Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Інфокомунікаціїі*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти *другий (магістерський)*

Дослідження особливостей адміністрування на операційній системі Linux
(тема)

Виконав:
студент 2 курсу, групи *ІМІм-20-2*
Меркулов К. А.
Спеціальності *172 Телекомунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *Освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)
Керівник *доц. Скорик Ю.В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2022 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____ *Меркулов К. А.*
(підпис) (прізвище та ініціали)

Керівник _____ *Скорик Ю.В.*
(підпис) (прізвище та ініціали)

Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва)

Тип програми *Освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Інформаційно-мережна інженерія*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри ІМІ_____
(підпис)
“____” _____2022 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові *Меркулову Костянтину Андрійовичу*
(прізвище, ім’я, по батькові)

1. Тема роботи *Дослідження особливостей адміністрування на операційній системі Linux*

затверджені наказом університету від 14 березня 2022 року № 379 *Ст*

2. Термін подання студентом роботи до екзаменаційної комісії *13 травня 2022 р.*

3. Вихідні дані до роботи *Дослідити операційну систему Linux, UBUNTU. Виконати реалізацію тестового веб-серверу. Встановити Terraform та Ansible.*

4. Перелік питань, що потрібно опрацювати в роботі
Вступ
1. Системне адміністрування
2. Сервер як один з елементів системного адмістрування
3. Реалізація тестового веб сервера
4. Практична реалізація сервера на базі ОС UBUNTU
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) *Слайди у форматі Power Point (назва, мета і задачі роботи, дослідження можливостей використання ОС Linux у системному адмініструванні, сервер як елемент системного адміністрування, архітектура Cloud Computing, основні операційні системи, що використовуються, інструменти автоматизації, програмна реалізація проекту, Отриманий результат роботи у вигляді працюючого веб-сервера, висновки)*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ*	*14.03.22*	*виконано*
2	*Підбір літератури за темою роботи*	*15.03-18.03.22*	*виконано*
3	*Виконання розділу 1*	*19.03-29.03.22*	*виконано*
4	*Виконання розділу 2*	*30.03-09.04.22*	*виконано*
5	*Виконання розділу 3*	*10.04-20.04.22*	*виконано*
6	*Виконання розділу 4*	*21.04-01.05.22*	*виконано*
7	*Оформлення пояснювальної записки*	*02.05-08.05.22*	*виконано*
8	*Оформлення презентаційного матеріалу*	*09.05-19.05.22*	*виконано*

Дата видачі завдання *14.03.2022 р.*

Студент _____ *Меркулов К. А.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Скорик Ю.В.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 53 с.,16 джерел, 2 додатки.

Об'єкт дослідження – операційні системи сімейства Linux.

Мета роботи – Дослідження особливостей адміністрування на операційній системі Linux.

Результати – в якості проекту розглянуто систему автоматизації процесу розгортання кластеру веб-серверів під керуванням операційної системи Ubuntu Server на прикладі встановлення і конфігурації одного сервера. Роботу виконувато з використанням віртуальної машини на базі VirtualBox та програмною обгорткою Vagrant та під управлінням ОС Linux, завдяки якій ми зможемо досягти більш якісного та гнучкого налаштування цільового сервера.

LINUX, AWS, TERRAFORM, АДМІНІСТРУВАННЯ, UBUNTU, LINUX CLI, СЕРВЕР

THE ABSTRACT

Explanatory note: 53 p.,16 sources, 2 app.

The object of study is the Linux operating systems.

The purpose of the work is to study the the features of administration on the Linux operating system.

Results - as a project we consider the system of automation of the process of deploying a cluster of web servers running the Ubuntu Server operating system on the example of installing and configuring a single server. The work is performed using a virtual machine based on VirtualBox and Vagrant software wrapper and running Linux, thanks to which we can achieve better and more flexible configuration of the target server.

LINUX, AWS, TERRAFORM, ADMINISTRATION, UBUNTU, LINUX CLI, SERVER

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – програмне забезпечення;
ОС – операційна система;
ЦП – центральний процесор;
ІТ – інформаційні технології;
KVM – Kernel-based Virtual Machine;
RFC – Request for Comments;
IDE – Integrated Drive Electronics;
SCSI – Small Computer System Interface;
ПК – персональний комп’ютер;
RDP – Remote Desktop Protocol;
GUI – Graphical User Interface;
MSSQL – Microsoft SQL Server;
DNS – Domain Name System;
LTS – Long Term Support;
SSH – Secure Shell;
AWS – Amazon Web Services;
IaaS – Infrastructure as a Service;
PaaS – Platform as a Service;
SaaS – Software as a Service;
СУБД – система управління базами даних;
IaC – Infrastructure as Code;
ЦОД – центр обробки даних;
DevOps – Development & Operations;
API – Application Programming Interface;
VPC – Virtual Private Cloud;
CLI – Command Line Interface;
GPG – GNU Privacy Guard;
IAM – Identity and Access Management.

ВСТУП

Питання системного адміністрування як такого є досить актуальним у наш час. Тотальна комп'ютеризація у всіх сферах діяльності створює нові потреби як у спеціалістах відповідного рівня, так і у спеціалізованих програмних рішеннях для адміністрування комп'ютерних систем і мереж. Основними завданнями системного адміністрування є: підключення та настроювання апаратних пристроїв; встановлення та оновлення програмного забезпечення; запуск та настроювання загальносистемних сервісів (конфігурування системи); керування користувачами; управління процесами; розподіл ресурсів; забезпечення безпеки. Дуже важливо обрати правильну та якісну операційною системою.

Світовий ринок операційних систем є досить стабільним і давно не зазнавав радикальних змін. Основні операційні системи, що використовуються, вже давно відомі користувачам та зарекомендували себе у певних сферах діяльності. Операційна система сімейства Linux давно набула популярності у сферах IT, інженерії та навіть у якості домашніх систем.

Тому актуальним є дослідження особливостей саме операційної системи Linux.

У представленій роботі будуть висвітлені саме програмні рішення, спрямовані на покращення якісних показників адміністрування систем.

1 СИСТЕМНЕ АДМІНІСТРУВАННЯ

1.1 Загальні відомості про системне адміністрування

Системне адміністрування це цілий комплекс спеціалізованих послуг, які спрямовані на забезпечення безперервної роботи всієї мережевої інфраструктури, комп'ютерної техніки та обладнання. Також системне та серверне програмне забезпечення є зоною відповідальності системного адміністратора. Системний адміністратор має забезпечувати працездатну та максимально захищену від будь-яких загроз мережу в організації. Будь-яке офісне та інше ПЗ потребує встановлення, моніторингу, налаштування та оновлення. Крім того слід зазначити, що адміністрування передбачає своєчасне виконання резервного копіювання даних, а також відновлення пошкоджених або втрачених даних при виникненні такої необхідності фахівцем. При цьому він може користуватись різноманітними методами і програмним забезпеченням. [15]

Адміністрування також включає в себе фахівців, що мають займатись віртуалізацією. У випадку, коли на одному сервері розгорнуто одразу декілька ОС, системний адміністратор або системний інженер має налаштовувати роботу всіх наявних ОС. Фахівець розподіляє ресурси сервера між всіма ОС щоб кожна працювала коректно. Це стає у нагоді у тому випадку, коли на одному серверному обладнанні розгорнуто декілька різних операційних систем. [15]

Також існує такий напрямок діяльності системного адміністратора, як адміністрування баз даних. Бази даних зазвичай використовуються для збереження великих масивів інформації. Це може бути як особиста інформація користувача, так і інформація, що використовується різними сервісами, розгорнутими на серверах. Адміністратор баз даних займається інсталяцією та налаштуванням систем управління, обслуговує ці системи, включаючи діагностику, моніторинг, оптимізацію, усунення будь-яких проблем у їх роботі.

Також фахівець займається резервним копіюванням баз даних і, за потреби, їх відновленням. [15]

Ще одним напрямком є спеціалізація адміністратора системи об'єднаних комунікацій. Такі адміністратори встановлюють і проводять налаштування кожної окремої системи, пов'язаної з електронною поштою, телефонією, голосовими конференціями або із внутрішнім чатом. До системних адміністраторів також належать фахівці, які займаються різноманітним обладнанням. Це інженери, що займаються налаштуванням та ремонтом всього мережевого та комп'ютерного обладнання, яке має компанія. [16]

1.2 Системне адміністрування комп’ютерних мереж і серверів

Для детальнішого огляду специфіки адміністрування комп’ютерних мереж не буде зайвим дати визначення цьому типу мереж.

Комп'ютерна мережа являє собою сукупність комп'ютерів, що з'єднані між собою каналами зв’язку та засобами комутації в одну систему для обміну даними будь-якого типу та доступу користувачів до технічних, програмних, організаційних та інформаційних ресурсів мережі.

