ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Автоматизація розгортання віртуальної інфраструктури Proxmox VE та SDN-мереж з допомогою Ansible»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Ілля КОНОЧІНСЬКИЙ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-42
Ілля КОНОЧІНСЬКИЙ
Ім'я, ПРІЗВИЩЕ

Керівник: Борис КОНДРАТЮК
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____ Наталія ЛАЩЕВСЬКА
_____
«_____» _____ 2025р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Коночінський Ілля Ігорович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Автоматизація розгортання віртуальної інфраструктури Proxmox VE та SDN-мереж з допомогою Ansible
керівник кваліфікаційної роботи Борис КОНДРАТЮК,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026р. № 51
2. Строк подання кваліфікаційної роботи «___» _____ 2026 р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література; сереводище віртуалізації: Proxmox VE; мережева архітектура: Proxmox SDN; інструменти автоматизації: Ansible, Cloud-Init, Git; операційна система: Ubuntu Cloud Image; скрипт Postwebhook на базі Python 3.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Проаналізувати сучасні платформи віртуалізації та обґрунтувати вибір Proxmox VE
2. Дослідити принцип роботи програмно-визначених мереж (SDN)
3. Розробити скрипт Postwebhook для роботи з ISO-образом Proxmox VE
4. Створити Ansible Playbook для створення віртуальних машин та мереж
5. Перелік ілюстративного матеріалу: *презентація, блок-схема алгоритму роботи програмного коду, фрагменти програмного коду, структурна схема мережі, скріншоти тестування системи.*
6. Дата видачі завдання «___» _____ 20__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	12.03-17.03.2026	Виконано
2	Розробка теоритичної частини	18.03-05.04.2026	Виконано
3	Аналіз методів управління конфігураціями та вибір системи автоматизації	05.04-19.04.2026	Виконано
4	Практична реалізація: підготовка образу Proxmox VE та розробка скрипту Postwebhook	19.04-25.04.2026	Виконано
5	Написання Ansible-ролей для розгортання SDN та веб-сервера.	25.04-29.04.2026	Виконано
6	Проведення тестування та перевірка мережевої зв'язності	29.04-04.05.2026	Виконано
7	Висновки за результатами аналізу	04.05-07.05.2026	Виконано
8	Розробка презентації	07.05-11.05.2026	Виконано
9	Передзахист	11.05-14.05.2026	Виконано
10	Здача роботи на кафедру	18.05-21.05.2026	Виконано

Здобувач вищої освіти _____ Ілля КОНОЧІНСЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Борис КОНДРАТЮК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 59 стор., 18 рис., 2 табл., 26 джерел.

Мета роботи – підвищення ефективності та швидкості розгортання віртуальної IT-інфраструктури шляхом автоматизації встановлення Proxmox VE та налаштування програмно-визначених мереж (SDN) за допомогою Ansible.

Об'єкт дослідження – процес розгортання та адміністрування віртуальної інфраструктури і програмно-визначених мереж.

Предмет дослідження – програмні засоби та методи автоматизації розгортання Proxmox VE та SDN-мереж на базі Ansible та підходу Infrastructure as Code (IaC).

Короткий зміст роботи: У роботі досліджено підходи до автоматизації розгортання віртуальних машин та мереж з допомогою рішення на базі Proxmox VE і Ansible. Створено модифікований ISO-образ для автоматичного встановлення гіпервізора з використанням конфігурації answer.toml. Розроблено Ansible Playbook для базового налаштування системи та автоматичного створення SDN-мереж (Simple). Працездатність рішення підтверджено автоматизованим розгортанням тестової віртуальної машини з веб-сервером Nginx в ізольованому сегменті мережі, з збереженням коду в системі контролю версій Git.

КЛЮЧОВІ СЛОВА: PROXMOX VE, ANSIBLE, ПРОГРАМНО-ВИЗНАЧЕНІ МЕРЕЖІ, SDN, ВІРТУАЛІЗАЦІЯ, АВТОМАТИЗАЦІЯ РОЗГОРТАННЯ, INFRASTRUCTURE AS CODE, VLAN.

ЗМІСТ

ВСТУП **6**
1 ТЕОРЕТИЧНІ АСПЕКТИ АВТОМАТИЗАЦІЇ ВІРТУАЛЬНОЇ ІНФРАСТРУКТУРИ **7**
1.1 Огляд підходів управління інфраструктурою як кодом (Infrastructure As Code) 7
1.2 Архітектура, можливості та API платформи для віртуалізації Proxmox VE 8
1.3 Програмно-визначені мережі (SDN): принципи роботи та топологія 12
1.4 Аналіз інструментів автоматизації: Ansible та Cloud-Init 14
2 АНАЛІЗ ТА ДОСЛІДЖЕННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО РОЗГОРТАННЯ ВІРТУАЛЬНОЇ ІНФРАСТРУКТУРИ **16**
2.1 Аналіз існуючих проблем управління конфігураціями та обґрунтування вибору системи контролю версій 16
2.2 Дослідження методів конфігурації для автоматичного встановлення Proxmox VE 18
2.3 Дослідження процесу базового налаштування гіпервізора з допомогою Proxmox API 19
2.4 Моделювання логічної топології та автоматизація SDN-мереж 20
3 ПРАКТИЧНА РЕАЛІЗАЦІЯ, РОЗГОРТАННЯ СЕРВІСІВ ТА ТЕСТУВАННЯ **24**
3.1 Підготовка образу Proxmox VE та створення скрипту для роботи з post-installation-webhook 24
3.2 Створення файлів конфігурації Ansible, Cloud-Init та створення репозиторію в системи для контролю версій Git 32
3.3 Запуск та тестування роботи Ansible Playbook, перевірка мережевої зв’язності 37
ВИСНОВОК **41**
ПЕРЕЛІК ВИКОРИСТАННИХ ДЖЕРЕЛ **43**
Додаток А. Скрипт для отримання відповіді Postwebhook 46
Додаток Б. Управління змінними (group_vars/all.yml) 48
Додаток В. Початковий файл автоматизації (site.yml) 49
Додаток Г. Базове налаштування Proxmox VE (roles/proxmox_setup/tasks/main.yml) 50
Додаток Ґ. Налаштування SDN-мережі (roles/sdn_setup/tasks/main.yml) 51
Додаток Д. Створення віртуальної машини та встановлення веб-сервера NGINX (roles/ubuntu_vm/tasks/main.yml) 53
Додаток Е. Блок-схема скрипту Python, який отримує дані postwebhook 58

ВСТУП

В сучасних умовах цифровізації бізнес-процесів та розвитку інформаційних технологій, вимоги до побудови IT-інфраструктури постійно зростають. Звичайні методи ручного розгортання та адміністрування серверного обладнання і мереж стають неефективними, оскільки вони потребують значних часових витрат, погано масштабуються та мають високий ризик виникнення помилок через людський фактор. На зміну їм прийшли технології віртуалізації та підхід «Інфраструктура як код» (Infrastructure as Code, IaC).

Використання відкритих платформ віртуалізації, таких як Proxmox Virtual Environment (VE), дозволяє створювати економічно вигідні та функціональні on-premise рішення. Водночас, концепція програмно-визначених мереж (SDN) забезпечує гнучкість та ізольованість мережевого трафіку для віртуальних машин. Проте, базове встановлення гіпервізора та налаштування мережі вручну залишається вузьким місцем. Використання системи управління конфігураціями Ansible дозволяє повністю автоматизувати ці процеси, від встановлення ОС до конфігурації VLAN, забезпечуючи ідемпотентність, швидкість та відтворюваність інфраструктури. Отже, розробка комплексного рішення для автоматизації розгортання Proxmox VE та SDN-мереж за допомогою Ansible є вельми актуальним завданням для сучасного системного та мережевого адміністрування.

Разом з тим, більшість сучасних публікацій фокусується на автоматизації хмарних провайдерів (Amazon AWS, Microsoftr Azure, Google Cloud Platform). Питання ж глибокої автоматизації локальних рішень на базі Proxmox VE, зокрема з використанням новітніх інструментів автоматичного встановлення (proxmox-auto-install-assistant) та модулів спільноти (Community.Proxmox), висвітлені недостатньо і потребують практичного систематизування.

1 ТЕОРЕТИЧНІ АСПЕКТИ АВТОМАТИЗАЦІЇ ВІРТУАЛЬНОЇ ІНФРАСТРУКТУРИ

1.1 Огляд підходів управління інфраструктурою як кодом (Infrastructure As Code)

Інфраструктура як код (Infrastructure As Code, IaC) - це підхід до конфігурації ресурсів, мереж та сховищ даних через файли визначення, а не з допомогою ручного налаштування програмного та апаратного забезпечення[4, 11]. Впровадження принципів інфраструктури як код (IaC) є основоположним для методології DevOps, завдяки ньому можливо забезпечити швидке, узгоджене та відтворюване програмне середовище.

В минулому налаштування серверів та мережевого обладнання виконувалось вручну, що призводило до проблем несумісних конфігурацій систем (Configuration Drift) - ситуації, коли сервери мали різні незбіжні конфігурації[15, 22]. Підхід Інфраструктури як коду вирішує цю проблему, завдяки принципу ідемпотентності (Idempotency), який забезпечує що при застосуванні конфігурації буде досягнуто того ж самого результату.

В підході інфраструктура як код (IaC) існують два основні типи конфігурацій - декларативні та імперативні[4]. Декларативний тип конфігурації описує бажаний стан системи, а система управління повинні самостійно знайти шлях для досягнення цього стану. Імперативний тип конфігурації навпроти, визначає конкретні кроки, які потрібно виконати для досягнення бажаного результату. Такі системи управління як Ansible, Puppet, Chef та Terraform використовують декларативний тип конфігурації.

Особливої уваги в сфері автоматизації та налаштування програмного забезпечення заслуговує інструмент Ansible. Ansible не вимагає встановлення клієнтського програмного забезпечення на цільові системи, він використовує стандартні протоколи, наприклад протокол SSH для Linux-систем, для передачі

команд. Завдяки цьому можна знизити витрати на обслуговування інфраструктури та підвищити рівень безпеки систем.

Для ефективного впровадження управління інфраструктурою як кодом потрібно дотримуватися принципів розробки програмного забезпечення. Конфігурація повинна зберігатися в системи контролю версій, наприклад Git, з чіткою структурою. Успішними підходами для Ansible є використання ролей (roles), які коротко висловлюють змінні, шаблони та завдання, а також розділення змінних на рівні груп (group_vars) та окремих систем (host_vars)[17].

Одним з найважливіших компонентів Infrastucture as Code є управління секретами. Зберігання секретних даних таких як паролі до баз даних, SSH-ключі та API-токени у відкритому вигляді є критичною вразливістю. Для рішення проблем з секретними даними використовується інструмент Ansible Vault, який дають можливість безпечно використовувати секретні дані в процесі автоматизації[13].

До того ж, підхід інфраструктура як код (IaC) передбачає контроль якості коду, перед його виконанням у програмному середовищі. Завдяки інструментам статичного аналізу (наприклад Ansible-Lint для перевірки синтаксису YAML коду). Внаслідок цього можна виявляти синтаксичні та логічні помилки на етапі створення коду, гарантуючи надійність процесу розгортання інфраструктури.

1.2 Архітектура, можливості та API платформи для віртуалізації Proxmox VE

Proxmox Virtual Environment (Proxmox VE) - це платформа для віртуалізації корпоративного рівня з відкритим кодом (open source)[3, 9]. Визначальною архітектурною особливістю цієї системи є те, що вона опирається на стабільному дистрибутиві Debian GNU/Linux та використовує модифіковане ядро Linux. Завдяки цьому,, можна забезпечити високий рівень надійності та використовувати стандартні інструменти Linux для адміністрування системи (див. рис. 1.1).

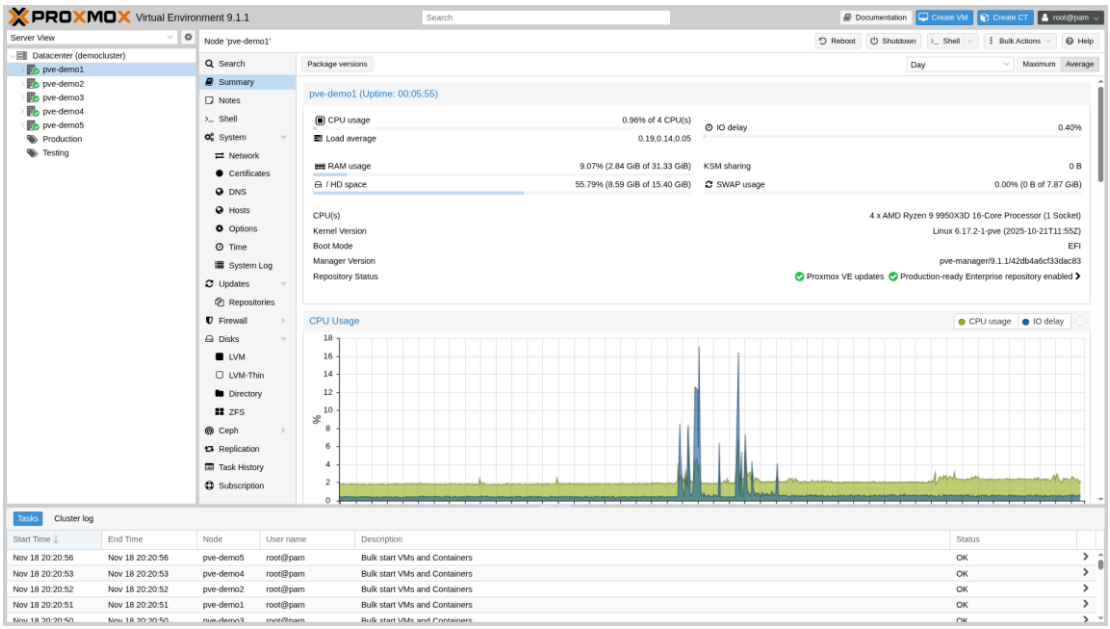


Рисунок 1.1 – Веб-інтерфейс управління платформою віртуалізації Proxmox VE

Порівнюючи Proxmox VE з іншими платформами для віртуалізації, такими як VMWare vSphere (ESXi) та Microsoft Hyper-V, становляться очевидними перевагами саме Proxmox VE перед конкурентами.

