ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ANSIBLE З МЕТОЮ ОПТИМІЗАЦІЇ ІТ-ІНФРАСТРУКТУРИ»**

на здобуття освітнього ступеня магістр
за спеціальності 123 Комп'ютерна інженерія
(код,найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

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

Вадим СТЕПАНЧУК
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр.КСДМ-62
Вадим СТЕПАНЧУК
(ім'я, ПРІЗВИЩЕ)
Керівник: Анастасія ВЄЧЕРКОВСЬКА
к.т.н., доцент *(ім'я, ПРІЗВИЩЕ)*

Рецензент:
науковий ступінь, вчене звання *(ім'я, ПРІЗВИЩЕ)*

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти «Магістр»

Спеціальність 123 Комп'ютерна інженерія
Освітньо-професійна прорама Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерної інженерії
Наталія ЛАЩЕВСЬКА
(ім'я, ПРІЗВИЩЕ)
"____" _____2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Степанчуку Вадиму Ігоровичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Дослідження використання Ansible з метою оптимізації IT-інфраструктури
керівник роботи к.т.н., доцент, Вєчерковська А.С.
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від "19" 10 2023 р. №145
2. Строк подання кваліфікаційної роботи
3. Вихідні дані кваліфікаційної роботи:
3.1. Інтернет ресурси стосовно розробки веб-серверів.
3.2. Хмарне середовище.
3.3. Науково-технічна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1. Аналіз типових помилок при розгортанні інфраструктури та управлінні життєвим циклом

4.2. Впровадження рішень.
4.3. Розробка розгортання веб-сервера.

5.Перелік ілюстраційного матеріалу: *презентація*

6. Дата видачі завдання "19" жовтня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	**Примітка**
1.	Підбір технічної літератури	.2023р. .2023р.	Виконано
2.	Аналіз типових помилок при розгортанні інфраструктури та управлінні життєвим циклом	.2023р. .2023р.	Виконано
3.	Впровадження рішень	.2023р. .2023р.	Виконано
4.	Розробка розгортання веб-сервера	.2023р. .2023р.	Виконано
5.	Оформлення роботи, висновки	.2023р. .2023р.	Виконано
6.	Розробка демонстраційного матеріалу, доповідь	.2023р. .2023р.	Виконано

Здобувач вищої освіти Вадим СТЕПАНЧУК
(підпис) *(ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи Анастасія ВЄЧЕРКОВСЬКА
(підпис) *(ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступеня магістр: 96 стор., 7 рис., 23 джерел.

Мета роботи – Покращення продуктивності та надійності систем шляхом оптимізації IT-інфраструктури за допомогою Ansible.

Об'єкт дослідження – Оптимізація IT-інфраструктури за допомогою Ansible.

Предмет дослідження – Інструмент Ansible.

Короткий зміст роботи: В даній магістерській роботі представлено розробку рішення для використання автоматизації розгортання інфраструктури в хмарному середовищі для компанії з розробки веб-додатків з використанням Ansible.

У роботі було проведено аналіз найбільш поширених помилок, що допускаються в процесі розгортання інфраструктури, а також аналіз інфраструктурних послуг, що було реалізовано в практичній частині дипломної роботи.

У практичному розділі дипломної роботи було розроблено рекомендації Ansible для автоматизації розгортання інфраструктури, який допомагає підтримувати послідовність майбутніх розгортань.

КЛЮЧОВІ СЛОВА: ANSIBLE, IT ІНФРАСТРУКТУРА, ANSIBLE CONTROL, AWS, ВЕБ-СЕРВЕР, ХМАРНА ІНФРАСТРУКТУРА, ДІАГРАМИ ІНФРАСТРУКТУРИ AWS

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 96 pages, 7 figures, 23 sources.

The purpose of the work is Improving system performance and reliability by optimizing IT infrastructure using Ansible.

The object of research is optimizing IT infrastructure using Ansible.

The subject of research is The Ansible tool.

