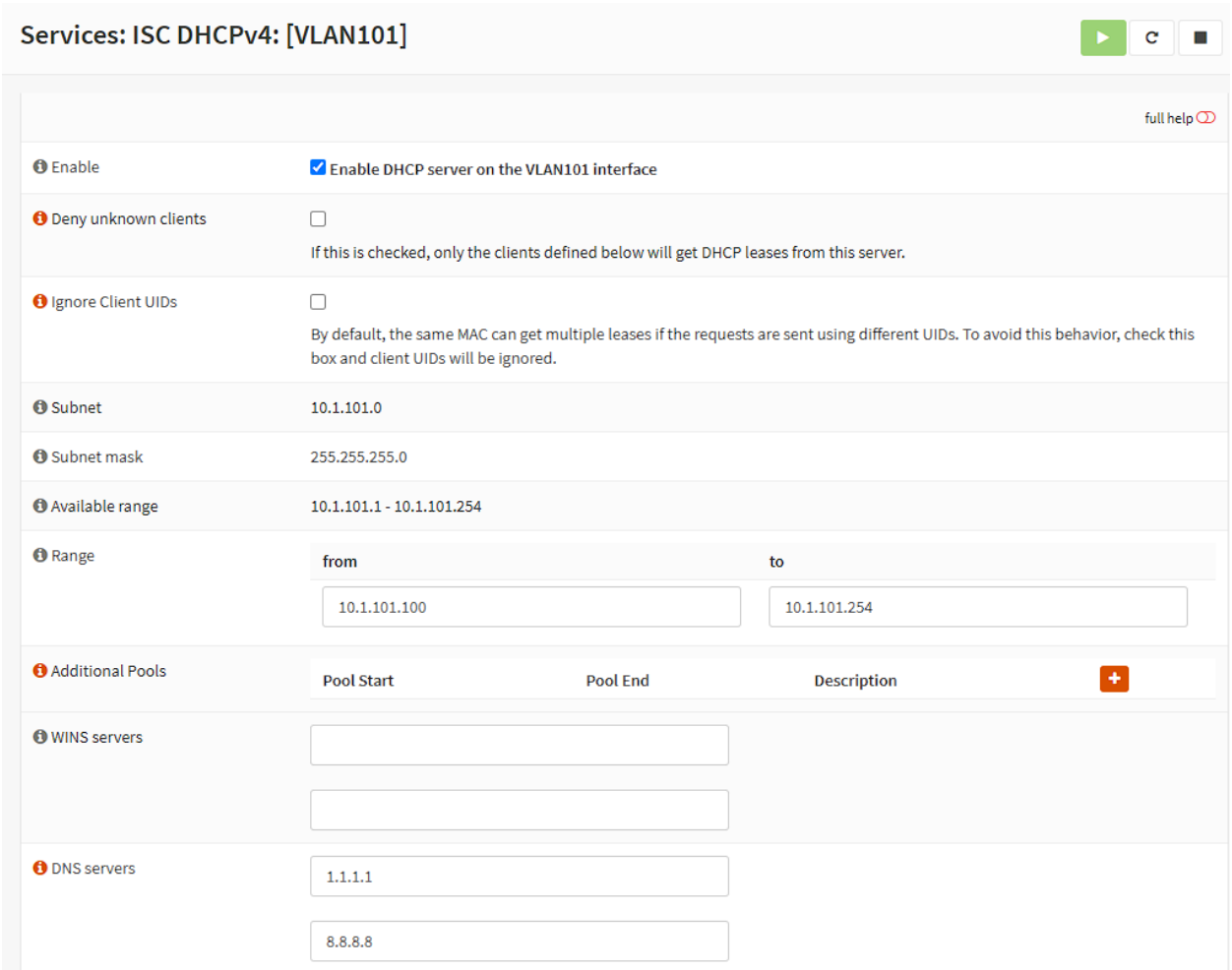

Рисунок 3.17 – Налаштування DHCP

Наступним кроком є налаштування правила для трафіку HTTP/HTTPS, для цього необхідно повернутись до вкладки «Firewall», де обрати «Rules» і натиснути «WAN», тут створюється нове правило з параметрами, вказаними на рисунку 3.18.