VMWare vSphere (ESXi) - це гіпервізор типу один, яка встановлюється на апаратне забезпечення серверу. Основною цього гіпервізора є мікро ядро VMkernel. VMWare vSphere відома завдяки таким технологіями як vMotion (жива міграція віртуальних машин) та DRS (автоматичне балансування навантаження). Але VMWare vSphere (ESXi) має особливість в вимогі використання окремого серверу для керування vCenter Server, який є додатковим модулем що використовує важливі обчислюванні ресурси. До того ж, після поглинання VMWare, Inc. до складу компанії Broadcom, продукт VMWare vSphere (ESXi) перейшов до моделі ліцензування суто через підписку, що призвело до підвищення цін для підтримання інфраструктури на базі цієї платформи для віртуалізації[18, 23].

Microsoft Hyper-V, як і VMWare vSphere (ESXi), є гіпервізором типу один. Він реалізує концепцію розділів, де основною системою є Windows Server, а вже над нею створено середовище для віртуалізації. Основною перевагою Hyper-V перед іншими платформами для віртуалізації є його тісна взаємодія з Active Directory та екосистемою Microsoft загалом. Проте, як і в VMWare vSphere (ESXi),

для управління кластерами серверів потрібен додатковий продукт, а саме System Center Virtual Machine Manager (SCVMM). Політика ліцензування Hyper-V є відповідних для тих, де система Windows Server є основною на кінцевих серверах[23].

Головною перевагою Proxmox VE є поєднання двох технологій віртуалізації в єдиній платформі віртуалізації: апаратної віртуалізації, яка базується на технології KVM (Kernel-based Virtual Machine) та контейнеризації (LXC) (див. рис. 1.2), яка забезпечується на рівні операційної системи з допомогою технології LXC (Linux Containers)[9]. Гіпервізор KVM дозволяє запускати повноцінні віртуальні машини з власним ізольованим ядром (як і Windows, так і Linux операційні системи), використовуючи апаратні розширення процесорів Intel VT чи AMD-V. З допомогою архітектури LXC можливо запустити ізольовані Linux-контейнери користуючись ядром хостової системи з мінімальними витратами на ресурси. Шляхом такого поєднання технологій можна гнучко розгортати сервіси залежно від вимог до продуктивності та безпеки.

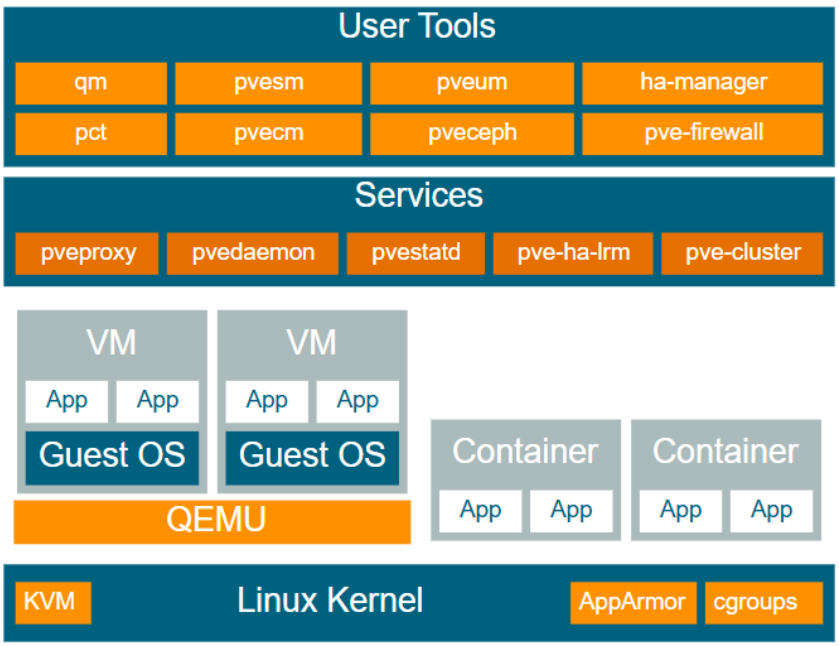


Рисунок 1.2 – Архітектура гіпервізора Proxmox VE[3]

Значною складовою архітектури системи Proxmox VE є підсистема зберігання даних. В Proxmox VE реалізована підтримка локальних та мережевих сховищ, наприклад LVM, ZFS, Ceph, NFS та iSCSI. Файлова система ZFS виконати функції миттєвих знімків (snapshot), стиснення даних на рівні блоків та вбудовану підтримку RAID[3]. Це є основою для надійного механізму резервного копіювання та реплікації віртуальних машин.

Мережева архітектура Proxmox VE основується на технології Linux Bridges (віртуальних комутаторів), але в останніх версіях платформи для віртуалізації було додано підтримку програмно-визначених мереж (Software-Defined Networking, SDN). Технологія SDN дає можливість відділити логічну топологію мережі від фізичного обладнання. В Proxmox VE адміністратор може створювати зони (VLAN, VxLAN, EVPN) та віртуальні мережі (VNet), забезпечуючи ізоляцію трафіку між різними групами віртуальних машин[7].

Головним елементом для автоматизації Proxmox VE є його API. Кожна дія, яка виконується з допомогою веб-інтерфейсу виконує виклик до відповідного методу API. Дякуючи цьому, можливості для зовнішньої автоматизації з допомогою таких інструментів як Ansible[19]. API-токени забезпечують безпечний доступ для управління ресурсами без використання паролів. Завдяки API можливе ідемпотентне конфігурування: створення віртуальних машин, налаштування мережевих стеків та правил мережевого екрану.

В платформі для віртуалізації Proxmox VE приділено увагу процесу первинного розгортання. Інструмент promox-auto-install-assistant та підтримка конфігураційних файлів у форматі answer.toml дозволяє повністю автоматизувати встановлення гіпервізора на фізичне обладнання[3]. Можливість визначити параметри дискової системи, мережі та безпеки усуває необхідність ручного втручання під час встановлення системи. У поєднанні з Cloud-Init та Ansible можна створити повноцінний цикл розгортання інфраструктури згідно принципів IaC (Infrastucure as Code).

1.3 Програмно-визначені мережі (SDN): принципи роботи та топологія

Програмно-визначені мережі (SDN) - це новітній підхід до побудови комп'ютерних мереж, який полягає у відокремленні фізичного та логічного рівня управління (Control Plane) від рівня передачі даних (Data Plane)[2, 21]. Завдяки такому підходу можливо керувати мережею з допомогою програмного API централізованим методом, що робить мережу масштабованою та придатною для автоматизації відповідно до принципів Infrastucture as Code (Iac)

Гіпервізор Proxmox VE надає інструментарій для створення сучасних SDN-мереж, який можливо налаштувати без складного процесу ручного налаштування. Відповідно до рисунку 1.3, архітектура SDN у рамках гіпервізора Proxmox VE складає з таких елементів: Fabric (глобальна мережева фабрика), Zones (зони, які визначають тип мережевої ізоляції) та VNets (віртуальні мережі)[7, 10].

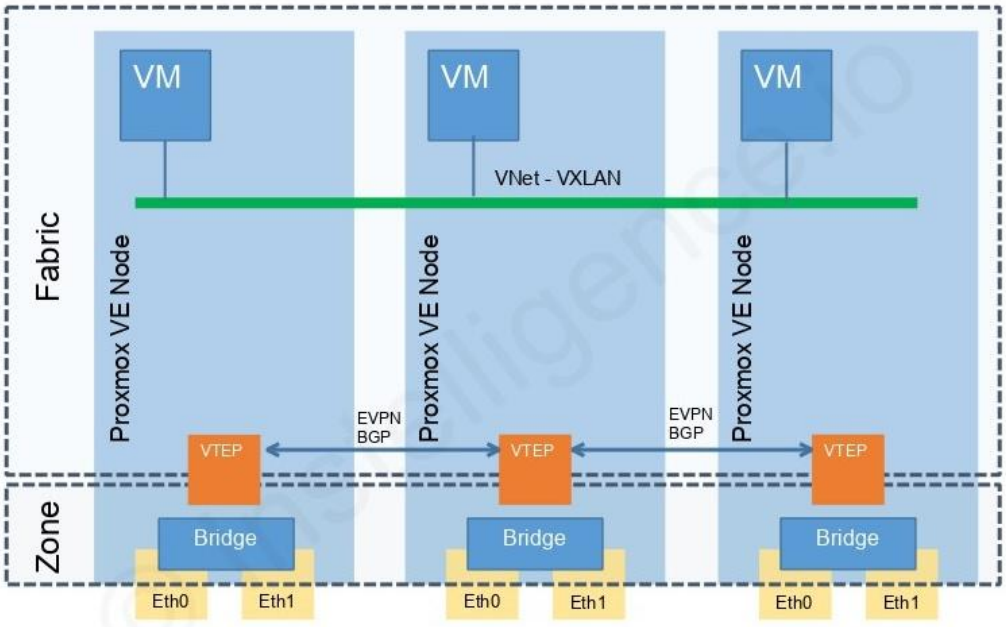


Рисунок 1.3 – Архітектура програмно-визначених-мереж (SDN) у середовищі Proxmox VE[7]

Зона (SDN Zone) є логічним об'єднанням вузлів кластера (Proxmox VE Nodes), в результаті чого вони формують єдину технологію для організації мережі. Можливо створити такі типи зони як: VLAN, VXLAN, EVPN та Simple.

В межах обраної зони створюються віртуальні мережі (VNet), які працюють як окремі сегменти мережі. Віртуальні машини або контейнери LXC підключаються до віртуальних мереж (VNet) з допомогою віртуальних інтерфейсів (vNIC). Правильне налаштування забезпечить, що мережевий трафік між різними мережами VLAN буде заблокований на рівні гіпервізора, а в всередині однієї VNet буде проходити вільний обмін мережевими даними. Для повноцінної роботи мережі для віртуальних машин чи контейнерів LXC також забезпечити автоматичне надавання IP-адрес та IP-адресацій в цілому. З допомогою вбудованого модулю proxmox_subnet, який виділяє блоки маршрутизації (CIDR-блоки) для кожної віртуальної мережі (VNet). Вбудований DHCP-сервер автоматично надає IP-адрес з блоків маршрутизації CIDR дял віртуальних машин під час їхнього запуску, що дозволяє не виконувати ручне налаштування IP-адресації.

Забезпечення доступу до зовнішньої мережі без порушення мережевої ізоляції є важливим для працездатності та безпеки віртуальних мереж. На рівні SDN-підмереж використовується механізм трансляції вихідних мережевих адрес (Source Network Address Translation, SNAT)[21]. SNAT створює вихідну маршрутизацію (Outbound NAT) без ручного налаштування шлюзів. В наслідок цього віртуальні машини чи контейнери отримують безпечний доступ до оновлень пакетів та можуть взаємодіяти з зовнішніми веб-сервісами.

Замість ручного налаштування, можливо автоматизувати задачі пов'язанні з налаштуванням мереж з допомогою Ansible Playbook, які працюють з Proxmox API. Завдяки цьому, можливо забезпечити надійне створення зон (SDN Zone) та приєднання до них віртуальних мереж (VNet), що гарантує відповідність мережевої інфраструктури згідно коду налаштувань, який існує в системи контролю версій (Git).

1.4 Аналіз інструментів автоматизації: Ansible та Cloud-Init

Перехід від ручного налаштування систем до принципу Infrastucture as Code (Iac) вимагає інструментів, які забезпечують швидкість та відтворюваність середовища. Одним з таких найбільш зручних інструментів для управління конфігураціями є Ansible, а інструментом для ініціалізації Cloud-Init. Їх поєднання дає можливість створити процес управління: від встановлення гіпервізора до конфігурації програмного забезпечення всередині віртуальних машин чи контейнерів LXC.

Ansible – це система автоматизації, яка використовує протокол SSH для управління кінцевими віртуальними машинами[13]. З допомогою нього можливо визначати тільки необхідне програмне забезпечення для гіпервізора Proxmox VE, при цьому зберігаючи цілісність системи. Основними перевагами Ansible для роботи з гіпервізором Promxox VE є такими:

– *декларативність:* Адміністратор описує бажаний стан системи, а Ansibl визначає самостійно необхідний шлях для його досягнення, при цьому гарантуючи однаковий результат при кожному запуску;

– *модульність:* Використання структури папок roles/, group_vars/ та host_vars/ дозволяє розділити конфігурацію на різні блоки (налаштування мережі, веб-сервера та безпеки). Така структура полегшує підтримку коду;

– *безпека:* Ansible Vault забезпечує безпечне зберігання секретних даних (API-токени чи паролі) і запобігати потрапляння цих даних у відкритий доступ. Використання ansible-lint надає перевірку синтаксису YAML-файлів конфігурації під час написанні коду;

– *взаємодія з Proxmox API*: завдяки використанню публічних бібліотек community.general та community.proxmox Ansible може виконувати виклики до Proxmox VE API дял створення віртуальних машин чи контейнерів, а також налаштування віртуальних мереж.

Для автоматизації встановлення гіпервізора Proxmox VE використовується інструмент proxmox-auto-install-assistant. Він використовує файл answer.toml для

визначення таких параметрів як: встановлення публічних SSH-ключів, параметри дискової системи (LVM/ZFS), налаштування мережі та інші параметри. Для правильної реалізації та сумісності з Ansible Playbook буде використано post-webhook, механізм, який після завершення встановлення системи надішле статус системи, що є індикатором для запуску Ansible Playbook[1, 6].

Cloud-Init – це інструмент для ініціалізації віртуальних машин під час першого запуску[20]. Процес автоматизації з допомогою нього складає з декількох кроків:

– *створення образу:* Завантаження Cloud-образів дистрибутивів Linux та їх змінення з допомогою утиліт, наприклад virt-customize;

– *створення шаблону:* перетворення налаштованого образу у Proxmox Template, що є прикладом для швидкого створення та клонування віртуальних машин;

– *динамічне налаштування:* передача параметрів user-data (користувачі, мережі налаштування, hostname) з допомогою Ansible-модулю proxmox_kvm, який забезпечує готовність віртуальної машини до роботи одразу після запуску.

Завдяки Ansible-модулю proxmox_subnet можна автоматизувати процес створення SDN Zone (наприклад типу VLAN), віртуальних мереж (VNet) та CIDR-блоків[16, 19]. Також це включає автоматичне надання IP-адрес через DHCP та налаштування SNAT для забезпечення вихідної маршрутизації.

