ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення автоматизованих систем

Пояснювальна записка

до кваліфікаційної роботи на
ступінь вищої освіти магістр

на тему: «Розгортання та автоматизована конфігурація віртуальних серверів на базі ОС Linux»

Виконав: студент 6 курсу, групи ІСДМ-61 спеціальності
126 Інформаційні системи та технології
освітня програма «Інформаційні системи та технології»

(шифр і назва спеціальності)

Сирота М.П.

(прізвище та ініціали)

Керівник Ткаленко О.М

(прізвище та ініціали)

Рецензент_____

(прізвище та ініціали)

Нормоконтроль _____

(прізвище та ініціали)

Київ – 2022

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти - «Магістр»
Спеціальність - 126 Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри Інженерії
програмного забезпечення
автоматизованих систем
_____ К.П.Сторчак
“____” _____2022 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ
Сирота Максим Павлович
(прізвище, ім’я, по батькові)

1.Тема роботи: «Розгортання та автоматизована конфігурація віртуальних серверів на базі ОС Linux»»
Керівник роботи: Ткаленко Оксана Миколаївна, к.т.н., доценткафедри ІПЗАС.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від року 12.12.2022 №-
2. Строк подання студентом роботи ----- 2022 року.
3. Вхідні дані до роботи:
3.1 Вимоги до кваліфікаційної роботи магістра з актуальних завдань спеціальності;
3.2 Існуючі інструменти автоматизованої конфігурації серверів;
3.3 Технічні вимоги;
3.4 Науково-технічна література з питань, пов’язаних з темою роботи.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно вирішити):
4.1 Аналіз засобів автоматизованої конфігурації;
4.2 Створення конфігураційного файлу для атоматизованої конфігурації віртуальних серверів;
4.3 Результати дослідження;
4.4 Висновки.

5. Перелік графічного матеріалу:
6. Дата видачі завдання.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури		
2	Вивчення матеріалів для подальшої взаємодії з ними		
3			
4	Визначення технічного завдання		
5			
6			
7	Вступ, висновки, реферат		
8			
9	Попередній захист роботи		
10	Подання роботи в деканат		

Студент_____ Сирота М.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____Ткаленко О.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи: - сторінки, - рисунків, - таблиці, - джерел.

Об'єкт дослідження – автоматизована конфігурація віртуальних серверів.

Предмет дослідження – технологія автоматизованої конфігурації серверів.

Мета роботи – розгорнути та с конфігурувати віртуальні сервери.

Методи дослідження – теорія ОС Linux, теорія конфігурування серверів.

В роботі досліджено архітектуру віртуального серверу, проаналізовано існуючі засоби автоматизованої конфігурації віртуальних серверів на базі ОС Linux.

Також проведено порівняння існуючих засобів автоматизованої конфігурації віртуальних серверів. Розгорнуто віртуальне середовище. Налаштовано файл Playbook для автоматизованої конфігурації.

Галузь використання – компанії, що розробляють ПЗ, та компанії, яким необхідна власна структура серверів.

АВТОМАТИЗАЦІЯ, ВІРТУАЛІЗАЦІЯ, ОС LINUX, КОНФІГУРАЦІЯ ВІРТУАЛЬНИХ СЕРВЕРІВ, КОНФІГУРАЦІЙНИЙ ФАЙЛ.

ЗМІСТ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ 1
ПЕРЕЛІК УМОВНИХ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ ПОЗНАЧЕНЬ 8
ВСТУП 9
1 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ АВТОМАТИЗОВАНОГО РОЗГОРТАННЯ 9
1.1 Огляд існуючих технологій автоматизованої конфігурації серверів 9
1.1.1 Puppet Enterprise 10
1.1.2 Enterprise Chef 11
1.1.3 Ansible 11
1.1.4 SaltStack 15
1.2 Порівняння існуючих технологій автоматизованої конфігурації серверів 17
1.3 Обґрунтування вибору засобу 21
2 ОГЛЯД ТЕХНІЧНИХ ЗАСОБІВ 23
2.1 Огляд операційної системи 23
2.1.1 Ubuntu 23
2.2 Огляд засобів віртуалізації 27
2.2.1 Hyper-V 34
2.2.2 DigitalOcean 36
2.2.3 Hetzner 37
3 РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ КОНФІГУРАЦІЇ 39
ВИСНОВОК 58
Список використаної літератури 59

ПЕРЕЛІК УМОВНИХ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ ПОЗНАЧЕНЬ

VPS	virtual private server
YAML	Yet Another Markup Language
IP	Internet Protocol

ВСТУП

З моменту появи засобів інформаційних технологій пройшло багато часу, і все більша частина людського життя охоплена зручними різновидами комп'ютерної техніки. Наразі неможливо собі уявити життя, в якому була б відсутня така різноманітна кількість гаджетів. Зокрема, велику користь інформатизація та автоматизація приносить багатьом видам бізнесу.

Автоматизація є головним аспектом у будь-якій сфері. В IT-сфері автоматизація відіграє важливу роль у швидкості та якості створення продукту.

Метою даного дипломного проекту є дослідження та практична реалізація автоматизованої конфігурації.

Задачами проекту є:

- дослідити існуючі засоби віртуалізації;
- дослідити існуючі технології автоматизації;
- дослідити існуючі технічні засоби для реалізації проекту;
- обґрунтувати вибір засобу;
- створення конфігураційного файлу вибраними засобами;
- на основі проведених досліджень сформувати остаточні висновки.

Підготовка конфігураційного файлу дозволить спростити роботу фінансової компанії, а також полегшить роботу системного адміністратора через спрощену модель керування пристроями та їхньою взаємодією.

1 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ АВТОМАТИЗОВАНОГО РОЗГОРТАННЯ

1.1 Огляд існуючих технологій автоматизованої конфігурації серверів

Оскільки компанії прагнуть стати більш гнучкими та використовувати досягнення цифровізації, автоматизація IT-процесів перетворилася в певну обов'язкову норму. Автоматизація IT-процесів, яка нещодавно сприймалася як виключно набір простих інструментів або сценаріїв для допомоги в бізнесі, зараз розглядається як стратегічна ініціатива та довгострокова IT-стратегія.

Опитування Harvard Business Review Analytic Services [6] серед 338 керівників компаній у всьому світі підкреслює важливість автоматизації IT: 80% респондентів кажуть, що впровадження IT-автоматизації є «надзвичайно важливим» або «дуже важливим», від цього залежить майбутній успіх їхньої організації.

Серед IT-процесів, які найчастіше автоматизують, є прості рутинні IT-процеси.

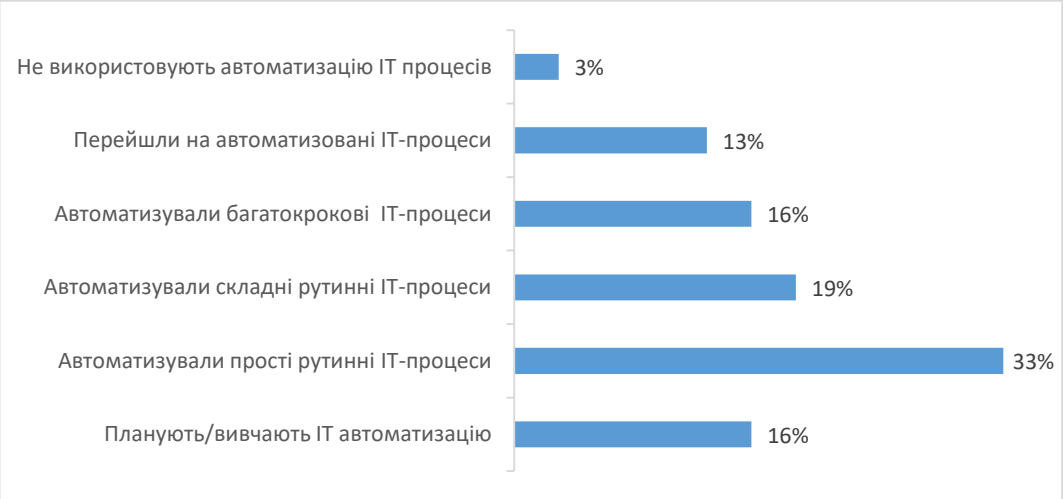

Рисунок 1.1 - Поширення автоматизації IT-процесів

1.1.1 Puppet Enterprise

Puppet вважається найбільш використовуваним із чотирьох. Він найбільш повний з точки зору можливих дій, модулів і інтерфейсів користувача, представляючи повну картину ЦОД, охоплюючи майже кожну операційну систему і надаючи утиліти для всіх основних ОС. Початкова установка відносно проста, вимагає розгортання головного сервера та клієнтських агентів на кожній керованій системі.

Інтерфейс командного рядка дозволяє завантажувати та встановлювати модулі за допомогою команди puppet. Потім потрібні зміни в конфігураційних файлах, необхідні для налаштування модуля під необхідне завдання, а клієнти, які повинні отримати інструкції, отримають їх при наступному зверненні до головного сервера або через запит від сервера, який ініціює процес зміни негайно.

Також є модулі, за допомогою яких виконується налаштування віртуальних та розміщених у «хмарах» серверів. Всі модулі та конфігурації будуються за допомогою вбудованого, заснованого на ruby, мови, або ж на самому ruby. Це потребує деяких знань програмування, крім навичок системного адміністрування.

У Puppet Enterprise найбільш повний веб-інтерфейс з усіх, що дозволяє контролювати керовані вузли в реальному часі за допомогою попередньо створених модулів та «куховарських книг» (cookbooks), розміщених на головних серверах. Вебінтерфейс хороший для керування, але не дозволяє проводити «тонке» налаштування модулів. Інструменти для побудови звітів добре розроблені, надаючи глибоку деталізацію щодо поведінки агентів та про внесені зміни.

1.1.2 Enterprise Chef

Chef схожий на Puppet з точки зору загальної концепції, в ньому також є головний сервер та агенти, встановлені на керованих вузлах. Крім головного сервера, установка Chef також вимагає робочої станції, для управління ним. Агенти можуть бути встановлені з робочої станції за допомогою утиліти knife, яка використовує протокол SSH для розгортання, полегшуючи тягар установки. Після цього керовані вузли аутентифікуються з головним за допомогою сертифікатів.

Конфігурація Chef тісно пов'язана із системою керування версіями Git, тому знання того, як працює Git необхідно для роботи. Так само як і Puppet, Chef заснований на ruby, тому знадобиться і ця мова. Як і у випадку з Puppet. Модулі можуть бути завантажені або написані з нуля, після чого встановлені на керовані вузли, відповідно до необхідних налаштувань.

На відміну від Puppet, Chef поки немає добре реалізованої функції push, хоча доступна бета-версія коду. Це означає, що агенти повинні бути налаштовані на періодичну синхронізацію з головним сервером, і негайне застосування змін неможливе.

Інтерфейс користувача функціональний, але не надає можливості модифікувати конфігурації. Він не такий повний, як веб-інтерфейс Puppet Enterprise, поступається у побудові звітів та деяких інших функціях, але дозволяє вести облік обладнання та організацію вузлів.

Як і у Puppet, Chef має великий набір модулів і рецептів налаштувань, переважно на ruby. З цієї причини Chef добре підходить для інфраструктур, орієнтованих на розробку.

1.1.3 Ansible

Термін «Ansible» був введений Урсулою К. Ле Гуїн у її романі «Світ Роканнона» 1966 року і стосується вигаданих систем миттєвого зв'язку.

Інструмент Ansible був розроблений Майклом ДеХааном, автором серверної програми ініціалізації Cobbler і співавтором фреймворку Fedora Unified Network Controller (Func) для віддаленого адміністрування.

Ansible, Inc. (спочатку AnsibleWorks, Inc.) була компанією, заснованою в 2013 році Майклом Де Хааном, Тімоті Герлою та Саїдом Зіуані для комерційної підтримки та спонсорства Ansible. Red Hat придбала Ansible у жовтні 2015 року.

Ansible входить до складу дистрибутива Fedora Linux, що належить Red Hat, а також доступний для Red Hat Enterprise Linux, CentOS, openSUSE, SUSE Linux Enterprise, Debian, Ubuntu, Scientific Linux і Oracle Linux через додаткові пакети для Enterprise Linux (EPEL), а також для інших операційних систем.

Ansible допомагає керувати кількома машинами, вибираючи частини інвентарю Ansible, що зберігається в простих текстових файлах ASCII. Інвентаризацію можна налаштувати, і інвентар цільової машини можна отримати динамічно або з хмарних джерел у різних форматах (YAML, INI).

Конфіденційні дані можна зберігати в зашифрованих файлах за допомогою Ansible Vaultз 2014 року. На відміну від інших популярних програм для керування конфігураціями, таких як Chef, Puppet, Salt і CFEngine, Ansible використовує безагентну архітектуру, при цьому програмне забезпечення Ansible зазвичай не працює і навіть не встановлюється на контрольованому вузлі. Натомість Ansible керує вузлом, тимчасово встановлюючи та запускаючи модулі на вузлі через SSH. Протягом виконання завдання оркестровки процес, який запускає модуль, спілкується з керуючою машиною за допомогою протоколу на основі JSON через стандартний вхід і вихід. Коли Ansible не керує вузлом, він не споживає ресурси на вузлі, тому що не запускаються демони чи не інсталюється програмне забезпечення.

Ansible вимагає встановлення Python на всіх керуючих машинах, включаючи менеджер пакетів pip разом із програмним забезпеченням для керування конфігурацією та залежними від нього пакетами. Керовані мережеві пристрої не потребують додаткових залежностей і не мають агентів.

Вузол управління

Контрольний вузол (головний хост) призначений для керування (оркестрування) цільових машин (вузлів, які називаються «інвентаризацією», див. нижче). Вузли керування доступні лише для Linux тощо; ОС Windows не підтримуються. Дозволено кілька вузлів керування. Ansible не потребує єдиної керуючої машини для оркестровки, що гарантує просте аварійне відновлення. Вузлами керує керуючий вузол через SSH.

Цілі проектування

Цілі дизайну Ansible включають:

Мінімальний характер. Системи управління не повинні накладати додаткові залежності від середовища.

Послідовний. За допомогою Ansible можна створити узгоджене середовище.

Безпечний. Ansible не розгортає агентів на вузлах. На керованих вузлах потрібні лише OpenSSH і Python.

Надійний. При ретельному написанні підручник Ansible може бути ідемпотентним, щоб запобігти неочікуваним побічним ефектам на керовані системи. Можна писати книги ігор, які не є ідемпотентними.

Потрібно мінімальне навчання. Playbooks використовують просту та описову мову на основі шаблонів YAML та Jinja.

Модулі здебільшого автономні та можуть бути написані на стандартній мові сценаріїв (такій як Python, Perl, Ruby, Bash тощо). Однією з головних цілей модулів є ідемпотентність, яка означає, що навіть якщо операція повторюється кілька разів (наприклад, після відновлення після збою), вона завжди переводитиме систему в той самий стан.

Розташування цільових вузлів вказується за допомогою списків конфігурації інвентаризації (у форматі INI або YAML), розташованих у /etc/ansible/hosts (у Linux). У файлі конфігурації перелічено IP-адресу або ім’я хоста кожного вузла, доступного Ansible. Крім того, вузли можуть бути призначені групам.