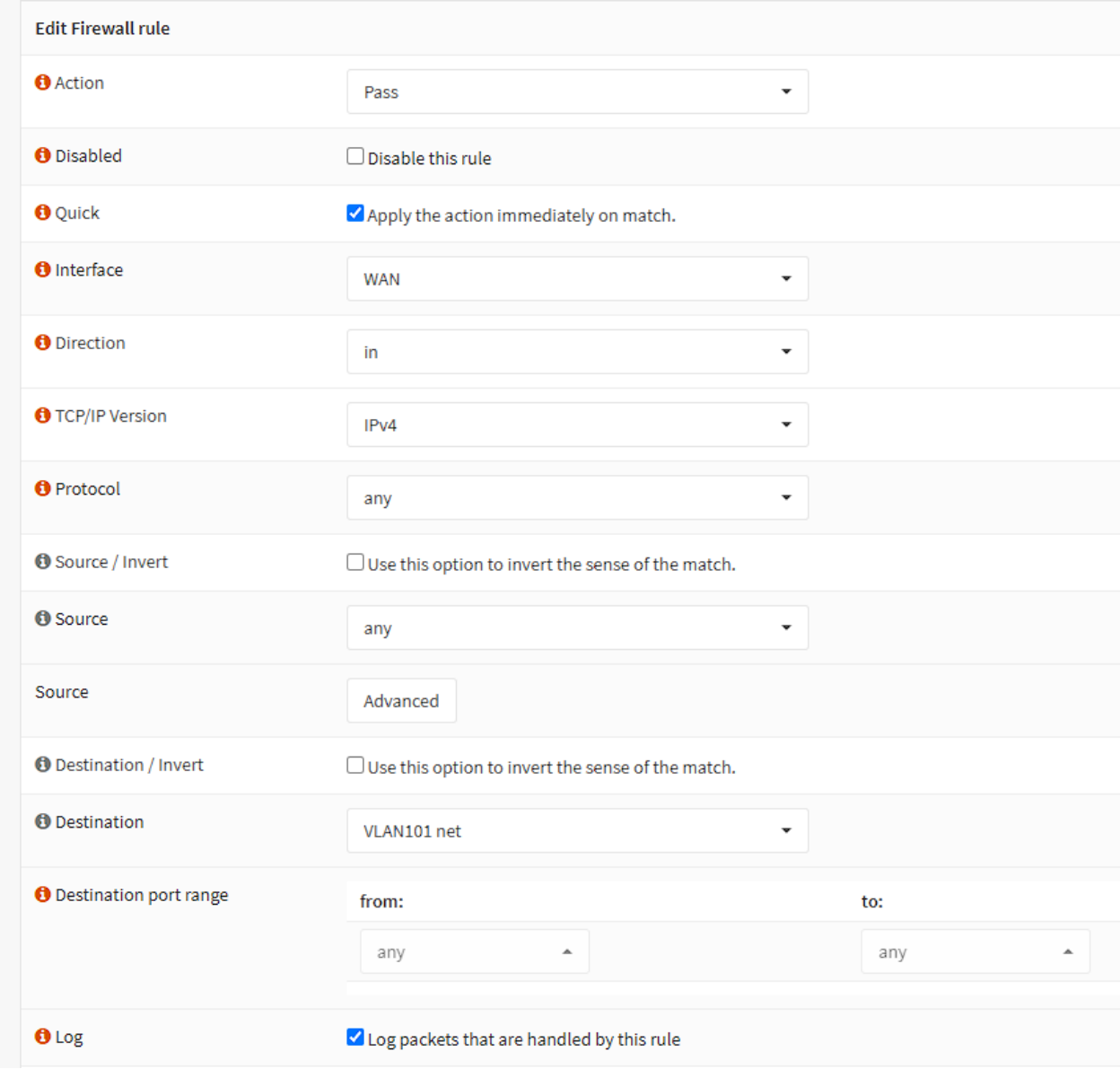

Рисунок 3.18 – Налаштування трафіку HTTP/HTTPS

Далі необхідно налаштувати NAT для вихідних підключень, тобто щоб внутрішні адреси мали доступ до глобальної мережі, для цього треба перейти до вкладки «Firewall:NAT», де обрати «Outbound», треба обрати гібридну генерацію правил «Hybrid outbound NAT rule generation». Це необхідно для того, щоб правила, автоматично згенеровані OPNsense, могли працювати разом зі створеними користувачем. Такий підхід гарантує коректність роботи NAT і можливість ручного налаштування.

Після цього треба створити нове правило з параметрами, вказаними на рисунку 3.19.

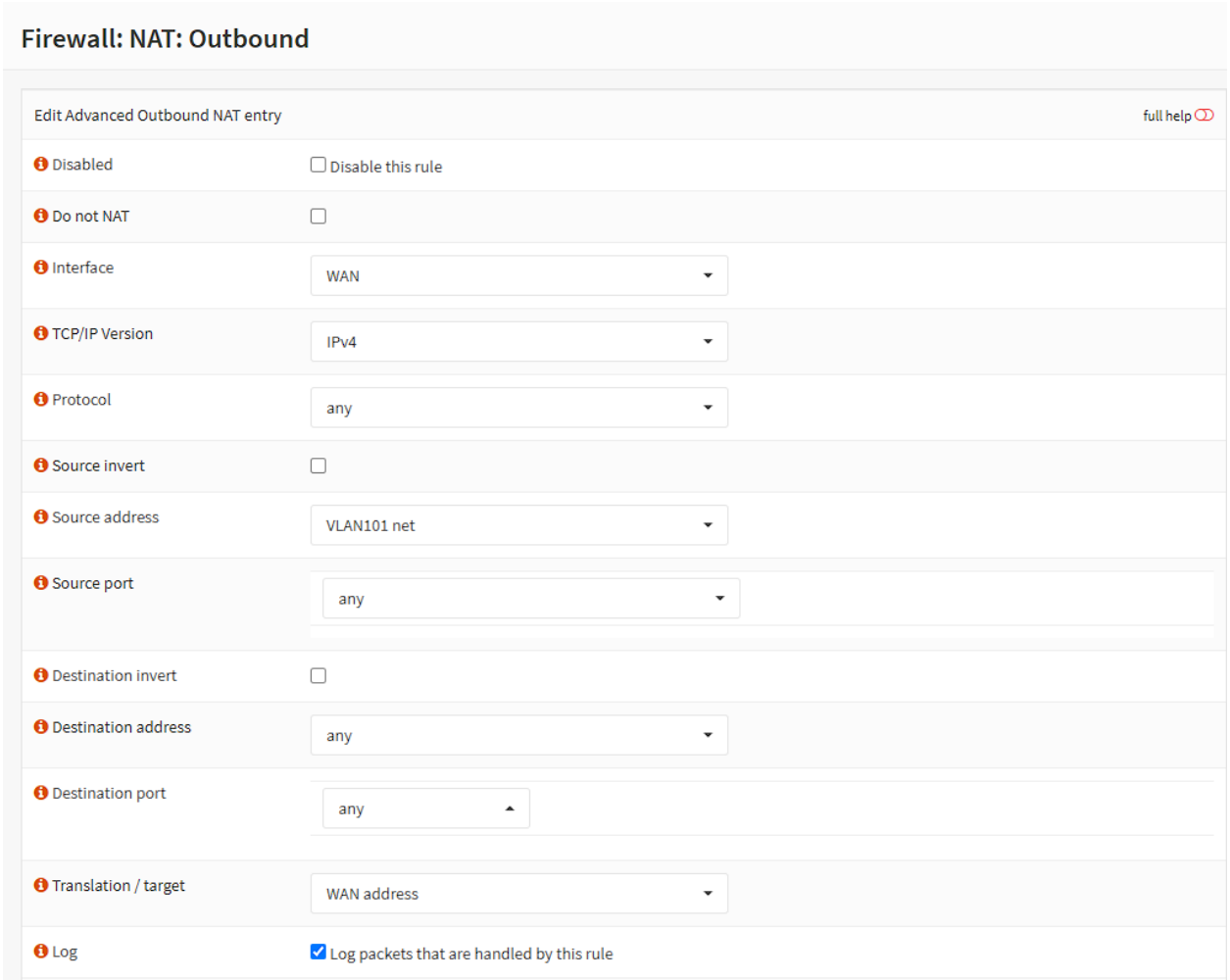

Рисунок 3.19 – Налаштування вихідних підключень NAT

Далі треба налаштувати фаєрвол для створеного раніше інтерфейсу, для цього треба перейти до «Firewall:Rules», де обрати створений інтерфейс і додати нове правило з параметрами, що вказано на рисунку 3.20.