2 АНАЛІЗ ТА ДОСЛІДЖЕННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО РОЗГОРТАННЯ ВІРТУАЛЬНОЇ ІНФРАСТРУКТУРИ

2.1 Аналіз існуючих проблем управління конфігураціями та обґрунтування вибору системи контролю версій

Сучасна IT-інфраструктура вимагає високої швидкості розгортання, для підтримання високого рівня надійності та масштабованості. Ручний метод налаштування серверів, який основується на взаємодії з графічними інтерфейсами чи командні рядки, має велику кількість критичних недоліків. На основі фактологічного аналізу випадків під час роботи інфраструктури (наприклад Annual Outage Analysis Report від Uptime Institute за 2025 рік)[24], показує, що понад 40% системних перебоїв пов'язанні з людським фактором при ігноруванні чи неадекватно виконаними процедурами.

Основною проблемою ручного управління є явище дрейфу конфігурацій (Configuration Drift)[22]. Така ситуація виникає, коли конфігурації серверів з часом поступово відхиляються від зразкової конфігурації. Таке явище відбувається через локальні зміни налаштувань без створення відповідних тікетів чи встановлення конфліктуючих залежностей. На підставі досліджень IBM та Broadcom щодо управління інфраструктурою, із-за явище дрейфу конфігурацій створюються ризики та вразливості для безпеки, внаслідок чого неможливо буде виконати аварійне відновлення (Disaster Recovery)[15].

Рішенням проблеми дрейфу конфігурацій, потрібно використовувати принцип інфраструктура як код (Infrastucture as Code)[11]. Але впровадження такого принципу неможливе без використання системи контролю версій (Version Control System)[5]. Було проведено порівняльний аналіз різних систем для контролю версій для вибору найкращого програми для системи контролю версій. (див. табл. 2.1).

Таблиця 2.1 – Порівняльний аналіз систем контролю версій для рішень IaC.

Критерій	Git	Subversion (SVN)	Mercurial
Архітектура	Розподілена	Централізована	Розподілена
Локальна автономність	Повна (гілки, коміти працюють без мережі)	Відсутня (потребує зв'язку з сервером)	Повна (гілки, коміти працюють без мережі)
Швидкість розгалуження (branching) та злиття (merging)	Дуже висока (підтримка rebase та fast-forward)	Низька (високий ризик конфліктів)	Висока
Інтеграція з CI/CD (GitHub, GitLab)	Вбудована (домінуюча екосистема)	Обмежена (додаткове налаштування)	Середня
Підтримка криптографічної цілісності	SHA-256, SHA-1	Відсутня	SHA-1
Механізм блокування файлів	Відсутній (паралельна робота)	Присутній (суворе блокування)	Відсутній

Дослідження переваг та недоліків різних систем для контролю версій показав, що Git, як децентралізована система для контролю версій гарантує найвищий рівень відмово стійкості, оскільки кожен розробник має повну локальну копію репозиторію, включно з історією змін коду. Завдяки цьому, ризики повної втрати коду у разі перебоїв центрального сервера (що є недоліком системи SVN). Маючи на увазі інтеграцію Git іншими інструментами для автоматизації, наприклад Ansible чи Terraform, та підтримка концепція GitOps, що автоматизує процес доставки програмного забезпечення.

2.2 Дослідження методів конфігурації для автоматичного встановлення Proxmox VE

Proxmox VE є рішення для віртуалізації корпоративного рівня з відкритим кодом, яке поєднує гіпервізор KVM та контейнеризацію LXC[9]. Однак базове встановлення Proxmox загалом виконується з допомогою графічного інсталятора шляхом ISO-образу. В контексті великого кластеру серверів чи всього дата-центру виконання процесу встановлення системи вручну стає проблемою, яка перешкоджає автоматичному розгортанню інфраструктури.

Було проаналізовано, що існують два основні шляхи для встановлення для встановлення Proxmox VE:

– *встановлення Proxmox VE на вже існуючу встановлену систем Debian GNU/Linux.* Такий метод передбачає існування вже встановленої системи Debian, на яку з допомогою Ansible плей буку додаються репозиторії з програмним забезпеченням Proxmox, після чого встановлюються необхідні пакети. Перевагою цього методу є висока гнучкість для програмного середовища. Недоліком такого підходу є висока ймовірність помилок, тривалість процесу та складність підтримки;

– *використання інструменту proxmox-auto-install-assistant.* Застосовуючи цей метод можна створити конфігураційний файл answer.toml, який додається до ISO-образу чи пережається мережею.

Використання Proxmox-Auto-installer є значно продуктивнішим методом встановлення Proxmox VE. Використовуючи його можна ввести до ISO-образу конфігураційного файлу answer.toml такі параметри системи:

– *використання ssh-ключа.* Він затосовується для безпечного доступу до системи;

– *налаштування розбивки та вибору файлової системи.* Форматування дисків використовуючи різні файлові системи (ZFS, LVM, Btrfs), а також створення RAID-масивів (RAID1, RAID10) для завантажувального диску;

– *налаштування мережі.* Призначення статичних IP-адрес, тегів VLAN, шлюзів та інших параметрів.

Використання автоматизованого встановлення нівелює ризик пов'язанні з людським фактором та значно скорочує час підготовки серверу до роботи.

2.3 Дослідження процесу базового налаштування гіпервізора з допомогою Proxmox API

Проте завершення встановлення системи не означає готовності гіпервізора до роботи. Після встановлення системи потрібно виконати дії по базовому налаштування системи. В цей процес входять такі дії як: підключення Cloud-Init шаблонів, створення спільних мережевих сховищ, створення програмно-визначених систем (Software-Defined Network) та інші налаштування[14, 20]. Наявність Proxmox REST API підтверджує, те що з допомогою програмного доступу можна змінювати та здійснювати функції. Для того щоб досягнути найбільш ефективної інтеграції Ansible, було здійснено дослідження щодо концепції Event-driven ініціалізації[1]. Postwebhook механізм є показником успішного встановлення системи Proxmox VE, який вбудований в інструменті proxmox-auto-install-assistant. Архітектура розгортання інфраструктури функціонує таким чином:

– *створення запиту (webhook):* після форматування дисків, встановлення SSH-ключів, встановлення часового поясу та налаштування мережі, гіпервізор генерує HTTP POST запит (webhook) на сервер, який вказаний в файлі answer.toml в розділі post-installation-webhook. В запиті розписаний стан системи та встановлена IP-адреса, отримана з допомогою DHCP-клієнта;

– *запуск сценарію Ansible Playbook:* сервер автоматизації отримує подію, яка запускає Ansible Playbook з IP-адресою, отриманою з запиту (webhook);

– *авторизація в Proxmox VE:* здійснюється авторизація до Proxmox VE з допомогою API-токену;

– *налаштування мережі через Ansible:* з допомогою модулю Ansible community.proxmox відбувається взаємодія з системою Proxmox[19]. Завдяки таким підмодулям як proxmox_node_network, proxmox_zone, proxmox_vnet. Використовуючи підмодуль proxmox_subnet створюються VLAN, віртуальні мережі (VNet) для ізоляції мережевого трафіку віртуальних машин[16].

Використання підходу Event-driven (використання postwebhook) в взаємодії з API Proxmox VE та інструменту для автоматизації Ansible, можливо стандартизувати налаштування інфраструктури, а також автоматизувати розгортання віртуальних машин з допомогою Cloud-Init.

2.4 Моделювання логічної топології та автоматизація SDN-мереж

В проєктування віртуальної мережевої інфраструктури моделювання логічної топології є важливим кроком для побудови надійної та відмово стійкої мережі. Вона визначає правила для мережевого трафіку, наприклад ізоляція пакетів до сервісів на рівні 2 та рівні 3 моделі OSI. Під час використання стандартних Linux Bridges (vmbr) на кожному окремому сервері ускладнює масштабування мережі в майбутньому, оскільки тоді буде потрібне налаштування та узгодження тегів VLAN між собою вручну. Завдяки використанню програмно-орієнтованих мереж (SDN) можливо відділити від фізичного рівня мережі та гнучко керувати структурою мережі[2, 21]. У Proxmox VE логічна модель SDN-мереж побудована таким чином[10]:

– *зони (Zones).* Вони визначають технологію ізоляції мережі (VLAN, VXLAN, EVPN);

– *віртуальні мережі (VNets).* Логічні світчі, яка належать до певної мережевої зони;

– *підмережі (Subnets).* Вони визначать блок CIDR IP-адресів, мережевий шлюз та конфігурацію DHCP-серверу всередині віртуальної мережі (VNets).

Архітектура SDN-зон у платформі для віртуалізації має декілька типів зон, які можливо створити для побудови віртуальних мереж для кінцевих віртуальних машин або контейнерів LXC[7]. Існують такі чотири основні зони:

– *мережева зона типу Simple.* Проста мережева зона, вона впроваджує просту форму ізоляції пакетів на другому рівні (L2) моделі OSI, створюючи віртуальний світч (Linux Bridge) в рамках одного серверу. Принцип його роботи полягає в тому, що віртуальні машини, які підключені до цієї мережі, обмінюються даними через віртуальний міст і не використовують фізичні інтерфейси напряму. Перевагою цього типу віртуальної мережі є простота в створенні, а недоліком є обмежена масштабованість цього типу мережі;

– *мережева зона типу VLAN (Virtual Local Area Network).* Тип мережі, який базується на тегування кадрів IEEE 802.1Q. Принцип роботи цього типу мережі в тому, що до кожної віртуальної мережі (VNet) всередині цієї мережевої зони призначається унікальний VLAN ID. Гіпервізор мітить Ethernet-пакети віртуальних машин тегом перед відправкою їх через фізичний інтерфейс. Основною перевагою є висока продуктивність цього типу зони. Недоліком такої мережі є обмеження максимум 4094 мереж, які можуть існувати;

– *мережева зона типу VXLAN (Virtual Extensible LAN).* Мережа, яка покликана вирішити проблеми масштабованості VLAN, використовуючи Overlay (оверлейні мережі) опираючись на стандарт RFC 7348 (VXLAN). Принципом роботи VXLAN є механізм MAC-in-UDP. Пакет рівня 2, створений віртуальною машиною, вбудовується до пакету UDP рівня 3 з додаванням заголовку VXLAN (24-бітний ідентифікатор мережі, VNI). Після чого мережевий пакет проходить процес маршрутизації між фізичними серверами (VXLAN Tunnel Endpoints) через звичайну мережі IP, незалежно від типу фізичної мережі. Головною перевагою є висока масштабованість (до 16 мільйонів можливих мереж), а недоліком є необхідність підвищення показника MTU (Maximum Transmission Unit) до значення 1550, для того щоб було достатньо місця для додаткових даних VXLAN;

– *мережева зона типу EVPN (Ethernet Virtual Private Network)*. Найбільш функціональний та масштабний тип мережевої зони. В цьому типу мережі поєднано передачу даних (Data Plane) з допомогою протоколу VXLAN, а також контролеру управління (Control Plane) базуючись на динамічному протоколі MP-BGP (Multiprotocol Border Gateway Protocol). Принцип роботи EVPN полягає в роботі EVPN як маршрутизатора BGP, який динамічно вивчає MAC-адреси та IP-адреси своїх віртуальних машин та сигналізує про них інші сервери кластери з допомогою BGP-сесій. Основним недоліком EVPN є його складність для налаштування, оскільки потрібно правильно налаштувати BGP-контролер. Проте, при правильному налаштуванні, процес маршрутизації завдяки EVPN можна організувати безпосередньо там, де запущена віртуальна машина.

Також було створено проведено порівняння основних типів SDN-зон, які доступні на вибір (див. табл. 2.2).

Таблиця 2.2 – Порівняння типів зон SDN у Proxmox VE

Тип зони	Рівень ізоляції	Масштабування	Необхідність у MTU 1500
Simple	Локальний (L2)	Лише один вузол	Ні
VLAN	Теги (802.1q)	До 4094 мереж, залежить від фізичної мережі	Ні
VXLAN	Інкапсуляція (L3)	Не залежить від фізичної мережі (необмежено)	Так
EVPN	Динамічна маршрутизація	Оптимально для великих кластерів (BGP контролер)	Так

Порівняльний аналіз показав те, що найбільш розумним вибором для розгортання мережі в рамках одного сегменту дата-центру чи кластерів серверів є

використання VLAN-зон, тому що вони не вимагають додаткових ресурсів для передачі пакетів. Втім для створення гнучких та сумісних з хмарними постачальниками послуг потрібно використовувати VXLAN тип зони, оскільки вони дозволяють створювати віртуальні мережі поверх будь-якої фізичної мережевої інфраструктури.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ, РОЗГОРТАННЯ СЕРВІСІВ ТА ТЕСТУВАННЯ

3.1 Підготовка образу Proxmox VE та створення скрипту для роботи з post-installation-webhook

Для створення процесу повністю автоматизованого циклу розгортання віртуальної інфраструктури в рамках підходу інфраструктура як код (Infrastucture as Code), варто повністю або в більшості випадків усунути ручне налаштування на кроці створення віртуальних машин[4, 11]. Методи встановлення систем вручну є неприйнятними для рішень у підході інфраструктура як код. Для вирішення такої проблеми, було використано режим автоматизованого встановлення системи (Automatted Installation), завдяки якому можна налаштувати гіпервізор без ручного втручання.

Основою для використання цього методу є файл конфігурації answer.toml. Він містить всі необхідні параметри системи, що підлягають налаштуванню, такі як: налаштування дисків, налаштування SSH-ключів системи, налаштування мережі та налаштування регіону[3].

Щоб забезпечити встановлення системи у автоматичному режимі, у середовищі розробки Visual Studio Code файл конфігурації answer.toml (див. рис. 3.1). Завдяки підсвічуванню синтаксису та зручним інструментам редактора процес написання конфігурації став значно простішим, що дозволило уникнути синтаксичниї помилок перед подальшим використанням.