Коли в мережі часто виникають збої, необхідно терміново шукати причини їх виникнення. Це може бути стара технічна база, погане системне адміністрування або ж низький рівень навичок роботи користувачів. Подібні проблеми впливають не лише на загальну працездатність комп'ютерної мережі, але й можуть стати причиною збоїв та проблем в бізнес-процесах компанії, тому ігнорувати подібні речі не варто. Можна сміливо стверджувати, що системне адміністрування і інженерія є одними з найважливіших потреб для роботи кожної компанії кожної компанії. [15]

Сюди входить весь спектр робіт, пов'язаних із встановленням, налагодженням, оптимізацією, профілактикою, моніторингом, а також іншими видами діяльності, що спрямовані на обслуговування інформаційної системи. Це не зовсім звичайне технічне обслуговування комп'ютерів та комп'ютерної техніки, а різнобічні процеси, що спрямовані на успішне використання

продуманої стратегії дій, і, звісно, вони потребують високої кваліфікації від фахівців що задіяні в них.

Основною перевагою процесів професійного системного адміністрування, які виконуються і контролюються кваліфікованими спеціалістами, насамперед є грамотно спланований процес, виконання поточних задач та подальший розвиток інфраструктури. Системний адміністратор має чітко визначати потреби компанії в тих чи інших інформаційних ресурсах. Зокрема він повинен враховувати специфіку бізнесу компанії, систему взаємодії між її окремими підрозділами та відділами. Важливо зазначити, що організація комп'ютерної мережі має проводитись з урахуванням усіх потреб та факторів і розглядатися з різних боків. Усі системи, що використовуються незалежно від їх конфігурації повинні бути надійними з технічної точки зору, а також зручними для користувачів. Для забезпечення надійної політики розвитку IT інфраструктури компанії, має місце проведення IT аудиту, який має бути основою такого розвитку. [15]

Якщо поглянути на системне адміністрування не загалом, а як на щоденне завдання, то тут проводиться забезпечення конфігурації всіх підключених пристроїв та систем, відбувається моніторинг стану мережі, виконується профілактика, усуваються різні неполадки. Також з точки зору кібербезпеки такі роботи включають у себе запобігання зовнішнім атакам, усунення різноманітних загроз, інших видів несанкціонованого доступу до мережі та внутрішніх систем, налаштування софту. [16]

2 СЕРВЕР ЯК ОДИН З ЕЛЕМЕНТІВ СИСТЕМНОГО АДМІНІСТРУВАННЯ

2.1 Загальні відомості про сервери

Сервер - це технічне рішення, яке забезпечує безліч комп'ютерів доступом до програм, файлів, ресурсів, принтерів, сканерів і т.п. Справедливим буде твердження, що від нього залежить робота користувачів, що підключені до сервера і працюють через свої робочі станції. На відміну від робочих станцій, що призначені для одного користувача, від сервера залежить велика кількість користувачів. Отже, головними пріоритетами роботи сервера є його надійність та безперебійність роботи. Докладаючи зусиль для підвищення загальної надійності сервера, фахівець має шукати можливості, які дозволять скоротити час простою і час відновлення, використовувати найкраще робоче оточення та використовувати методи, які дозволяють виконувати конфігурації, що забезпечать відмовостійкість. До сервера мають можливість підключатися сотні, тисячі або навіть мільйони користувачів одночасно. Усі зусилля щодо підвищення загальної продуктивності чи надійності сервера наштовхуються на бар'єр величезної кількості користувачів, що обслуговуються. [15]

Якщо говорити про фізичний сервер - обладнання, з якого він складається, якісно відрізняється від обладнання, призначеного для індивідуальних робочих станцій. У серверного устаткування зовсім інші і, зазвичай, ширші можливості, а у процесі його розробки враховано іншу економічну модель. При встановленні та підтримці серверів використовуються спеціальні процедури. У більшості випадків сервери постачаються з системами резервного копіювання, контрактом на обслуговування, операційною системою, а також можливістю віддаленого доступу до них. Крім того, сервери розміщуються у спеціально обладнаних обчислювальних центрах з контрольованим мікрокліматом та з обмеженим доступом до серверного обладнання. [16]

Наведена вище інформація виглядає простою і зрозумілою, але для кращого розуміння сутності сервера спробуємо розглянути його як дві складових: апаратну і програмну.

З апаратної точки зору сервером можна назвати комп'ютер, виділений із групи персональних комп'ютерів (чи робочих станцій) для виконання будь-якого сервісного завдання з мінімальною участю людини, або без участі людини. Сервер і робоча станція зазвичай мають однакову апаратну конфігурацію, оскільки різниця лише в участі у роботі людини за консоллю. Деякі сервісні завдання можуть бути виконані на робочій станції паралельно з роботою користувача. Таку робочу станцію можна назвати не виділеним сервером. Консоль (зазвичай це монітор/клавіатура/миша) та участь людини найчастіше необхідні серверам на стадії первинного налаштування, під час апаратно-технічного обслуговування, у випадках позаштатних ситуацій та управління (штатно, управління більшістю серверів відбувається віддалено). Для нештатних ситуацій сервери зазвичай мають один консольний комплект на групу серверів, що складається з комутатора, наприклад KVM-перемикача, або без такого. [15]

Серверне програмне забезпечення, у свою чергу, в інформаційних технологіях являє собою певний програмний компонент обчислювальної системи, що виконує функцію сервісу (обслуговування) за запитом до певних ресурсів чи послуг. [15]

Поняття клієнт та сервер і закріплені до них ролі утворюють програмну концепцію «клієнт-сервер». Для взаємодії з клієнтом (або клієнтами, якщо підтримується одночасна робота з декількома клієнтами) сервер виділяє необхідні ресурси міжпроцесної взаємодії (пам'ять, пайп, сокет і т. п.) і чекає запити на відкриття з'єднання (або, власне, запити на наданий сервіс). Залежно від типу ресурсу, що необхідно виділити, сервер може обслуговувати процеси в межах однієї комп'ютерної системи або процеси на інших робочих машинах через канали передачі даних (наприклад, COM-порт) або мережеве з'єднання. Формат запиту клієнта та відповідей сервера визначається протоколом.

Специфікації відкритих протоколів описані відкритими стандартами, наприклад протоколи Інтернету визначено в документах RFC. [16]

Отже, ми з’ясували, що сервер складається із спеціалізованого ПЗ та апаратних рішень, які в свою чергу мають утворювати єдиний потужний механізм, для вирішення певних задач, та задовольняти всі вимоги користувача. []

2.2 Способи реалізації сервера апаратно та за допомогою технологій хмарних обчислень

У цьому підрозділі розглянемо варіанти фізичного втілення сервера, а саме:

- придбання, встановлення і обслуговування свого фізичного кластеру серверів, що являє собою створення власного міні дата-центру;

- побудова хмарної серверної інфраструктури на платформах компаній, що надають такі послуги. Наприклад Microsoft Azure, Google Cloud Platform, Amazon Web Services, IBM Cloud та інші.

Для того щоб визначити який із двох варіантів є оптимальним у тому чи іншому випадку необхідно заглибитись в особливості реалізації кожного з них.

2.2.1. Апаратна реалізація сервера

Системи, що постачаються як сервери, значно відрізняються від систем, призначених для використання як настільні робочі станції. Інколи робляться спроби заощадити кошти за рахунок придбання комп'ютерного заліза та встановлення на нього серверного ПЗ. Таке рішення може допомогти на короткий період часу, але це недопустимо для довгострокових або великих проектів, які повинні бути надійними. Серверне обладнання завжди має вищу ціну, але додаткові можливості, які відкриваються після його придбання, виправдовують вкладення. Розглянемо деякі з цих можливостей. [1]

- Можливість розширюватись. Зазвичай, в серверах набагато більше слотів для карт розширення та центральних процесорів, фізичного простору для

жорстких дисків, також вони можуть бути оснащені роз'ємами з високою пропускною здатністю, що дозволяє підключати спеціалізовані периферійні пристрої. Також постачальники можуть надати додаткове ПЗ для кластеризації, конфігурації обладнання, розподілу навантаження, автоматизації перемикання на резервні потужності у випадку відмови обладнання та багато інших подібних можливостей.