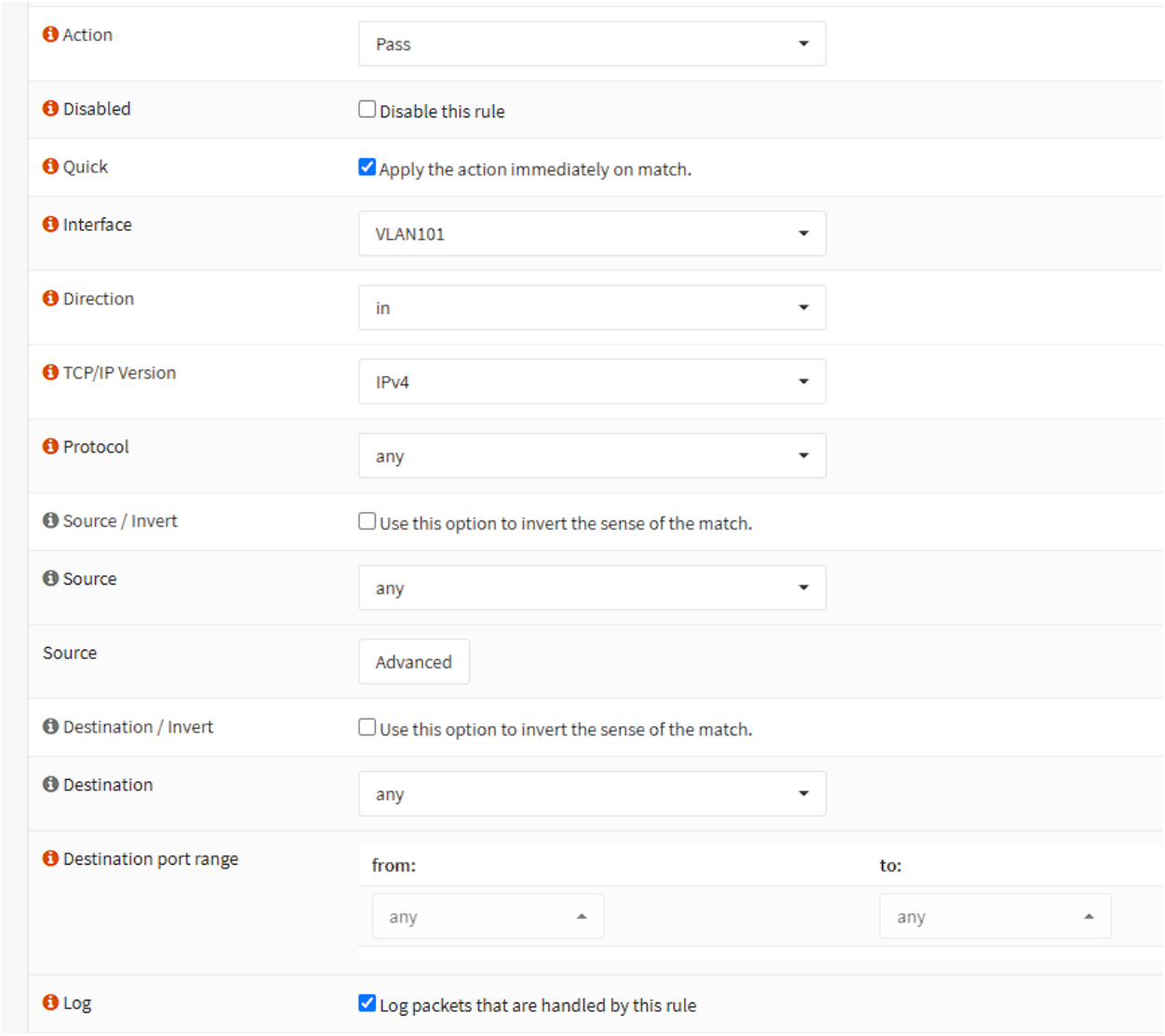

Рисунок 3.20 – Налаштування фаєрволу для внутрішньої мережі

Також треба додати адреси, що будуть використовуватись в NAT, для цього треба перейти у вкладку «Interfaces», «Virtual IPs», де обрати «Settings», тут необхідно додати адреси, якими користувач володіє, для подальшого їх присвоєння до VM. Додати треба адреси в режимі «Proxy ARP», так як цей режим

підтримує шлюз, що використовується, і який не відноситься безпосередньо до адрес внутрішньої мережі і не являється адресою для NAT.

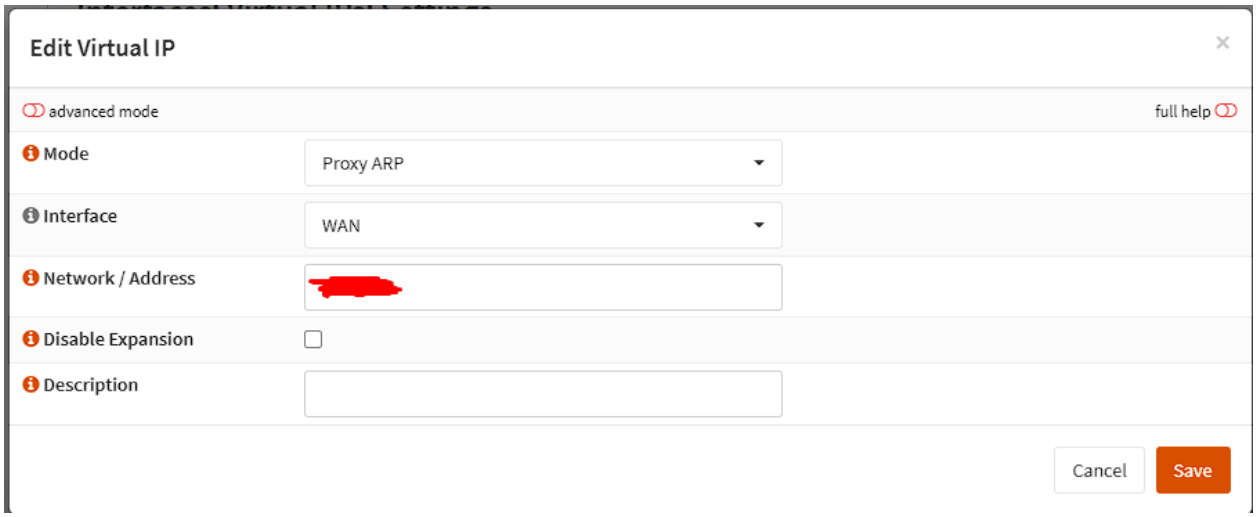

Рисунок 3.21 – Налаштування нової зовнішньої IP адреси

Також треба перевірити, чи ввімкнено NAT в OPNsense, для цього у вкладці «Firewall:Settings» треба натиснути «Advanced» і перевірити, чи виділено параметр «Reflection for 1:1».

Також варто зазначити, що зовнішні адреси присвоюються вручну, для цього у вкладці «Firewall:NAT» треба обрати «One-to-One» і додати або змінити існуюче правило. У полі «External» вказується зовнішня адреса, тоді як в полі «Internal» - внутрішня.