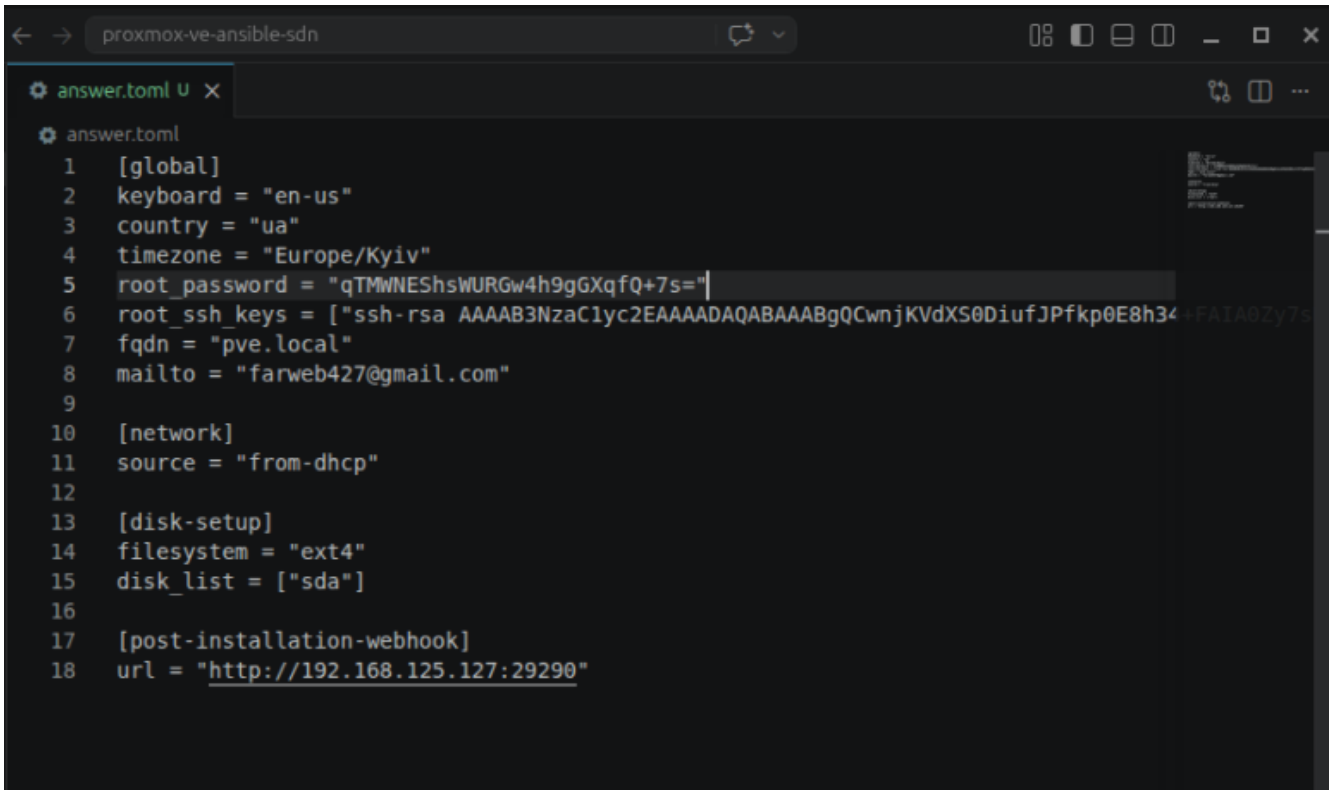


Рисунок 3.1 – Приклад файлу конфігурації answer.toml для встановлення Proxmox VE в режимі Automatted Install

В файлі конфігурації answer.toml використовується синтаксис TOML, який відомий простотою свого синтаксису та читабельністю. Структурно файл answer.toml поділений на блоки коду, кожен з блоків кодів відповідає за налаштування певного компоненту системи Proxmox VE. На рис. 3.1 показаний файл конфігурації answer.toml, який було використано для конфігурації образу для автоматичної інсталяції Proxmox VE, і він побудованих таких блоків:

– *Блок [global]*. Блок глобальних налаштувань, в якому знаходяться такі параметри як:

1. keyboard = “us” та country = “ua” - встановлення стандартної англійської розкладки клавіатури, а також встановлення регіону системи Україна, оскільки цей вибір впливає на вибір дзеркала для пакетів Debian (APT repositories).

2. fqdn = “pve.local” (Fully Qualified Domain Name) - повне доменне ім'я сервера. Воно є важливим для налаштування кластерів (corosync) та локальної бази даних конфігурацій (pmxcfs).

3. mailto = "farweb427@gmail.com” - адреса електронної пошти адміністратора сервера, на яку можливо надсилати системи сповіщення.

4. timezone = “Europe/Kyiv” - часовий пояс системи. Цей параметр є важливим для коректної синхронізації часу з серверами NTP (Network Time Protocol).

5. root_password та root_ssh_keys - встановлення паролю root та встановлення SSH-ключа для root. Можливо використання конфігурація хешованим паролем root для безпеки облікових даних користувача root.

– *Блок [network]*. Блок для налаштування мережі, в цій конфігурації було налаштовано лише параметр source, але існують ще декілька параметрів для мережі:

1. source = “from-dhcp” - вказує, як інсталятор системи повинен отримувати IP-адресу: від DHCP-сервера (опція from-dhcp) чи ігнорувати DHCP-сервер (опція from-answer).

2. cidr, dns, gateway - статичне налаштування IP-адреси, DNS-сервера та шлюзу мережі.

3. filter_match = “mac” - можливість отримувати IP-адресу, тільки на визначеному інтерфейсу, який має необхідну MAC-адресу.

– *Блок [disk-setup]*. У ньому підтримується декілька режимів управління дисків для встановлення Proxmox VE:

1. filesystem = “ext4”: встановлення файлової системи ext4 для диску, де буде встановлено Proxmox VE. Також існує вбудована підтримка файлової системи ZFS.

2. disk_list = [“sda”, “sdb”]: можливість встановлення масиву для встановлення гіпервізора та створення пулу ZFS.

– *Блок [postwebhook]*. Блок для відправки стану системи після встановлення. В параметрі url = “192.168.125.127:29290” вказано веб-сервер, де знаходиться webhook, який буде приймати повну інформацію про систему.

Для створення модифікованого ISO-образу Proxmox VE, який міститиме файл конфігурації answer.toml, потрібно застосувати утиліту proxmox-auto-install-

assistant. Цей інструмент працює найкраще в середовищі Ubuntu Linux, який побудований на основі Debian. З оглядом на це, було обрано дистрибутив Ubuntu Linux 26.04 LTS для виконання процесу інтеграції файлу answer.toml до ISO-образу.

Першим кроком є завантаження ISO-образу з офіційного сайту Proxmox VE[26]. Для подальшої роботи було обрано завантаження ISO-образу версії 9.1 (див. рис. 3.2).

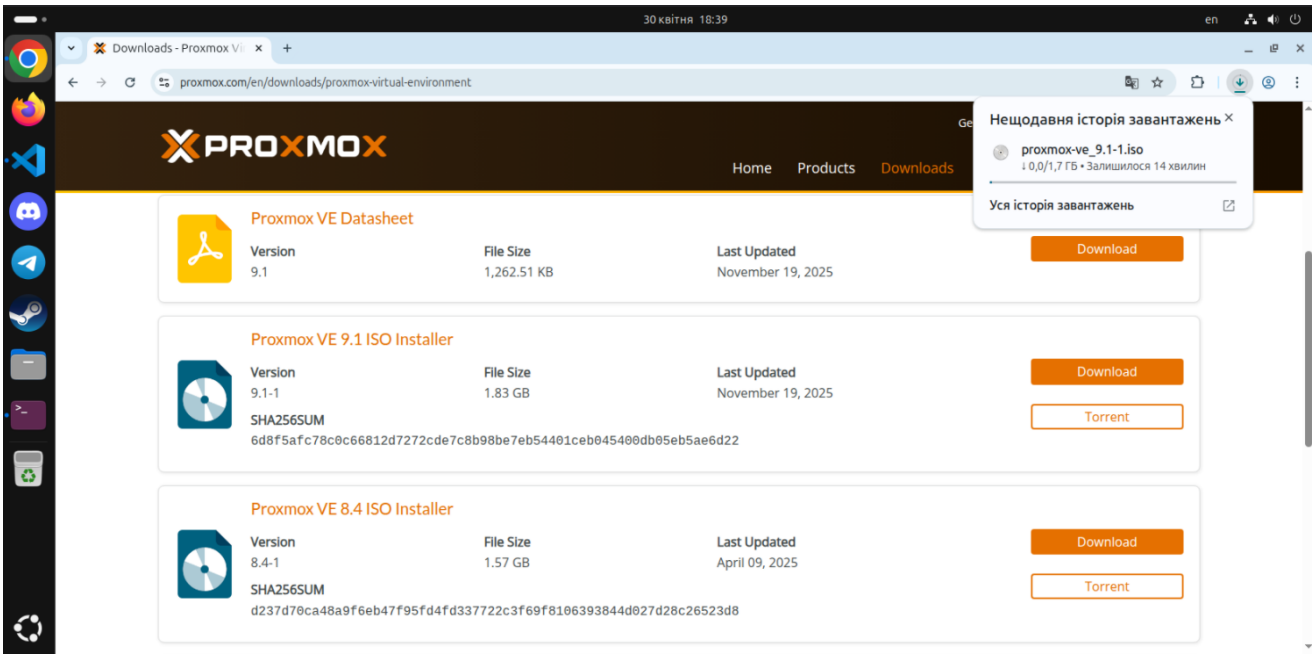


Рисунок 3.2 – Демонстрація завантаження ISO-образу з офіційного сайту Proxmox VE для подальшої модифікації

Отримавши оригінальний ISO-образ, потрібно інсталювати інструмент proxmox-auto-install-asssistant в операційній системі Ubuntu Linux. Спочатку виконано команду echo для додавання репозиторію Proxmox Releases, який додає джерело для встановлення proxmox-auto-install-assistant (див. рис.3.3). Наступним кроком стало завантаження GPG-ключа, який гарантує безпеку пакетів під час роботи з цим репозиторієм пакетів.



Рисунок 3.3 – Демонстрація додавання репозиторію Proxmox Releases до операційної системи Ubuntu Linux

Після цього було виконано оновлення пакетів системи Ubuntu Linux з допомогою sudo apt update, для того щоб оновити базу даних пакетів. Для установки інструменту proxmox-auto-install-asssitant було виконано команду sudo apt install proxmox-auto-install-assistant xorriso, де xorriso виступає як допоміжна програма для роботи з ISO-образами (див. рис. 3.4).



Рисунок 3.4 – Демонстрація процесу встановлення інструменту proxmox-auto-install-assistant

Сам процес модифікації ISO-образу Proxmox VE полягає в запуску інструменту proxmox-auto-install-asssistant в терміналі. Для цього було виконано наступну команду: proxmox-auto-install prepare-iso /home/illia-konochinskyi/Завантажене/proxmox-ve_9.1.1.iso --fetch-from iso –answer-file answer.toml --output /home/illia-konochinskyi/proxmox-autoinstall.iso. Під час виконання цієї команди програма розпаковує оригінальний ISO-образ, вбудовує до нього файл відповідей answer.toml та збирає новий модифікований ISO-образ (див. рис. 3.5)



Рисунок 3.5 – Процес модифікації ISO-образу Proxmox VE

Для забезпечення автоматизованого встановлення системи, потрібно мати обробку веб-хуку системи, який буде надіслано по закінченню встановлення системи. Цей скрипт, який написаний на мові Python 3, вирішує проблему отримання IP-адреси серверу Proxmox VE, оскільки для запуску інструменту Ansible потрібна IP-адреса, до якої Ansible завдяки протоколу SSH чи API-токену може під єднатися для подальшого налаштування. Повний програмний код наведено у Додатку А. Було обрано фреймворк Flask для написання скрипту, оскільки він є достатньо простим для розгортання та має підтримку роботи з JSON[6, 8].

Також наведено блок-схему скрипту (див. рис. 3.6), який побудований на послідовній перевірці вхідних даних. Скрипт отримує POST-запит у форматі JSON, перевіряє наявність даних JSON, після чого шукає в даних JSON значення network-interfaces. Після знаходження network-interfaces, виконується пошук інтерфейса з параметром is-management: true, вслід за чим відокремлює IP-адресу, видалає маску підмережі та запускає інструмент Ansible з знайденою IP-адресою.

Для користування скриптом необхідно виконати наступні кроки:

1. Переконатись, що встановлено Python та інструмент ansible-playbook, а також залежності скрипта як то flask (з допомогою команди pip install flask).

2. В одній папці мати і скрипт webhook.py так і файл конфігурації Ansible Playbook.

3. Запустити в терміналі командою python webhook.py, після чого було запущено скрипт.

4. Налаштовано відправку postwebhook у файлі конфігурацій answer.toml.

Для ілюстрації роботи скрипту, було створено блок-схему скрипту Python, який отримує дані postwebhook, яка знаходиться в Додатку Е.

Для запуску програми було відкрито вікно терміналу, де було здійснено в віртуальне середовище Python та запущено скрипт з допомогою команди python3 webhook.py (див. рис. 3.6).