Приклад інвентаризації (формат INI):

192.168.6.1
[веб-сервери]
foo.example.com
bar.example.com

Цей конфігураційний файл визначає три вузли: перший вузол визначається IP-адресою, а останні два вузли визначаються іменами хостів. Крім того, останні два вузли згруповано в групі веб-серверів.

Ansible також може використовувати спеціальний сценарій динамічної інвентаризації, який може динамічно отримувати дані з іншої системи і підтримує групи груп.

Playbooks — це файли YAML, які зберігають списки завдань для повторного виконання на керованих вузлах. Кожен Playbook відображає (асоціює) групу хостів із набором ролей. Кожна роль представлена викликами завдань Ansible.

Платформа автоматизації Ansible — це REST API, веб-служба та веб-інтерфейс (додаток), призначені для того, щоб зробити Ansible більш доступним для людей із широким набором IT-навичок. Це композиція платформи

Ansible більше схожий на Salt, ніж Puppet або Chef. Ansible фокусується на оптимізації та швидкості, і не потребує встановлення агентів на керовані вузли – всі функції виконуються за SSH. Ansible написаний на python, на відміну від Puppet і Chef, заснованих на ruby.

Установка Ansible може бути виконана шляхом клонування Git-репозиторію на головний сервер. Слідом за цим, вузли, над якими потрібне управління додаються в конфігурацію Ansible, і авторизовані ключі SSH «прив'язуються» до кожного вузла, ставлячись до користувача від імені якого запускатиметься Ansible. Як тільки це зроблено, головний сервер може з'єднуватися з вузлами протоколу SSH і виконувати всі необхідні завдання. Для роботи з системами, що не дозволяють доступ до прав суперкористувача (root) по SSH, Ansible використовує

облікові дані, що дозволяють виконувати дії від імені суперкористувача за допомогою команди sudo.

Ansible може використовувати Paramiko - реалізацію протоколу SSH2 на мові python, або стандартну реалізацію SSH, але є також прискорений режим для більш швидкої та широкомасштабної комунікації.

Ansible може бути запущений з командного рядка без використання конфігураційних файлів для простих завдань, таких як перевірка, що якийсь сервіс запущено, або оновлення тригерів і перезавантаження. Для комплексних завдань, конфігураційні файли створюються з допомогою YAML і називаються «сценарії» (playbook). Вони можуть бути використані шаблони для розширення функціональності.

У Ansible є набір модулів, які можуть використовуватися для управління різними системами, так само як і «хмарними» інфраструктурами, такими як Amazon EC2 та OpenStack. Додаткові модулі можуть бути написані практично будь-якою мовою програмування за умови, що висновок буде у форматі JSON.

Веб-інтерфейс доступний як AnsibleWorks AWX, але безпосередньо не пов'язаний з інтерфейсом командного рядка. Це означає, що елементи конфігурації, задані через командний рядок, не з'являться у веб-інтерфейсі, доки не буде запущено цикл синхронізації. Ви можете використовувати вбудовану утиліту, але необхідно запланувати її регулярний запуск. Сам по собі веб-інтерфейс досить функціональний, але не такий повний, як інтерфейс командного рядка, таким чином ви або перемикатиметеся між обома, або ж просто використовувати командний рядок.

1.1.4 SaltStack

Salt схожий з Ansible у тому, що заснований на командному рядку. Він використовує метод push для зв'язку із клієнтами. Він може бути встановлений через Git або через систему керування пакетами на головному сервері та клієнтах.

Клієнт робить запит до головного сервера, і якщо той дає дозвіл, дозволяє керувати цим вузлом за допомогою агента (у термінах Salt minion).

Salt може зв'язуватися з клієнтами за протоколом SSH, але масштабованість значно розширюється за рахунок клієнтських агентів. Також, Salt включає асинхронний файловий сервер для прискорення обслуговування агентів, дозволяючи створювати системи, що добре масштабуються.

Як і у випадку Ansible, можна віддавати команди, такі як запуск сервісів або встановлення пакетів агентам безпосередньо з командного рядка, або використовувати конфігураційні файли у форматі YAML (state), для обробки комплексних завдань. Також є централізовано розміщені набори даних (pillar) яких мають доступ конфігураційні файли під час роботи.

Ви можете запросити інформацію про конфігурацію, таку як версія ядра або детальну інформацію про мережний інтерфейс, безпосередньо від агентів через командний рядок. Агенти можуть також задаватися через використання елементів інвентарю, які називаються «зернами» (grain), що дозволяють легко передавати команди на певні сервери, безвідносно до налаштованих груп. Наприклад, однією командою можна надіслати запит до агентів, які розташовані на серверах з певною версією ядра.

Як і попередні продукти, Salt надає велику кількість модулів для різноманітного програмного забезпечення, операційних систем та «хмарних» сервісів. Допоміжні модулі можуть бути написані мовами Python або PyDSL. Salt надає можливість керувати і Windows-вузлами, так само як і Unix, але більше розрахований на системи Unix та Linux.

Веб-інтерфейс Salt - Halite - занадто новий і не повний, як інтерфейси користувачів інших систем. З його допомогою можна переглядати системні журнал повідомлень та статус керованих вузлів, а також є можливість виконувати на них команди. Цей інструмент активно розробляється, і обіцяє значні покращення, але поки це голий «скелет» і містить багато помилок.

Найбільша перевага Salt – масштабованість та гнучкість. У вас може бути кілька рівнів головних серверів, що організують пов'язану структуру, для

забезпечення розподілу навантаження та збільшення відмовостійкості. Головні сервери верхнього рівня можуть управляти нижчими в ієрархії та їх підлеглими вузлами. Інша перевага - однорангова система обміну повідомленнями, що дозволяє підлеглим вузлам ставити питання головним, а ті можуть отримувати відповіді від інших серверів для завершення картини. Це може бути корисним, якщо дані для завершення установки вузла знаходяться в базі даних реального часу.

1.2 Порівняння існуючих технологій автоматизованої конфігурації серверів

Якщо Puppet та Chef орієнтовані на розробників, то Salt та Ansible більше підходять для потреб системних адміністраторів. Простий інтерфейс та зручність використання Ansible підходять мисленню сісадмінів у компаніях з великою кількістю Unix та Linux систем. Ansible швидко і легко запускається «з коробки».

Salt найнадійніший із чотирьох і теж підійде системним адміністраторам. Добре масштабований і достатній функціонал, лише веб-інтерфейс залишає бажати кращого.

Puppet найбільш зрілий і, ймовірно, найдоступніший з чотирьох продукт, з точки зору зручності використання, хоча й настійно рекомендуються ґрунтовні знання ruby. Puppet не настільки оптимізований як Ansible або Salt, і його конфігурація часом може нагадувати «фількіну грамоту». Puppet найбільш безпечний для гетерогенного оточення, але ви можете вважати Ansible або Salt більш підходящим для великих або більш гомогенних інфраструктур.

Chef є стабільним і добре опрацьованим рішенням, але поки не дотягує до рівня Puppet з точки зору основних функцій, проте його можна розширити. Chef може бути важким вивчення адміністраторами з недостатнім досвідом у програмуванні, але може підійти компаніям, які займаються розробкою ПЗ.

Puppet 3.0

Плюси:

- Модулі можуть бути написані на ruby, або більш простою, похідною від ruby мовою;
- Команди Push дозволяють застосовувати зміни негайно;
- Веб-інтерфейс підтримує звіти, інвентаризацію та керування вузлами в реальному часі;
- Деталізовані звіти про роботу агентів та конфігурацію вузлів.

Мінуси:

- Потрібно вивчення вбудованої мови або ruby;
- Процесу встановлення немає звітів про помилки.

Ціна:

- Безкоштовна версія з відкритим вихідним текстом;
- Puppet Enterprise коштує $100 за комп'ютер на рік.

Chef 11.4

Плюси:

- «Поварені книги» та рецепти використовують усю міць ruby;
- Централізовані, засновані на JSON масиви даних, дозволяють скриптам заповнювати змінні під час роботи;
- Веб-інтерфейс дозволяє вести пошук та облік вузлів, переглядати їх активність, застосовувати «кухонні книги» та ролі.

Мінуси:

- Потрібне знання ruby;
- На даний момент немає функціональних команд push;
- Документація місцями неясна.

Ціна:

- Безкоштовна версія з відкритим вихідним кодом;
- Enterprise Chef безкоштовний для 5 комп'ютерів, $120 за місяць для 20 комп'ютерів, $300 за місяць для 50 комп'ютерів, $600 за місяць для 100 і так далі.

Ansible 1.3

Плюси:

- Модулі можуть бути написані майже будь-якою мовою;
- Не потрібні агенти на керованих вузлах;
- Веб-інтерфейс дозволяє настроювати користувачів, команди та обладнання, застосовувати сценарії;
- Дуже просто налаштовується та запускається.

Мінуси:

- Бракує підтримки клієнтів для Windows;
- Веб-інтерфейс автоматично не пов'язується з наявною установкою Ansible - дані мають бути імпортовані.

Ціна:

- Безкоштовна версія з відкритим вихідним кодом;
- AWX безкоштовний для 10 комп'ютерів, далі $100 або $250 за комп'ютер на рік, залежно від підтримки.

Salt 0.17

Плюси:

- Конфігураційні файли можуть бути простими YAML-шаблонами або скриптами на pyhton та PyDSL;
- Може зв'язуватися з клієнтами через SSH або за допомогою локально встановлених агентів;
- Веб-інтерфейс дозволяє переглядати запущені завдання, статус підлеглих вузлів та дозволяє виконувати команади на клієнтах;
- Вкрай добре масштабується.

Мінуси:

- Веб-інтерфейс не такий зрілий та повний як у конкурентів;
- Не вистачає інструментів для детальних звітів.

Ціна:

- Безкоштовна версія з відкритим вихідним кодом;
- SaltStack Enterprise коштує $150 за вузол на рік, зі знижками в залежності від кількості вузлів та корпоративними ліцензіями.

Загальна порівняльна характеристика засобів наведена у Таблиці 1.2.1.

Інструмент	Особливості
Puppet	Широка підтримка Рекомендовані навички написання коду Розширення за допомогою Ruby Потребує встановлення агентів на всіх клієнтах
Chef	Інтегрований з Git Рекомендовані навички написання коду Складність у вивченні Широка підтримка Потребує встановлення chef-агентов на всіх клієнтах
AnsiBle	Зручний для системних адміністраторів На базі Python Не потребує коду, нема агентів на хості Прості, швидкі з’єднання працюють через SSH Запуск через текстові файли Досить простий для вивчення
Salt	Працює через агенти Зданий до масштабування Зручний для системних адміністраторів

Таблиця 1.2.1 - Загальна порівняльна характеристика технологій автоматизації

1.3 Обґрунтування вибору засобу

Звичайно, кожен інструмент має ідеальні випадки використання, у яких він проявляється яскравіше, ніж інші. Наприклад, багато хто вважає Ansible простим у вивченні та використанні – його підручники читаються/зрозумілі людині, що дозволяє досягти результатів за коротший проміжок часу. Тим не менш, простота пропозиції може змусити досвідчених користувачів забажати більшої витонченості.

Простий/легкий у навчанні

Це, мабуть, найуспішніша характеристика Ansible: користувачі можуть швидко навчитись і продуктивно працювати з інструментом. Завдяки чіткій та зрозумілій документації можна за короткий проміжок часу вивчити робочий процес і логіку операцій Ansible. Відсутність системи залежностей означає, що завдання Ansible просто виконуються послідовно та зупиняються, коли виникає помилка. Це спрощує пошук несправностей, особливо на початку роботи з інструментом.

Ansible був написаний на Python, на відміну від інших конкуруючих рішень, створених за допомогою таких мов, як Ruby. Отже, запустити його та запустити легше, оскільки бібліотеки Python за замовчуванням присутні в більшості дистрибутивів Linux. Це також мова, яка більш поширена для завдань адміністрування та сценаріїв: інженери та системні адміністратори, швидше за все, знають Python, ніж Ruby. Однак модулі Ansible для розширення функціональності інструменту можна написати будь-якою мовою, якщо вони повертають дані у форматі JSON.

Для керування вузлами Ansible обробляє всі зв’язки між головним/агентом за допомогою стандартного SSH або модуля Paramiko, який є реалізацією SSH2 на Python. Інструмент не потребує встановлення будь-яких агентів на віддалених системах для керування, що означає менше витрат на обслуговування та збільшення продуктивності.

Playbooks – конфігураційні файли Ansible – написані на YAML, який краще підходить для керування конфігурацією та автоматизації, ніж інші формати, такі як JSON. Його легше читати, він підтримує коментарі та використовує прив’язки для посилань на інші елементи.

Існує також портал “Ansible Galaxy”, який служить центральним сховищем для пошуку, повторного використання та обміну вмістом Ansible. Наприклад, багаторазові ролі для конфігурації сервера або встановлення додатків можна завантажити для використання в своїх плейбуках, що значно прискорює час розгортання.

Враховуючи вище перераховані плюси, для автоматизованої конфігурації віртуальних серверів, було обрано Ansible.

2 ОГЛЯД ТЕХНІЧНИХ ЗАСОБІВ

2.1 Огляд операційної системи

2.1.1 Ubuntu

Ubuntu — це дистрибутив Linux, заснований на Debian і складається переважно з безкоштовного програмного забезпечення з відкритим вихідним кодом. Ubuntu офіційно випущено в трьох версіях: Desktop,Server і Core для пристроїв Інтернету речей і роботів. Усі випуски можуть працювати як на комп'ютері, так і на віртуальній машині. Ubuntu — популярна операційна система для хмарних обчислень із підтримкою OpenStack. У 2017 році робочий стіл Ubuntu за замовчуванням змінився з внутрішньої Unity на GNOME майже через 6,5 років після випуску версії 17.10.

Ubuntu випускається кожні шість місяців, з довгостроковою підтримкою (LTS) кожні два роки. Станом на жовтень 2022 року найновішим випуском є 22.10 («Kinetic Kudu»), а поточним довгостроковим випуском підтримки є 22.04 («Jammy Jellyfish»).

Ubuntu розробляється британською компанією Canonical та спільнотою інших розробників за моделлю меритократичного управління. Canonical надає оновлення безпеки та підтримку для кожного випуску Ubuntu, починаючи з дати випуску й доки випуск не досягне визначеної дати завершення терміну служби (EOL). Canonical отримує дохід за рахунок продажу преміальних послуг, пов'язаних з Ubuntu, і пожертвувань від тих, хто завантажує програмне забезпечення Ubuntu.

Ubuntu названо на честь філософії ubuntu Nguni, яка, за словами Canonical, означає «людяність до інших» із відтінком «Я такий, який я є, тому що ми всі є».

Ubuntu побудовано на архітектурі та інфраструктурі Debian і включає серверну, настільну та зняту з виробництва версію операційної системи Linux для телефонів і планшетів. Ubuntu випускає оновлені версії передбачувано кожні шість місяців[30], і кожен випуск отримує безкоштовну підтримку протягом дев'яти

місяців (вісімнадцять місяців до 13.04)[31] з виправленнями безпеки, виправленнями серйозних помилок і консервативними, суттєво корисними виправленнями помилок з низьким рівнем ризику. [32] Перший випуск відбувся в жовтні 2004 року.