Summary of the work: This master's thesis presents the development of a solution for the use of infrastructure deployment automation in a cloud environment for a web application development company using Ansible.

The work analyzed the most common mistakes made in the process of infrastructure deployment, as well as the analysis of infrastructure services, which was implemented in the practical part of the thesis.

In the hands-on section of the thesis, Ansible guidelines were developed for automating infrastructure deployment, which helps to maintain the consistency of future deployments.

KEY WORDS: ANSIBLE, IT INFRASTRUCTURE, ANSIBLE CONTROL, AWS, WEB SERVER, CLOUD INFRASTRUCTURE, AWS INFRASTRUCTURE DIAGRAMS

ЗМІСТ

ВСТУП

Дана магістерська робота присвячена аналізу автоматизації процесу налаштування та розгортання інфраструктури, необхідної для невеликої компанії з розробки веб-додатків. Середовище буде розгорнуто в хмарному рішенні. Автоматизація всього процесу за допомогою інструменту керування конфігурацією зробить розгортання послідовним, безпечним, портативним і простим в управлінні. У роботі буде проведено аналіз найбільш поширених помилок, що допускаються в процесі розгортання інфраструктури в малому бізнесі та стартапах, а також аналіз інфраструктурних послуг, що буде реалізовано в практичній частині дипломної роботи.

Інфраструктура, про яку йде мова, міститиме такі послуги:

- веб-сервери
- баз даних
- контроль
- резервне копіювання та відновлення

Результат цієї роботи повинен стосуватися починаючої або вже існуючої компанії з розробки веб-додатків, яка хотіла б підвищити безпеку і полегшити управління своєю інфраструктурою.

Рішення буде розроблено з використанням найкращих практик безпеки, кодування та інфраструктури, щоб його можна було впровадити для вже існуючих компаній, що працюють у сфері розробки веб-додатків.

Дослідницькими питаннями, на які звертається увага під час роботи:

- які найпоширеніші помилки допускаються під час розгортання інфраструктури?

- які помилки, пов'язані з кібербезпекою, допускаються в управлінні життєвим циклом інфраструктури?

- які оптимальні програмні рішення для інфраструктурних сервісів (веб-сервер, база даних, моніторинг, резервне копіювання)?

Актуальність даної роботи зумовлена наступними пунктами.

- зростання складності та обсягу IT-інфраструктур вимагає впровадження ефективних інструментів для автоматизації та управління ними. Ansible є одним з таких інструментів, який дозволяє забезпечити швидку та надійну автоматизацію процесів управління інфраструктурою.

- використання Ansible дозволяє знизити час та зусилля, необхідні для налаштування та управління IT-інфраструктурою. Це дозволяє звільнити IT-спеціалістів від рутинних задач та сконцентруватися на більш складних та стратегічних завданнях.

- Ansible є відкритим та гнучким інструментом, який може бути використаний для автоматизації різних аспектів IT-інфраструктури, включаючи налаштування серверів, розгортання програмного забезпечення, моніторинг та багато іншого. Дослідження використання Ansible дозволить виявити його потенціал та ефективність у різних сценаріях використання.

- оптимізація IT-інфраструктури є актуальною задачею для багатьох організацій, оскільки вона дозволяє знизити витрати, покращити продуктивність та забезпечити більшу надійність систем. Дослідження використання Ansible з метою оптимізації IT-інфраструктури може бути корисним для організацій, які прагнуть покращити ефективність своєї інфраструктури та забезпечити більшу автоматизацію процесів управління.

1 АНАЛІЗ ТИПОВИХ ПОМИЛОК ПРИ РОЗГОРТАННІ ІНФРАСТРУКТУРИ ТА УПРАВЛІННІ ЖИТТЄВИМ ЦИКЛОМ

1.1 Актуальність проблеми розгортання інфраструктури

Ручне розгортання інфраструктури – це дуже трудомісткий процес, під час якого можуть статися неправильні конфігурації. Ці помилки можуть призвести до дірок у безпеці або до того, що системи не працюватимуть належним чином. Це означає, що в майбутньому буде витрачено більше часу на налагодження або перепроектування всієї інфраструктури, що може означати можливі простої протягом тривалого часу.