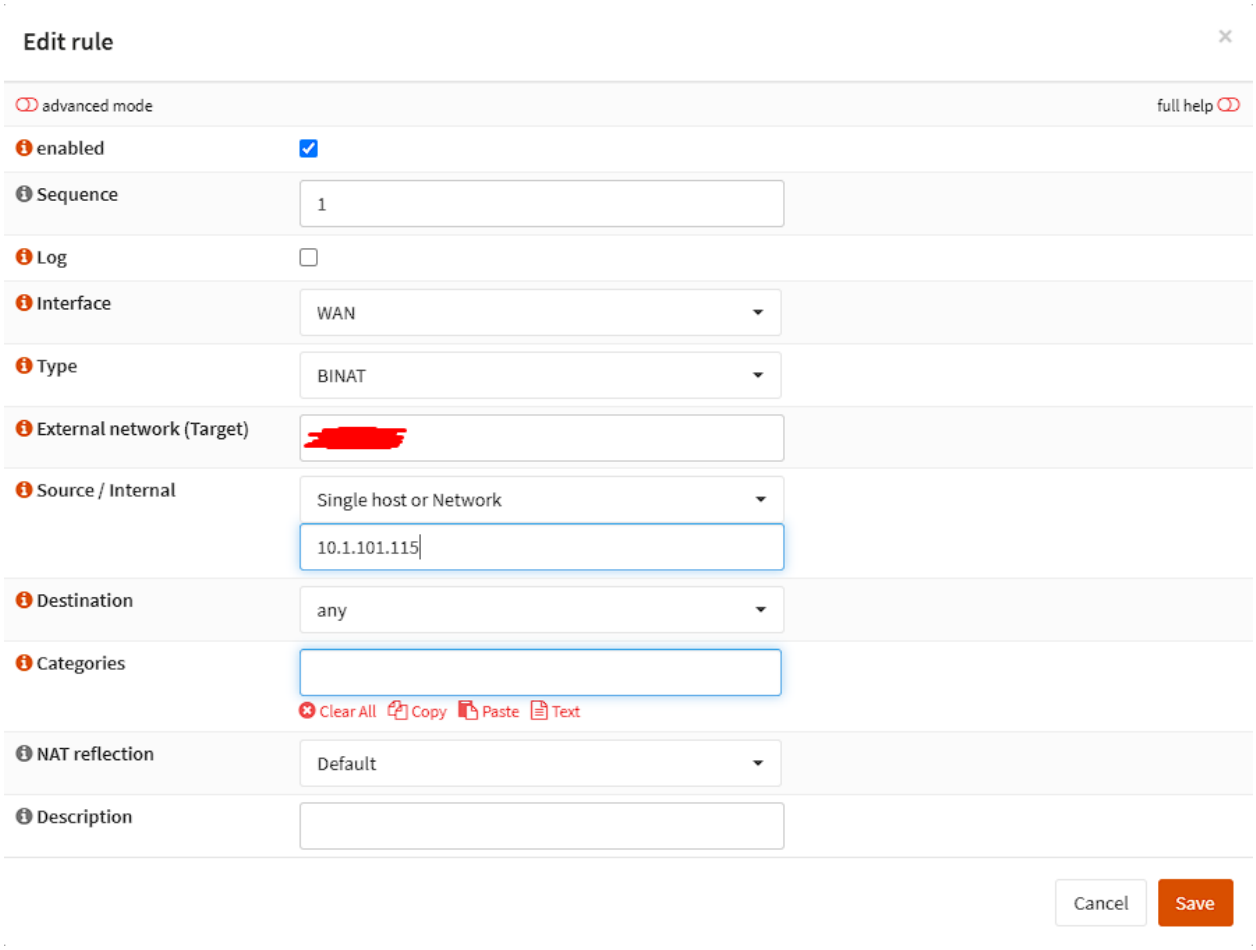

Рисунок 3.22– Присвоєння зовнішньої адреси до VM

Тепер створена інфраструктура має повноцінну внутрішню мережу з доступом в Інтернет, що значно полегшить розгортання і підтримку роботи. На жаль, всі ці кроки автоматизувати неможливо з деякими винятками.

На даному етапі залишилось створити кілька віртуальних машин або контейнерів, на яких буде розгорнуто частини інфраструктури, обов'язково має бути як мінімум одну віртуальну машину (зі збільшенням потреб варто створити додаткові), яка матиме в собі основний функціонал, а саме GitHub Runners, одну машину, що матиме в собі всі необхідні інструменти для моніторингу стану інфраструктури (стек Prometheus, Grafana, Alertmanager, Loki), а також одну машину для запуску скриптів автоматизації, так званий «бастіон».

Оптимальні параметри для цих машин вказано в таблиці 3.1.

Таблиця 3.1 – Оптимальні параметри для машин

Призначення	Тип віртуалізації	Об’єм диску	Кількість процесорів	Об’єм оперативної пам’яті
GitHub Runner	VM	100 ГБ	Мінімум 2	Мінімум 8 ГБ
Моніторинг	LXC (для збереження дискового простору сервера)	100 ГБ	4	8 ГБ
Бастіон	LXC	100 ГБ	2	8 ГБ

Варто зазначити, що зі збільшенням потреб існує можливість змінити як розміри диску, так і об’єми ОЗП і кількість процесорів.

Рисунок 3.23– Склад серверної інфраструктури

Після створення віртуальних машин користувач матиме повноцінну серверну інфраструктуру з можливістю масштабування в рамках наявних на сервері ресурсів, з налаштованою і ізольованою внутрішньою мережею з динамічним розподіленням IP-адрес і доступом в Інтернет.

4 НАПИСАННЯ ANSIBLE PLAYBOOK

4.1 Аналіз вимог до playbook

Перед початком написання скрипту для автоматизації розгортання інфраструктури необхідно визначити, що саме необхідно автоматизувати. Це ставить перед інженером завдання передбачити більшість можливих сценаріїв, наприклад, можливість встановлення програм на різні ОС.

Очевидно, що скрипт має автоматизувати встановлення і налаштування GitHub runner і всіх необхідних інструментів для моніторингу. Для забезпечення більш надійної безпеки як файлів, так і системи в цілому краще встановлювати інструменти, подібні до Prometheus, як контейнери, тобто скрипт також має перевіряти, чи встановлено Docker і встановлювати його в разі відсутності.

Також необхідно врахувати, що Docker не може самостійно проводити очищення деяких файлів, тому необхідно створити скрипт, що буде доставлятись та налаштовуватись на цільовій машині для роботи з планувальником завдань Cron.

Також для ізоляції функціоналу та файлової системи кращим вибором буде розгортання програм для моніторингу у вигляді контейнерів. Однак в такому випадку треба врахувати те, в якому вигляді контейнери будуть розгортатись. Існує два варіанти: як окремі контейнери або контейнери, що внесені до файлу Docker Compose. Другий варіант є кращим, оскільки з ним можливо змінювати параметри контейнерів та додавати нові прямо на віддаленій машині без необхідності запуску додаткових скриптів.

Оскільки програми моніторингу повинні мати можливість до зміни конфігурації, необхідно у файлі Docker Compose визначити файли, що будуть містити в собі конфігурацію цих програм.

Для оптимізації використання скрипту необхідно передбачити, які параметри будуть змінюватись часто і винести їх як змінні, що задаються перед запуском скрипту.

До таких параметрів відносяться логіни, паролі, ключі автентифікації, а також шляхи, до яких копіюються або які створюються при роботі скрипту.

Окрім наповнення необхідно приділити увагу структурі playbook, так, необхідно розподілити частини скрипту на «ролі» задля ізоляції коду, його читабельності і можливості запуску лише необхідних частин за потреби. Також в теках певних ролей треба створити теки «files» та/або «templates», що будуть містити в собі заготовлені початкові конфігурації для деяких програм.

Слідуючи цим вимогам інженер скрипт, що місить «ролі» для початкового оновлення пакетів на машині, перевірки і встановлення за потреби Docker, встановлення і налаштування програм для моніторингу, встановлення GitHub runner.

4.2 Огляд модулів Ansible та написання playbook

Для виконання поставлених задач необхідно визначити, які Ansible модулі необхідно використати.