Поточні випуски довгострокової підтримки (LTS) підтримуються протягом п’яти років і випускаються кожні два роки. З моменту випуску Ubuntu 6.06 кожен четвертий випуск отримує довгострокову підтримку.[30] Довгострокова підтримка включає оновлення для нового апаратного забезпечення, виправлення безпеки та оновлення «стеку Ubuntu» (інфраструктури хмарних обчислень). Перші випуски LTS підтримувалися протягом трьох років на робочому столі та п’яти років на сервері; починаючи з Ubuntu 12.04 LTS, підтримку випусків LTS на робочому столі також було збільшено до п’яти років. Випуски LTS отримують регулярні точкові випуски з підтримкою нового обладнання та інтеграцією всіх оновлень, опублікованих у цій серії на сьогоднішній день.

Пакунки Ubuntu базуються на пакетах із нестабільної гілки Debian, які синхронізуються кожні шість місяців. Обидва дистрибутиви використовують формат пакетів deb Debian та інструменти керування пакетами (наприклад, програмне забезпечення APT і Ubuntu). Пакунки Debian і Ubuntu не обов'язково сумісні один з одним у бінарному вигляді, тому для використання в Ubuntu пакунки може знадобитися перезбирати з вихідного коду. Багато розробників Ubuntu також супроводжують ключові пакети в Debian. Ubuntu співпрацює з Debian, повертаючи зміни до Debian, хоча критикують, що це відбувається не досить часто. Ян Мердок, засновник Debian, висловив занепокоєння з приводу того, що пакети Ubuntu потенційно надто відрізняються від Debian, щоб залишатися сумісними. Перед випуском пакунки постійно імпортуються з Debian unstable і об’єднуються зі специфічними для Ubuntu модифікаціями. За місяць до випуску імпорт заморожено, а розробники пакетів працюють над тим, щоб заморожені функції добре взаємодіяли.

Зараз Ubuntu фінансується Canonical Ltd. 8 липня 2005 року Марк Шаттлворт і Canonical оголосили про створення Ubuntu Foundation і надали початкове

фінансування в розмірі 10 мільйонів доларів США. Метою фонду є забезпечення підтримки та розробки для всіх майбутніх версій Ubuntu. Марк Шаттлворт описує основну мету як забезпечення безперервності проекту Ubuntu.

12 березня 2009 року Ubuntu оголосила про підтримку розробниками сторонніх хмарних платформ керування, таких як ті, що використовуються в Amazon EC2.

32-розрядні процесори x86 підтримувалися до Ubuntu 18.04. Було вирішено підтримувати «застаріле програмне забезпечення», тобто вибрати 32-розрядні пакети i386 для Ubuntu 19.10 і 20.04 LTS.

Стандартна інсталяція Ubuntu містить широкий спектр програмного забезпечення, яке включає LibreOffice, Firefox, Thunderbird, Transmission і кілька легких ігор, таких як Sudoku і Mines. Багато додаткових програмних пакетів доступні за допомогою вбудованого програмного забезпечення Ubuntu (раніше Ubuntu Software Center), а також будь-яких інших інструментів керування пакетами на основі APT. Багато додаткових програмних пакетів, які більше не встановлені за замовчуванням, наприклад Evolution, GIMP, Pidgin і Synaptic, все ще доступні в репозиторіях і можуть бути встановлені за допомогою основного інструменту або будь-якого іншого інструменту керування пакетами на основі APT. Також доступні крос-дистрибутивні пакети знімків і плоских пакетів, які дозволяють інсталювати програмне забезпечення, наприклад, програмне забезпечення Microsoft, у більшості основних операційних систем Linux (таких як будь-яка версія Ubuntu, яка зараз підтримується, і у Fedora). Файловим менеджером за замовчуванням є GNOME Files, раніше називався Nautilus.

Усе програмне забезпечення, встановлене за замовчуванням, є безкоштовним. Крім того, Ubuntu перерозповсюджує деякі драйвери апаратного забезпечення, які доступні лише у двійковому форматі, але такі пакунки чітко позначені в обмеженому компоненті.

Ubuntu прагне бути безпечним за замовчуванням. Програми користувача працюють із низькими привілеями та не можуть пошкодити операційну систему чи файли інших користувачів. Для підвищення безпеки інструмент sudo

використовується для призначення тимчасових привілеїв для виконання адміністративних завдань, що дозволяє обліковому запису root залишатися заблокованим і допомагає запобігти недосвідченим користувачам від випадкового внесення катастрофічних змін у систему або відкриття дірок у безпеці. Polkit також широко впроваджується в робочий стіл.

Більшість мережевих портів закрито за замовчуванням, щоб запобігти злому. Вбудований брандмауер дозволяє кінцевим користувачам, які встановлюють мережеві сервери, контролювати доступ. GUI (GUI для нескладного брандмауера) доступний для його налаштування. Ubuntu компілює свої пакунки, використовуючи функції GCC, такі як PIE і захист від переповнення буфера, щоб посилити своє програмне забезпечення. Ці додаткові функції значно підвищують безпеку за рахунок зниження продуктивності на 0,01% у 64-бітній версії.

Ubuntu також підтримує повне шифрування диска, а також шифрування домашніх і приватних каталогів.

Системні вимоги залежать від продуктів Ubuntu. Для настільної версії Ubuntu 22.04 LTS рекомендується ПК із двоядерним процесором принаймні 2 ГГц, 4 ГБ оперативної пам’яті та 25 ГБ вільного місця на диску. Для менш потужних комп’ютерів існують інші дистрибутиви Ubuntu, такі як Lubuntu та Xubuntu. Ubuntu також підтримує архітектуру ARM. Він також доступний на Power ISA, тоді як старіша архітектура PowerPC неофіційно підтримувалася , а тепер підтримуються нові процесори Power ISA (POWER8). Також офіційно підтримується архітектура x86-64 (AMD64).

Живі образи є типовим способом для користувачів оцінити та згодом інсталювати Ubuntu. Їх можна завантажити як образ диска (.iso), а потім записати на DVD або флеш-накопичувач USB і потім завантажити. Інші методи включають запуск живої версії через UNetbootin або Startup Disk Creator (попередньо встановлений інструмент на Ubuntu, доступний на машинах, на яких уже працює ОС) безпосередньо з USB-накопичувача (створення, відповідно, живого DVD або живого USB-носія). Запуск Ubuntu у такий спосіб повільніший, ніж запуск із жорсткого диска, але не змінює комп’ютер, якщо користувач спеціально не вказав

це. Якщо користувач вирішить завантажити живий образ, а не запустити інсталятор під час завантаження, все ще є можливість використовувати інсталятор під назвою Ubiquity для встановлення Ubuntu після завантаження в живе середовище.Образи дисків усіх поточних і минулих версій доступні для завантаження на веб-сайті Ubuntu.

Крім того, інсталяції флеш-накопичувача USB можна використовувати для завантаження Ubuntu і Kubuntu таким чином, щоб дозволити постійне збереження налаштувань користувача та перенесення встановленої через USB системи між фізичними машинами (однак BIOS комп'ютера має підтримувати завантаження з USB). У новіших версіях Ubuntu програму створення Ubuntu Live USB можна використовувати для встановлення Ubuntu на USB-накопичувачі (з Live CD або DVD або без них). Створити завантажувальний USB-накопичувач із збереженням так само просто, як перетягнути повзунок, щоб визначити, скільки місця зарезервувати для збереження; для цього Ubuntu використовує casper.

2.2 Огляд засобів віртуалізації

На Рис.2.1 наведено загальну архітектуру віртуального серверу.

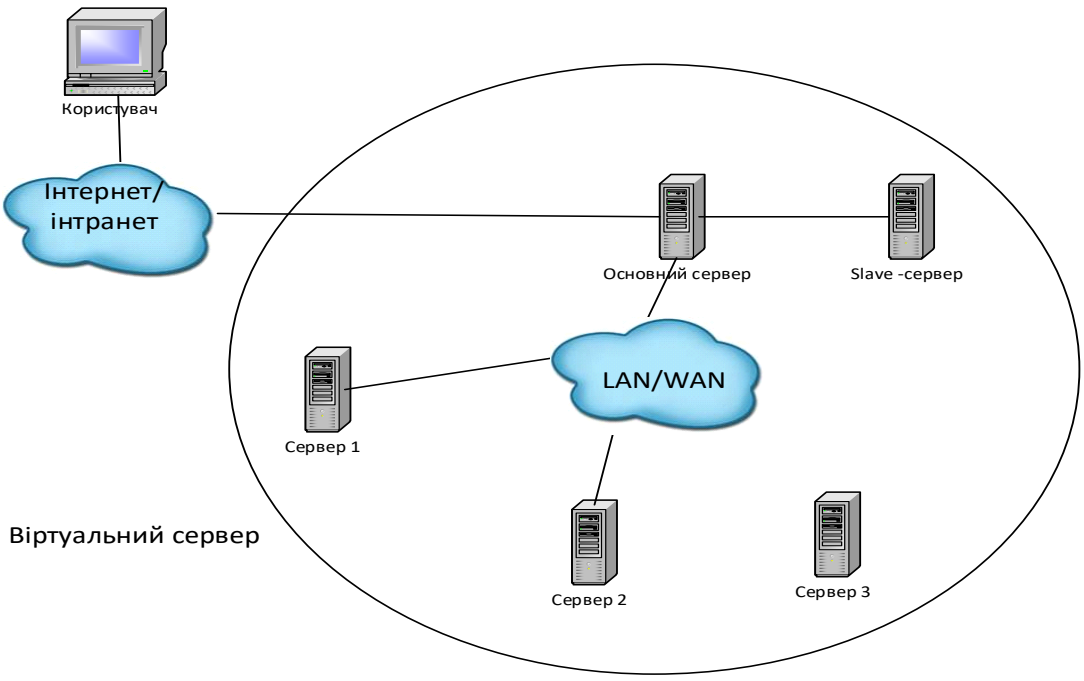

Рисунок 2.1 - Компоненти віртуального серверу

Віртуальний сервер – набір компонентів інтегрованого програмного забезпечення, за допомогою якого створюють виділені віртуальні середовища та розподіляється навантаження між реальними серверами. Віртуальний сервер – високодоступний сервер, що легко масштабується, і побудований на кластері реальних серверів. Архітектура серверного кластеру повністю прозора для кінцевого користувача, і користувачі можуть взаємодіяти з кластерною системою так, ніби з окремим реальним сервером. Чисельні запити від кінцевих користувачів поступають на віртуальний сервер і він відіграє роль балансувальника навантаження. Віртуальний сервер розподіляє запити між реальними серверами і таким чином розвантажує реальні сервери.

Реальні сервери мають однакову конфігурацію, вони не мають зв'язку з зовнішнім середовищем і мають лише private IP. Ці сервери комунікують з іншими реальними серверами та з основним сервером (балансувальником навантаження). Основний сервер має чотири IP-адреси – одну public IP-адресу для кластеру, другу private IP-адресу – для взаємодії з реальними серверами. Крім того, public IP-адресу та private IP -адресу для веб-серверу. Причому останні дві IP-адреси конфігуруються у конфігураційному файлі, і називаються віртуальними. На випадок, якщо основний сервер вийде з ладу, кластер має так званий slave-сервер. Взаємодія між основним та slave-сервером відбувається завдяки активації наступних технологій:

- Pulse service
- Heartbeat
- Nanny process
- Ipvsadm

Pulse service - це процес керування, який запускає всі інші образи, пов'язані з маршрутизаторами кластера. Під час завантаження демон запускається сценарієм /etc/rc.d/init.d/pulse. Потім він читає файл конфігурації /etc/sysconfig/ha/lvs.cf. На активному маршрутизаторі pulse server запускає віртуальний сервер Linux. На резервному маршрутизаторі імпульс визначає працездатність активного

маршрутизатора, передаючи простий контрольний сигнал із заданим користувачем інтервалом. Якщо активний маршрутизатор не відповідає після встановленого користувачем інтервалу, він ініціює відмову. Під час відновлення після відмови pulse на резервному маршрутизаторі дає вказівку pulse на активному маршрутизаторі вимкнути всі служби віртуального сервера, запускає програму send_arp, щоб перепризначити плаваючі IP-адреси MAC-адресі резервного маршрутизатора, і запускає віртуальний сервер.

Heartbeat являє собою пакети даних невеликого розміру, призначення яких – перевірка активного стану сервера та його роботи.

Щодо nanny process, то його призначення - перевірка стану сконфігурованого сервісу на реальному сервері та передача віртуальному серверу інформації, що реальний сервер в роботі. Nanny process дозволяє “вимкнути” slave-сервер, який на даний момент не задіяний. Він перезапускає образ послуги, що не була надана, на реальному сервері. Цей процес також змінює таблицю IP-маршрутизації в Linux kernel залежно від конфігурації віртуального серверу.

Ipvsadm використовується для підтримки, контролю таблиці віртуального серверу. Тобто завдяки Ipvsadm віртуальні адреси з основного серверу передаються на slave-сервер і таким чином, slave-сервер вступає в роботу.

Компоненти віртуального серверу Linux представлені на Рис.2.2.

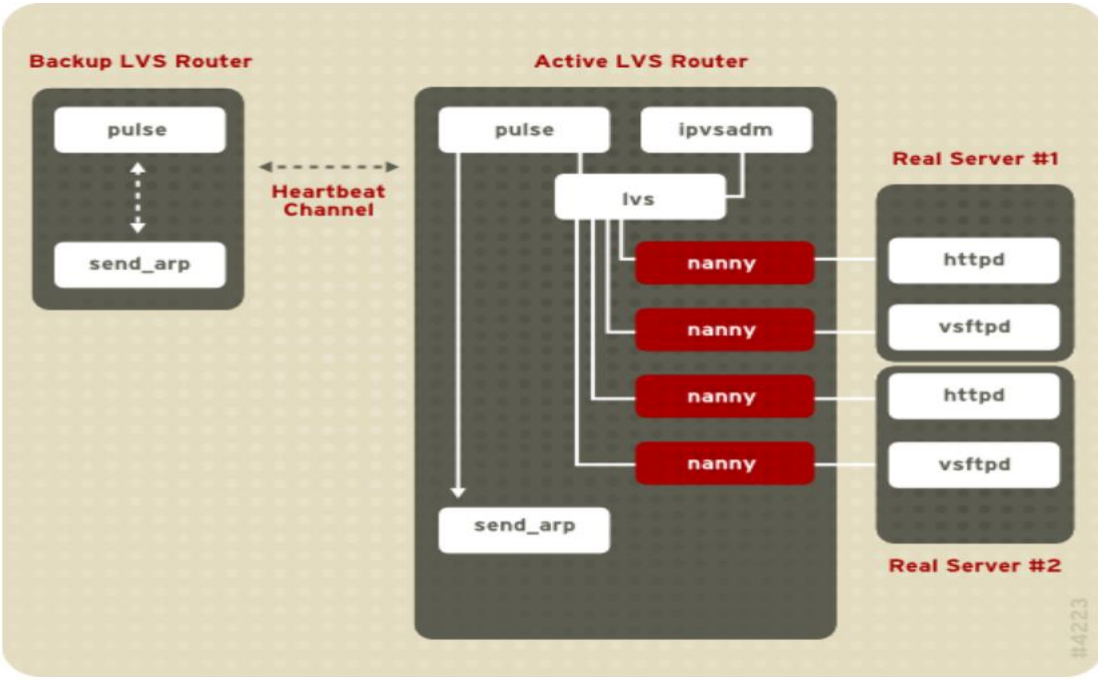

Рисунок 2.2 - Компоненти віртуального серверу Linux

На Рис.2.3 представлена діаграма розгортання віртуального сервера.