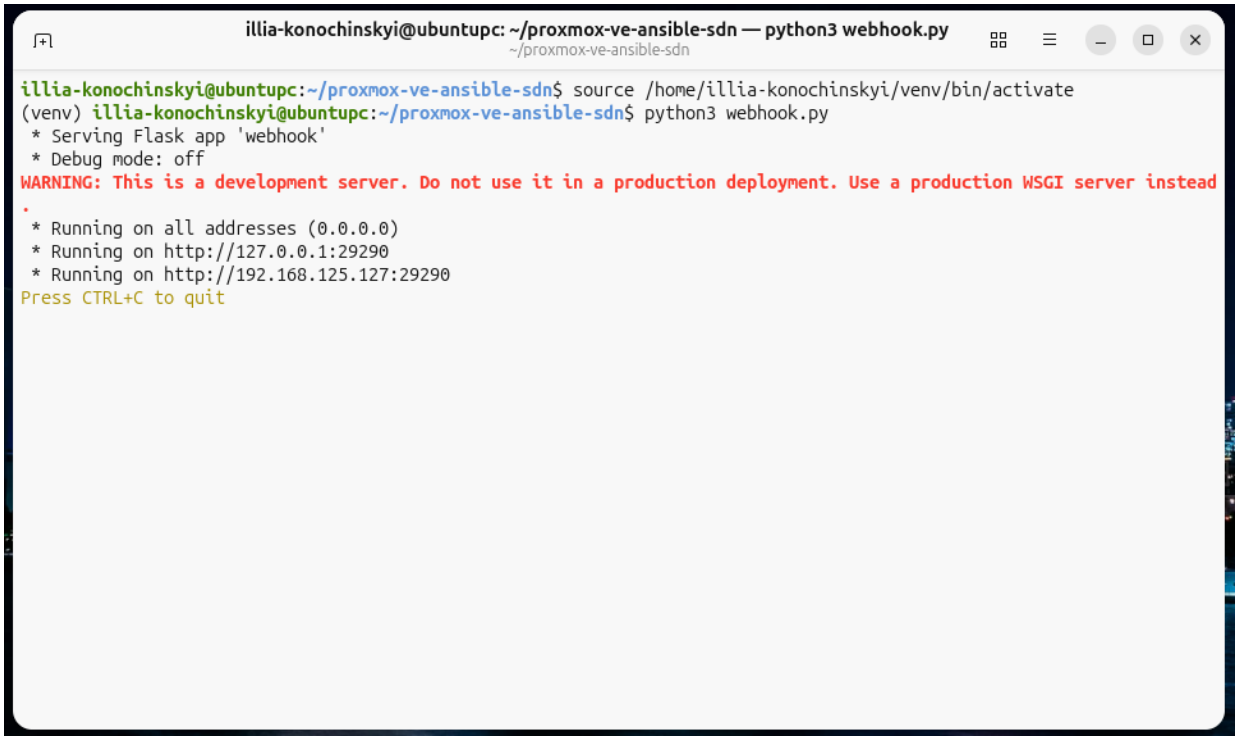


Рисунок 3.6 – Демонстрація запуску скрипта для отримання відповіді postwebhook

3.2 Створення файлів конфігурації Ansible, Cloud-Init та створення репозиторію в системи для контролю версій Git

Збереження, відстеження та аудит змін є важливим компонентом для реалізації підходу Інфраструктура як Код (Infrastucture as Code)[4, 11]. Для того щоб створити автоматизований процес розгортання веб-серверу NGINX, необхідно почати з організації системи контролю версій Git.

Першип кроком для цього є створення папки, де будуть знаходитись файли, і вона буде називатися proxmox-ve-ansible-sdn. У цій папці будуть розміщені конфігураційні файли Ansible, Cloud-Init та Ansible Playbook.

У терміналі було виконано наступні команди: git init, git config user.name та git.config user.email (див. рис. 3.7). Оскільки система для контролю версій Git вимагає чіткого авторства кожної зміни, завдяки цім командам було ініціалізованого git у папці proxmox-ve-ansible-sdn, надано ім’я (user.name) та електрону пошту (user.email) для подальшого використання у системі Git[5].



Рисунок 3.7 – Демонстрація початкового налаштування системи для контролю версій Git

Завдяки системі для контролю версій Git буде можливо тестувати та скасовувати зміни коду, Тестування конфігурацій становиться більш ефективним завдяки тому, що інші існують старі та нові версії конфігураційних файлів.

Після ініціалізації локального репозиторію Git було розділено на різні блоки, щоб код був більш читабельним та масштабованим у майбутньому. У редакторі коду Visual Studio Code було створено структуру папок та конфігурацій (див рис. 3.8)

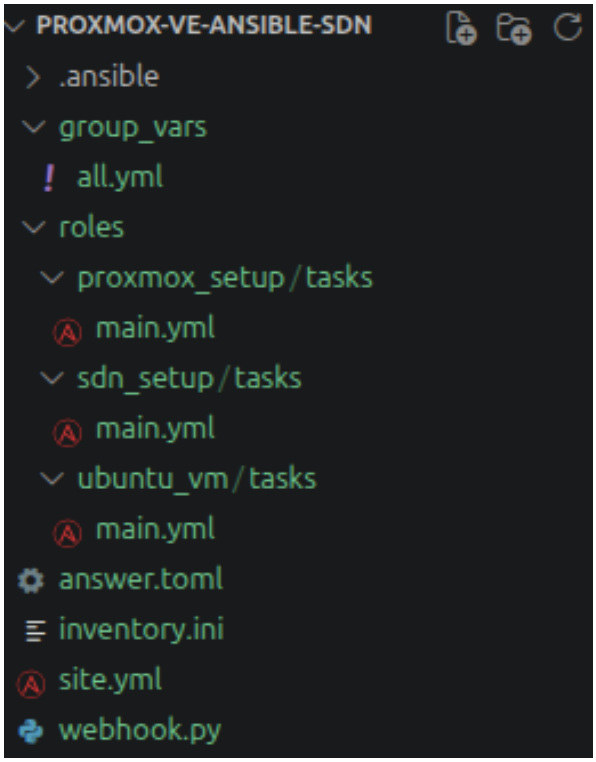


Рисунок 3.8 – Структура папок та даних у папці proxmox-ve-ansible-sdn

Усі ключові глобальні змінні знаходяться в файлі конфігурацій group_vars/all.yml. Через це можна легко змінювати змінн та інші базові налаштування, індефікатори для мереж, а також параметри віртуальної машини.

Головним є файл конфігурація site.yml, після чого він декларує послідовність застосування ролей для серверів:

– *роль proxmox_setup (базове налаштування Proxmox)*. Забезпечення оновлення пакетів та встановлення необхідних системних утиліт. Для цього кроку

важливим є наявність в системі бібліотек python3-proxmoer та python3-requests, для правильної роботи Ansible з Proxmox VE API[19];

– *роль sdn_setup (конфігурація програмно-визначеної мережі).* Автоматизація налаштування SDN-зони типу vlan та віртуальну мережу (VNet) з тегом 10 (VLAN 10)[16, 21];

– *роль ubuntu_vm (створення віртуальної мережі з допомогою cloud-init).* Завантаження образу Ubuntu Cloud Image до гіпервізора, створення конфігураційного файлу cloud-init, де задано встановлення веб-серверу nginx та відкриття порту TCP 443 для доступу до веб-сервера[14, 20].

Завершевши розробку та тестування коду для створення віртуальної інфраструктури (Ansible Playbook), надавши йому назву proxmox-ve-ansible-sdn. Проте для збереження коду необхідно зберегти зміни у системі контролю версій. Для цього в графічному інтерфейсі редактору коду Visual Studio Code було виконано фіксацію коду (git commit). Це дозволило задокументувати поточний стан коду та підготувати його до подальшого використання (див. рис. 3.9).

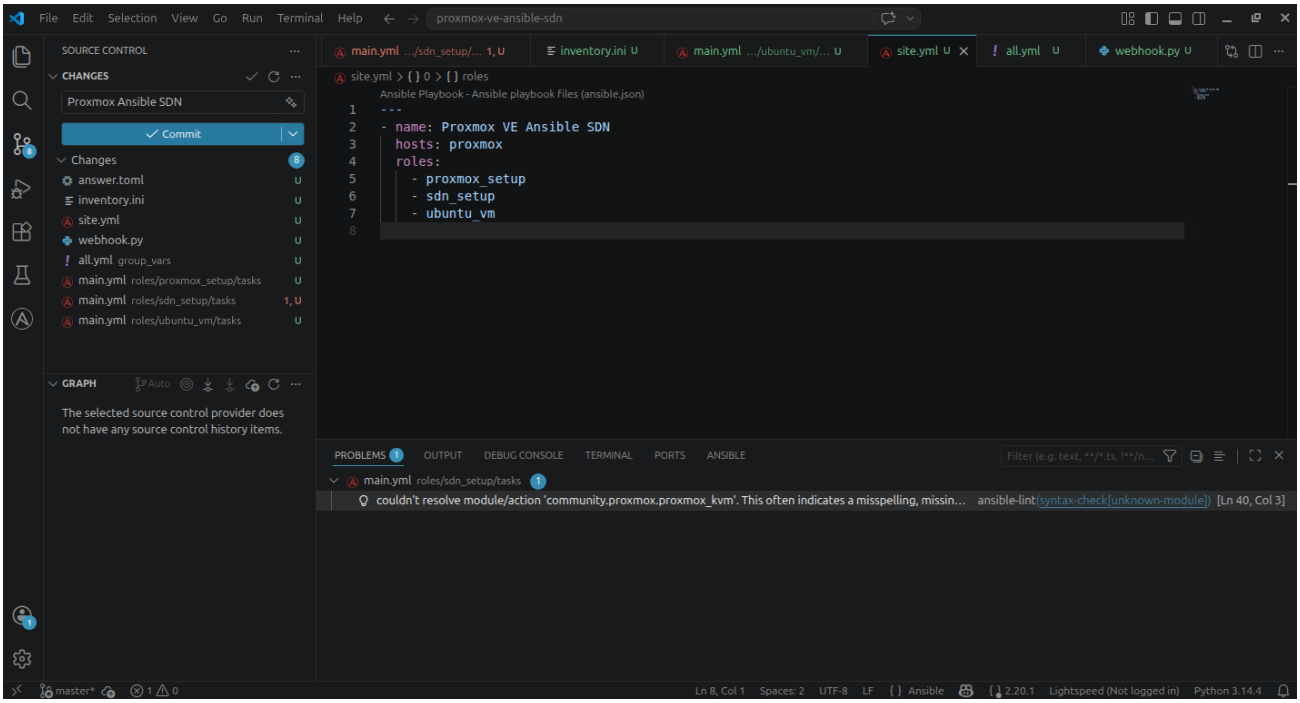


Рисунок 3.9 – Виконання git commit у редакторі коду Visual Studio Code

Після збереження коду в локальному репозиторії git, потрібно його завантажити на онлайн сервіс системи для контролю версій GitHub. Для цього в лівому меню було обрано опцію Publish Branch, тобто опублікування коду до віддаленого репозиторію (див. рис. 3.10).

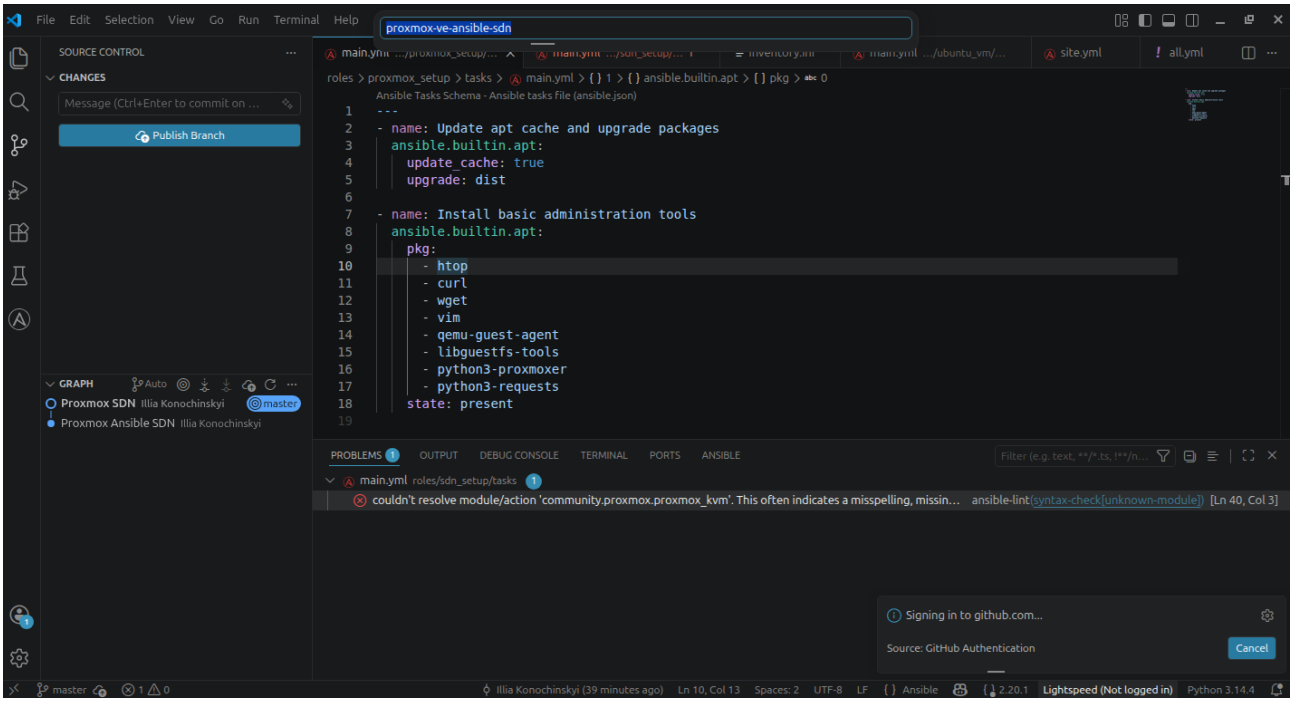


Рисунок 3.10 – Процес опублікування коду до віддаленого репозиторію git

Наступним етапом стала авторизація на онлайн-платформі GitHub. Вбудовані інструменти редактора коду запропонували обрати формат майбутнього репозиторію: приватний чи публічний. Зважаючи на наявність конфіденційних даних та облікових даних, було обрано приватний тип репозиторію. В результаті успішно був створений приватний репозиторій, куди були завантажені всі файли та конфігурації Ansible (див. рис. 3.11).