Для виконання операції оновлення пакетів в системі існує модуль «apt», вбудований в стандартний пакет Ansible і створений саме для роботи з пакетами в ОС Linux. Він дозволяє виконувати всі можливі операції з пакетами. Однак саме для оновлення пакетів і дистрибутиву потрібні лише аргументи «update_cache» та «upgrade». Серед інших корисних можливостей даного модуля є аргумент «allow_downgrade», що дає можливість встановити більш стару версію вказаних пакетів, аргумент «state» має кілька можливих значень, серед яких «present» встановлює пакет з останньою рекомендованою версією, «absent» видаляє, «latest» встановлює останню можливу версію, «fixed» запускає

перевірку залежностей пакетів і намагається їх виправити в разі появи помилки [7].

Як додаткову міру надійності варто впровадити повторення виконання в разі невдачі, так як іноді машина може займати пакетний менеджер на деякий час, наприклад одразу після перезапуску машини. Для цього в ansible вбудовані команди для створення циклів «retries» і «until», перша визначає кількість повторень, за потреби можна вказати частоту повторень в секундах, додавши команду «delay», за замовчуванням цей час складає 0 секунд, друга задає умову для закінчення циклу [8].

Також одразу варто визначити, чи треба запускати дану роль при кожному запуску скрипту, в даному випадку очевидно потрібно, тому додається тег «always». Це спеціальний тег в Ansible саме для вказання, що певну задачу або «роль» необхідно виконувати незалежно від того, які теги вказано при запуску скрипту [9].

Для «ролі», що встановлюватиме за потреби Docker одразу додається тег «docker».

В цій «ролі» також використовується модуль «apt» для встановлення залежностей Docker, для встановлення пакетів в модулі використовується аргумент «state: present» і «name» для переліку пакетів, що треба встановити [7].

Для перевірок наявності файлів, що необхідні для встановлення Docker використовується модуль «stat» з аргументами «path» та «register» для задання шляху до файлу і назви змінної, в яку записується інформація про файл [10].

Оскільки встановлення Docker неможливе з використанням пакетного менеджера без завантаження певних файлів, використовується модуль «shell» для виконання команди для додавання GPG-ключа Docker і додавання репозиторію до системи. До цих «завдань» додається аргумент «when» з вказанням змінної, згідно якої Ansible вирішить, пропустити «завдання» чи виконати його [11].

Для додавання Docker до автозапуску при запуску системи використовується модуль «systemd_service» з аргументами «name: docker» для задання ім’я сервісу, з яким треба виконати певні дії, «enabled: true» безпосередньо для додавання сервісу до автозапуску та «state: started» для ввімкнення сервісу. Серед інших корисних функцій даного модуля аргумент «daemon_reload», що перезапускає менеджер system, це необхідно у випадках, коли додається новий сервіс або змінюється вже існуючий, аргумент «masked», що робить неможливим запуск вказаного сервісу, це потрібно для вимкнення сервісів без їх видалення [12].

Для надання користувачу, відмінному від root, можливості виконувати команди Docker без використання sudo треба використати модуль «user» з аргументами «name: “{{ ansible_user }}”», «groups: docker» та «append: yes». Таке визначення ім’я необхідне для використання зазначеного в файлі inventory.ini користувача, однак за потреби це можна винести як окрему змінну, для цього треба лише змінити «ansible_user» на іншу назву і додати її до файлу group_vars/main.yml, тому що ця змінна є вбудованою в Ansible [13].

Для перевірки встановлення всіх необхідних компонентів Docker використано «shell» модуль. До цього модуля додається аргумент «changed_when», що визначає, якою буде позначка при виконанні «завдання» [14].

Для відправки скрипту по видаленню непотрібних файлів Docker спочатку додається скрипт до теки «files» всередині «ролі» по встановленню Docker, для доставки цього скрипту на віддалену машину в Ansible передбачено модуль «copy», аргументами якого є «src» для визначення джерела копіювання, «dest» для визначення місця призначення копіювання та «mode» для встановлення прав для файлу, оскільки скрипти за замовчуванням не можуть бути виконані [15].

Скрипт поміщується саме до цієї теки за кількома причинами, серед яких незмінність файлу скрипту, тобто дуже малоймовірно, що команди, введені в

ньому, будуть замінені розробниками, а також простота у вказанні шляху до файлу, а точніше відсутність необхідності його вказання.

Після операції копіювання використовується модуль «cron» для створення та налаштування завдань Cron. Так як скрипт є і необхідно запускати його щодня опівночі, використано аргументи «name» для вказання імені завдання, «minute» та «hour» для вказання точного часу запуску скрипту, в даному випадку обидва встановлюються «0», «job» для вказання інтерпретатора та шляху до скрипту, «state» для вказання, що нове завдання саме додається [16].

Часто при роботі з Docker в Ansible виникає проблема відсутності пакету Docker в пакетному менеджері Python «pip», що призводить до виникнення помилок при роботі плейбуку, тому створюється «завдання» на його встановлення, для чого використовується модуль «pip» з аналогічними «apt» аргументами [17].

Таким чином було налаштовано автоматичне встановлення та налаштування бази для налаштування системи та встановлення моніторингових інструментів.

Наступною створено «роль» для встановлення і налаштування програм для моніторингу, а саме Prometheus, Grafana, Alertmanager та Loki, додається тег «monitoring». Prometheus збирає данні з експортерів, до яких він підключається у файлі конфігурації, Loki збирає файли логів, що надходять до нього з машин зі встановленим та налаштованим експортером логів. Grafana є візуалізатором даних, в якому налаштовується вид, в якому дані подаються та які саме дані подаються. Alertmanager надсилає повідомлення до вказаних у файлі конфігурації точок, якщо виконується умова правил повідомлень у Prometheus.

Також для коректної роботи контейнерів, до яких приєднуються певні теки, диски або файли треба створити їх на віддаленій машині перед розгортанням контейнерів. Тому створені файли конфігурації для Prometheus, Loki та Alertmanager додаються до теки «templates» всередині «ролі» для розгортання моніторингових програм як файли шаблонів з розширенням .j2, а

для Grafana заготовлені файли панелей візуалізацій додаються до теки «files» в корені проекту, таке рішення обумовлене тим, що в разі появи нових файлів графіків користувачу не доведеться шукати теку цієї «ролі» для додавання нових файлів. До «ролі» додано «завдання» на створення цих файлів з шаблонів за вказаною текою з перевіркою існування даної теки та її створенням в разі відсутності для кожної програми.

Аналогічним чином створено «роль» для розгортання та налаштування за необхідності моніторингових експортерів, що збирають дані з віддалених машин і з яких моніторингові програми, такі як Prometheus, ці дані забирають. Даній ролі надано тег «monitoring_exporters».

До стандартного набору експортерів входять Node Exporter для збору даних про стан фізичних компонентів машини, тобто навантаження процесора, дисків тощо, Cadvisor для моніторингу стану контейнерів, Aggregator exporter для збору всіх даних з інших експортерів для компактності конфігурації Prometheus, Blackbox для моніторингу підключення до серверу та/або окремих програм, що мають веб інтерфейс.