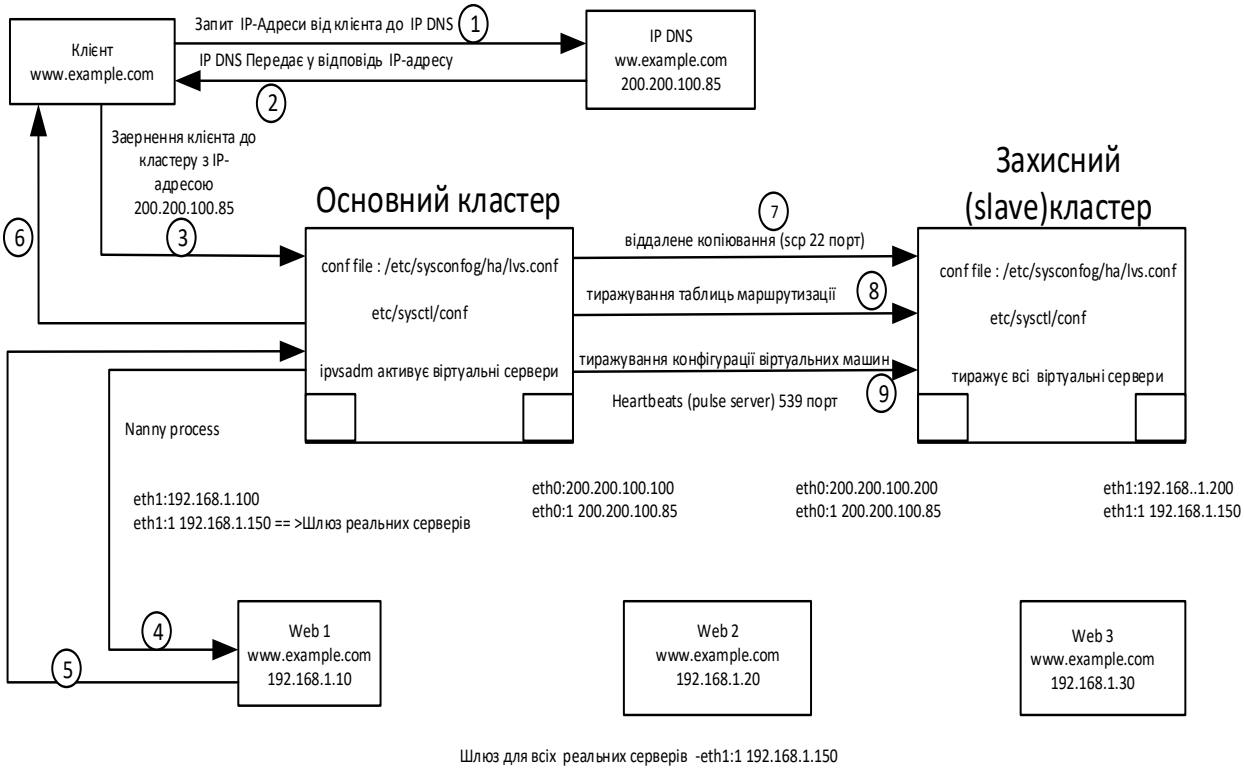

Рисунок 2.3 - Діаграма розгортання віртуального сервера

Маємо два кластери – основний та захисний (slave). Також представлені три реальні сервіси - web1, web2, web3. Всі три web-сервіси мають однакову конфігурацію, і на них працює один веб-сайт. Всі web-сервіси мають private IP і в них нема public IP. Public IP веб-сервіса сконфігурована в основному кластері. В основному сервері працюють дві плати-контролери (NIC -Network Controller Card). Основний сервіс має public IP - eth0:200.200.100.100 – та private IP - eth1:192.168.1.100. Завдяки цій IP-адресі основний сервер може взаємодіяти з реальними веб-сервісами - web1, web2, web3.

Коли сервіс розпочинається на основному сервісі, він бачить тільки дві IP-адреси - eth0, eth1. Але там присутні ще дві IP-адреси - eth0:1 (public IP), eth1:1 (private IP), які є IP-адресами віртуального серверу. І їх також треба додати у конфігураційний файл conf file : /etc/sysconfog/ha/lvs.conf.

Отже, дві public IP-адреси (одна – public IP основного сервера, друга – public IP веб-сервера) конфігуруються командою config відповідного конфігураційного інструмента. А дві віртуальні IP-адреси зазначаються у конфігураційному файлі conf file : /etc/sysconfog/ha/lvs.conf.

Щодо slave-сервера - ми бачимо public IP - eth0:200.200.100.200. Отже, щоб сконфігурувати віртуальний сервер Linux вже потрібно мати три public IP: одна - public IP-адреса основного сервера, друга - public IP-адреса slave-сервера і третя - public IP-адреса веб-сервера.

Під час запуску pulse service активується ipvsadm і ми побачимо чотири IP-адреси. Четвертою IP-адресою в конфігураційному файлі буде private IP веб-сервіса - eth1:192.168.1.100.

Коли клієнт відкриває веб-сайт www.example.com на ISP DNS спрямовується запит на видачу IP-адреси. ISP DNS видає IP-адресу 200.200.100.85, яка є віртуальною адресою веб-сервера. Отримавши цю адресу, клієнт робить запит до основного кластеру. Завдяки конфігураційному файлу etc/sysctl/conf основний сервер функціонує як сервер маршуртизації, тому він спрямує клієнтський запит з IP-адреси eth0:1 200.200.100.85 на з IP-адресу eth0:200.200.100.200, яка є public IP основного серверу. Таким чином почнеться взаємодія, яка полягає в тому, що запит

з IP-адреси eth0:200.200.100.200 спрямується на eth1:192.168.1.100. Відповідно з IP-адреси eth1:192.168.1.100 запит перейде на eth1:1 192.168.1.150, який є шлюзом для реального сервісу.

Таким чином, запит пройде маршрут 3-4. І коли запит досягне веб-сервіс 1, він буде визначений як такий, що надійшов від основного серверу. Пакет запиту не матиме інформації щодо клієнта, який є ініціатором запиту. Натомість, матиме посилання на основний сервер - IP-адреса джерела буде IP-адреса основного сервера. На який веб-сервер потрапить запит - вирішуватиме nanny process, залежно від того, який з веб-серверів на даний момент буде активний. Відповідний активний веб-сервер надішле відповідь основному серверу, а той, відповідно - клієнту. Так само і клієнт не матиме інформації щодо того, з якого веб-сервера надійшла йому відповідь на запит. Клієнт матиме лише дані основного сервера.

Таким чином, це дозволяє ізолювати веб-сервери від мережі Інтернет.

У випадку, коли основний сервер недоступний, всі віртуальні адреси транслюються від основного серверу до slave-серверу.

Віртуальний приватний сервер (VPS) — це віртуальна машина, яка продається як послуга службою Інтернет-хостингу. Подібне значення має і віртуальний виділений сервер (VDS).

Віртуальний приватний сервер запускає власну копію операційної системи (ОС), і клієнти можуть мати доступ до цього екземпляра операційної системи на рівні суперкористувача, тому вони можуть інсталювати майже будь-яке програмне забезпечення, яке працює на цій ОС. Для багатьох цілей він функціонально еквівалентний виділеному фізичному серверу, і, оскільки він визначається програмним забезпеченням, його можна створити та налаштувати набагато легше. Віртуальний сервер коштує набагато дешевше, ніж еквівалентний фізичний сервер. Однак, оскільки віртуальні сервери спільно використовують базове фізичне обладнання з іншими VPS, продуктивність може бути нижчою залежно від робочого навантаження будь-яких інших виконуваних віртуальних машин.

Сила віртуалізації сервера подібна до тієї, яка призвела до розвитку розподілу часу та мультипрограмування в минулому. Незважаючи на те, що ресурси все ще є

спільними, оскільки в моделі розподілу часу віртуалізація забезпечує вищий рівень безпеки, залежно від типу віртуалізації, який використовується, оскільки окремі віртуальні сервери здебільшого ізольовані один від одного і можуть запускати власні повноцінні операційна система, яку можна самостійно перезавантажити як віртуальний екземпляр.

Розбиття одного сервера на кілька серверів стає все більш поширеним на мікрокомп'ютерах після запуску VMware ESX Server у 2001 році. Фізичний сервер зазвичай працює під керуванням гіпервізора, завданням якого є створення, звільнення та керування ресурсами «гостьових» операційних систем, або віртуальні машини. Цим гостьовим операційним системам виділяється частка ресурсів фізичного сервера, як правило, таким чином, що гостьовий сервер не знає про будь-які інші фізичні ресурси, крім тих, які йому надає гіпервізор. Оскільки VPS використовує власну копію своєї операційної системи, клієнти мають доступ до цього екземпляра операційної системи на рівні суперкористувача та можуть інсталювати майже будь-яке програмне забезпечення, яке працює на ОС; однак через кількість клієнтів віртуалізації, які зазвичай працюють на одній машині, VPS зазвичай має обмежений процесорний час, оперативну пам'ять і дисковий простір.

Багато компаній пропонують хостинг віртуального приватного сервера або хостинг віртуального виділеного сервера як розширення послуг веб-хостингу. Під час ліцензування пропрієтарного програмного забезпечення у багатокористувацьких віртуальних середовищах необхідно враховувати кілька проблем.

При некерованому або самокерованому хостингу замовнику залишається адмініструвати власний екземпляр сервера.

Нелімітований хостинг, як правило, пропонується без обмеження обсягу даних, що передаються по лінії з фіксованою пропускною здатністю. Зазвичай безлімітний хостинг пропонується зі швидкістю 10 Мбіт/с, 100 Мбіт/с або 1000 Мбіт/с (декотрі досягають 10 Гбіт/с). Це означає, що клієнт теоретично може використовувати ~3 ТБ на 10 Мбіт/с або до ~300 ТБ на лінії 1000 Мбіт/с на місяць, хоча на практиці значення будуть значно меншими. У віртуальному приватному

сервері це буде спільна смуга пропускання, і слід застосовувати політику справедливого використання. Необмежений хостинг також широко продається, але зазвичай обмежений прийнятною політикою використання та умовами обслуговування. Пропозиції щодо необмеженого дискового простору та пропускної здатності завжди помилкові через вартість, ємність носія та технологічні межі.

Багато фірм надають хостинг віртуального приватного сервера або хостинг виділеного сервера як доповнення до своїх послуг веб-хостингу.

В якості прикладу автоматизованої конфігурації було використано вбудований у Windows гіпервізор Hyper-V.

2.2.1 Hyper-V

Microsoft Hyper-V під кодовою назвою Viridian і коротко відомий до випуску як Windows Server Virtualization — це рідний гіпервізор; він може створювати віртуальні машини в системах x86-64 під керуванням Windows.[2] Починаючи з Windows 8, Hyper-V замінив Windows Virtual PC як компонент апаратної віртуалізації клієнтських версій Windows NT. Комп'ютер-сервер, на якому запущено Hyper-V, можна налаштувати для надання доступу до окремих віртуальних машин одній або декільком мережам. Hyper-V було вперше випущено з Windows Server 2008 і доступно без додаткової плати з Windows Server 2012 і Windows 8. Автономний сервер Windows Hyper-V є безкоштовним, але має лише інтерфейс командного рядка. Останньою версією безкоштовного сервера Hyper-V є Hyper-V Server 2019, який базується на Windows Server 2019.

Hyper-V реалізує ізоляцію віртуальних машин за допомогою розділу. Розділ — це логічна одиниця ізоляції, яка підтримується гіпервізором, у якій виконується кожна гостьова операційна система. В екземплярі гіпервізора має бути принаймні один батьківський розділ із підтримуваною версією Windows Server (2008 і пізніших). Програмне забезпечення віртуалізації працює в батьківському розділі та має прямий доступ до апаратних пристроїв. Батьківський розділ створює дочірні

розділи, на яких розміщуються гостьові ОС. Батьківський розділ створює дочірні розділи за допомогою Hypercall API, який є інтерфейсом програмування додатків, відкритим Hyper-V.

Дочірній розділ не має доступу до фізичного процесора, а також не обробляє його реальні переривання. Замість цього він має віртуальне представлення процесора та працює з гостьовою віртуальною адресою, яка, залежно від конфігурації гіпервізора, може не обов'язково складати весь віртуальний адресний простір. Залежно від конфігурації віртуальної машини Hyper-V може виставляти лише підмножину процесорів для кожного розділу. Гіпервізор обробляє переривання процесора та перенаправляє їх до відповідного розділу за допомогою логічного синтетичного контролера переривань (SynIC). Hyper-V може апаратно прискорити трансляцію адрес гостьових віртуальних адресних просторів, використовуючи трансляцію адрес другого рівня, надану ЦП, що називається EPT на Intel і RVI (раніше NPT) на AMD.

Дочірні розділи не мають прямого доступу до апаратних ресурсів, натомість мають віртуальний перегляд ресурсів у термінах віртуальних пристроїв. Будь-який запит до віртуальних пристроїв перенаправляється через VMBus до пристроїв у батьківському розділі, який керуватиме запитами. VMBus — це логічний канал, який забезпечує зв'язок між розділами. Відповідь також перенаправляється через VMBus. Якщо пристрої в батьківському розділі також є віртуальними пристроями, він буде переспрямований далі, поки не досягне батьківського розділу, де отримає доступ до фізичних пристроїв. Батьківські розділи запускають постачальника послуг віртуалізації (VSP), який підключається до VMBus і обробляє запити на доступ до пристроїв від дочірніх розділів. Віртуальні пристрої дочірнього розділу внутрішньо запускають клієнт служби віртуалізації (VSC), який перенаправляє запит до VSP у батьківському розділі через VMBus. Весь цей процес прозорий для гостьової ОС.

Віртуальні пристрої також можуть використовувати переваги функції віртуалізації Windows Server під назвою Enlightened I/O для зберігання, мережевих і графічних підсистем, серед іншого. Enlightened I/O — це спеціалізована реалізація

комунікаційних протоколів високого рівня, яка підтримує віртуалізацію, як-от SCSI, що дозволяє обходити будь-який рівень емуляції пристрою та безпосередньо використовувати переваги VMBus. Це робить зв'язок ефективнішим, але вимагає, щоб гостьова ОС підтримувала Enlightened I/O.

Зараз лише такі операційні системи підтримують Enlightened I/O, що дозволяє їм працювати швидше як гостьові операційні системи під Hyper-V, ніж інші операційні системи, які потребують повільнішого емулюваного обладнання:

- Windows Server 2008 і новіших версій;
- Windows Vista і пізніші версії;
- Linux з ядром 3.4 або новішої версії;
- FreeBSD.

2.2.2 DigitalOcean

DigitalOcean, LLC – американський постачальник хмарної інфраструктури зі штаб-квартирою в Нью-Йорку з центрами обробки даних по всьому світу. DigitalOcean надає розробникам, стартапам і малим і середнім компаніям платформи хмарної інфраструктури як послуги.

DigitalOcean також проводить Hacktoberfest, одномісячне свято програмного забезпечення з відкритим кодом, що проводиться в жовтні. Щороку він співпрацює з різними розробниками програмного забезпечення, включаючи GitHub, Twilio, Dev.to, Intel, AppWrite і DeepSource.