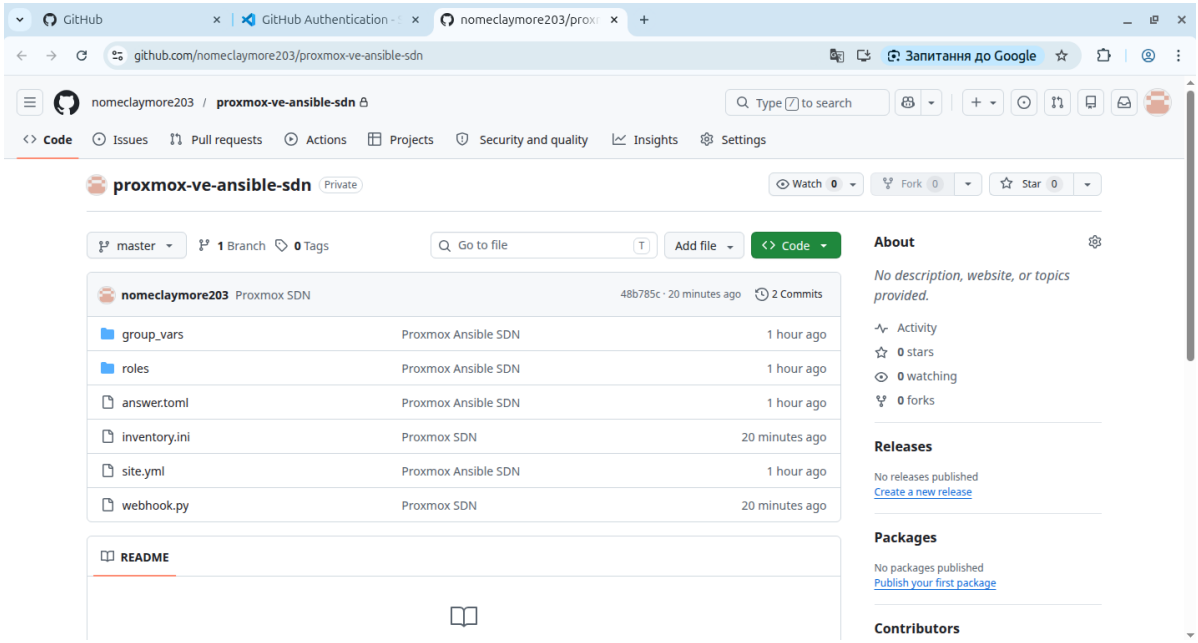


Рисунок 3.11 – Результат успішного створення закритого репозиторію

Код Ansible Playbook proxmox-ve-ansible-sdn поділений на ролі (roles), для того щоб розділити процес налаштування на декілька окремих компонентів, що є цінним для подальшої масштабованості коду. У proxmox-ve-ansible-sdn існують такі основі файли конфігурацій:

– *управління змінними (group_vars/all.yml)*. Файл, де зберігаються глобальні змінні для інших файлів конфігурації. В ньому містяться параметри доступу до API, параметри дискової системи для віртуальної машини, параметри SDN-мереж. Код цього конфігураційного файлу знаходиться в Додатку Б;

– *початковий файл автоматизації (site.yml)*. Це файл, який знаходиться на початку процесу автоматизації Ansible, в ньому вказані ролі, які потрібно запустити надалі. Код цього файлу знаходиться в Додатку В;

– *базове налаштування Proxmox VE (roles/proxmox_setup/tasks/main.yml)*. Цей файл виконує оновлення пакетів системи та встановлює необхідні пакети для роботи Ansible. Код цього файлу налаштування знаходиться в Додатку Г;

– *налаштування SDN-мережі (roles/sdn_setup/tasks/main.yml)*. Створення SDN-зони типу Simple, для тестування роботи функціоналу SDN в Proxmox VE. Після цього створюється віртуальна мережа (Vnet) в зоні sdnzone, вслід за цим

створюється підмережа і застосування SDN-мереж для налаштувань мережі. Код цього файлу конфігурації знаходиться в Додатку Ґ;

– *створення віртуальної машини та встановлення веб-сервера NGINX (roles/ubuntu_vm/tasks/main.yml)*. Ця роль виконує завантаження Ubuntu Cloud Image, який є основою дял створення віртуальної мережі. Після чого створюється Cloud-Init User Data, тобто інструкції для системи в віртуальній машині. Вслід за тим створюється, з допомогою community.proxmox.proxmox_kvm, віртуальна машина з заданими ресурсами. В фіналі системи диск розширюється до 32 ГБ, після чого запускається віртуальна машина. Файл створення віртуальної машини знаходиться в Додатку Д.

3.3 Запуск та тестування роботи Ansible Playbook, перевірка мережевої зв’язності

Завершальним кроком розробки сценаріїв (Ansible Playbook) є перевірка та тестування сценаріїв для розгортання віртуальної інфраструктури. Для цього в терміналі командою python3 webhook.py було запущено скрипт, який автоматично визначив IP-адресу сервверу Proxmox VE, і після чого запустив команду ansible-playbook, для початку роботи процесу автоматизованого розгортання віртуальної інфраструктури.

У ході роботи цієї програми виводились звіти щодо процесів програми, в яких було вказано, які процеси автоматизації були успішними (ok) та були застосовані, чи виводилися статуси failed чи unreachable, які означають не застосування автоматизації для цих компонентів (див. рис. 3.12).

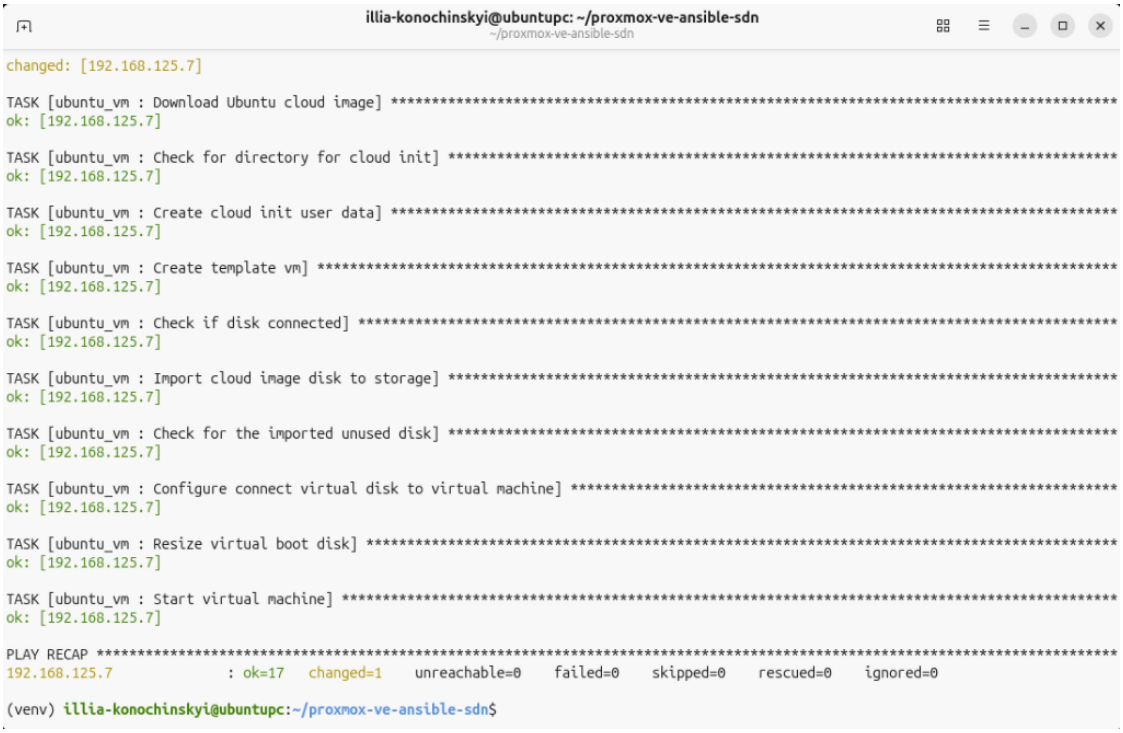


Рисунок 3.12 – Процес роботи Ansible Playbook

Як тільки всі сценарії Ansible завершили свою роботу, потрібно перевірити результат роботи сценаріїв Ansible. Було перевірено наявність створених віртуальних машин та віртуальних мереж (див. рис. 3.13).

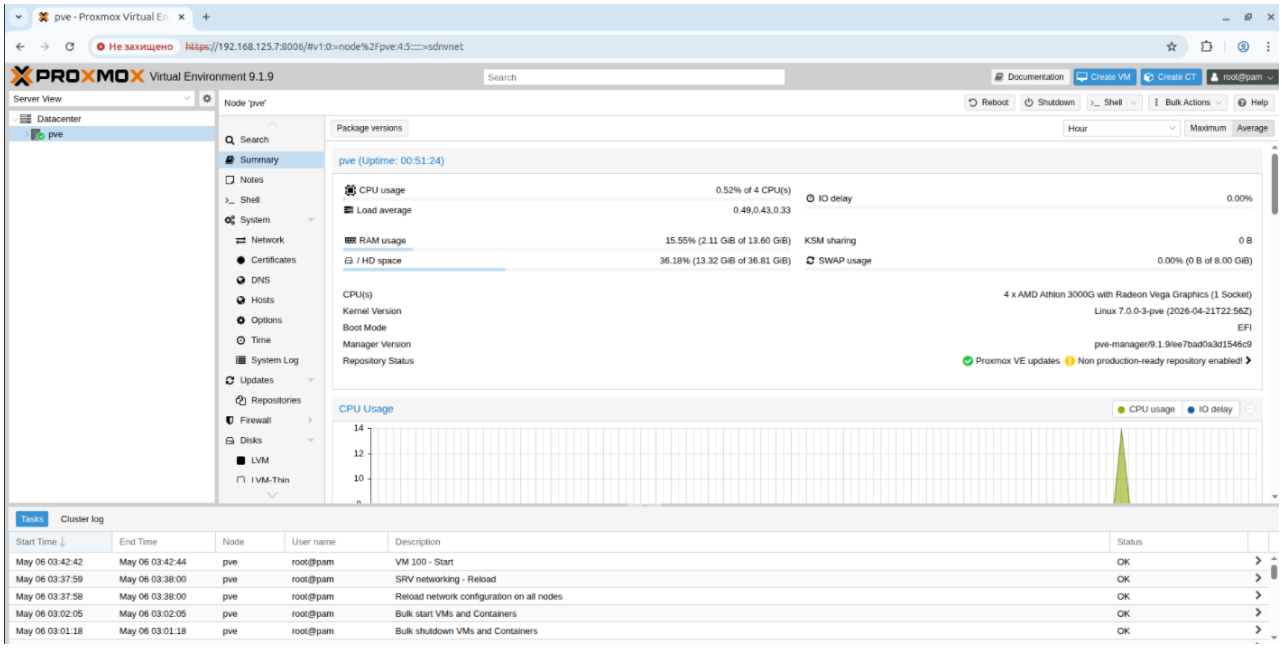


Рисунок 3.13 – Перевірка наявності створеної віртуальної машини

Зважаючи на те, що створена віртуальна машина знаходиться в ізольованій SDN-мережі, прямий доступ до неї ззовні за замовчуванням відсутній. Щоб забезпечити можливість тестування мережевої зв'язності та доступу до внутрішніх сервісів, потрібно було виконанти маршрутизацію вхідного трафіку. Для цього було налаштовано правило цільової трансляції мережевих адрес (DNAT) з допомогою системного інструменту iptables (див. рис. 3.14).

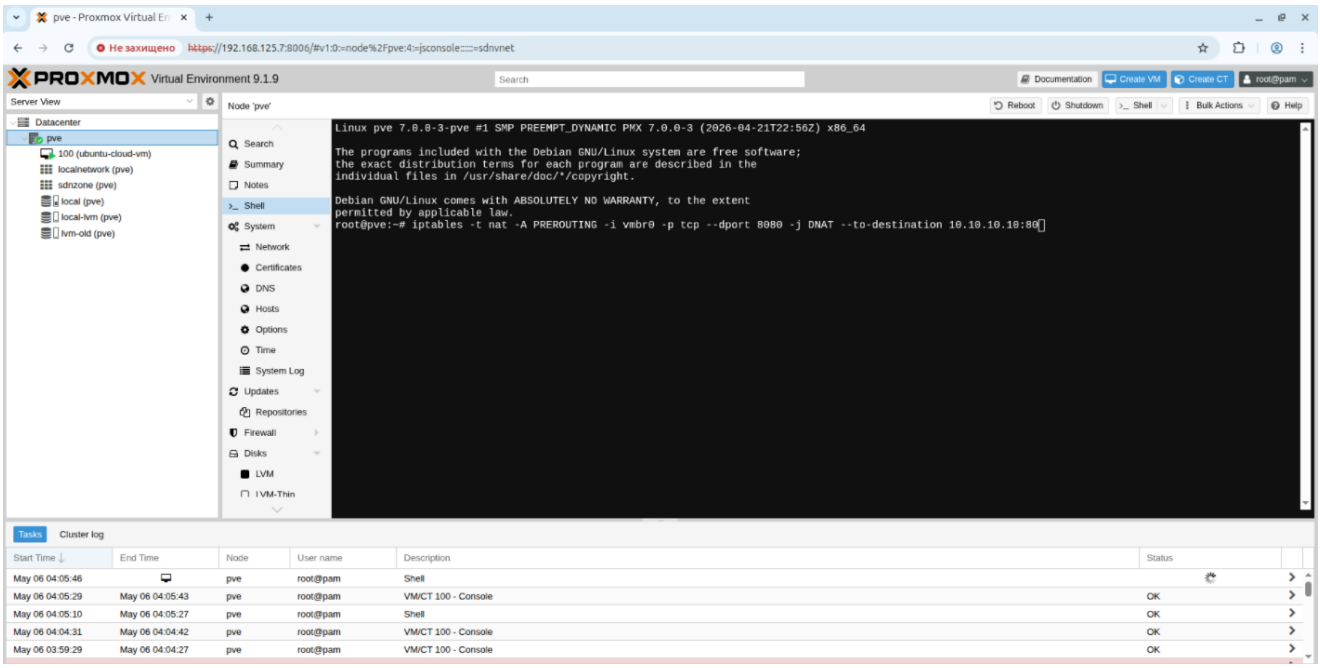


Рисунок 3.14 – Процес виконання трансляції мережевих адрес (DNAT) для перевірки мережевої зв'язності.

Щоб переконатися в коректній роботі мережі та працездатності роботи веб-сервера NGINX, у браузері було відкрито адресу http://192.168.125.7:8080. Коректне відображення стандартної сторінки веб-серверу NGINX підтверджує правильну роботу веб-серверу та мережної зв'язності (див рис. 3.15).