Ця робота зосереджена переважно на стартапах та малому бізнесі, оскільки станом на 2022 рік 47% малих підприємств зазнавали принаймні однієї кібератаки на рік. З них 44% пережили два, три або чотири напади за той самий період, а 8% мали п'ять і більше нападів. Головними страховими претензіями, пов'язаними з кібернетикою, були програми-вимагачі, хакери та втрата або неправомірне використання даних. Крім того, 7 з 10 бізнесів не готові до кібератаки.

Найпоширенішими помилками, яких припускаються стартапи та малий бізнес при розгортанні хмарної інфраструктури та управлінні життєвим циклом, є:

- збереження облікових даних у системах контролю версій, таких як Git;
- неправильне керування доступом;
- неправильно налаштовані процедури резервного копіювання та відновлення;
- системи/процедури виправлень та вразливостей не застосовуються.

Автоматизація розгортання забезпечить узгодженість середовища, скоротить витрати часу, виключить можливість неправильних конфігурацій і забезпечить більш безпечну та стійку інфраструктуру. Зміни конфігурації також будуть

швидшими та послідовнішими, оскільки все середовище описується в коді та керується інструментом керування конфігурацією.

Принципи, за якими допоможе досягти автоматизація розгортання інфраструктури:

- можливість постійно вносити покращення, а не робити все одразу в певний час;

- системні адміністратори витрачають менше часу на рутинні та повторювані завдання;

- користувачі можуть керувати ресурсами та визначати те, що їм потрібно, без допомоги системного адміністратора;

- системи та сервіси можна швидко та легко відновити.

Результат практичної частини цієї дипломної роботи допоможе стартапам та малому бізнесу якнайшвидше розгорнути свою інфраструктуру у хмарі, зберігаючи найкращі практики безпеки та інфраструктури. Безпечне налаштування відповідних сервісів, не маючи попереднього досвіду, було б складним завданням.

1.2 Проблеми в малих IT-компаніях

У всьому світі понад 95% підприємств вважаються малими (<100 співробітників). Найбільш складними завданнями для створення IT-інфраструктури в невеликих IT-компаніях є аналіз вимог, фінансові та бюджетні розрахунки, вирішення питань впровадження та експлуатації. Після цього компанія повинна зрозуміти продукт, який вона створює, і технологію, необхідну для цього.

Крім того, створення компанії може бути дуже стресовим процесом, під час якого деякі частини процесу неминуче поспішають, а потім забувають. Ці деталі можуть стати величезними проблемами, якщо вони прив'язані до

IT-інфраструктури. Коли власник бізнесу не є фахівцем в IT-сфері і не вважає пріоритетом правильне планування IT-стратегії, масштабування в майбутньому може стати чималим викликом для компанії.

Основними помилками безпеки, допущеними в хмарі, згідно з дослідженням, проведеним компанією Netskope, є:

- надання публічного доступу до відер для зберігання;
- використання облікового запису root для повсякденних дій;
- невиконання моніторингу;
- питання інтеграції.

Коли компанія не сприймає ці проблеми серйозно, одного разу вона може припинити свою діяльність через витік даних, що може призвести до величезних штрафів. У 2022 році в Сполучених Штатах було зареєстровано приблизно 1579 витоків даних.

Витік даних може бути результатом помилки користувача. Найбільш типовим сценарієм є ситуація, коли оператор неправильно налаштував платформу або сервер, що призвело до можливості зовнішнього суб'єкта отримати несанкціонований доступ до даних. Витік даних може вплинути на особисті дані мільйонів людей, що в Європі через GDPR може призвести до величезних штрафів.