DigitalOcean пропонує віртуальні приватні сервери (VPS) або «droplets» за термінологією DigitalOcean, використовуючи KVM як гіпервізор і можуть бути створені в різних розмірах (розділені на 2 класи: стандартний і оптимізований) у 13 різних регіонах центрів обробки даних (станом на грудень 2020 р.) і з різними опціями, включно з шістьма дистрибутивами Linux і десятками програм, що працюють одним клацанням миші.

На початку 2017 року DigitalOcean розширив набір функцій, додавши до своєї пропозиції балансувальники навантаження. Їхня платформа є альтернативною

хмарною пропозицією, і компанія націлена на невеликих розробників, дозволяючи їм витрачати лише п’ять доларів на свою платформу.

DigitalOcean можна керувати через веб-інтерфейс або за допомогою командного рядка doctl.

DigitalOcean також пропонує блочне та об’єктне сховище, а з травня 2018 року контейнерний сервіс на основі Kubernetes.

Рецензенти відзначили, що DigitalOcean вимагає від користувачів певного досвіду системного адміністратора та DevOps. У своїй рецензії для ScienceBlogs письменник Грег Ладен попередив: «DigitalOcean не для всіх. Ви повинні бути принаймні трохи підкованими в Linux...».

У 2021 році DigitalOcean запустив керовану службу баз даних MongoDB.

2.2.3 Hetzner

Hetzner Online GmbH – це компанія та оператор центру обробки даних, яка базується в Гунценхаузені, Німеччина.

Її не слід плутати з її колишньою південноафриканською тезкою та партнерською компанією xneelo (раніше Hetzner (Pty) Ltd) — дві окремі компанії зареєстровані на власні права відповідно до чинного законодавства країни. У них не однакові акціонери.

Проекти центрів обробки даних Hetzner Online координуються та впроваджуються власними силами з мінімальним залученням аутсорсингу. Блоки центру обробки даних обслуговуються кількома резервованими висхідними лініями, включаючи 1300 Гбіт/с до DE-CIX і оптоволоконними лініями зв’язку з Нюрнбергом і Франкфуртом. Об’єкти спільного розташування розташовані в усіх парках центрів обробки даних у Нюрнберзі, Фалькенштейні/Фогтланді в Німеччині та Гельсінкі у Фінляндії.] У 2021 році було відкрито центр обробки даних в Ешберні, штат Вірджинія, що стало першим американським сервером Hetzner.

Магістраль створена у вигляді кільцевої мережі між центрами обробки даних у Нюрнбергу та Фалькенштейні, а також найважливішим місцем Інтернету у

Франкфурті. Усі локації підключені до центральних вузлів обміну, таких як DE-CIX, AMS-IX, DATA-IX і V-IX, через власну волоконно-оптичну мережу компанії. Усі виділені сервери Hetzner мали мінімальний місячний ліміт у 20 ТБ для повної швидкості на своїх серверах з можливістю додаткової плати за повну швидкість після цього моменту, однак з 1 жовтня 2018 року було знято обмеження пропускної здатності на швидкості з’єднання 1 Гбіт/с.

3 РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ КОНФІГУРАЦІЇ

Як вже зазначалося, для прикладу було використано гіпервізор Hyper-V. Для роботи Ansible не важливо де знаходяться віртуальні сервери, локально чи у хмарі - головне, щоб до них був доступ. Нижче наведений приклад створення віртуального серверу в гіпервізору Hype-V.

При створенні віртуального серверу була введена назва цього серверу.

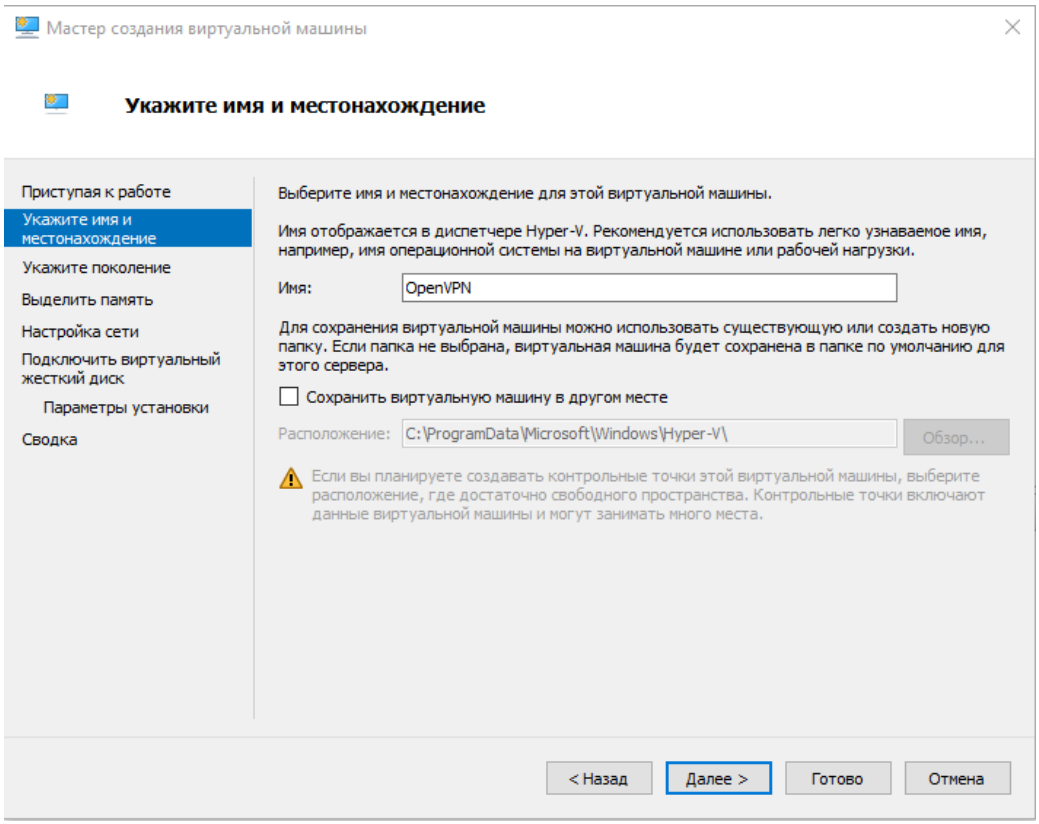

Рисунок 3.1 - Введення назви віртуального серверу

Далі було обрано перше покоління віртуальної машини - це покоління вважається вже застарілим і призначене для запуску більш старих версій операційних систем, але в конкретному випадку, вибір покоління не впливає на роботу ОС, яка встановлюється.

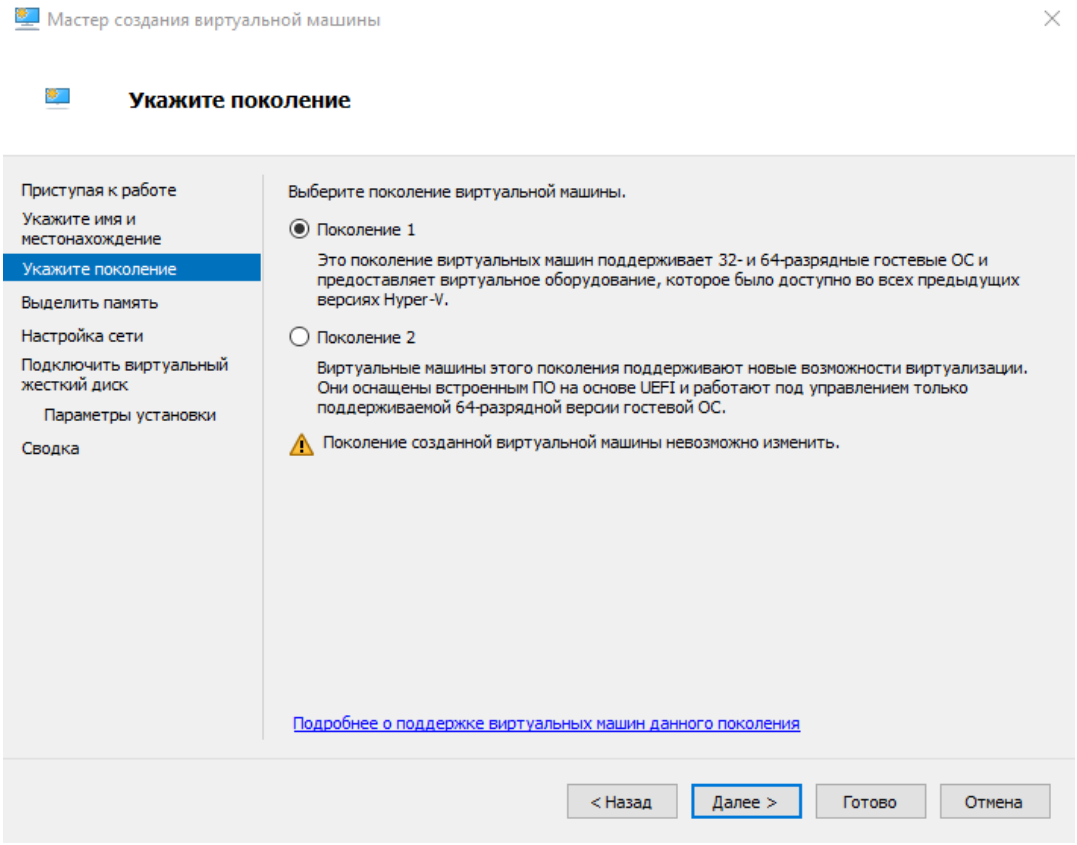

Рисунок 3.2 – Вибір покоління віртуальної машини

На Рис. 3.3 візуалізовано вибір кількості оперативної пам’яті для віртуальної машини. Для виконуваних задач вистачить і 1024мб, але на всякий випадок було включено опцію динамічного розширення оперативної пам’яті - при потребі сервер може використати більше пам’яті, наприклад, при інсталяції програмного забезпечення.

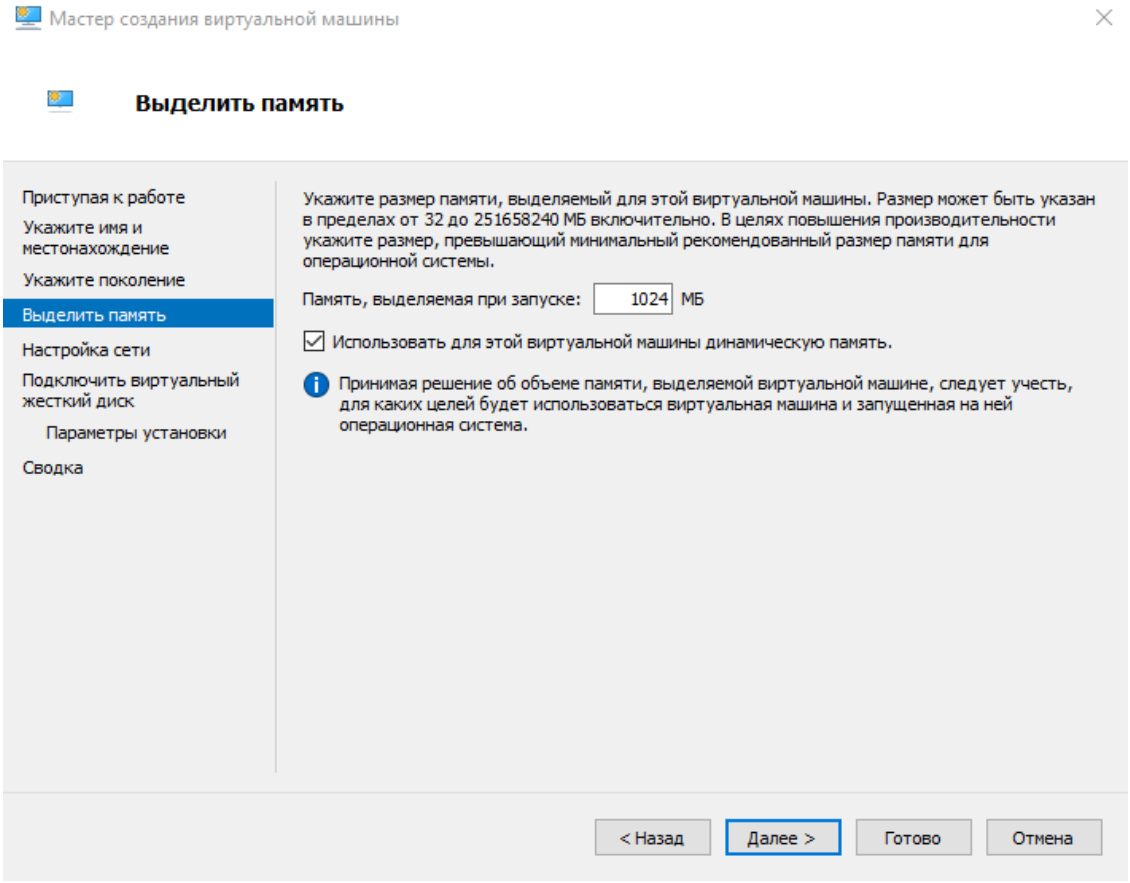

Рисунок 3.3 – Вибір параметрів віртуальної машини

Після цього, необхідно, обрати віртуальний адаптер для підключення віртуальної машини для мережі (Рис. 3.4). За замовчуванням створюється адаптер, який дозволяє ВМ “бачити” один одного, але не дозволяє підключатися до глобальної мережі та з ОС, на якій встановлений гіпервізор (наприклад по ssh). Тому було створено новий віртуальний адаптер “My network”, який дозволяє ВМ підключатися до глобальної мережі.