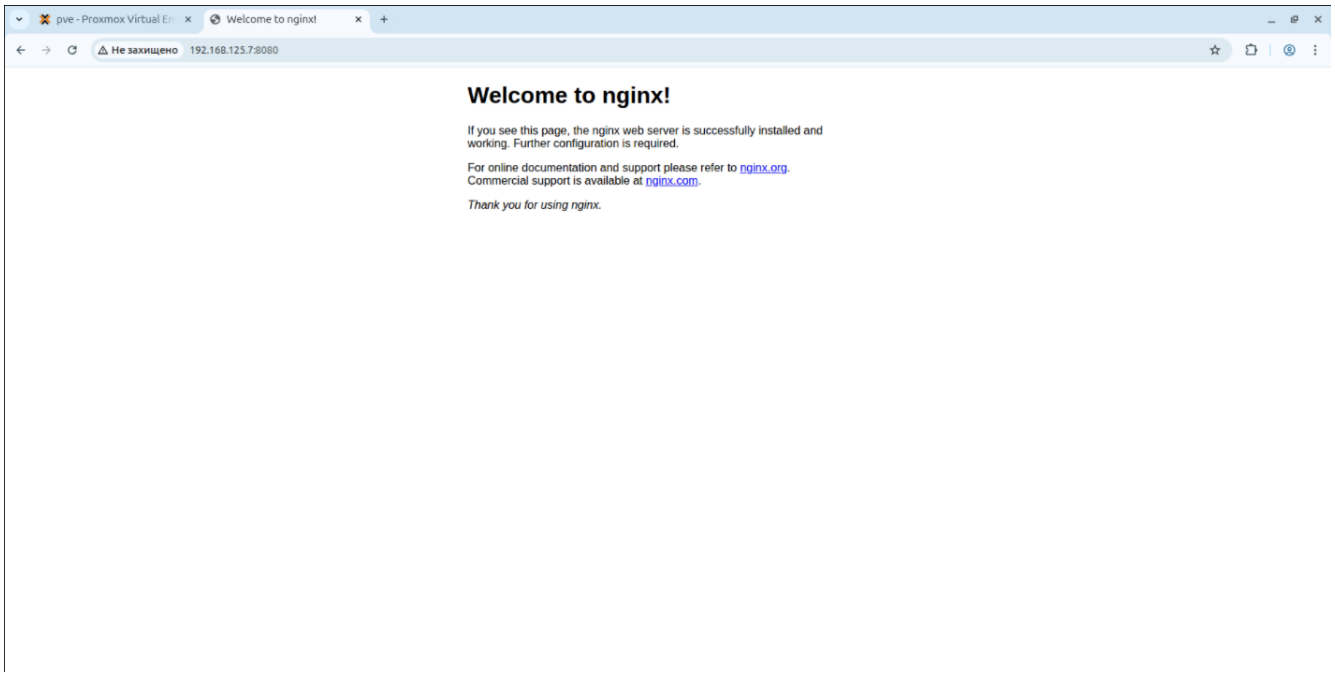


Рисунок 3.15 – Перевірка функціонування веб-серверу NGINX.

ВИСНОВОК

Ефективне управління та швидке розгортання є важливою вимогою для побудови сучасної IT-інфраструктури. У випадку віртуальних машин та мереж, ручне налаштування серверів є тривалим процесом, який може призвести до помилок та проблеми дрейфу конфігурацій. Оскільки необхідність ручного втручання може стати проблемою при масштабуванні систем, виникає необхідність в автоматизації процесу створення віртуальної інфраструктури, що дає можливість працювати за підходом Інфраструктура як код (Infrastructure as Code). Використання інструментів таких як Ansible для автоматизації процесів в інтеграції з гіпервізором Proxmox VE дозволяють реалізувати створення віртуальної інфраструктури за цим підходом.

Аналізуючи вище сказане та дослідивши сучасні системи віртуалізації, вимоги до їх розгортання та налаштування, можна зробити висновок, що автоматизації віртуальної інфраструктури є актуальним практичним завданням. Можливо побудувати гнучкі рішення під потреби компанії, ефективно використовуючи платформи та інструменти з відкритим кодом з підтримкою API-інтерфейсу для створення таких рішень.

Використання підходу інфраструктура як код у поєднанні з системою для контролю версій Git дозволяє зробити процес управління віртуальною інфраструктурою зручним та надійним. Система контролю версій Git дозволяє контролювати зміни коду, завдяки чому можна зберігати конфігураціцї та усувати проблеми їх дрейфу. Було досліджено сучасні платформи для віртуалізації та обрано Proxmox VE як найбільш доцільну для використання. Наведено переваги платформи та її API для побудови віртуальної інфраструкри.

Практично впроваджено концепцію SDN, де створено просту тополігію зони Simple, забезчевши автоматичний розподіл IP-адрес та доступ до мережі з допомогою механізму SNAT. Створення сценарій конфігурацій (Ansible Playbook) для забезпечення повного циклу автоматизації: від роботи з Proxmox API до

створення віртуальних мереж та віртуальних машин. Також розроблено механізм для автоматичного встановлення системи без ручного втручання Proxmox VE. Здійснено перевірку роботи створеної віртуальної інфраструктури, де було підтверджено функціонування розгорнутого веб-сервера, що доводить правильність обраних архітектурних рішень.

Подальший розвиток та впровадження інстурментів автоматизації, розвиток SDN-мереж та підходів інфраструктура як код є актуальним та необхідним, тому що вони забезпечують масштабованність, швидкість та надійність розгортання віртуальної інфраструктури. Вдосконалення та розвиток методів управління конфігураціями може вкрай удосконалити існуючі віртуальні інфраструктури та уникнути рутиних процесів адміністрування.

ПЕРЕЛІК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Особливості event-driven architecture на прикладах з практики : DOU. – [Електронний ресурс] – URL: https://dou.ua/forums/topic/38182/

2. Що таке програмно-визначені мережі (SDN) : Netwave. – [Електронний ресурс] – URL: https://netwave.ua/blog/software-defined-networking-sdn-viznachennya-j-osoblivosti-programno-viznachenih-merezh/

3. Proxmox VE Administration Guide : Proxmox. – [Електронний ресурс] – URL: https://pve.proxmox.com/pve-docs/pve-admin-guide.html

4. What is infrastructure as code (IaC)? - Azure DevOps : Microsoft Learn. – [Електронний ресурс] – URL: https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

5. Git Vs. Other Version Control Systems : GeeksforGeeks. – [Електронний ресурс] – URL: https://www.geeksforgeeks.org/git/git-vs-other-version-control-systems-whygit-stands-out/

6. How to build a webhook receiver with a headless flask app : Oak Tree. – [Електронний ресурс] – URL: https://www.oak-tree.tech/blog/flask-webhook-receiver

7. Introducing Proxmox SDN: Simplifying Networking with BGP, EVPN, and VXLAN : Instelligence. – [Електронний ресурс] – URL: https://www.instelligence.io/blog/2025/08/proxmox-ve-sdn-a-practical-guidefor-private-cloud-networking/

8. Welcome to Flask — Flask Documentation (3.1.x) : Flask. – [Електронний ресурс] – URL: https://flask.palletsprojects.com/en/stable/#user-s-guide

9. What Is Proxmox VE? Features and Benefits : Acronis. – [Електронний ресурс] – URL: https://www.acronis.com/en/blog/posts/what-is-proxmox/

10. Software Defined Network (SDN) Setup in Proxmox : DB Tech Reviews. – [Електронний ресурс] – URL: https://dbtechreviews.com/2024/05/06/software-defined-network-sdn-setup-in-proxmox/

11. What is Infrastructure as Code? - IaC Explained : AWS. – [Електронний ресурс] – URL: https://aws.amazon.com/what-is/iac/

12. How to Use Ansible to Manage Proxmox Virtual Machines : OneUptime. – [Електронний ресурс] – URL: https://oneuptime.com/blog/post/2026-02-21-how-to-use-ansible-to-manage-proxmox-virtual-machines/view

13. Ansible Documentation — Ansible Community Documentation : Ansible. – [Електронний ресурс] – URL: https://docs.ansible.com/projects/ansible/latest/index.html

14. A Comprehensive Guide to Cloud-Init: Automating Cloud Instance Initialization : DEV Community. – [Електронний ресурс] – URL: https://dev.to/adityapratapbh1/a-comprehensive-guide-to-cloud-init-automating-cloud-instance-initialization-4l28

15. The Silent Sabotage of Configuration Drift : Broadcom. – [Електронний ресурс] – URL: https://academy.broadcom.com/blog/network-observability/network-monitoring/the-silent-sabotage-of-configuration-drift

16. GitHub - gabrielmorin1/proxmox-sdn-modules: Ansible modules for Proxmox VE SDN feature : GitHub. – [Електронний ресурс] – URL: https://github.com/gabrielmorin1/proxmox-sdn-modules

17. Ansible File Structure, destructured : DEV Community. – [Електронний ресурс] – URL: https://dev.to/robogeek95/ansible-file-structure-destructured-27cb

18. Proxmox vs. VMware: an Ottawa SMB's guide post-Broadcom : 4UIT. – [Електронний ресурс] – URL: https://4uit.ca/blog/proxmox-vs-vmware-ottawa

19. Community.Proxmox — Ansible Community Documentation : Ansible. – [Електронний ресурс] – URL:

https://docs.ansible.com/projects/ansible/latest/collections/community/proxmox/index.html

20. Introduction to cloud-init - cloud-init 26.1 documentation : Cloud-init. – [Електронний ресурс] – URL: https://docs.cloud-init.io/en/latest/explanation/introduction.html

21. Software-Defined Network : Proxmox. – [Електронний ресурс] – URL: https://pve.proxmox.com/pve-docs/chapter-pvesdn.html

22. What is Configuration Drift? The Silent Risk to IT Infrastructure and Solutions : Josys. – [Електронний ресурс] – URL: https://www.josys.com/article/article-saas-security-what-is-configuration-drift-the-silent-risk-to-it-infrastructure-and-solutions

23. Proxmox vs VMware vs Hyper V Guide : Tbourke Solutions. – [Електронний ресурс] – URL: https://tbourke-solutions.com.au/proxmox-vs-vmware-vs-hyper-v-guide/

24. Uptime Announces Annual Outage Analysis Report 2025 : Uptime Institute. – [Електронний ресурс] – URL: https://uptimeinstitute.com/about-ui/press-releases/uptime-announces-annual-outage-analysis-report-2025

25. Automated Installation : Proxmox VE. – [Електронний ресурс] – URL: https://pve.proxmox.com/wiki/Automated_Installation

26. Downloads - Proxmox Virtual Environment : Proxmox VE. – [Електронний ресурс] – URL: https://proxmox.com/en/downloads/proxmox-virtual-environment/

Скрипт для отримання відповіді Postwebhook

```
from flask import Flask, request, jsonify
import subprocess import ipaddress
app = Flask(name)
# Створення webhook для Proxmox
@app.route('/', methods=['POST'])
def handle_webhook():
data = request.get_json(force=True, silent=True) # force для
прийому json без заголовку, silent для прийому json без
правильної структури
if not data:
   return jsonify({"error": "No JSON"}), 400

target_ip = None # Змінна IP-адреси

# Пошук мережевого інтерфейсу з значенням management в json
if 'network-interfaces' in data:
   for interface in data['network-interfaces']:
       if interface.get('is-management') == True:
           address_with_cidr = interface.get('address')
           if address_with_cidr:
               # Видалення маски підмережі
               raw_ip = address_with_cidr.split('/')[0]
               target_ip = raw_ip
               break
if target_ip:
   print(f"IP: {target_ip}. Запуск Ansible")
```

```
   # Запуск Ansible
   try:
       subprocess.run([
           'ansible-playbook',
           '-i', f'{target_ip},',
           'site.yml'
       ], capture_output=True, text=True)
       return jsonify({"status": "success",
"message": f"Успішний запуск Ansible {target_ip}"}), 200
   except FileNotFoundError:
        return jsonify({"error": "ansible-playbook недоступний в
системі"}), 500
   except Exception as e:
       return jsonify({"error": str(e)}), 500

return jsonify({"error": "IP адреси не знайдено в JSON"}), 400

if __name__ == '__main__':
app.run(host='0.0.0.0', port=29290)
```

Управління змінними (group_vars/all.yml)

```
proxmox_api_host: 192.168.125.7
proxmox_api_user: root@pam
proxmox_api_password: "5K*U'Yr'PH'DLzL"
proxmox_node: pve

# Virtual Machines Variables
storage_pool: local-lvm
snippet_storage: local

# SDN Variables
sdn_zone: sdnzone
sdn_vnet: vnet10
sdn_subnet: 10.10.10.0/24
sdn_gateway: 10.10.10.1

# Cloud-Init Variables
vmid: 100
vm_name: ubuntu-cloud-vm
vm_admin_password: “vm_admin_password”
ubuntu_image_url: "https://cloud-
images.ubuntu.com/jammy/current/jammy-server-cloudimg-amd64.img"
ubuntu_image_dest: "/tmp/jammy-server-cloudimg-amd64.img"
```

ДОДАТОК В

Початковий файл автоматизації (site.yml)

```
---
- name: Proxmox VE Ansible SDN
  hosts: proxmox
  roles:
    - proxmox_setup
    - sdn_setup
    - ubuntu_vm
```

ДОДАТОК Г

Базове налаштування Proxmox VE (roles/proxmox_setup/tasks/main.yml)

```
---
- name: Update apt cache and upgrade packages
  ansible.builtin.apt:
    update_cache: true
    upgrade: dist

- name: Install basic administration tools
  ansible.builtin.apt:
    pkg:
      - htop
      - curl
      - wget
      - vim
      - qemu-guest-agent
      - libguestfs-tools
      - python3-proxmoxer
      - python3-requests
    state: present
```

Налаштування SDN-мережі (roles/sdn_setup/tasks/main.yml)

```
---
- name: Create SDN Simple Zone
  ansible.builtin.command: >
    pvesh create /cluster/sdn/zones
    --zone {{ sdn_zone }}
    --type simple
    --ipam pve
  register: sdn_setup_zone_create
  failed_when: false
  changed_when: "'already exists' not in
sdn_setup_zone_create.stderr"

- name: Create SDN VNet ({{ sdn_vnet }})
  ansible.builtin.command: >
    pvesh create /cluster/sdn/vnets
    --vnet {{ sdn_vnet }}
    --zone {{ sdn_zone }}
  register: sdn_setup_vnet_create
  failed_when: false
  changed_when: "'already exists' not in
sdn_setup_vnet_create.stderr"

- name: Create SDN Subnet with SNAT
  ansible.builtin.command: >
    pvesh create /cluster/sdn/vnets/{{ sdn_vnet }}/subnets
    --subnet {{ sdn_subnet }}
```

```
    --gateway {{ sdn_gateway }}
    --snat 1
  register: sdn_setup_subnet_create
  failed_when: false
  changed_when: "'already exists' not in
sdn_setup_subnet_create.stderr"

- name: Apply SDN Configuration
  ansible.builtin.command: pvesh set /cluster/sdn
  when: sdn_setup_zone_create.changed or
sdn_setup_vnet_create.changed or sdn_setup_subnet_create.changed
  changed_when: true

- name: Give Proxmox moment to apply network changes
  ansible.builtin.pause:
    seconds: 5
  when: sdn_setup_zone_create.changed or
sdn_setup_vnet_create.changed or sdn_setup_subnet_create.changed
```