Створено «роль» для встановлення та налаштування GitHub Runner, для цього використовується «роль» «github_actions_runner» з публічного репозиторію Ansible Galaxy [18].

Дана роль дозволяє створити, оновити та видалити runner відносно певного репозиторію або відносно організації або акаунта.

Також треба визначити, які параметри необхідно винести як змінні. Так, треба передбачити, що користувач захоче обрати, куди йому зберігати файли конфігурації для певних інструментів, тому виносимо шлях для копіювання у змінну «copy_path».

Для Alertmanager в даному скрипті за замовчуванням передбачено лише відправку повідомлень до соціальної платформи Discord, тому виноситься посилання до Webhook як змінна «discord_webhook».

Для конфігурації Prometheus необхідно винести як змінні IP-адреси Prometheus, Grafana та Alertmanager, так було отримано змінні «prometheus_ip», «grafana_ip» та «alertmanager_ip», для підключення до Alertmanager необхідно вказувати логін та пароль, тому також винесено їх як змінні «alertmanager_user» та «alertmanager_password». Так само і логін та пароль самого Prometheus «prometheus_user» та «prometheus_password».

Для Grafana необхідно ввести логін та пароль, ці дані виносяться як змінні «grafana_user» та «grafana_password».

Для ролі зі встановлення та налаштування GitHub runner в даному випадку необхідні змінні «github_account» для визначення акаунту, відносно якого виконати операцію з runner, «runner_name» для встановлення імені, «runner_labels» для встановлення тегів, що допомагає відрізняти ранери між собою, а також для використання лише певних ранерів у файлах CI, «runner_user» для вказання користувача, від імені якого буде працювати ранер, «runner_version» для вказання версії ранеру, при вказанні «latest» скрипт автоматично вкаже саме останню версію, «access_token» для отримання доступу до акаунту, «github_repo» для вказання репозиторію, до якого буде прив’язано ранер. Окрім вказання певного репозиторію є змінна «runner_org», яка прив’язує ранер до акаунту в цілому, що робить ранер доступним для використання будь-яким репозиторієм.

Після написання плейбуку необхідно перевірити коректність синтаксису і провести тестовий запуск для виявлення помилок в роботі. Для цього необхідно скористатись інструментом Ansible Lint для перевірки наявності синтаксичних помилок і запустити плейбук з аргументом «–check» для перевірки наявності проблем в виконанні. Код кожної ролі приведено в додатку А.

Оскільки було застосовано теги для кожної ролі, можна запускати тести кожної частини окремо. Для цього виконується команда у вигляді «ansible-playbook -i inventory.ini playbook.yml –check –tags “<теги>”».

Рисунок 4.1 – Результати тестування ролі для встановлення Docker

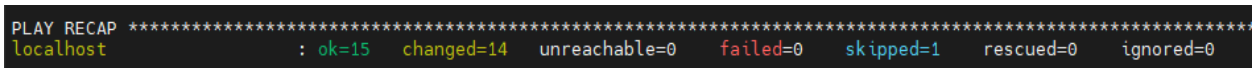

Рисунок 4.2 – Результати тестування ролі для встановлення та налаштування моніторингових інструментів

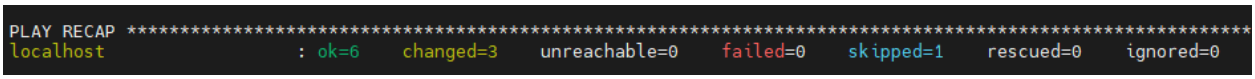

Рисунок 4.3 – Результати тестування ролі для встановлення та налаштування моніторингових експортерів

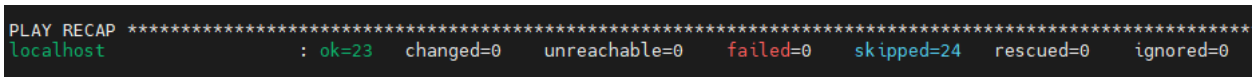

Рисунок 4.4 – Результати тестування ролі для встановлення та налаштування GitHub Runner

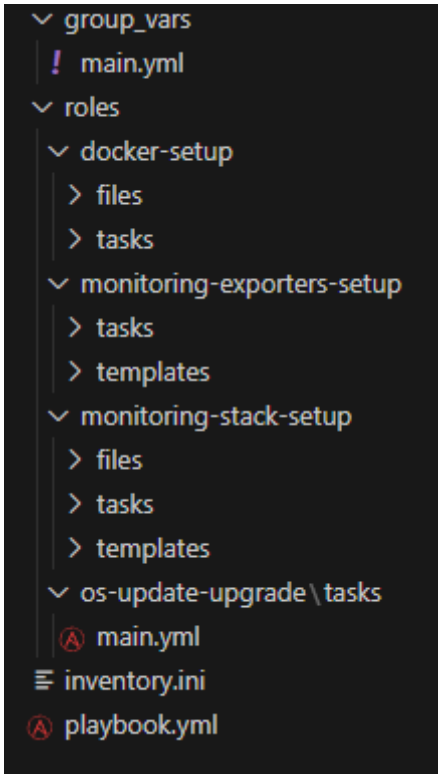

Рисунок 4.5 – Файлова структура плейбуку

4.3 Результати роботи плейбуку

Після перевірки було додано всі необхідні змінні, хости до файлу inventory.ini та запущено плейбук в роботу. Для перевірки результатів необхідно дочекатись закінчення роботи скрипту. Спершу перевірено доступність моніторингових сервісів, для цього в пошуку браузера вказано IP-адресу та порт сервісу.