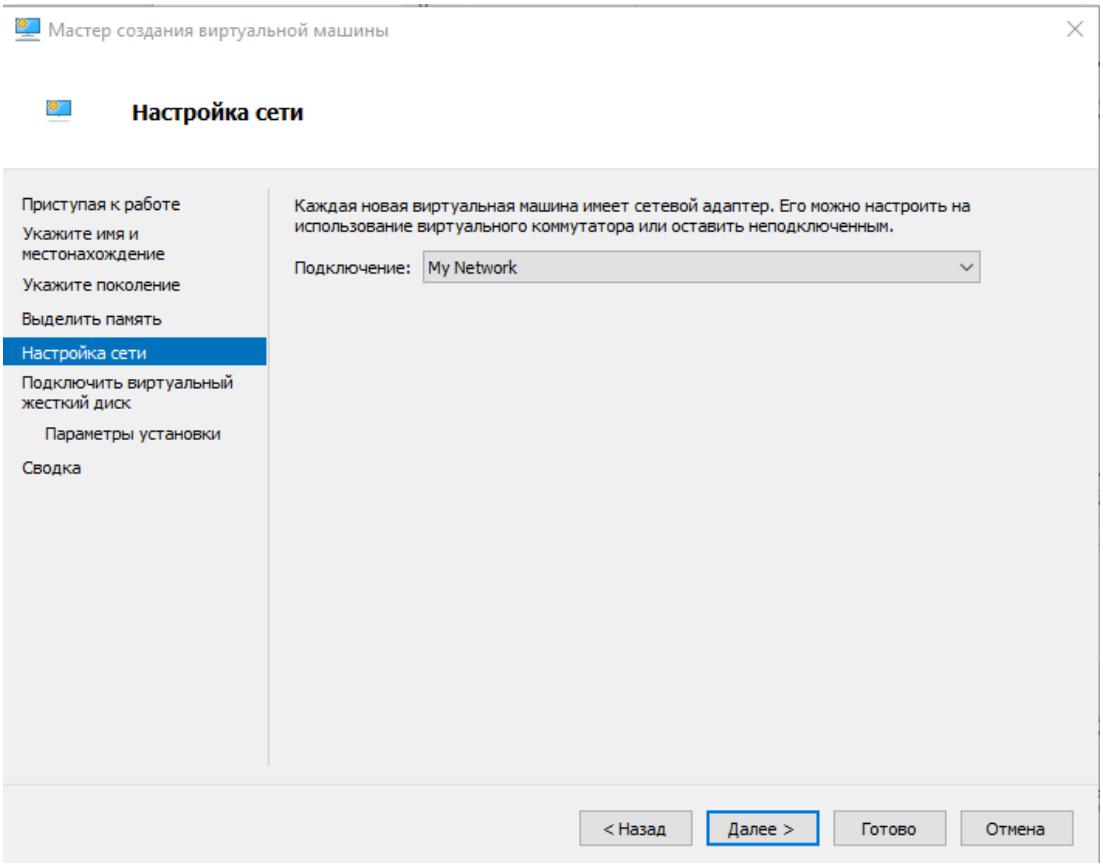

Рисунок 3.4 - налаштування мережевого адаптера

На наступному етапі (Рис. 3.5) було обрано “Файл образу” (ISO-файл) для встановлення самої ОС на віртуальну машину. Після збереження конфігурації, створена ВМ з’явиться в основному списку (Рис. 3.6). Після цього, віртуальна машина була запущена і пройдена стандартна установка ОС. Під час встановлення, було обрано додаткову функцію OpenSSH, для того щоб було одразу підключитися через протокол SSH з основної робочої машини.

В якості операційної системи було обрано дистрибутив Linux Ubuntu Server 22.04 - останню стабільну версію, без UI інтерфейсу, так як всі дії виконуються в терміналі. Ubuntu є досить популярною ОС, яка широко використовується в компаніях.

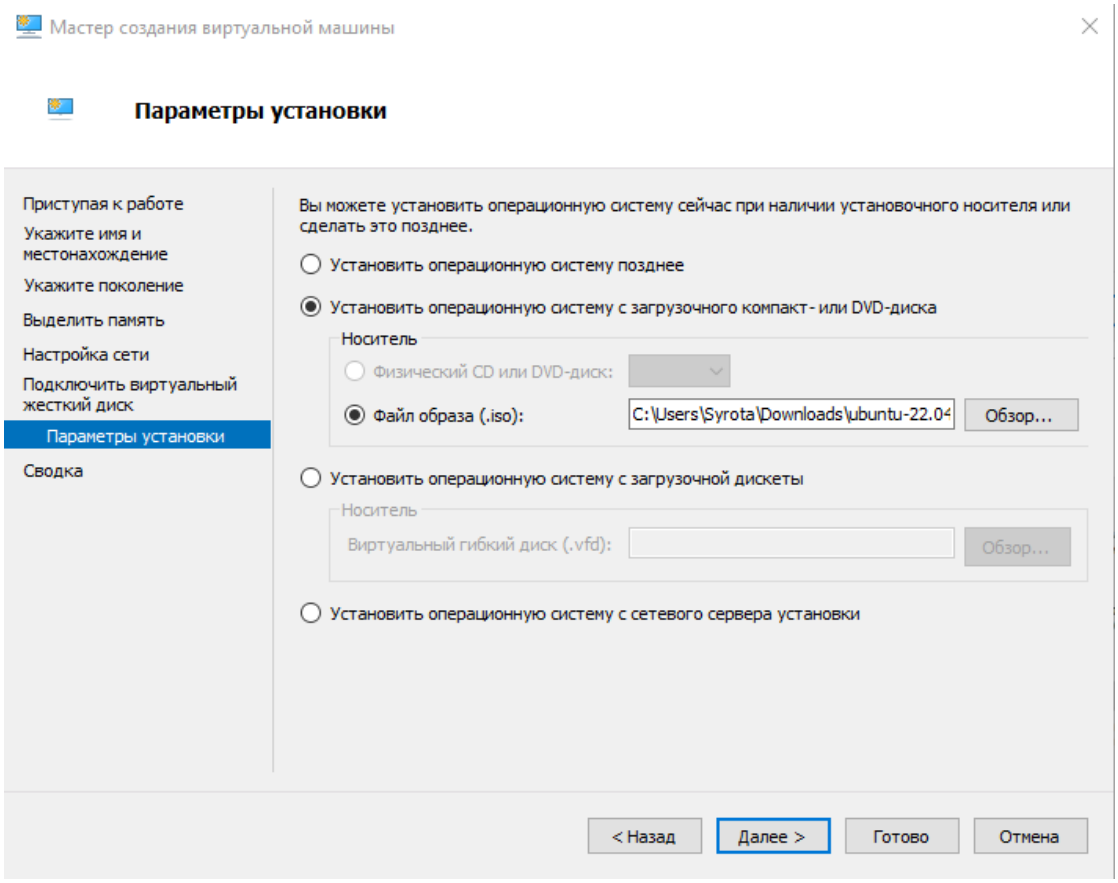

Рисунок 3.5 – Вибір файлу образу

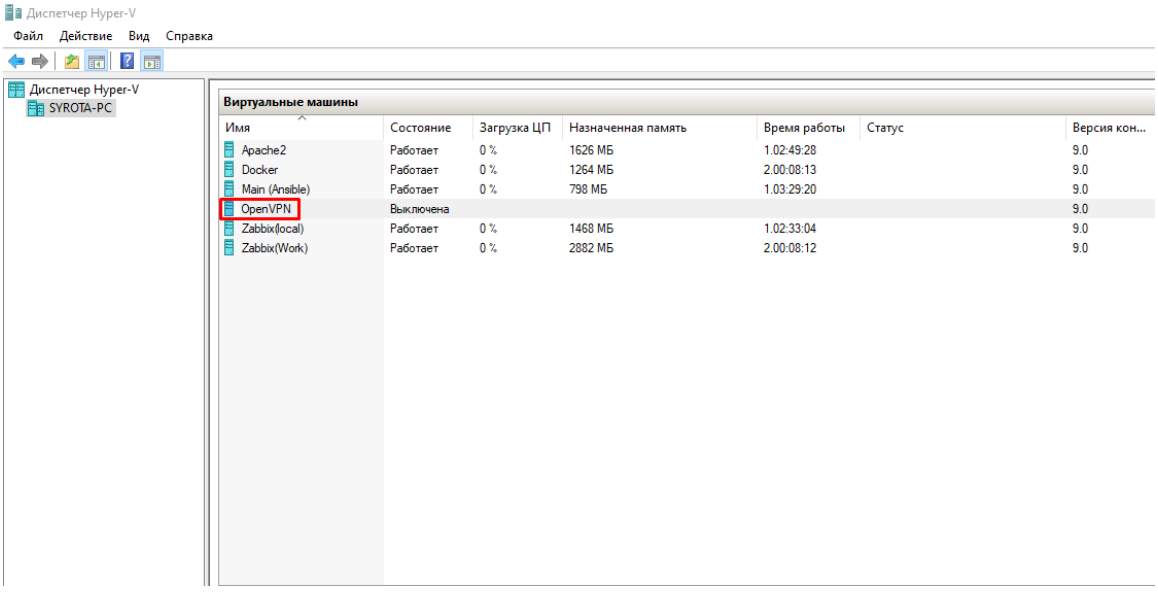

Рисунок 3.6 – Створення віртуальної машини

Для того щоб підключитися до віртуальної машини, на якій буде встановлено Annsible, було використано программу PuTTy. PuTTy - клієнт для протоколів SSH,

Telnet, rlogin і чистого TCP. Для підключення необхідно ввести IP сервера, порт (за замовчуванням порт - 22), ім’я користувача та пароль (Рис. 3.7, Рис. 3.8)

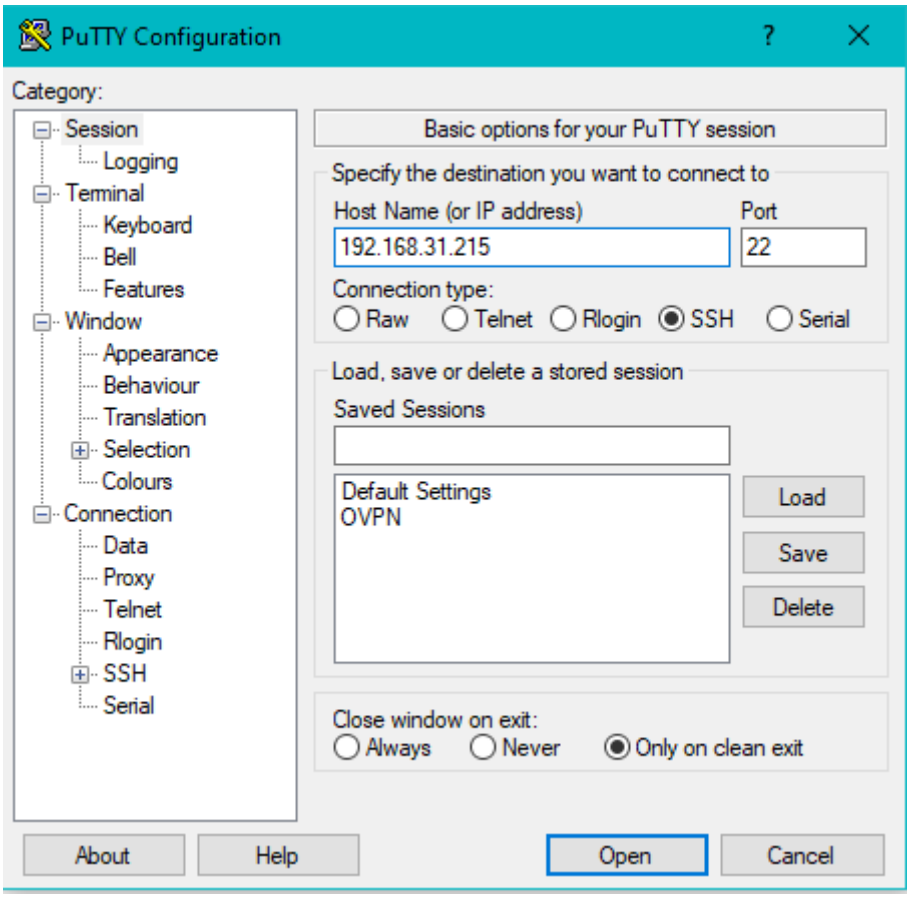

Рисунок 3.7 – Встановлення PuTTy -сесії

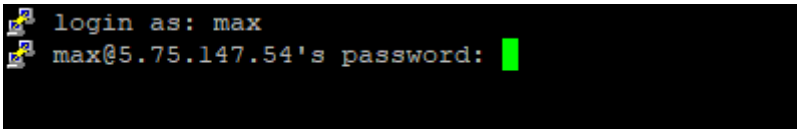

Рисунок 3.8 – Ініціалізація входу

Після підключення до головного серверу, було встановлено Ansible за допомогою команди:

$ sudo apt install ansible

Наступний крок, це створення папки для необхідних файлів:

$ mkdir ansible

Далі було створено файл “hosts.txt” (Рис. 3.9). Цей файл призначений для зберігання даних для підключення до віртуальних серверів за допомогою Ansible.

У файлі “hosts.txt” були прописані IP адреси ВМ, ім’я користувача та шлях до згенерованих ключів SSH до кожного серверу (Рис. 10).

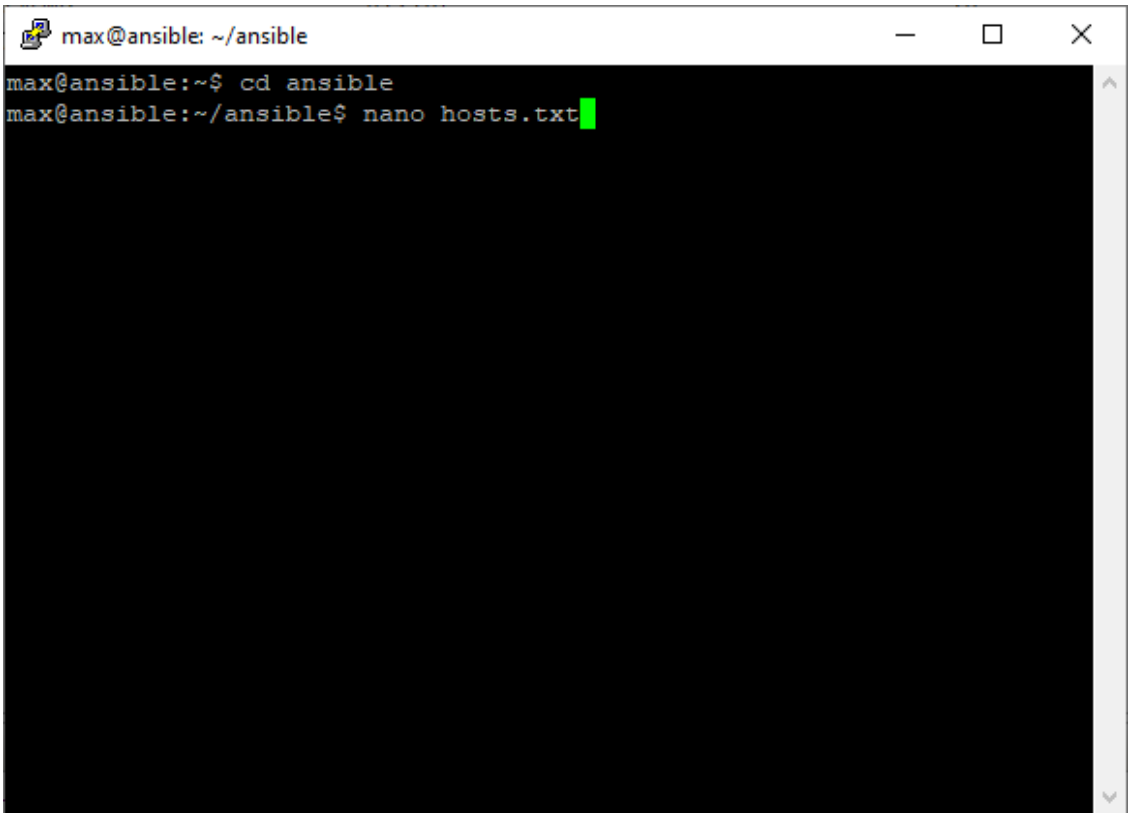

Рисунок 3.9 – Створення файлу “hosts.txt”

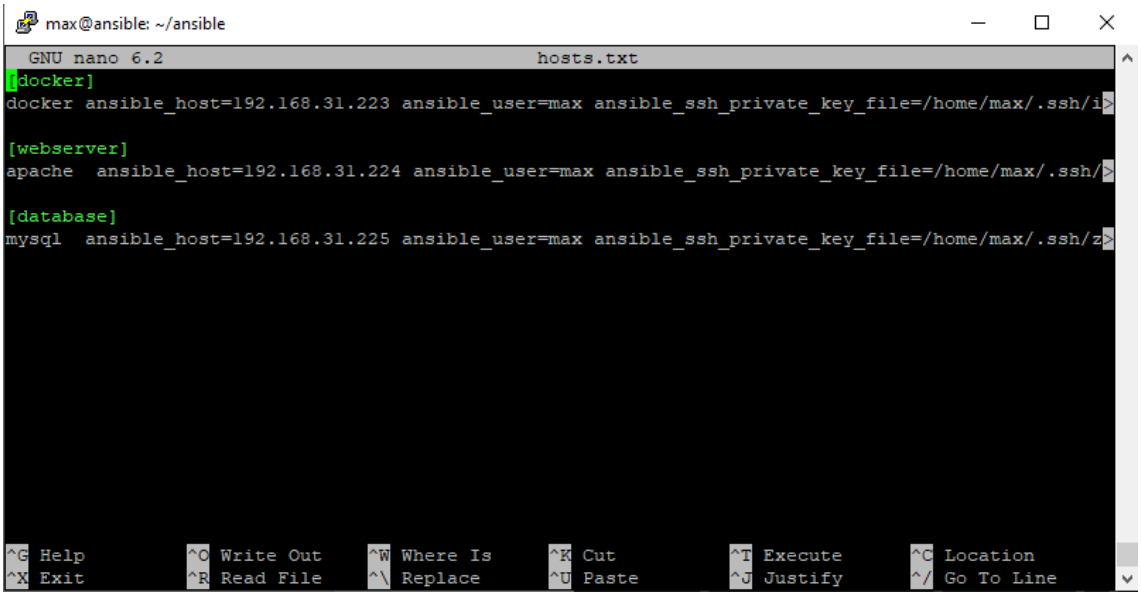

Рисунок 3.10 - Зміст файлу “hosts.txt”

Також, було створено файл “ansible.cfg” (Рис. 3.11). Це конфігураційний файл, в якому можна задати параметри за замовчуванням. В даному випадку було прописано наступні параметри:

“host_key_checking = false” - цей параметр вимикає перевірку певних сертифікатів, які потрібно підтвердити при першому підключенні до серверу, через Ansible. Якщо серверів небагато, це не є проблемою, але при великій кількості потрібно дуже багато часу щоб підтвердити сертифікати. Наступній параметр - “inventory = ./hosts.txt” - дозволяє Ansible визначити шлях до файлу “hosts.txt”, що дозволяє додатково не прописувати цей файл при виклику команди “ansible-playbook”.