Створення віртуальної машини та встановлення веб-сервера NGINX

(roles/ubuntu_vm/tasks/main.yml)

```
---
- name: Download Ubuntu cloud image
  ansible.builtin.get_url:
    url: "{{ ubuntu_image_url }}"
    dest: "{{ ubuntu_image_dest }}"
    mode: '0644'

- name: Check for directory for cloud init
  ansible.builtin.file:
    path: /var/lib/vz/snippets
    state: directory
    mode: '0755'

- name: Create cloud init user data
  ansible.builtin.copy:
    dest: /var/lib/vz/snippets/{{ vm_name }}-user-data.yaml
    mode: '0644'
    content: |
      #cloud-config
      hostname: {{ vm_name }}
```

```
      manage_etc_hosts: true
users:
        - name: admin
          sudo: ALL=(ALL) NOPASSWD:ALL
          shell: /bin/bash
          lock_passwd: false
          passwd: “{{ vm_admin_password |
password_hash(‘sha512’) }}"
      package_upgrade: true
      packages:
        - qemu-guest-agent
        - nginx
        - iptables-persistent
      Runcmd:
        - iptables –I INPUT –p tcp –dport 443 –j ACCEPT
        - netfilter-persistent save

- name: Create template vm
  community.proxmox.proxmox_kvm:
    api_host: "{{ proxmox_api_host }}"
    api_user: "{{ proxmox_api_user }}"
    api_password: "{{ proxmox_api_password }}"
    node: "{{ proxmox_node }}"
    vmid: "{{ vmid }}"
    name: "{{ vm_name }}"
    memory: 2048
    cores: 2
    net:
      net0: "virtio,bridge={{ sdn_vnet }}"
```

```
    scsihw: virtio-scsi-pci
    ide:
      ide2: "{{ storage_pool }}:cloudinit"
    boot: "order=scsi0"
    serial:
      serial0: socket
    vga: serial0
    agent: 1
    cicustom: "user={{ snippet_storage }}:snippets/{{ vm_name
}}-user-data.yaml"
    state: present

- name: Check if disk is connected to virtual machine
  ansible.builtin.shell: "set -o pipefail && qm config {{ vmid
}} | grep -q '^scsi0:'"
  args:
    executable: /bin/bash
  register: ubuntu_vm_scsi0_check
  failed_when: false
  changed_when: false

- name: Import cloud image disk to storage
  ansible.builtin.command: "qm importdisk {{ vmid }} {{
ubuntu_image_dest }} {{ storage_pool }}"
  when: ubuntu_vm_scsi0_check.rc != 0
  changed_when: true

- name: Check for the imported unused disk
```

```
  ansible.builtin.shell: "set -o pipefail && qm config {{ vmid
}} | grep -o 'unused[0-9]: [^,]*' | head -n 1 | cut -d' ' -f2"
  args:
    executable: /bin/bash
  register: ubuntu_vm_unused_disk_info
  changed_when: false
  when: ubuntu_vm_scsi0_check.rc != 0

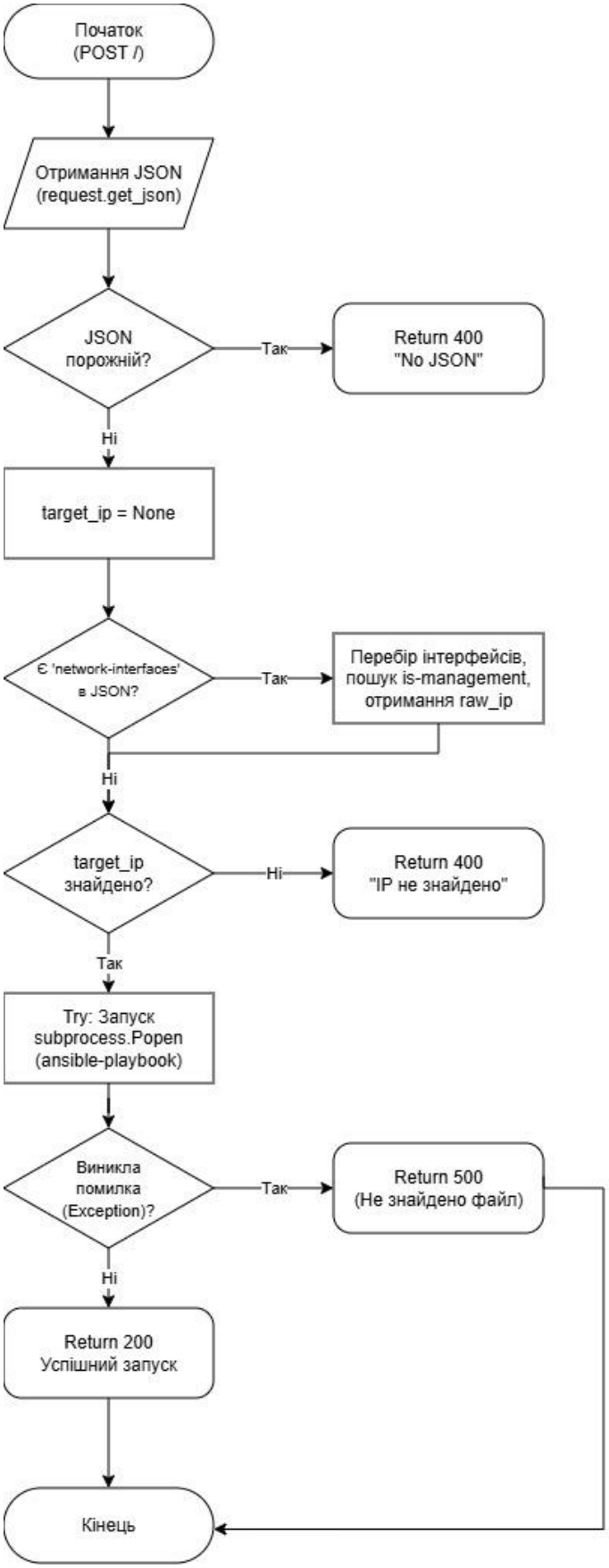
- name: Connect virtual disk to virtual machine
  community.proxmox.proxmox_kvm:
    api_host: "{{ proxmox_api_host }}"
    api_user: "{{ proxmox_api_user }}"
    api_password: "{{ proxmox_api_password }}"
    node: "{{ proxmox_node }}"
    vmid: "{{ vmid }}"
    update: true
    scsi:
      scsi0: "{{ ubuntu_vm_unused_disk_info.stdout }}"
  when:
    - ubuntu_vm_scsi0_check.rc != 0
    - ubuntu_vm_unused_disk_info.stdout | default('') != ""

- name: Resize virtual boot disk
  ansible.builtin.shell: |
    qm disk resize {{ vmid }} scsi0 20G
    qm unlock {{ vmid }}
  register: ubuntu_vm_maintenance_result
  failed_when: false
  changed_when: true
```

```
- name: Start virtual machine
  community.proxmox.proxmox_kvm:
    api_host: "{{ proxmox_api_host }}"
    api_user: "{{ proxmox_api_user }}"
    api_password: "{{ proxmox_api_password }}"
    node: "{{ proxmox_node }}"
    vmid: "{{ vmid }}"
    state: started
    timeout: 600
```

ДОДАТОК Е

Блок-схема скрипту Python, який отримує дані postwebhook



ДОДАТОК Є

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій
Кафедра Комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

на тему: «Автоматизація розгортання віртуальної інфраструктури Proxmox VE та SDN-мереж з допомогою Ansible»

Виконав студент групи КІД-42
Коночінський Ілля Ігорович
Керівник кваліфікаційної роботи:
Кондратюк Борис Олександрович, викладач кафедри

Київ – 2026

Мета роботи – Підвищення ефективності розгортання IT-інфраструктури шляхом автоматизації встановлення гіпервізора та налаштування SDN-мереж

Об`єкт дослідження – Розгортання віртуальної IT-інфраструктури і програмно-визначених мереж

Предмет дослідження – Програмні засоби та методи автоматизації розгортання Proxmox VE та SDN-мереж на базі Ansible та підходу Infrastructure as Code

Актуальність обраної теми

Вимоги до IT-інфраструктури постійно зростають через безперервний розвиток цифрових технологій.

Методи ручного розгортання серверів стають неефективними, оскільки потребують часових витрат та мають ризики помилок через людський фактор.

Базове встановлення гіпервізора та налаштування мережі вручну залишається вузьким місцем адміністрування. Використання інструменту Ansible дозволяє автоматизувати ці процеси розгортання.

Моделювання SDN-мереж у Proxmox VE

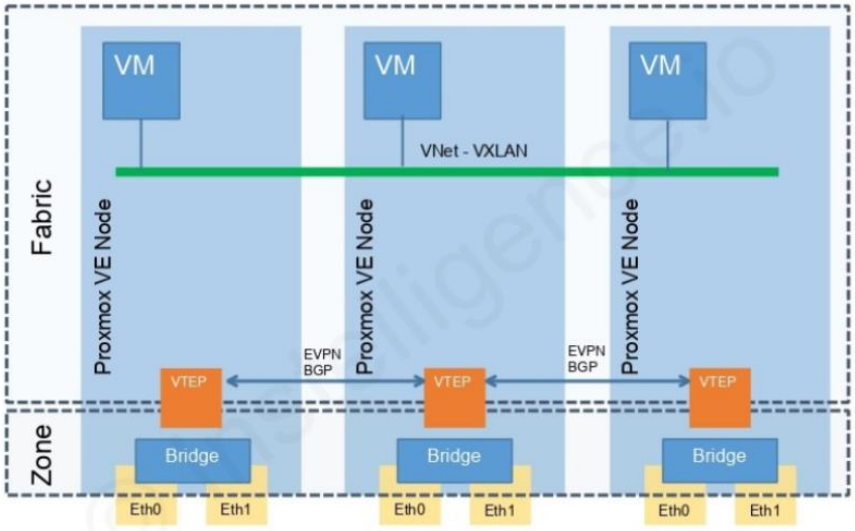


Рисунок 1.3 – Архітектура програмно-визначених-мереж (SDN) у середовищі Proxmox VE

Практична реалізація та програмна інтеграція

Створено ISO-образ для встановлення системи Proxmox VE з використанням файлу відповідей answer.toml.

Розроблено скрипт на мові Python з використанням фреймворку Flask для обробки post-installation-webhook. Скрипт автоматично витягує IP-адресу з отриманого JSON-запиту та запускає інструмент Ansible.

Написано Ansible Playbook для автоматизації основних налаштувань гіпервізора Proxmox VE та створення програмно-визначених мереж (SDN) з допомогою модуля community.proxmox.

Використані технології та компоненти

Середовище віртуалізації: Proxmox VE з підтримкою KVM та LXC.

Мережева архітектура: Proxmox SDN, механізм SNAT.

Інструменти автоматизації: Ansible, Cloud-Init.

Система контролю версій: Git з віддаленим репозиторієм на GitHub.

Стек інтеграцій: автоматизація через Webhook на базі Python 3 та POST-запитів.

Гнучкість конфігурацій та організація коду

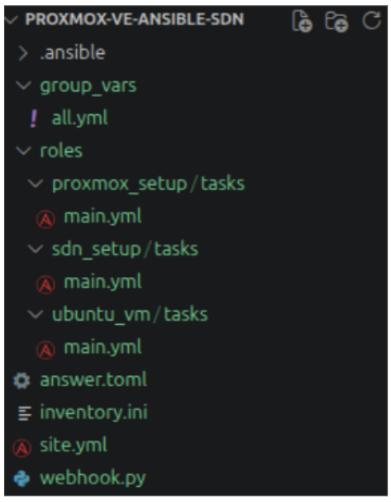


Рисунок 3.9 – Структура папок та даних у папці proxmox-ve-ansible-sdn

Інструмент Ansible дозволяє розділити процес налаштування на кілька компонентів:

- *proxmox_setup:* базове налаштування Proxmox;
- *sdn_setup:* налаштування SDN-мереж;
- *ubuntu_vm:* створення віртуальної машини.

В інтеграції з Git можливо швидко розгортати віртуальну інфраструктури, а також відстежувати та зберігати зміни.

Результати тестування системи

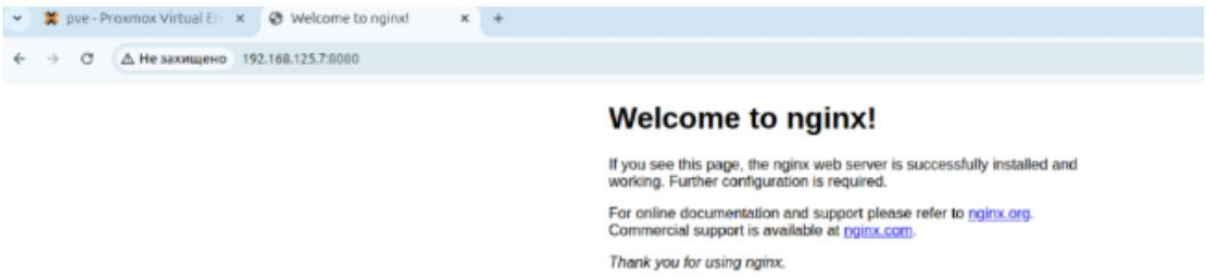


Рисунок 3.16 – Перевірка функціонування веб-серверу NGINX

Висновки

Було вирішено актуальне завдання автоматизації процесу створення віртуальної IT-інфраструктури на базі Proxmox VE за допомогою Ansible.

Охоплено повний цикл автоматизації: від налаштування встановлення Proxmox до створення програмно-визначених мереж (SDN) та кінцевих сервісів.

Вирішено проблеми ручного адміністрування шляхом впровадження парадигми Infrastructure as Code у інтеграції з Git.