Оскільки хмарні обчислення часто спрощують процес розгортання IT-інфраструктури, багато компаній переносять свої послуги в хмару. Однак користувачі повинні розуміти концепції безпеки обраного ними хмарного провайдера.

Оскільки компанії-початківці, як правило, мають обмежені ресурси, вони залишають розгортання та управління новими додатками в хмарній інфраструктурі розробникам. Вони вважають це менш дорогим рішенням у порівнянні з наймом співробітника, який би відповідав за хмарну інфраструктуру.

Коли додаток починає набирати популярність і стає одним з основних джерел доходу компанії, базова інфраструктура стає занедбаною, оскільки розробники більше зосереджуються на додатку. Рутинні процеси залишаються невиконаними, належний моніторинг не здійснюється, а документація пишеться.

Крім того, якщо виникнуть проблеми з взаємозв'язком між інстансами або зовнішніми службами, розробники можуть зробити неправильну конфігурацію мережі, що може залишити всю інфраструктуру в вразливому стані. Як згадувалося раніше, одним із найгірших сценаріїв було б залишити бази даних або сегменти AWS S3 відкритими для громадськості, що може призвести до витоку даних.

1.3 Витік даних, спричинений неправильно налаштованими службами IT-інфраструктури

У жовтні 2022 року дані API, облікові дані автентифікації, які включали 40 000 відкритих текстових паролів, сертифікатів, ключів дешифрування, конфігурацій, інформації про клієнтів Accenture були залишені відкритими для публічного доступу через неправильно налаштовані сегменти зберігання AWS S3. До списку їхніх клієнтів увійшли 94 компанії зі списку Fortune Global 100.

BJC Healthcare повідомила, що незахищений сервер залишався відкритим для доступу до Інтернету в період з травня 2021 року по січень 2022 року. Повідомляється, що витік торкнувся інформації про 33 000 пацієнтів. На сервері зберігалася така інформація, як водійські права пацієнтів, дані про страховку, документи про лікування. Персональні дані, такі як імена, адреси, дата народження, номери телефонів та номери соціального страхування, також були вразливими.

1.4 Узгодженість конфігурації

Як згадувалося раніше, ручне розгортання інфраструктурних сервісів може бути довгим і складним завданням. При розгортанні декількох середовищ одночасно, в залежності від розміру і компетенції команди, що працює над ним, присутня можливість невідповідностей і неправильних конфігурацій. Невиявлені помилки конфігурації, які не виявлені вчасно, можуть призвести до порушень безпеки, непрацюючих або несправних систем.

У разі аварії відновлення віртуальних машин або служб буде складним завданням, якщо не буде запроваджено належних процедур резервного копіювання та аварійного відновлення. Якщо процедура передислокації виконується без автоматизації, процес може зайняти години або дні, залежно від розміру інфраструктури. Під час великих передислокацій неправильні конфігурації та невідповідності є дуже ймовірними через тиск, пов'язаний із відновленням роботи. Виявлення помилок, допущених протягом цього періоду, може бути надзвичайно важко знайти та виправити, що може збільшити загальний час простою інфраструктури.

Якщо компанія використовує кілька постачальників хмарних послуг, важливість узгодженої інфраструктури є критично важливою. Це знижує ризик операційної неефективності та забезпечує плавний перехід між постачальниками хмарних послуг, якщо в цьому коли-небудь виникне необхідність.

1.5 Трудомісткість розгортань

Як зазначалося раніше, ручне розгортання інфраструктури може бути дуже трудомістким процесом. Якщо говорити про інфраструктуру, на якій зосереджена ця магістерська робота, то процес розгортання цієї інфраструктури вручну може зайняти кілька днів у кількох середовищах. Дуже важливо також задокументувати

налаштування інфраструктури для майбутньої стійкості, що також збільшить витрати часу на все розгортання.

Ця робота буде зосереджена на розгортанні інфраструктури в хмарному середовищі. Результат практичної частини може бути модифікований у майбутньому для розгортання інфраструктури в інших середовищах.