- Велика продуктивність центральних процесорів. Сервери часто оснащені кількома ЦП, а також мають багато додаткових можливостей обладнання, таких як багатоступінчаста перевірка процесорів, попереджувальна вибірка даних, і динамічний розподіл ресурсів між ЦП. Процесори різняться робочими частотами; їхня ціна напряму залежить від частоти ЦП. Однак слід зазначити, що ціна на найшвидкісніші процесори часто буває непропорційно завищеною, що можна пояснити платою за використання передових технологій при розробці і виробництві ЦП. Такі великі додаткові витрати можна виправдати лише на сервері, який має дуже велике навантаження і підтримує одночасно багато користувачів. Оскільки сервери мають на увазі досить тривалий термін використання, інколи має сенс купувати ЦП з більшими швидкісними показниками, які довше залишатимуться актуальними. Слід зазначити, що швидкість процесора на серверах не завжди є об’єктивним показником ефективності, оскільки швидкість роботи багатьох додатків залежить не від частоти ЦП, а від швидкості обміну інформацією (введення-виведення). [1]

- Високопродуктивні системи обміну інформацією (введення-виведення). Сервери у більшості випадків продуктивніші ніж клієнти, коли йдеться про обмін інформацією (введення-виведення). Можливості введення-виведення зазвичай пропорційні кількості користувачів, що в свою чергу виправдовує застосування швидкісних підсистем введення-виведення. Доцільним є використання жорстких дисків з інтерфейсами FC-AL або SCSI замість IDE, високошвидкісних внутрішніх шин або мережевих інтерфейсів, що значно швидші, ніж інтерфейси користувачів. [1]

- Можливості модернізації. Сервери часто модернізують, а не замінюють, вони призначені для потреб що зростають. У більшості випадків на серверах є можливість додавати або замінювати процесори на більш швидкі, що не потребують додаткових змін у апаратній структурі. Серверні процесори розміщуються на окремих роз'ємах (слотах) у шасі або знаходяться в знімних роз'ємах на системній платі. [1]

- Можливість монтування у серверну стійку. Сервери обов’язково повинні мати можливість встановлення у стійку. Хоча сервери не зовсім призначені для стійок, їх можна встановити на полиці у стійці. Але це марнування простору, і в деяких випадках це навіть небезпечно. Якщо настільний комп'ютер може розміщуватись у майже будь-якому корпусі, то сервер бажано монтувати у корпус прямокутної форми для того, щоб використання простору в стійці було максимально ефективним. Дуже бажано, щоб усі кришки, які мають зніматися під час обслуговування та ремонту, знімалися без необхідності виймати сервер зі стійки. Одним з найважливіших факторів є те, що конструкція корпусу сервера має бути виконана з урахуванням всіх вимог до охолодження та вентиляції. Система охолодження, в якій вентиляційні отвори розташовуються тільки з одного боку, апріорі не здатна витримати температурне навантаження у стійці так само добре, як система з наскрізною системою вентиляції від передньої до задньої панелі. Отже при встановленні потрібно обов’язково подбати про відповідність сервера місцю, що виділене для нього. [1]

Замість того, щоб купувати, володіти та підтримувати фізичні центри обробки даних і сервери, завжди можна отримати доступ до технологічних послуг, таких як обчислювальна потужність, сховище та бази даних, за потребою від хмарного постачальника.

2.2.2. Реалізація сервера шляхом застосування технологій хмарних обчислень

Хмарні обчислення — це надання широкого спектру обчислювальних послуг, таких як сервери, мережі, моніторинг, сховища, ПЗ, бази даних, аналітика та багато інших сервісів через інтернет. Такий підхід допомагає швидше впроваджувати будь-які інновації, забезпечує гнучкість ресурсів та можливість їх швидкого масштабування, а також економію коштів. Ви завжди платите лише за ті послуги хмарного обчислення, якими користуєтеся, що значно знижує операційні витрати, а також дозволяє ефективно керувати інфраструктурою в цілому та розширюватися в залежності від змін та потреб вашого бізнесу. [1]

Розглянемо плюси і мінуси застосування хмарних обчислень більш детально. Власне до плюсів можна віднести наступне:

- Економічна ефективність. Замість інвестицій у власні центри обробки даних і обладнання, перш ніж ви знаєте, як ви збираєтеся їх використовувати, ви платите лише тоді, коли використовуєте обчислювальні ресурси, і платите лише за те, скільки ви використовуєте. Хмарні обчислення дозволяють зосередитися на своїх клієнтах, а не на важкій роботі зі встановлення стелажів, штабелювання та живлення фізичної інфраструктури. Це часто називають недиференційованим підняттям тяжкості. Використовуючи хмарні обчислення, ви можете отримати також нижчу собівартість, ніж використовуючи власний дата-центр. Оскільки ресурси, які використовуються сотнями або тисячами клієнтів об’єднуються в хмарі, можна досягти значної економії від масштабування, що означає значно нижчу оплату за ці ресурси по мірі їх використання. [1]

- Масштабованість. Завдяки передовим системам моніторингу навантаження можна з великою точністю передбачати потреби інфраструктури. Коли необхідно приймати рішення про необхідну ємність ресурсів для розгортання програми, можна задіяти велику кількість цих ресурсів, які залишаться не задіяними, або навпаки помилитись з обмеженою ємністю. Завдяки великій кількості найрізноманітніших метрик і балансерів навантаження можна отримати доступ до такої кількості чи мінімальної ємності

ресурсів, яка буде оптимальна в цей час, і збільшити та зменшити за потреби лише за кілька хвилин. [1]

- Доступність. Завдяки використанню хмарних технології ми маємо повну свободу доступу до будь-яких IT-ресурсів, чим можна скоротити час, щоб зробити ці ресурси доступними для ваших розробників протягом кількох хвилин. Це призводить до підвищення продуктивності оперативності для компанії, оскільки часу необхідного на розробку і експерименти витрачається значно менше. [1]

- Швидкість розгортання нових ресурсів. Вихід на глобальний ринок за лічені хвилини в новому регіоні є простою задачею за допомогою використання хмарних обчислень. Розгортання програми на новій серверній інфраструктурі в кількох регіонах майже будь-якої точки світу можливе протягом кількох хвилин. Це означає, що можна забезпечити меншу затримку та кращий досвід використання клієнтами продукту компанії за мінімальних витрат. [1]

2.3. Серверні операційні системи

Операційні системи завжди були приводом для дискусій у сфері інформаційно-комунікаційних технологій. З одного погляду, вони споживають ресурси сервера, які є цінними і їх можна було б використати на інші процеси. З іншої точки зору, операційна система є своєрідним оркестратором для кожного додатку сервера і дозволяє перетворити однозадачний обчислювальний комплекс у багатозадачну платформу, а також полегшити взаємодію з обладнанням всіх зацікавлених сторін. Для того щоб обрати ОС відповідно до цілей користувача необхідно визначити особливості систем, що представлені зараз на ринку. Зараз основними представниками ринку серверних ОС є Windows Server та кілька Linux-дистрибутивів різної спрямованості. Кожна з цих операційних систем має свої плюси, мінуси, особливості та ніші застосування. [2]

Розглянемо найпоширеніші з них.

Windows Server

Ця операційна система є дуже популярною в корпоративному сегменті, незважаючи на асоціації більшості рядових користувачів виключно з робочою версією Windows для ПК. Залежно від поставлених задач і необхідної для підтримки інфраструктури на даний момент компаніям надається вибір з одразу кількох версій Windows Server, починаючи з Windows Server 2003 і закінчуючи останньою версією - Windows Server 2019. [2]

Windows Server 2003 в основному використовують для підтримки корпоративних систем та мереж, що побудовані на базі Windows XP. Знята близько п'яти років тому з підтримки версія десктопної ОС від Microsoft до цього часу використовується деякими компаніями, оскільки свого часу під неї було написано багато софту для виробництва. Така сама ситуація з Windows Server 2008 R2 і Windows Server 2016 - вони є сумісними зі старим, але робочим програмним забезпеченням, яке задіяно у різного роду виробництві, тому використовуються досі. [2]

Основними плюсами серверів під керуванням Windows є відносна простота адміністрування таких серверів, досить велика кількість інформації, мануалів та ПЗ. Також неможливо обійтися без сервера на базі Windows, якщо в екосистемі компанії присутнє програмне забезпечення або рішення, що використовує бібліотеки та частини ядра будь-яких систем Microsoft. До переваг також можна додати технологію RDP, яка надає доступ користувачу до серверних програм та загальну універсальність системи. Що важливо, Windows Server має дуже полегшену версію Windows Server Core, у якій відсутній GUI, така версія за споживанням ресурсів знаходиться на рівні Linux-дистрибутива. [2]

До мінусів WS можна віднести одразу два параметри: споживання ресурсів та висока вартість ліцензії. Windows Server серед усіх серверних ОС є найбільшим споживачем апаратних ресурсів і вимагає щонайменше одне ядро процесора і близько трьох гігабайтів оперативної пам'яті лише для роботи ядра