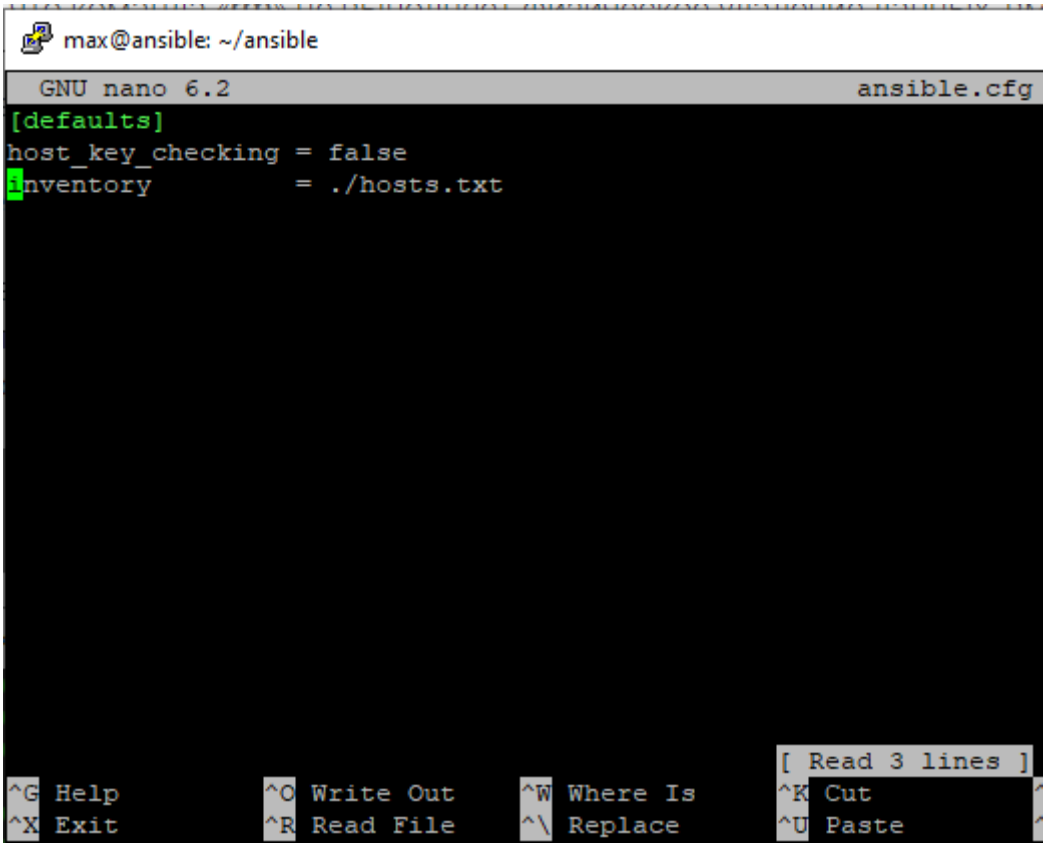

Рисунок 3.11 - Створення конфігураційного файлу

Рисунок 3.12 – Зміст конфігураційного файлу

Далі представлений зміст файлу “playbook.yml”, в якому виконується скрипт для автоматизації:

```
#Install webserver

  - hosts: apache
    become: yes
    tasks:
      - name: install apache2
        apt: name=apache2 update_cache=yes state=latest

#Install Docker

  - hosts: docker
    become: yes
```

```
vars:
  container_count: 4
  default_container_name: docker
  default_container_image: ubuntu
  default_container_command: sleep 1

tasks:
- name: Install aptitude
  apt:
    name: aptitude
    state: latest
    update_cache: true

- name: Install required system packages
  apt:
    pkg:
      - apt-transport-https
      - ca-certificates
      - curl
      - software-properties-common
      - python3-pip
      - virtualenv
      - python3-setuptools
    state: latest
    update_cache: true

- name: Add Docker GPG apt Key
  apt_key:
    url: https://download.docker.com/linux/ubuntu/gpg
    state: present
```

```
  - name: Add Docker Repository
    apt_repository:
      repo: deb https://download.docker.com/linux/ubuntu focal stable
      state: present

  - name: Update apt and install docker-ce
    apt:
      name: docker-ce
      state: latest
      update_cache: true

  - name: Install Docker Module for Python
    pip:
      name: docker

  - name: Pull default Docker image
    community.docker.docker_image:
      name: "{{ default_container_image }}"
      source: pull

  - name: Create default containers
    community.docker.docker_container:
      name: "{{ default_container_name }}{{ item }}"
      image: "{{ default_container_image }}"
      command: "{{ default_container_command }}"
      state: present
    with_sequence: count={{ container_count }}

#Install Database
```

```
- hosts: database
  becaome: yes
  vars:
    mysql_root_password: maxsyrota

  tasks:
  - name: install mysql
    apt: name=mysql update_cache=yes cache_valid_time=3600 state=present

  - name: start up the mysql servic
    shell: "service mysql start"
  - name: ensure mysql is enabled to run on startup
    service: name=mysql state=started enabled=true

  - name: update mysql root password for all root accounts
    mysql_user:
      name: root
      host: database
      password: "{{ mysql_root_password }}"
      login_user: root
      login_password: "{{ mysql_root_password }}"
      check_implicit_admin: yes
      priv: "*.*:ALL,GRANT"
    with_items:
    - "{{ ansible_hostname }}"
    - 127.0.0.1
    - ::1
    - localhost
  - name: create a new database
```

```
    mysql_db: name=testdb state=present login_user=root login_password="{{ mysql_root_password }}"
  - name: add sample data to database
    copy: src=dump.sql dest=/tmp/dump.sql
  - name: insert sample data into database
    mysql_db: name=testdb state=import target=/tmp/dump.sql login_user=root login_password="{{ mysql_root_password }}"
```

Після запуску конфігураційного файлу “playbook.yml” командою:
“ansible-playbook playbook.yml”
На екрані було відображено результат виконання команд, які містяться у файлі (Рис. 3.13 та Рис. 3.14). На екрані терміналу поетапно показано виконання та підсвічено кольорами результат виконання кожної «Задачі».

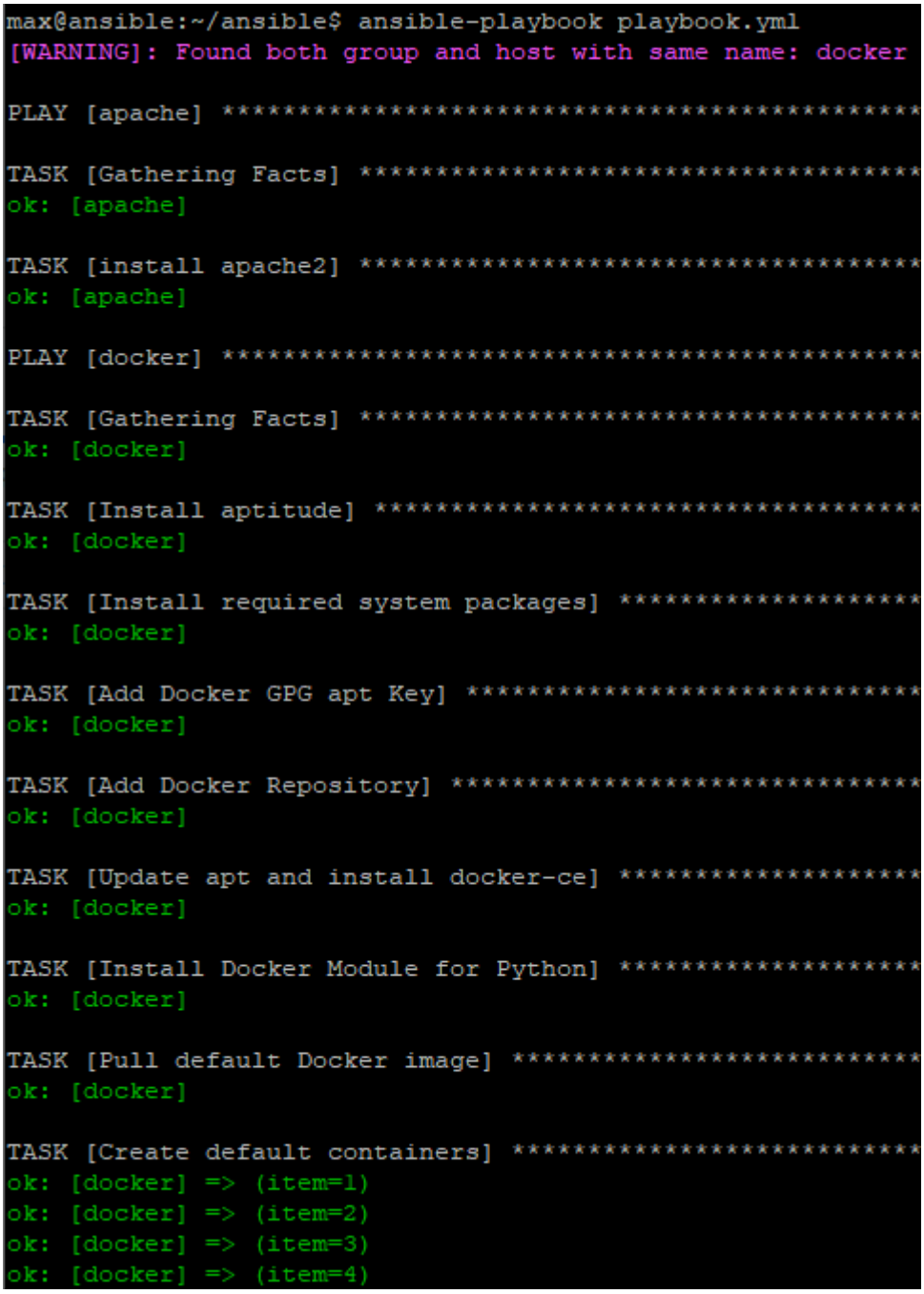

Рисунок 3.13 – Результат виконання файлу “playbook.yml” (Частина 1)

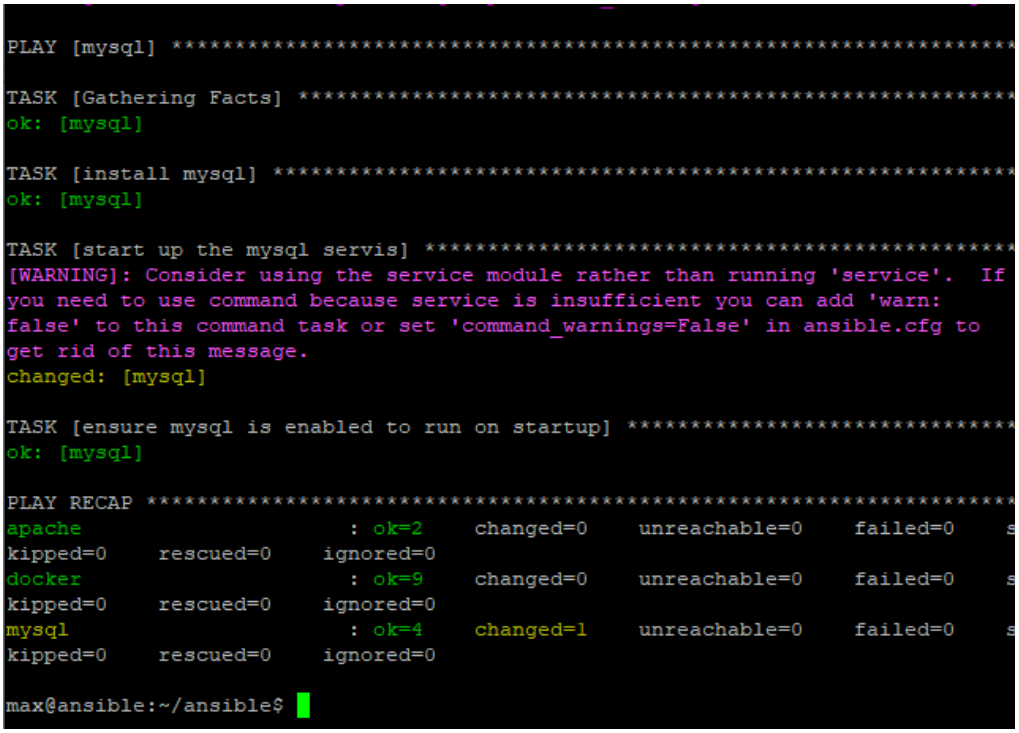

Рисунок 3.14 – Результат виконання файлу “playbook.yml” (Частина 2)

Було введено IP адресу вебсерверу в браузері комп’ютера, що знаходиться в одній локальній мережі, для перевірки його роботи (Рис. 3.15). До вебсерверу доступ налаштований тільки з локальної мережі, що підходить для розробки та тестування свого вебсайту та робить неможливим втручання з «зовні».

Для перевірки роботи “Docker” була введена команда безпосередньо на сервері:

“sudo systemctl status docker”

Результатом є те що, служба “Docker”, що відповідає за його роботу, мала статус “active” (Рис. 3.16).