Аналітична частина цієї дипломної роботи дасть огляд найбільш поширених помилок під час розгортання інфраструктури та управління життєвим циклом. Наступні проблеми були обрані через вплив на кібербезпеку, який вони можуть мати для бізнесу.

Розділ присвячений таким питанням: процедури резервного копіювання та відновлення, збереження облікових даних у системах контролю версій, неналежне керування доступом та керування виправленнями.

1.6 Процедури резервного копіювання та відновлення

Дані є одним із найважливіших активів незалежно від розміру бізнесу, і завжди існує ризик втратити дані. Незалежно від того, чи сталася втрата даних через апаратний збій, кібератаку або неправильне управління доступом, це створить багато проблем для компанії або, в гіршому випадку, закриє компанію. Наявність хорошої стратегії резервного копіювання має важливе значення для безпеки даних, оскільки не існує гарантованого методу запобігання витоку даних.

Втрата даних може статися внаслідок:

- випадкове видалення або зміна даних;
- апаратний збій;
- неналежне управління правами доступу;
- загублені або вкрадені пристрої.

Оскільки компанії залежать від наявності та захисту своїх даних, необхідно регулярно впроваджувати та тестувати належні процедури резервного копіювання та відновлення. Планування починається з визначення того, для яких даних потрібно створювати резервні копії та як регулярно має відбуватися резервне копіювання. Резервне копіювання таких даних, як резервні копії баз даних, має здійснюватися принаймні раз на день.

Найпоширенішими помилками, яких припускаються стартапи та малий бізнес з резервним копіюванням, є:

- покладаючись на людей, які створюють резервні копії вручну;
- відновлювальні процедури відсутні;
- для хмарного сховища не встановлено жодних правил;
- шифрування резервних копій.

Покладатися на людей, які роблять регулярні резервні копії, не є стійкою стратегією резервного копіювання. Стартапи та малий бізнес, як правило, мають одну ключову особу, яка відповідатиме за цю процедуру. Якщо виникне потреба у відновленні даних, а ключова особа опиниться у відпустці, не написавши жодної документації щодо процедури відновлення, компанія може зіткнутися з величезними проблемами.

Оскільки все більше компаній зберігають свої дані в хмарі, необхідно впровадити належне управління та управління доступом. Якщо цього не зробити, це може призвести до несанкціонованого доступу, витоку даних і втрати даних. Крім того, критично важливе сховище, яке зберігається в хмарному сховищі, має бути зашифроване за замовчуванням.

Оскільки вся інфраструктура буде розгорнута та керована Ansible, немає необхідності робити резервні копії конфігураційних файлів. Вихідні коди програм, які присутні в системах, повинні зберігатися в системах контролю версій, а це означає, що немає необхідності робити їх резервні копії. Однак автоматизоване резервне копіювання баз даних має бути налаштоване. В ідеалі автоматизоване

резервне копіювання має створювати резервні копії всіх баз даних, які присутні в системі, і зберігати їх протягом 30 днів. За потреби резервні копії, старіші за 30 днів, можна зберігати в довгостроковому сховищі.

Найефективнішим способом створення резервної копії баз даних є використання вбудованої функції експорту бази даних. Наприклад, в базах даних MySQL функція називається mysqldump, а в PostgreSQL функція називається pg_dump.

1.7 Закріплення облікових даних у системах контролю версій

Одним з найпоширеніших способів зберігання вихідного коду додатків є використання публічних систем контролю версій, таких як GitHub і GitLab. Якщо розробники не звертають уваги на те, що вони завантажують у ці публічні середовища, вони можуть завантажити секретні дані помилково. Секретні дані, такі як ключі API, ключі шифрування, токени OAuth, сертифікати, PEM-файли, паролі та парольні фрази, можуть бути використані для доступу до інших ресурсів, якими керує компанія.