та функціонування стандартних служб. Однозначно можна стверджувати, що дана система зовсім не підходить для малопотужних конфігурацій. Також серйозним мінусом є наявність ряду вразливостей, які пов'язані із RDP та різними політиками груп та користувачів. [2]

Найчастіше Windows Server використовують для адміністрування інтранетів компаній та підтримки працездатності різноманітного специфічного софту, роботи баз даних MSSQL, та іншого програмного забезпечення, створеного спеціально для Windows. Це все ще повноцінна серверна ОС, за допомогою якої можна підняти службу DNS, розгорнути маршрутизацію, та забезпечити роботу будь-якої службу. [2]

Ubuntu

На сьогодні саме Ubuntu є одним з найпопулярніших дистрибутивів сімейства Linux, що мають стабільний розвиток. Ubuntu стала найбільш популярною серверною ОС насамперед завдяки великому ком'юніті та безперервному вдосконаленню. [2]

Якщо Windows Server зазвичай використовується як ОС для підтримки спеціалізованого та windows-орієнтованого софту, то Ubuntu як Linux-дистрибутив – це найкращий вибір для веб-розробки та open source. Саме сервери на базі Linux прийнято використовувати для розміщення веб-серверів на базі Apache або nginx, для роботи з базами даних PostgreSQL та MySQL або скриптовими мовами розробки, які популярні зараз. На Ubuntu Server відмінно інсталюються всі необхідні служби управління трафіком та маршрутизації. []

До значних плюсів варто віднести споживання ресурсів, яке є значно меншим ніж у Windows Server, а також нативну для всіх систем на базі Linux роботу в консолі та пакетними менеджерами. Також, Ubuntu, будучи спочатку простою домашньою ОС, досить зручна для не досвідченого користувача, що полегшує її адміністрування. [2]

Основним мінусом цієї системи є те що для роботи з нею, особливо коли мова про повноцінну серверну конфігурацію, тобто виключно використовуючи

термінал, потрібні певні професійні навички. Ще однією особливістю є те, що Ubuntu більше підходить для персонального використання, та не завжди є підходящим інструментом для вирішення корпоративних задач. [2]

Debian

Debian є одним із найстаріших дистрибутивів систем сімейства Linux. При всій схожості з системою Ubuntu, в основу якої ліг саме код Debian, на відміну від свого спадкоємця не отримав такого ж рівня доброзичливості та зручності для користувача, як молодша система. Однак, незважаючи на такі особливості, система має певні переваги. Debian є більш гнучкою системою у порівнянні з Ubuntu, завдяки чому має можливості для більш поглибленого конфігурування та може ефективніше вирішувати цілий ряд специфічних завдань корпоративного характеру. [2]

Основним плюсом Debian є великі показники сек'юрності та стабільності у порівнянні з Ubuntu і, звісно з ОС сімейства Windows. Так само, як і будь-яка інша Linux-система має низькі показники споживання апаратних ресурсів. Особливо це помітно при використанні його як серверної системи під управлінням терміналу. Ще однією перевагою є те, що Debian-спільнота це open source, завдяки чому ці системи орієнтовані на ефективну та коректну роботу із застосуванням безкоштовних рішень. [2]

Але гнучкість системи має свої нюанси, звісно у такої системи є свої недоліки. Розробкою Debian займається open-source спільнота через систему майстрів гілок без наявності чіткого ядра. Одночасно з цим Debian має одразу три версії: стабільна, нестабільна та тестова. Головна проблема полягає в тому, що розробка stable-гілки має серйозне відставання від тестової, тобто час від часу в ядрі зустрічаються вже застарілі його модулі та частини. Наслідком чого є постійне ручне перескладання ядра, чи взагалі перехід на нестабільну тестову гілку, у тому випадку коли ваші завдання йдуть попереду можливостей стабільної версії системи Debian. У Ubuntu відсутні подібні проблем із

розривом версій: стабільно раз на два роки розробники системи випускають її стабільну LTS-версію. [2]

З наведеного вище можна зробити висновок,що для кожної конкретної задачі можна обрати ОС що відповідатиме вимогам користувача.

3 РЕАЛІЗАЦІЯ ТЕСТОВОГО ВЕБ-СЕРВЕРА

3.1 Обґрунтування вибору ОС Linux

Серверне використання є однією з найважливіших областей застосування ОС Linux. Все це завдяки тому,що ця ОС добре масштабується, здатна підлаштовуватись під різні програмно-апаратні умови, та розрахована на роботу багатьох користувачів. Сюди варто додати підвищену надійність і стабільність, високий рівень захисту та інформаційної безпеки, широкі можливості для налаштування. Також у IT сфері широко розповсюджена практика використання саме Linux систем у якості десктопної ОС. Ця система завдяки своїм перевагам добре підходить для розробки софту, конфігурації мереж, розгортання серверної інфраструктури, моніторингу, тощо. Серверна версія ОС без звичного графічного інтерфейсу дозволяє працювати за допомогою термінальних команд. Це є зручним рішенням, коли ви працюєте з віддаленим сервером по SSH. [14]

Сервери на базі Ubuntu зазвичай використовуються для розміщення ПЗ, сайтів для роботи в мережі Інтернет, тощо. В даному випадку в якості робочої ОС для створення тестового сервера використаємо дистрибутив Ubuntu Server. Дистрибутив Ubuntu використовується як серверне рішення найчастіше. За статистикою сервісу The Cloud Market, на хмарних серверах Amazon EC2 загальна кількість екземплярів з ОС Ubuntu перевищує 300 тисяч. Цей показник є втричі більшим у порівнянні з Debian, наприклад. Найпопулярнішими версіями ОС є саме Ubuntu Server LTS-версії.

3.2 Обґрунтування вибору хмарної технології AWS

Вибір найоптимальнішого способу розгортання кластеру серверів є одним з основних факторів нормальної майбутньої роботи цього кластеру. В даному процесі важливо уникати зайвих витрат фінансових та часових ресурсів. В нашому випадку замість фізичного встановлення та обслуговування обладнання, його налаштування і підключення кабелів, логічним буде

використання технологій хмарного обчислення з можливістю керування готовою фізичною інфраструктурою, наданою провайдером, через Інтернет.

Використання хмарних обчислень не лише заощаджує час з точки зору налаштування, але і усуває іншу недиференційовану роботу. Якщо поглянути на будь-який додаток, то можна побачити, як його аспекти дуже важливі для будь-якого бізнесу, наприклад таким аспектом є код. Звісно існують інші аспекти, які не відрізняються від будь-якого іншого додатку, який можна створити: наприклад, так званий екземпляр обчислення, який представляє собою хмарний сервер, на якому виконується код. Тож використаємо ці служби AWS для створення масштабованої, високодоступної та економічно ефективної інфраструктури для розміщення веб-сервера. Використовуючи можливості хмарних обчислень можна дуже швидко і якісно розгортати робочі кластери серверів без необхідності керувати важким фізичним обладнанням. [14]

AWS має більш ніж 10 років досвіду роботи у хмарних обчисленнях, що у свою чергу гарантує безліч переваг. Основними з яких є:

Найширша мережа дата-центрів по всьому світу: вона забезпечує високу надійність та безпеку у послугах та мережній інфраструктурі, яку Amazon пропонують клієнтам. Їх багатий досвід також дозволяє розширити спектр послуг, які пропонує ця платформа порівняно з іншими аналогами. [14]

Гнучкість: незалежно від того, який тип продукту або рішення вам потрібен для інтеграції у будь-який бізнес процесс, у AWS знайдуться необхідні для цього сервіси. Інструменти досить легко адаптуються та підлаштовуються відповідно до потреб середовища. [14]

Бюджетність: У AWS платите тільки за використання сервісів, плата за які може змінюватись відповідно до їх масштабування.

3.3 Вибір автоматизованого підходу до розгортання серверів

Як ми вже з’ясували хмарні обчислення при правильній організації роботи з ними можуть заощадити наш час і кошти завдяки відсутності великої фізичної мережевої інфраструктури. Існує велика кількість інструментів для

роботи з хмарними технологіями, які буде розглянуто далі. Для кращого розуміння роботи хмарних обчислень слід розглянути моделі послуг, які надаються провайдерами.

Більшість служб хмарних обчислень прийнято поділяти на три групи: інфраструктура як послуга (IaaS), платформа як послуга (PaaS) та програмне забезпечення як послуга (SaaS). Такі служби можна називати хмарним обчислювальним стеком, через їх зв’язок між собою. Для більшої простоти реалізації тієї чи іншої ідеї необхідно мати розуміння роботи кожної з них.