Для перевірки бази даних була введена команда безпосередньо на сервері:

“sudo systemctl status mysql”

Результатом є те що, служба “mysql”, що відповідає за його роботу, мала статус “active” (Рис. 3.17).

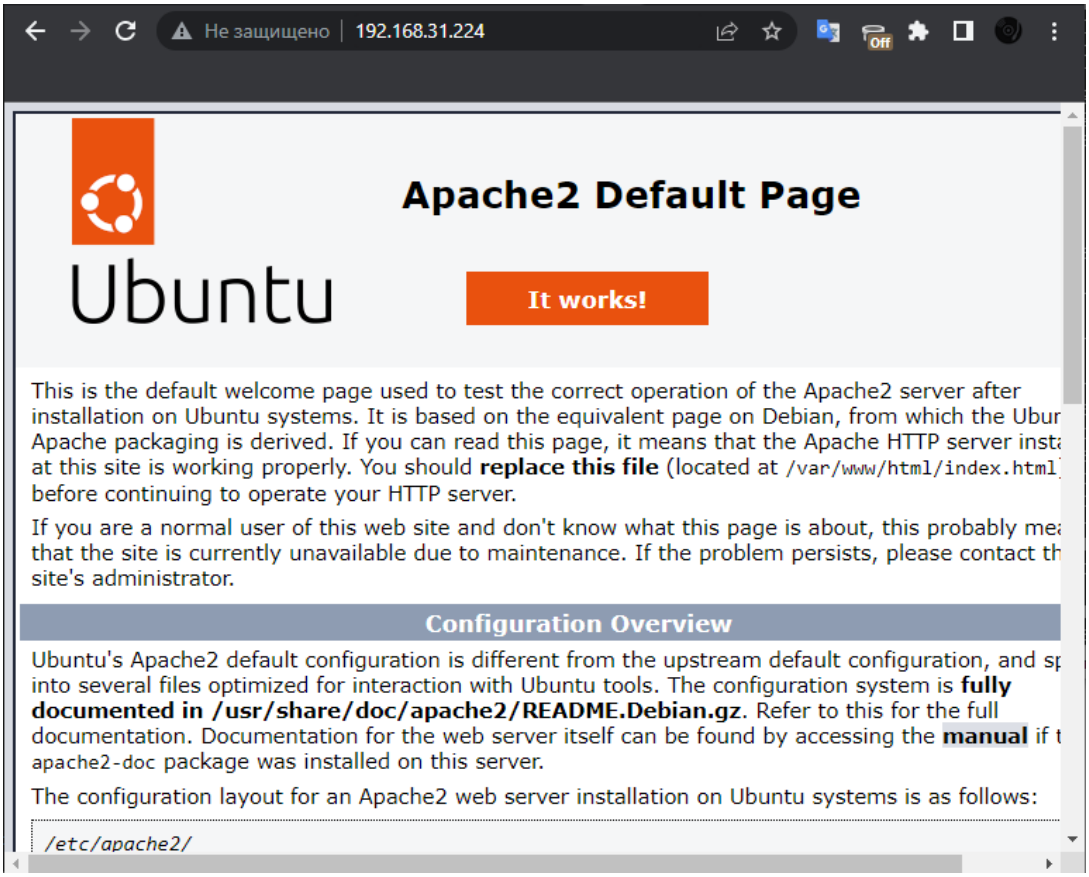

Рисунок 3.15 – Результат роботи вебсерверу

```
max@docker:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2022-12-24 09:26:15 UTC; 2h 43min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 810 (dockerd)
      Tasks: 10
     Memory: 88.0M
        CPU: 1.749s
     CGroup: /system.slice/docker.service
             └─810 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Рисунок 3.16 – Результат роботи “Docker”

```
max@database:~$ sudo systemctl status mysql
[sudo] password for max:
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2022-12-24 10:31:47 UTC; 1h 39min ago
    Process: 2260 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
   Main PID: 2268 (mysqld)
     Status: "Server is operational"
      Tasks: 38 (limit: 952)
     Memory: 360.1M
        CPU: 30.496s
     CGroup: /system.slice/mysql.service
             └─2268 /usr/sbin/mysqld
```

Рисунок 3.17 – Результат роботи “MySQL”

У царині віртуалізації було проведено безліч досліджень, які дозволили дійти висновку, що за належного планування та впровадження підхід до віртуалізації та, отже, віртуальні сервери можуть потенційно допомогти обчислювальній інфраструктурі сприяти інноваційним обчисленням, таким як хмарні обчислення, без шкоди для продуктивності, і, отже, сприяти популяризації методів віртуалізації в дата-центрах і в області хмарних обчислень. Існує значна кількість різних підходів, за допомогою яких можна реалізовувати віртуалізацію сервера; їхня різноманітність базується на вимогах до продуктивності загальної віртуалізованої інфраструктури, включаючи хмарні інфраструктури.

Продуктивність віртуалізованого середовища безпосередньо пов'язана з рядом факторів - загальна кількість віртуальних пристроїв, зайнятих у віртуальному середовищі, а також операційні показники віртуального середовища. Для центрів обробки даних консолідація серверів у вигляді віртуалізованих серверів є необхідною нормою, якщо налаштуванню продуктивності було надано пріоритет при реалізації віртуальної інфраструктури і приділено увагу до аналізу необхідних показників функціонування.

Вибір обладнання для віртуальної інфраструктури завжди був основною проблемою, і це було обумовлено тим, що при розгортанні високопродуктивних мереж із низькою затримкою, де високий рівень пропускної здатності був вирішальним, спостерігалось зниження продуктивності віртуальних обчислень.

Так, з метою порівняння параметрів продуктивності було проведено дослідження, в рамках якого було створено два середовища з однаковими налаштуваннями - з розгорнутим фізичним сервером та, відповідно, з віртуальним сервером. Однакові сервіси було активовано в різних середовищах. Було розгорнуто DNS-сервер, поштовий сервер, сервер бази даних та веб-сервер. В той час, як всі вище перераховані сервіси були активовані в одному віртуальному середовищі, для кожної служби використовувались різні фізичні сервери. Фізичні сервери було сконфігуровано однаково так, щоб можна було виокремити точні показники навантаження та функціонування.

Отже, час відгуку серверу бази даних, розгорнутому на реальному сервері, та на віртуальному сервері був досліджений протягом певного відрізку часу. Лінії тренду для часу відгуку реального та віртуального серверів представлена на Рис.3.18.

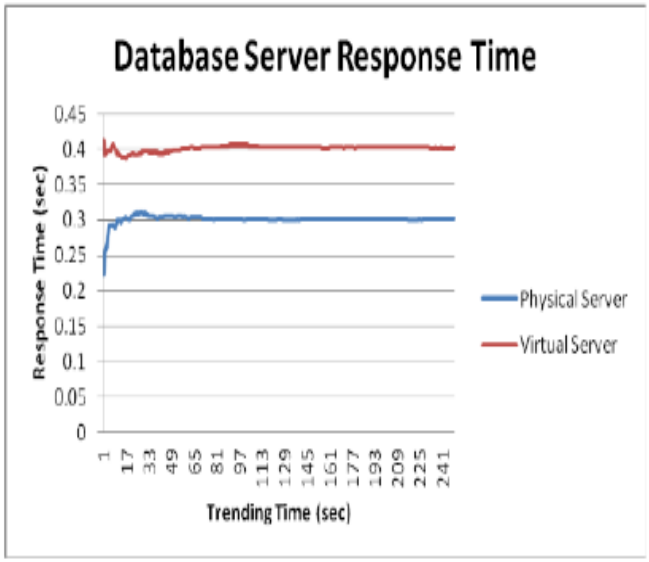

Рисунок 3.18 - Порівняльний графік часу відповіді на запит бази даних для віртуального та фізичного серверів

Наведена вище ілюстрація наочно демонструє, що час відповіді від віртуального сервера становить довший період часу, ніж від фізичного сервера у будь-який момент часу.

Обсяги Ethernet-трафіку необхідно визначити для кожного випадку середовища віртуалізації та фізичного серверного середовища, щоб визначити рівень оптимального використання ресурсів і встановити, чи було якесь із середовищ тестування пов’язане з проблемами перевантаження. Графік на Рис.3.19 показує, що потужність Ethernet може бути використана більш оптимально у віртуалізованому середовищі. Іншими словами, віртуалізоване середовище обробляє більше Ethernet-трафіку, ніж у фізичний сервера:

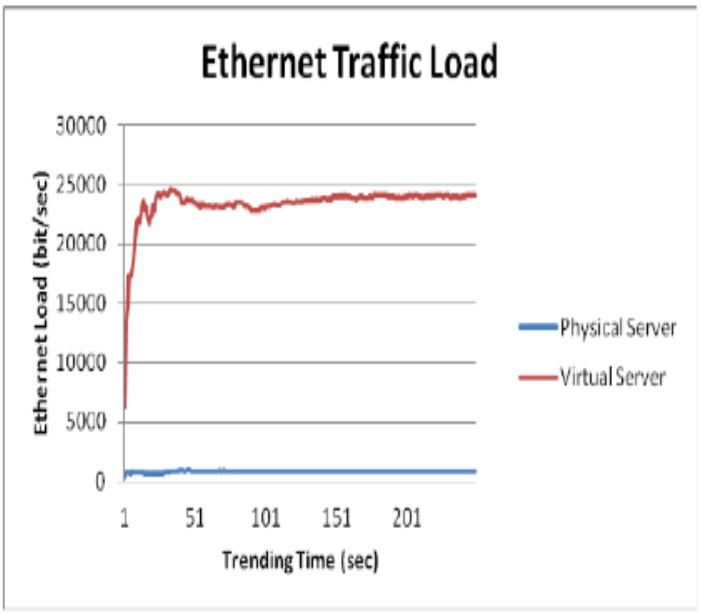

Рисунок 3.19 - Обсяг Ethernet-трафіку

Якщо оптимальне використання ресурсів буде прийнято як фактор, віртуальні сервери будуть у виграшній ситуації за умови, що загальна потреба в ресурсах і потужностях буде правильно проаналізована та досліджена на етапі планування впровадження інфраструктури віртуалізації. Ці важливій аспекти впливають на вибір потужності процесора, залежно від операційного навантаження та обсягу ресурсів, необхідних для обробки інформації, які вимагали реальні та віртуальні сервери, де віртуальні сервери завжди потребують більше ресурсів порівняно з ресурсами фізичних серверів.

Але безумовно, однією з відмінних рис віртуалізації було те, що, хоча вона не поступалася в продуктивність порівняно з фізичними серверами, віртуальний сервер є більш екологічним, ніж реальний сервер, що в наш час набуває важливого значення.

ВИСНОВОК

В дипломній роботі отримано такі результати:

1. Встановлено, що автоматизація IT-процесів, яка нещодавно сприймалася як виключно набір простих інструментів або сценаріїв для допомоги в бізнесі, зараз розглядається як стратегічна ініціатива та довгострокова IT-стратегія.
2. IT-спеціалісти мають широкий вибір засобів для автоматизації як простих рутинних, так і складних операційних процесів компаній різного розміру, різного спрямування та різної кількості співробітників.
3. Найбільш поширеним завдяки своїм доступним кодам, наявності навчальної літератури, простоті використання, є інструмент Ansible, що написаний на Python.
4. На даний час однією з найпоширеніших напрямків IT-підрозділів компаній є створення загального IT-середовища, його підтримка та підключення та інтеграція співробітників в єдине середовище. З метою мінімізації затрат людської робочої сили, та охоплення якомога більшої кількості співробітників одночасно, а також для оптимізації витрат на IT-ресурси, IT-спеціалісти компанії переходять на автоматизоване розгортання віртуальних серверів.
5. Результатом виконання даної роботи є розгорнуте віртуальне середовище та створений конфігураційний файл, який дозволяє автоматично розгортати віртуальний сервер.

   Представлений результат відповідає найсучаснішим викликам, які ставить перед IT-спеціалістами та системними адміністраторами епоха автоматизації та віртуалізації та “хмарних” технологій.

Список використаної літератури

1. *Рамський Ю. "Адміністрування комп'ютерних мереж і систем: навчальний посібник "/. Рамський Ю, Олексюк В., Балик А. – Тернопіль; Навчальна книга "Богдан", 2010. - 196 с.*
2. *Bertram A. PowerShell for sysadmins: a guide for sysadmins. No Starch Press, Incorporated, 2019. - 320 p.*
3. *Blum R., Bresnahan C. Mastering Linux system administration. Wiley & Sons, Incorporated, John, 2021. - 576 p.*
4. *Cannon J. Linux administration: the Linux operating system and command line guide for Linux administrators. CreateSpace Independent Publishing Platform, 2016. - 202 p.*
5. *Clinton D. Linux in action / David Clinton. – [S. l.] : Manning Publications, 2018. – 384 p.*
6. *Harvard Business Review Analytic Services, sponsored by Red Hat. "Taking the lead on IT automation: IT leaders as evangelists for their automation strategies," Jan. 2022.- https://hbr.org/resources/pdfs/comm/RedHat/CRE2480_HBR_PS_RedHat_4%20final.pdf*
7. *Heap M. Ansible [Electronic resource] / Michael Heap. – Berkeley, CA : Apress, 2016. – Mode of access: https://doi.org/10.1007/978-1-4842-1659-0 (date of access: 06.12.2022).*
8. *Hitchcock K. Linux system administration for the 2020s: the modern sysadmin leaving behind the culture of build and maintain. Apress L. P., 2022 – 368 p.*
9. *Hochstein L. Ansible: up and running: automating configuration management and deployment the easy way / Lorin Hochstein, Rene Moser. – [S. l.] : O'Reilly Media, 2017. – 430 p.*
10. *Frisch A. Essential system administration: tools and techniques for Linux and Unix administration. O'Reilly Media, Incorporated, 2002. – 1178 p.*
11. *Keating J. Mastering Ansible. Packt Publishing - ebooks Account, 2015. - 207 p.*

12. *Limoncelli T., Hogan C., Chalup S. Practice of system and network administration : volume 1: DevOps and other best practices for enterprise IT. Pearson Education, Limited, 2016. – 1231 p.*
13. *Mallett A. Writing YAML and basic playbooks.* Red Hat certified engineer (RHCE) study guide. *Berkeley, CA, 2021. P. 63–77. URL: https://doi.org/10.1007/978-1-4842-6861-2_5 (date of access: 18.12.2022).*
14. *Naik G. Learning Linux shell scripting:Leverage the power of shell scripts to solve real-world problems, 2nd Edition. Packt Publishing, 2018. - 332 p.*
15. *Negus C. Linux Bible, 2005 Edition / Christopher Negus. – 2nd ed. – [S. l.] : Wiley, 2005. – 830 p.*
16. *Ray D. S. Unix and Linux / Deborah S. Ray. – 4th ed. – Berkeley, CA : Peachpit Press, 2009. – 393 p.*
17. *Ryan M., Lucifredi F. AWS system administration: best practices for sysadmins in the amazon cloud. O'Reilly Media, 2018. 384 p.*
18. *Smith S., Looney J. Automating junos administration: doing more with less. O'Reilly Media, Incorporated, 2016. – 682 p.*
19. *Unix (R) and Linux (R) system administration handbook / Garth Snyder [et al.]. – [S. l.] : Pearson Education, Limited, 2011. – 1344 p*
20. *Wang K. C. Systems programming in Unix/Linux / K. C. Wang. – [S. l.] : Springer, 2019. – 476 p.*