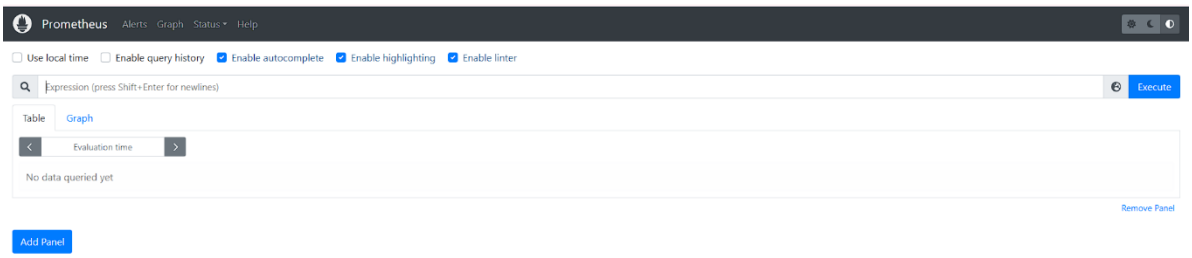

Рисунок 4.6 – Головна сторінка Prometheus

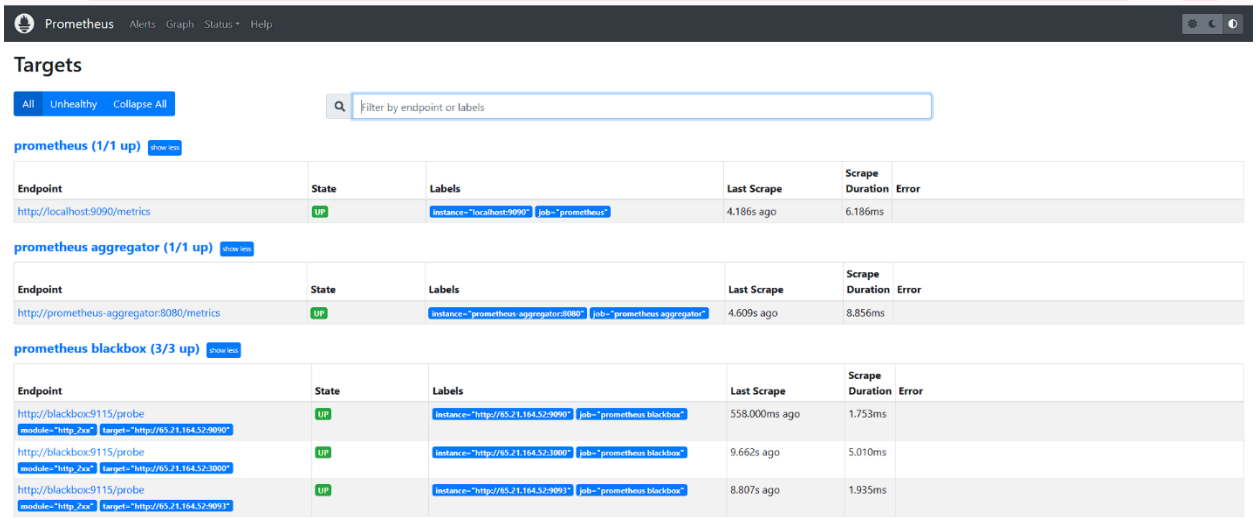

Рисунок 4.7 – Список цілей, чий стан перевіряється Prometheus

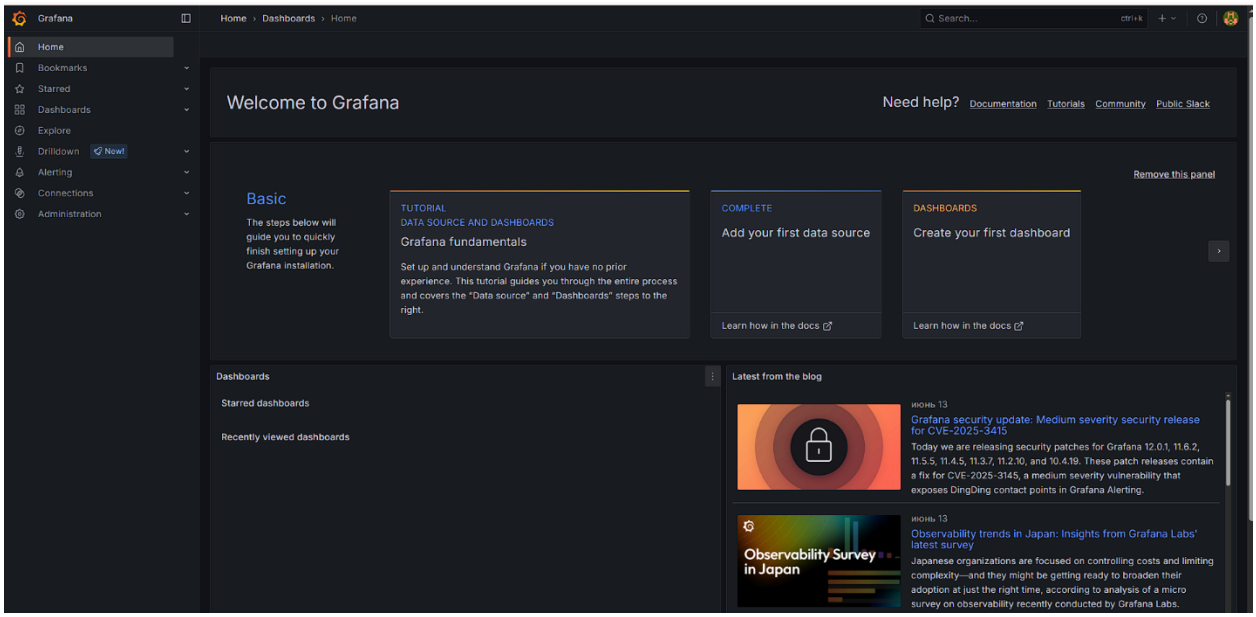

Рисунок 4.8 – Головна сторінка Grafana

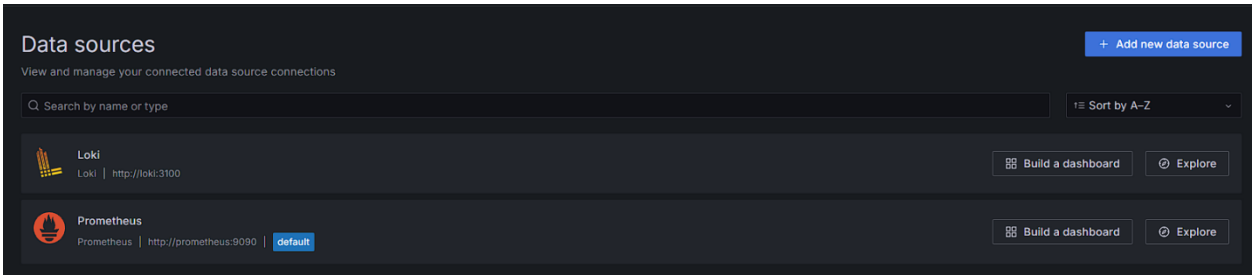

Рисунок 4.9 – Список джерел даних, до яких підключено Grafana

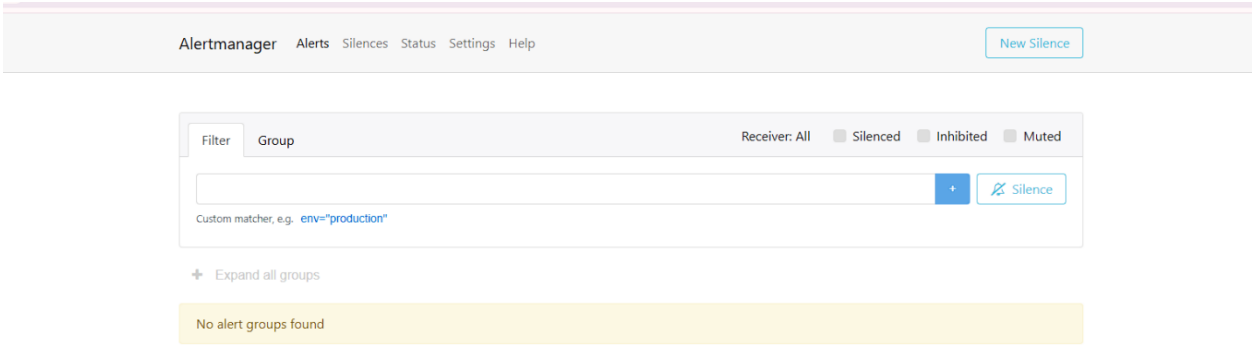

Рисунок 4.10 – Головна сторінка Alertmanager

Після перевірки моніторингових сервісів перевірено підключення GitHub Runner до аккаунту GitHub.