Інфраструктура як послуга (IaaS) є однією з моделей обслуговування в хмарних обчисленнях. За цією моделлю споживачу за підпискою надаються інформаційно-технологічні ресурси, такі як віртуальні сервери з певною обчислювальною потужністю, доступом до мережі та операційною системою (часто встановленою провайдером з певного шаблону). OpenStack є найпопулярним програмним рішенням для створення IaaS. [5]

IaaS знаходиться на найнижчому рівні серед хмарних моделей обслуговування, на відміну від моделі PaaS (де провайдер надає готове програмне забезпечення, СУБД, засоби розробки) і SaaS (на якому надається прикладне програмне забезпечення), в IaaS не передбачається контроль з боку постачальника послуг за програмним забезпеченням, яке встановлює користувач, він контролює лише фізичну та віртуальну інфраструктуру. [9]

Головною особливістю IaaS є масштабування відповідно до потреб. Хмарні обчислення дають набагато більше можливостей у цьому відношенні, ніж звичайні стратегії горизонтального та вертикального масштабування. Для кращої реалізації цього потенціалу, всі системи та програми краще проектувати максимально незалежними одна від одної, керуючись сервісною архітектурою та чергами повідомлень.

Платформа як послуга (PaaS) це модель надання хмарних обчислень, де споживач отримує можливість використовувати інформаційно-технологічні платформи, такі як: операційні системи, системи управління базами даних, сполучне ПЗ, засоби розробки та тестування, розміщені у провайдера. У

представленій моделі всією інформаційно-технологічною інфраструктурою, включаючи сервери, обчислювальні мережі, системи зберігання даних, повністю керує провайдер, провайдер визначає який набір видів платформ доступний споживачу і набір керованих параметрів платформ. Споживачу ж надається можливість використання платформи, створення її віртуальних екземплярів, встановлення, розробка, тестування, експлуатація ними прикладного ПЗ, з можливістю динамічної зміни кількості споживаних обчислювальних ресурсів. [4]

Програмне забезпечення як послуга (SaaS) одна з моделей обслуговування хмарних обчислень, за якої користувачу надається готове прикладне ПЗ, яке повністю обслуговується провайдером. Провайдер здійснює самостійне керування програмою, надаючи клієнту повний доступ до функцій програми з клієнтських пристроїв, зазвичай через веб-браузер або мобільну програму. [7]

Основною перевагою такої моделі для споживача послуги є відсутність витрат, які пов'язані із встановленням, оновленням та підтримкою обладнання у працездатному стані та ПЗ працюючого на ньому.

Як і у всіх моделях обслуговування хмарних обчислень, користувач платить за оренду ПЗ (використання через веб-інтерфейс або мобільний додаток), а не за володіння цим ПЗ як таким . Таким чином клієни має порівняно невеликі періодичні витрати, але таким чином у нього немає потреби робити значні грошові інвестиції у придбання прикладних програм та програмно-платформних та апаратних засобів для його розгортання, а також підтримувати працездатність всього технічного комплексу. Модель SaaS з точки зору розробника пропрієтарного ПЗ дозволяє ефективно боротися з таким явищем, як неліцензійне використання цього ПЗ, оскільки саме пПЗ не потрапляє до кінцевого замовника. Також модель SaaS дозволяє зменшити витрати спрямовані на розгортання та впровадження систем консультаційної та технічної підтримки продукту, але не виключає їх у повному обсязі. [6]

Як найкраще рішення для розгортання тестового сервера ефективним буде підхід IaC, який широко використовується популярний у моделі обслуговування хмарних обчислень IaaS.

Інфраструктура як код (IaC) є моделлю управління інфраструктурою (мережі, віртуальні машини, підсистеми балансування навантаження і топології підключень). Цей підхід застосовується для опису та управління інфраструктурою ЦОД через файли конфігурації, яке є набагото єфективнішим від ручного редагування конфігурацій на серверах. Такий підхід включає в себе декларативний та імперативний спосіб опису інфраструктури. Саме декларативний підхід застосовується найчастіше. Основний принцип декларативного підходу полягає в тому, що один вихідний код створює один двійковий файл, а сама модель IaC під час кожного застосування формує те ж середовище. IaC це ключова методика DevOps, яку використовують у поєднанні з безперервним постачанням (тобто процесом, у якому створення, складання,налаштування, тестування, та розгортання відбувається у робочому середовищі). [5]

3.4 Інструменти реалізації підходу IaC

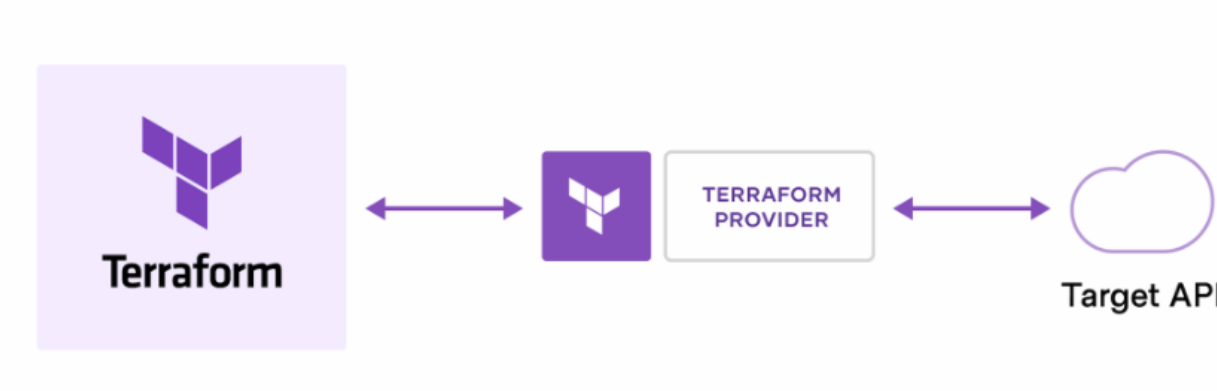
Поглянувши на поточний стан речей у сфері високих технологій можна зрозуміти, що всі моделі управління хмарних обчислень потребують своїх особливих інструментів. Завдання, такі які управління ресурсами сервера та інфраструктурою, аж ніяк не можна назвати легкими. Наприклад, якщо необхідно внести зміни у конфігурації на одному сервері, це буде досить простою і зрозумілою задачею. Але постає питання як внести подібні зміни на десятки чи навіть сотні серверів. Деякі компанії використовують підхід розгортання всіх ресурсів інфраструктури вручну, що у свою чергу створює багато проблем та сильно підвищує вірогідність збою по мірі кожного нового оновлення. Саме для мінімізації часових витрат та зменшення вірогідності помилок під час розгортання та налаштування ресурсів інфраструктури ми і використовуємо IaC. Тож саме для реалізації моделі управління

інфраструктурою IaC потрібні спеціалізовані інструменти для досягнення бажаного результату.

3.4.1 Terraform

HashiCorp Terraform є інструментом для кодування інфраструктури. Він дозволяє визначати хмарні і локальні в зрозумілих для сприйняття файлах конфігурації, з яких можна створювати різні версії конфігурацій, а також повторно використовувати та ділитися ними. За допомогою тих самих файлів можна використовувати послідовний робочий процес для керування всією інфраструктурою протягом її життєвого циклу. Terraform пристосований до керування компонентами низького рівня, такими як обчислювальні ресурси, сховища та мережеві ресурси. Також він може керувати записами DNS та функціями SaaS, які в свою чергу є високорівневими компонентами. [12]

За допомогою Terraform можна створювати та керувати ресурсами на багатьох хмарних платформах та інших сервісах за допомогою вбудованих інтерфейсів прикладного програмування (API). Провайдери дозволяють Terraform працювати практично з будь-якою платформою або сервісом із доступним API. [12]



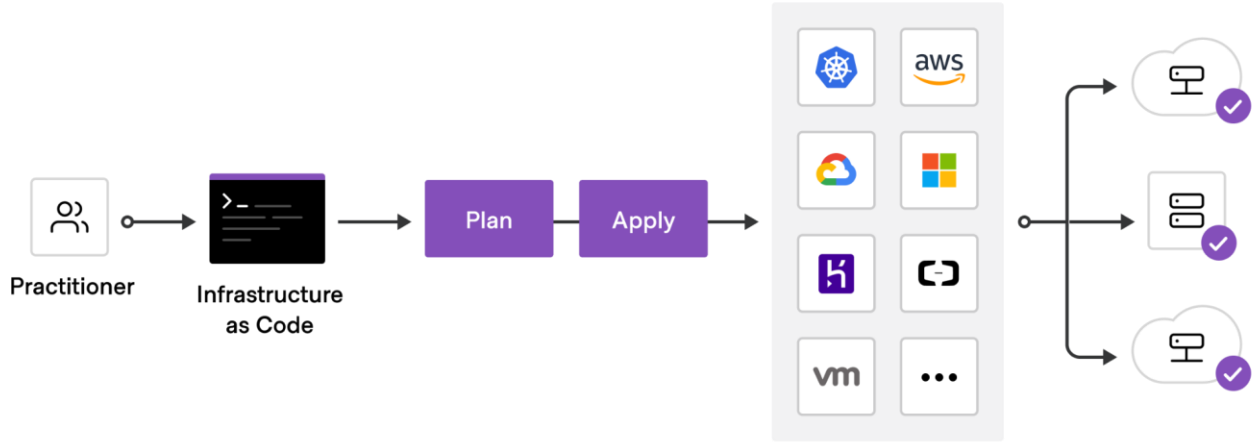
Серед таких загальнодоступних провайдерів у реєстрі Terraform є такі гіганти як Amazon Web Services (AWS), Google Cloud Platform (GCP), Azure, Helm, GitHub, Kubernetes, Splunk, DataDog та інші. Основний процес роботи у Terraform поділено на три етапи:

- Write: визначення ресурсів, які можуть бути у різних хмарних постачальників і служб. Наприклад, можна створити конфігурацію, яка виконуватиме розгортання програми на декількох віртуальних машинах у мережі віртуальної приватної хмари (VPC) з групами безпеки, підмережами, базою даних та балансувальником навантаження. [12]

- Plan: Terraform створює план виконання конфігурації, що повністю описує інфраструктуру, яку він може створити, оновити або видалити на основі файлів, як описують наявну інфраструктуру та наявної конфігурації. [13]

- Apply: після підтвердження в консолі Terraform виконує запропоновані на минулому етапі операції в правильному порядку, дотримуючись залежностей від ресурсів, прописаних в конфігурації. Наприклад, якщо потрібно оновити властивості VPC і змінити кількість віртуальних машин у тому самому VPC, Terraform спочатку відтворить VPC перед тим як виконувати масштабуванням віртуальних машин. [13]

Провайдери визначають окремі одиниці інфраструктури, такі як обчислювальні екземпляри, їх тип або групи безпеки, як ресурси. Ми маємо змогу складати ресурси різних постачальників у багаторазові конфігурації, які називаються модулями, і керувати цими модулями за допомогою узгодженої мови та робочого процесу. Мова конфігурації Terraform є декларативною. Це означає опис бажаного кінцевого стану інфраструктури, на відміну від процедурних мов програмування, які вимагають детальних покрокових інструкцій для виконання поставлених завдань. Постачальники Terraform автоматично обчислюють всі наявні залежності між ресурсами, для їх створення або знищення у правильному порядку.



Terraform відстежує реальну інфраструктуру за допомогою файлу стану, який є джерелом інформації про поточний стан середовища. Terraform використовує цей файл, щоб визначити які зміни потрібно внести у вашу інфраструктуру, щоб вона відповідала бажаній конфігурації.

3.4.2. Ansible

Ansible це система для управління конфігураціями, оркестрації ті централізованого встановлення застосунків, написана мовою програмування Python. Вона використовує декларативну мову розмітки для опису конфігурацій. Сферою її застосування є автоматизація розгортання та налаштування програмного забезпечення. Зазвичай використовується для керування вузлами Linux, але також має підтримку Windows. До особливостей Ansible можна віднести відносно просту мову управління конфігурацією, яка легко читається. Також до плюсів можна віднести можливість розпаралелювання робіт, відсутню необхідність встановлювати на віддалені системи програми-агенти та можливість роботи без root прав. Ansible не на стільки ускладнений, як CFEngine, puppet і Chef (програма), і при цьому має широкі можливості використання та високу гнучкість в управлінні. [11]

Ansible може працювати, підключаючись до вузлів завантажуючи на них невеликі програми, які називаються модулями Ansible. Ці модулі написані як ресурсні моделі цільового стану системи. Після виконання цих модулів (за

замовчуванням через SSH) вони автоматично видаляються. У Ansible використовується так званий push mode: конфігурація проштовхується (push) з керуючого хоста. Інші CM-системи зазвичай працюють інакше - вузли підтягують (pull) файли конфігурації з управляючої машини. Це важливо тому, що не обов’язково мати головну машину у публічному доступі для віддаленого налаштування вузлів. Бібліотека модулів може бути розташована на будь-якій машині, для якої не потрібні сервери чи бази даних. Зазвичай процес конфігурації відбувається в терміналі, текстовому редакторі і, час від часу - системі контролю версій, для відстеження зміни у вмісті. [11]

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ СЕРВЕРА НА БАЗІ ОС UBUNTU

4.1 Встановлення робочого оточення для конфігурації сервера

Для коректної роботи та кращого розкриття потенціалу використання необхідних у роботі інструментів було прийняте рішення створити віртуальну машину на базі ОС Ubuntu. Слід зазначити, що Terraform має версію для Windows, але ця система не є добре оптимізованою для його використання. У користувачів доволі часто виникають проблемі навіть на етапі встановлення. Саме тому використання Ubuntu є оптимальним в даному випадку

Для створення ВМ використаємо ПЗ VirtualBox та Vagrant. VirtualBox надає нам своєрідну платформу для створення віртуальної машини використовуючи ресурси вашого ПК. Цей софт є добре знайомим для більшості системних інженерів і досить часто використовується завдяки широким функціональним можливостям. Віртуальна машина створює власне ізольоване оточення на реальній машині, яке складається з віртуальних компонентів того ПК, на якому вона функціонує: процесора, жорсткого диска,, оперативної пам'яті, тощо. [10]

Інсталяція VirtualBox є досить простою і зрозумілою. Користувачі вінди можуть завантажити актуальну версію програми на офіційному сайті, після чого виконати завантажений exe файл, дотримуючись підказок програми встановлення. Після цих нескладних кроків програма готова до використання.

Встановлення ж Vagrant, на відміну від VirtualBox, потребує певних теоретичних знань користувача, та його навичок роботи з командною строкою вінди. Але спочатку трохи інформації. Vagrant є продуктом компанії HashiCorp, яка є спеціалістом в області інструментів для автоматизації розробки та експлуатації. Vagrant дозволяє створювати та конфігурувати легке, переносне та повторюване оточення для розробки. Він забезпечує:

- Ізольованість. Уникаються можливі конфлікти з іншими програмним оточеннями (наприклад, основною операційною системою) залишаючи основну систему чистою. [10]

- Повторюваність. Можливість перестворити робоче середовище за кілька хвилин, використовуючи при цьому кілька команд. Будь-яка зміна поширюється одразу для всіх. [10]

- Переносність. Потрібне оточення розгортається на будь-якій системі універсальним способом. [10]

В нашому випадку Vagrant буде використано сумісно с VirtualBox, де VirtualBox виступатиме у ролі платформи, а встановлення і налаштування ОС виконуватиметься Vagrant.

Для встановлення перейдемо до командної строки Windows. Перш за все створюємо директорію для Vagrant, в якій зберігатиметься конфігурація віртуальної машини Ubuntu.

Після створення директорії заходимо до неї і запускаємо інсталяцію нашої ОС. Робиться це командою

```
C:\Users\Dualism\vagrant_files>vagrant init ubuntu/bionic64
```

Залишається запустити віртуальну машину за допомогою, та увійти за допомогою SSH. Оскільки основні налаштування Vagrant виконує автоматично під час установки ми одразу отримуємо готову робочу машину Ubuntu.

```
 System load:  1.09              Processes:             102
 Usage of /:   7.2% of 38.71GB   Users logged in:       0
 Memory usage: 9%                IP address for enp0s3: 10.0.2.15
 Swap usage:   0%

* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.

  https://ubuntu.com/blog/microk8s-memory-optimisation

4 updates can be applied immediately.
To see these additional updates run: apt list --upgradable


Last login: Tue May 17 17:51:34 2022 from 10.0.2.2
vagrant@ubuntu-bionic:~$
```

4.2 Встановлення інструментів Terraform та Ansible

Після успішного виконання установки робочого оточення Ubuntu наступним кроком є встановлення необхідного ПЗ. Почнемо з установки

інтерфейсу командного рядка AWS. Представлений інтерфейс це єдиний інструмент, що дозволяє керувати сервісами AWS. Встановивши його можна контролювати велику кількість сервісів AWS з командного рядка. Також даний інтерфейс є засобом автоматизації цих сервісів за допомогою скриптових команд. [13]

Інсталяцію AWS CLI на ОС Linux можна виконати з командного рядка. Цей процес має кілька кроків:

- Використовуючи команду curl – параметр “-o” вказує на ім’я файлу, в який записується завантажений пакет. Відповідно до параметрів команди, наведеної нижче, завантажений файл буде записано у поточний каталог з локальною назвою файлу “awscliv2.zip”.[13]

```
vagrant@ubuntu-bionic:~$ $ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv
2.zip"
```