Якщо репозиторій міститиме секрети або облікові дані для таких ресурсів, як сегменти AWS S3, бази даних, системи управління взаємовідносинами з клієнтами або щось подібне, це залишить компанію в вразливому стані до витоку даних. Єдиним способом видалення секретів є заміна поточного репозиторію на новий.

Створення нового репозиторію є важливим, оскільки існують доступні інструменти, такі як Gitrob, які використовуються для пошуку потенційно конфіденційних файлів, надісланих до публічних репозиторіїв на GitHub. Він може клонувати репозиторії, що належать організації, до певної глибини, і перебирати історію комітів і в процесі, позначати потенційно конфіденційні файли.

Оскільки принцип інфраструктури як коду слідує практиці розробки програмного забезпечення, вихідний код рішення повинен зберігатися в системі контролю версій, такій як Github. Під час початкового налаштування рішення користувачеві пропонується ввести облікові дані автентифікації постачальника хмарних послуг, які будуть збережені в зашифрованому вигляді за допомогою ansible-vault. Файл не буде записано до віддаленого сховища, оскільки файл реєстраційних даних буде типово проігноровано. gitignore.

1.8 Неправильне керування доступом

Належний контроль доступу є одним із найважливіших компонентів хмарної безпеки. Доступ до самої хмарної платформи повинен бути мінімальним, а ті, хто має до неї доступ, повинні мати мінімальні права доступу для виконання своїх завдань. Для забезпечення належної безпеки повинен бути реалізований протокол привілейованого доступу.

Коли стартапи або невеликі компанії починають використовувати сховище AWS S3 без попередніх знань, вони можуть налаштувати сегмент зберігання з занадто публічними правами доступу. Одна з найбільших помилок, яку може зробити користувач хмари, — це налаштування сегмента сховища за допомогою параметра «автентифікований users», вважаючи, що це користувачі, які належать до організації або до відповідної програми. Це неправильно, оскільки «автентифіковані користувачі» означають будь-кого з автентифікацією AWS, тобто будь-якого клієнта AWS. Це непорозуміння може призвести до того, що об'єкти зберігання залишаться у відкритому доступі.

Оскільки розробники повинні мати мінімальний обсяг доступу до хмарної консолі, необхідно розробити рекомендації для створення облікових записів розробників. Рекомендації створюватимуть облікові записи користувачів у

екземплярах EC2 і додадуть їхні відкриті ключі SSH до свого облікового запису, щоб користувачі могли отримати доступ до системи без використання паролів.

1.9 Керування патчами

Основною метою виправлення є забезпечення стабільного та безпечного середовища. Мета стратегії виправлення полягає в тому, щоб максимізувати продуктивність, пом'якшити проблеми з безпекою та покращити доступність системи. Оскільки деякі системи можуть бути схильні до нещодавно виявлених вразливостей безпеки, дуже важливо вирішити ці проблеми якомога швидше. В іншому випадку системи, які залежать від нього, можуть працювати неоптимально або погіршуватися з точки зору продуктивності.

Виправлення повинно проводитися регулярно, наприклад, кожні 30 днів і виконуватися автоматично. Під час процесу в системі встановлюються засоби безпеки та виправлення помилок. Патчі слід застосовувати спочатку в середовищах розробки або тестування, щоб переконатися, що на системи не впливають зміни, внесені під час процесу. Якщо після завершення процесу виникнуть будь-які проблеми, з ними можна впоратися, а виправлення можна застосувати до того, як виправлення відбудеться у виробничому середовищі.

Однією з головних переваг правильного керування виправленнями є безпека мережі. На жаль, управління виправленнями зазвичай впроваджується після того, як компанія зіткнулася з витоком даних і хоче гарантувати, що дані інших компаній залишаться недоторканими. Завчасний захист мережі допоможе запобігти крадіжці даних, юридичним проблемам та репутаційній шкоді.

Ще однією перевагою керування виправленнями є відповідність. У міру того, як зростає кількість порушень безпеки, збільшується і кількість правил, яких повинні дотримуватися компанії. Це змушує компанії впроваджувати найкращі

практики безпеки. Недотримання правил потенційно може призвести до юридичних санкцій.