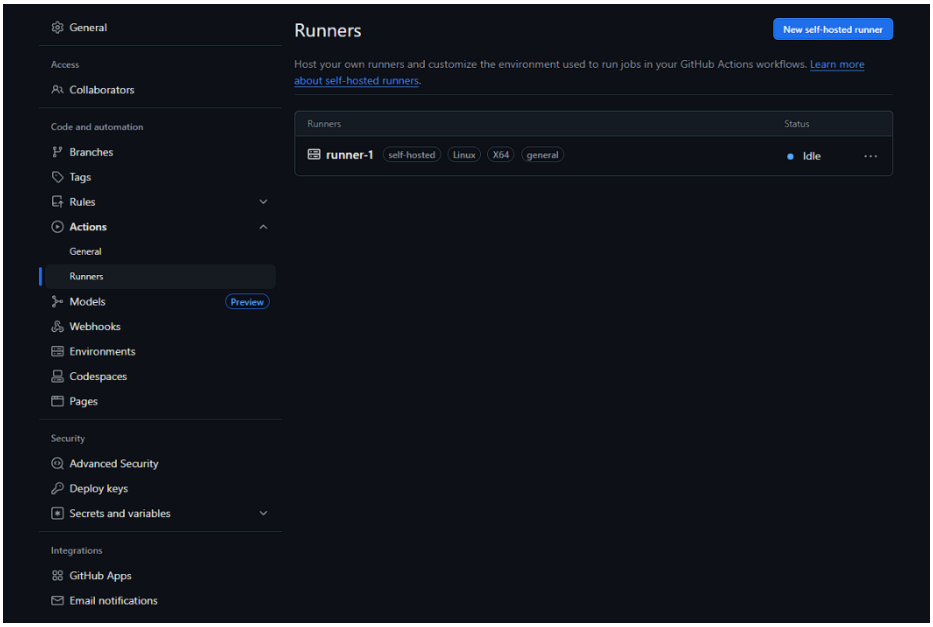

Рисунок 4.11 – Наявність GitHub Runner у вказаному репозиторії

Таким чином було показано, що всі «ролі» було виконано, як очікувалось, зі всіма вказаними параметрами та налаштуваннями.

ВИСНОВКИ

В ході виконання роботи було проаналізовано методи розгортання серверної інфраструктури, Ansible як інструмент автоматизації, порівняно його з іншими популярними інструментами автоматизації та математично доведено перевагу Ansible над конкурентами з використанням методу аналізу ієрархій.

Пройдено повний шлях для розгортання сервера на базі Proxmox Virtual Environment і провели налаштування програми маршрутизатора OPNsense для налаштування мережі всередині сервера і для надання йому доступу в Інтернет, детально розглянувши всі можливі параметри та їх призначення. Дані кроки було виконано через неможливість їх повної автоматизації.

Проведено аналіз вимог до плейбуку Ansible і визначено, які модулі краще застосувати та які параметри використовувати. Створено та протестовано плейбук перед безпосереднім використанням для розгортання програмної частини інфраструктури.

Після застосування плейбуку перевірено виконання всіх вказаних «завдань», відкривши веб інтерфейси встановлених програм та переглянувши їх підключення до експортерів та джерел даних.

Ці дії з автоматизації значно зменшують кількість часу, необхідну для налаштування подібної інфраструктури, особливо у випадках, коли необхідно налаштувати одразу велику кількість однакових машин.

І хоча у разі необхідності налаштування більш складної інфраструктури, наприклад такої, що складається з більшої кількості сервісів, складність написання скрипту автоматизації збільшується, такий крок необхідний для забезпечення надійності та швидкості налаштування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kadima S. Ansible: History Of Ansible. Medium. 10.07.2023. URL: https://medium.com/@kadimasam/ansible-history-of-ansible-4b16208c9188 (дата звернення: 04.05.2025).

2. Manage packages using chocolatey. Ansible Community Documentation. URL: https://docs.ansible.com/ansible/latest/collections/chocolatey/chocolatey/win_chocolatey_module.html (дата звернення: 04.05.2025).

3. Бондарчук А., Срочинська Г., Твердохліб М. ОСНОВИ ІНФОКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ. Київ, 2015. 75 с.

4. What is Ansible?. Medium. 07.11.2023. URL: https://medium.com/@techlatest.net/what-is-ansible-4b0c5afadc7d (дата звернення: 04.05.2025).

5. Saaty R. W. The analytic hierarchy process–what it is and how it is used. Mathematical modelling. 1987. Vol. 9, no. 3-5. P. 161–176. URL: https://doi.org/10.1016/0270-0255(87)90473-8.

6. Windows Server Installation Guide 21.05.2025. URL: https://learn.microsoft.com/en-us/windows/wsl/install-on-server (дата звернення: 22.05.2025).

7. ansible.builtin.apt module – Manages apt-packages. Ansible Community Documentation. URL: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html (дата звернення: 01.06.2025).

8. Loops. Ansible Community Documentation. URL: https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_loops.html (дата звернення: 01.06.2025).

9. Tags. Ansible Community Documentation. URL: https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_tags.html (дата звернення: 01.06.2025).

10. ansible.builtin.stat module – Retrieve file or file system status. Ansible Community Documentation. URL: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/stat_module.html (дата звернення: 01.06.2025).

11. ansible.builtin.shell module – Execute shell commands on targets. *Ansible Community Documentation*. URL: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html (дата звернення: 01.06.2025).

12. ansible.builtin.systemd_service module – Manage systemd units. *Ansible Community Documentation*. URL: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/systemd_service_module.html (дата звернення: 01.06.2025).

13. ansible.builtin.user module – Manage user accounts. *Ansible Community Documentation*. URL: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html (дата звернення: 01.06.2025).

14. Error handling in playbooks. *Ansible Community Documentation*. URL: https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_error_handling.html (дата звернення: 01.06.2025).

15. ansible.builtin.copy module – Copy files to remote locations. *Ansible Community Documentation*. URL: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/copy_module.html (дата звернення: 01.06.2025).

16. ansible.builtin.cron module – Manage cron.d and crontab entries. *Ansible Community Documentation*. URL:

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/cron_module.html (дата звернення: 01.06.2025).

17. ansible.builtin.pip module – Manages Python library dependencies. *Ansible Community Documentation*. URL: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/pip_module.html (дата звернення: 01.06.2025).

18. MonolithProjects.github_actions_runner. *Ansible Galaxy*. URL: https://galaxy.ansible.com/ui/standalone/roles/MonolithProjects/github_actions_runner/install/ (дата звернення: 01.06.2025).