- Розпаковуємо інсталятор. Якщо у вашому дистрибутиві Linux немає вбудованої команди розархівування, скористайтеся еквівалентом, щоб розпакувати його. Наведений нижче приклад команди розпаковує пакунок і створює каталог з іменем aws у поточному каталозі. [13]

```
vagrant@ubuntu-bionic:~$ unzip awscliv2.zip
```

- Запускаємо програму встановлення. Команда встановлення використовує файл з іменем install у щойно розархівованому каталозі aws.

```
vagrant@ubuntu-bionic:~$  sudo ./aws/install
```

- Переконатися в тому, що програму встановлено успішно можна перевіривши поточну версію CLI за допомогою команди.

```
vagrant@ubuntu-bionic:~$ aws --version

aws-cli/2.6.4 Python/3.9.11 Linux/4.15.0-177-generic exe/x86_64.ubuntu.18 prompt/off
```

Наступним кроком є встановлення головного інструменту для розгортання нашого веб-сервера - Terraform. Для початку встановлення необхідно переконатись, що систему оновлено, також мають бути інстальовані пакети gnupg, curl та software-properties-common. Ці пакети буде використано для перевірки підпису HashiCorp GPG та встановлення репозиторію пакетів HashiCorp Debian.

```
vagrant@ubuntu-bionic:~$ sudo apt-get update && sudo apt-get install -y gnupg software-properties-comm
on curl
```

Додаємо ключ HashiCorp GPG.

```
vagrant@ubuntu-bionic:~$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
```

Додаємо офіційний Linux репозиторій, після чого оновлюємо щоб додати репозиторій та встановлюємо Terraform.

```
vagrant@ubuntu-bionic:~$ sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com
$(lsb_release -cs) main"
```

```
vagrant@ubuntu-bionic:~/tf-ansible$ sudo apt-get update && sudo apt-get install terraform
```

І останнім кроком є встановлення Ansible на нашу віртуальну машину Linux, яка використовується в якості вузла управління.

```
vagrant@ubuntu-bionic:~$ sudo apt update
```

```
vagrant@ubuntu-bionic:~$ sudo apt install ansible
```

За допомогою цих команд ми успішно встановили Ansible. Для вирішення наших задач він не потребує додаткового налаштування, тож він повністю готовий до роботи.

4.3 Побудова серверної інфраструктури інструментами реалізації підходу IaC

Після встановлення всіх необхідних складових для розгортання тестового сервера можна переходити до самого процесу реалізації IaC . Він складається з кількох послідовних етапів, тісно пов’язаних між собою.

Почнемо з отримання доступ до заздалегідь створеного аккаунту AWS за допомогою встановленої раніше AWS CLI. Це потрібно для отримання повного доступу до сервісів AWS, за допомогою яких ми будемо створювати наші ресурси. Щоб використовувати свої облікові дані IAM для аутентифікації постачальника Terraform AWS, потрібно встановити змінну ідентифікатора ключа доступу AWS_ACCESS_KEY_ID за допомогою команди: [13]

```
vagrant@ubuntu-bionic:~$ export AWS_ACCESS_KEY_ID="<YOUR_AWS_ACCESS_KEY_ID>"
```

Далі встановлюємо значення цього ключа

```
vagrant@ubuntu-bionic:~$ export AWS_SECRET_ACCESS_KEY="<YOUR_AWS_SECRET_ACCESS_KEY>"
```

Останнім кроком є встановленням регіон за замовчуванням. Оскільки AWS має величезну глобальну інфраструктуру, вона поділена на зони доступу та регіони. Визначення регіону потрібне для оптимальної роботи інфраструктури на виділених ресурсах. Тож встановимо змінну AWS_DEFAULT_REGION [11]

```
vagrant@ubuntu-bionic:~$ export AWS_DEFAULT_REGION="<YOUR_AWS_DEFAULT_REGION>"
```

Після налаштування командного рядку AWS можна переходити до наступного етапу налаштування інструментів, а саме до Terraform.

Набір файлів, що використовуються для чіткого опису і створення інфраструктури в Terraform, називають конфігурацією Terraform. У нашому випадку таку конфігурацію буде написано для створення сервера на базі екземпляру AWS EC2. Важливо зазначити, що будь-яка конфігурація Terraform для коректної роботи має перебувати у відповідному робочому каталозі. З цих міркувань першим кроком буде створення каталогу, де якому буде розміщено нашу конфігурацію. Створюємо каталог та проходимо до нього за допомогою команд[13]

```
vagrant@ubuntu-bionic:~$ mkdir tf-ansible
vagrant@ubuntu-bionic:~$ cd tf-ansible
```

Тепер створюємо головний файл Terraform, який описуватиме нашу інфраструктуру.

```
vagrant@ubuntu-bionic:~/tf-ansible$ touch main.tf
```

У цьому ж каталозі послідовно створюємо два файли, які використовуватиме Ansible для налаштування сервера.

```
vagrant@ubuntu-bionic:~/tf-ansible$ touch nginx.yaml
vagrant@ubuntu-bionic:~/tf-ansible$ touch ansible.cfg
```

Файл nginx.yaml є так званим плейбук-файлом Ansible. Ці файли викликають головну роль, посилаючись на неї.

Наступний файл ansible.cfg може містити в собі велику кількість параметрів, які можна змінювати в залежності від тих чи інших потреб налаштування інфраструктури.

Також у раніше створеному каталозі tf-ansible за допомогою засобів командного рядка Linux відтворюємо наступну структуру каталогів

```
vagrant@ubuntu-bionic:~/tf-ansible$ mkdir roles
vagrant@ubuntu-bionic:~/tf-ansible$ cd roles
vagrant@ubuntu-bionic:~/tf-ansible/roles$ mkdir nginx
vagrant@ubuntu-bionic:~/tf-ansible/roles$ cd nginx
vagrant@ubuntu-bionic:~/tf-ansible/roles/nginx$ mkdir tasks
```

Заходимо у директорію tasks і створюємо там файл main.yaml який є роллю Ansible, яка в свою чергу є частиною майбутньої конфігурації нашого Linux-сервера.

```
vagrant@ubuntu-bionic:~/tf-ansible/roles/nginx$ cd tasks
vagrant@ubuntu-bionic:~/tf-ansible/roles/nginx/tasks$ touch main.yaml
```

Роль дозволяє завантажувати всі пов’язані завдання, змінні, файли та решту необхідних Ansible ресурсів на основі відомої файлової структури повністю автоматично. Після того, як вміст згруповано за ролями, його можна з легкістю використовувати повторно у проектах та надати доступ до нього іншими адміністраторам.

Після створення необхідної нам структури директорій та файлової структури можна переходити до написання коду для розгортання сервера та його конфігураційних файлів.

Розпочнемо з головного конфігураційного файлу Terraform - main.tf. У цьому файлі користувач за допомогою коду може створити необхідні ресурси, доступні для створення та надані провайдером хмарних обчислень, у нашому випадку AWS. Синтаксис мови Terraform є досить зрозумілим для користувача знайомого з основними принципами програмування. Слід зазначити, що однією з особливостей Terraform є виконання коду не по порядку, а за пріоритетністю ресурсів, які потрібно створити. Це саме одна з причин чому Terraform є таким гарним інструментом автоматизації. [12]

Розглянемо основні моменти написання нашого коду файла main.tf. У роботі наведено фрагменти коду, повний код наведено у додатку А. Для оптимізації коду та більшої зручності його читання оголосимо локальні значення ресурсів та розглянемо їх. [12]

```
locals {
  vpc_id           = "vpc-0feeb3c3484f5818d"
  subnet_id        = "subnet-03ec1ab36d2cbda77"
  ssh_user         = "ubuntu"
  key_name         = "nginx-key"
  private_key_path = "~/aws/nginx-key.pem"
}
```

- vpc_id є унікальним ідентифікатором існуючої віртуальної приватної мережі,у якій буде розміщено наш сервер.

- subnet_id - ідентифікатор підмережі, прив'язаної до VPS. Завдяки цьому сервісу елементи, які знаходяться всередині цієї мережі мають вихід до глобальної мережі інтернет.

- ssh_user - ім'я користувача SSH

- key_name - унікальний ключ SSH, створений через консоль AWS, потрібний для підключення до створеного сервера.

- private_key_path - шлях у файловій структурі, за яким можна знайти ключ.