Однією з головних помилок компаній є неактуальність програмного забезпечення. Хакери завжди шукають вразливості, якими можна скористатися та отримати доступ до внутрішніх даних та ресурсів компанії. Впровадження управління виправленнями як на серверах, так і на робочих станціях є важливою частиною безпеки даних.

Рішення включатиме посібник, який буде виправляти екземпляри кожні 30 днів. Буде два окремих завдання виправлення для веб-серверів та екземплярів баз даних. Ці завдання будуть налаштовані на виконання кожні 30 днів за допомогою Cron на керуючому вузлі Ansible.

1.10 Розгортання інстансу

Базовою операційною системою, яка працюватиме на екземплярах, є Ubuntu. Це найпопулярніша ОС для ведення веб-сайтів. Він заснований на Debian і може бути завантажений з офіційного сайту Ubuntu. Версія, яка буде використовуватися, - 20.04 LTS (кодова назва Focal Fossa), яка буде отримувати оновлення до 2030 року.

Автоматизація розгортання інстансів допомагає забезпечити узгодженість середовища, усуває необхідність розгортання з консолі AWS, скорочує час, необхідний для застосування початкової конфігурації на інстансі.

У компаніях, де розгортання EC2 є повсякденним явищем, це допоможе заощадити значну кількість часу, витраченого на це завдання. Розробники можуть швидко підготувати нові екземпляри без потреби в системному адміністраторі.

1.11 Розгортання веб-сервера

Основні принципи, яких допоможе досягти автоматизація розгортання веб-сервера:

- узгодженість конфігураційних файлів;

- більш швидке розгортання нових веб-додатків;

- розробники можуть розгортати нові віртуальні хости без необхідності системного адміністратора.

Хоча плюси автоматизації завдань, пов'язаних з веб-сервером, переважують мінуси налаштування всього вручну, все ж є деякі завдання, які можуть вимагати ручного втручання.

Наприклад, додавання нових шаблонів конфігураційних файлів, внесення змін до наявних конфігураційних файлів, створення та внесення змін до користувацьких конфігураційних файлів, які не створюються за допомогою стандартних шаблонів розгортання.

Основними вимогами до веб-сервера є:

- легкий;

- висока продуктивніст;

- доступно з типових репозиторіїв.

У цій дипломній роботі Nginx буде використовуватися як програмне забезпечення веб-сервера через полегшену структуру та здатність обробляти веб-сайти з високим трафіком у порівнянні з Apache. Якщо порівнювати складність налаштування Apache і Nginx, то Apache є явним переможцем. Це робить автоматизацію розгортання Nginx ще більш важливою, оскільки можуть статися помилки конфігурації.

1.12 Розгортання баз даних

Основні принципи автоматизації розгортання баз даних допоможуть досягти:

- нові бази даних створюються без використання командного рядка;

- реплікація бази даних налаштована послідовно;

- під час процесу встановлення користувачі, які не використовуються, і бази даних видаляються за замовчуванням, під час ручного процесу встановлення, який потрібно виконати вручну;

- резервне копіювання бази даних налаштовано за замовчуванням.

Коли потрібно створити нову базу даних, користувач повинен увійти в керуючу базу даних і виконати команду для створення бази даних. Якщо реплікація також потрібна, це вимагатиме додаткових кроків для двох або більше екземплярів бази даних. Автоматизація цього процесу допоможе заощадити час, витрачений на створення бази даних і процес налаштування реплікації. Крім того, конфігурація як бази даних, так і реплікації будуть виконуватися послідовно.

Коли виникає потреба у відновленні з резервної копії, цей процес також можна автоматизувати, щоб забезпечити послідовне відновлення бази даних. Якщо є додаткові дії, які необхідно виконати перед відновленням, інший варіант - виконати команду відновлення вручну.