Далі ми повідомляємо Terraform, що у якості провайдера для розгортання інфраструктури обрано AWS. Оскільки ми зробили авторизацію в AWS раніше - обираємо профіль за замовчуванням, а також вказуємо бажаний для розгортання ресурсів регіон. [13]

```
provider "aws" {
  profile = "default"
  region  = "eu-central-1"
}
```

Тепер ми можемо створювати ресурси для нашої інфраструктури. Першим з таких є група безпеки, яка є набором обраних нами правил взаємодії з інтернет мережею.

```
resource "aws_security_group" "nginx" {
  name   = "nginx"
  vpc_id = local.vpc_id
```

Даємо групі безпеки ім'я та даємо інструкцію щодо того, до якої підмережі потрібно застосувати прописані правила.

Створивши групу безпеки ми переходимо до створення платформи для нашого сервера - екземпляру AWS.

```
resource "aws_instance" "nginx" {
  ami                         = "ami-02584c1c9d05efa69"
  subnet_id                   = "subnet-03ec1ab36d2cbda77"
  instance_type               = "t2.micro"
  associate_public_ip_address = true
  security_groups             = [aws_security_group.nginx.id]
  key_name                    = local.key_name
```

Оголошуємо про створення ресурсу типу aws_instance і даємо йому ім’я. Головним параметром, який потрібно вказати при цьому є ami - ідентифікатор ОС з каталогу AWS, за яким Terraform визначає її тип, версію, архітектуру та інші параметри. Для нашого сервера через свої численні переваги обрано ОС Ubuntu, 20.04 LTS, amd64 focal image build on 2022-04-19. Далі знов вказуємо ідентифікатор підмережі, до якої буде підключено наш екземпляр, а також його тип, у нашому випадку t2.micro. Обов’язково додаємо параметр автоматичного отримання публічної IP-адреси, необхідної для підключення до нашого сервера. Далі асоціюємо екземпляр с групою безпеки та ключем доступу. [13]

Екземпляр створено і він готовий до роботи, але тепер постає питання підключення до нього. Для цього ми створюємо провайдера дистанційного підключення по SSH. Провайдер підключається до екземпляру отримуючи його публічну IP-адресу і виконує з’єднання за протоколом SSH.

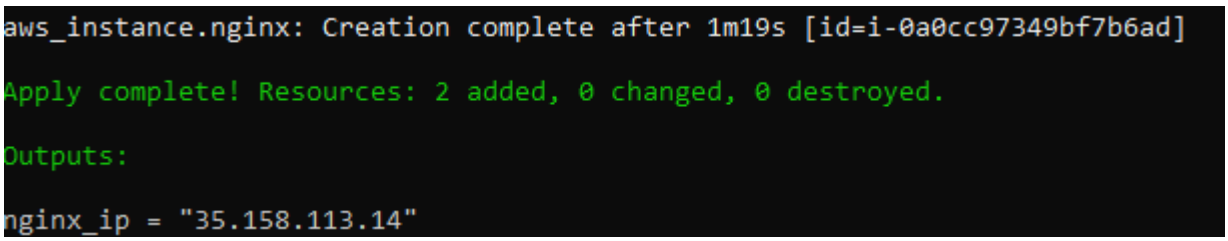
```
provisioner "remote-exec" {
  inline = ["echo 'Wait until SSH is ready'"]

  connection {
    type        = "ssh"
    user        = local.ssh_user
    private_key = file(local.private_key_path)
    host        = aws_instance.nginx.public_ip
  }
}
```

Наступним кроком є створення локального провайдера, за допомогою якого Ansible може завантажити і виконати свої файли конфігурації.

```
provisioner "local-exec" {
  command = "ansible-playbook  -i ${aws_instance.nginx.public_ip}, --private-key ${local.private_key_path} nginx.yaml"
}
```

Після успішного виконання коду програма має вивести до командної строки IP-адресу нашого сервера.



Слід звернути особливу увагу на верхню частину скріншоту, на якій вказано час створення сервера. Це час від застосування коду командою terraform apply до того, як сервер готовий до роботи. Це є реальністю завдяки злагодженій роботі обраної нами ОС та інструментів для розгортання серверу. Ubuntu завдяки легкості серверної версії системи як ми бачимо може розгортатись менш ніж за дві хвилини, що є чудовим показником та стає у нагоді під час виникнення загрози перевантаження існуючих кластерів серверів. Такий малий аптайм дозволяє миттєво реагувати на навантаження, що стрімко збільшується. Також системи Linux дуже легко конфігурувати за допомогою засобів автоматизації і підходу IaC завдяки їх гнучкості і швидкодії. [13]

ВИСНОВКИ

У виконаній кваліфікаційній роботі було досліджено використання в оболонці Linux-системи спеціалізованих інструментів системного адміністрування системи автоматизації розгортання веб-серверів на базі ОС Linux. В ході роботи було використане програмне забезпечення, яке є особливо ефективним і зручним у використанні саме на системах Linux-сімейства. Інструменти, які було задіяно для автоматизації, конфігурації та адміністрування серверів на базі ОС Ubuntu Server також були розгорнуті на базі віртуальної машини Ubuntu, завдяки всім перевагам її використання. В процесі роботи було виявлено, що такі показники як швидкість розгортання та загалом роботи ОС Ubuntu, є надзвичайно гарними, оскільки на тестовому сервері систему було встановлено і налаштовано за 80 секунд.

Одним з головних плюсів є ядро, яке має відкритий вихідний код, що дозволяє модернізувати програмне забезпечення та саме ядро, покращуючи тим самим показники стабільності та загальної роботи системи.

В свою чергу слід зазначити високі показники безпеки Linux систем, оскільки вірусне програмне забезпечення, написане під цю систему, майже відсутнє.

Налаштування спеціалізованого програмне забезпечення для найрізноманітніших проектних цілей у більшості випадків легке та гнучке, оскільки ця система створена розробниками для таких самих розробників.

Саме тому можна автоматизувати найменші задачі засобами bash, вбудованого в систему.

Зазначені вище якості обумовлені особливостями Unix-систем, завдяки яким ці системи є популярними серед системних адміністраторів, розробників ПО, інженерів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лимончелли Т., Хоган К., Чейлап С. Системное и сетевое администрирование. Практическое руководство, 2-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2009 с101-103
2. Windows Server или Linux-дистрибутивы? Выбираем серверную ОС [Електронний ресурс] / Режим доступу: URL: https://habr.com/ru/company/ruvds/blog/499322/
3. Что такое облачные вычисления? [Електронний ресурс] / Режим доступу: URL: https://azure.microsoft.com/ru-ru/overview/what-is-cloud-computing/#benefits
4. Платформа как услуга [Електронний ресурс] / Режим доступу: URL: https://ru.wikipedia.org/wiki/Платформа_как_услуга
5. Инфраструктура как код [Електронний ресурс] / Режим доступу: URL: https://ru.wikipedia.org/wiki/Инфраструктура_как_код
6. Что такое "Инфраструктура как код"? [Електронний ресурс] / Режим доступу: URL: https://docs.microsoft.com/ru-ru/devops/deliver/what-is-infrastructure-as-code
7. Программное обеспечение как код [Електронний ресурс] / Режим доступу: URL: https://ru.wikipedia.org/wiki/Программное_обеспечение_как_услуга
8. Каковы плюсы и минусы перехода в облако AWS? - Веб-сервисы Amazon [Електронний ресурс] / Режим доступу: URL: https://www.informatique-mania.com/ru/applications/quels-sont-les-avantages-et-les-inconvenients-de-la-migration-vers-le-cloud-aws-services-web-amazon/
9. Infrastructure as code: обзор опенсорсных инструментов [Електронний ресурс] / Режим доступу: URL: https://habr.com/ru/company/otus/blog/570926/
10. Что такое Vagrant: установка, запуск, использование [Електронний ресурс] / Режим доступу: URL: https://guides.hexlet.io/ru/vagrant/
11. Ansible для сетевых инженеров [Електронний ресурс] / Режим доступу: URL: https://ansible-for-network-engineers.readthedocs.io/ru/latest/book/07_playbooks/roles.html
12. Terraform documentation [Електронний ресурс] / Режим доступу: URL: https://www.terraform.io/intro#manage-any-infrastructure
13. How to create reusable infrastructure with Terraform modules [Електронний ресурс] / Режим доступу: URL: https://blog.gruntwork.io/how-to-create-reusable-infrastructure-with-terraform-modules-25526d65f73d

14. 10 основных преимуществ и недостатков облачных вычислений | Риски | Выгоды [Електронний ресурс] / Режим доступу: URL: https://tehnografi.com/10-основных-преимуществ-и-недостатков-о/

15. Системное администрирование - это менеджмент или инженерия? [Електронний ресурс] / Режим доступу: URL: https://frameboxxindore.com/ru/android/is-system-administration-a-management-or-engineering.html

16. Что должен знать и уметь системный администратор [Електронний ресурс] / Режим доступу: URL: https://itfb.com.ua/chto-dolzhen-znat-i-umet-sistemnyj-administrator/