Основними вимогами до бази даних є:

- структура бази даних для запобігання дублюванню даних;

- доступне керування користувачами;

- простота в налаштуванні;

- реплікація даних повинна бути доступною;

- доступно з типових репозиторіїв.

Було розглянуто використання наступних баз даних: PostgreSQL, MySQL і MariaDB. З трьох була обрана MySQL. Станом на березень 2022 року це друге за популярністю програмне забезпечення для баз даних після Oracle, тоді як PostgreSQL займає 4-е місце, MariaDB займає 13-е місце. У порівнянні з

PostgreSQL, продуктивність і більш проста конфігурація переважають розширені функції, які пропонує PostgreSQL. Крім того, в Ansible є модуль для налаштування реплікації MySQL.

1.13 Моніторинг розгортання

Основні переваги автоматизації розгортання та налаштування рішення для моніторингу допоможуть досягти таких принципів:

- швидке та послідовне розгортання;
- конфігурацією сервера/агентів можна керувати централізовано;
- додавання нових вузлів не потребуватиме ручного введення.

Основними вимогами, що пред'являються до рішення для моніторингу, є:

- широкі можливості налаштування;
- інтеграція з інструментами візуалізації повинна бути доступною.

Було розглянуто використання наступних рішень моніторингу: Zabbix, Prometheus і Nagios. З трьох рішень буде використовуватися Zabbix. Потужний і сучасний веб-графічний інтерфейс переважує можливості Nagios. Простий і швидкий процес налаштування важливіший за приріст продуктивності, який пропонує Prometheus.

1.14 Резервне копіювання та відновлення

Основні переваги автоматизації розгортання та налаштування рішення для моніторингу допоможуть досягти таких принципів:

- процес відновлення не вимагає ручного втручання;

- резервне копіювання налаштовується послідовно на інстансах;

- у складних налаштуваннях менше часу витрачається на налагодження конфігурації резервного копіювання/агента;

- не покладається на людей, які виконують резервне копіювання вручну.

Хоча ручне резервне копіювання баз даних може бути виконано, це не є стійкою стратегією. Люди роблять помилки і іноді можуть забути про створення резервної копії бази даних або якоїсь іншої системи. Автоматизація цього процесу гарантує, що резервне копіювання буде здійснюватися регулярно та послідовно. Рішення автоматично налаштує резервне копіювання серверів баз даних. Резервне копіювання конфігураційних файлів не знадобиться, як згадувалося раніше, Ansible налаштує служби, а вихідний код плейбука зберігається на окремій машині, а в ідеалі також у системі керування версіями.

У складних системах резервного копіювання, таких як Bacula або Bareos, керування конфігураційними файлами вручну може зайняти багато часу через складність системи. Якщо є проблема з конфігурацією, яка призведе до помилки резервного копіювання, налагодження цієї проблеми може бути дуже довгим і стресовим завданням.

Основною вимогою до рішення для резервного копіювання є:

- хмарне сховище має підтримуватися.

Borg, Bareos і Duplicity відповідають критеріям. У цій дипломній роботі буде використано дублювання, оскільки єдиною метою рішення є зберігання резервних копій баз даних, а розширені можливості Bareos не потрібні.

1.15 Постачальники послуг хмарної інфраструктури

За останні кілька років використання хмарних обчислень значно зросло. У 2022 році впровадження публічних хмар зросло, тоді як використання приватної хмари скоротилося. Компанії збільшують інвестиції в публічну хмару. Оскільки все більше підприємств використовують Azure як постачальника хмарних послуг,

вони скорочують розрив між лідером AWS. Протягом останніх трьох років управління витратами на хмару було одним із головних пріоритетів для компаній, які використовують постачальників хмарних послуг.

Станом на 4 квартал 2022 року світовий ринок послуг хмарної інфраструктури досяг рекордного рівня, оскільки витрати зросли на 37% і склали понад 30 мільярдів доларів США. AWS залишався домінуючим постачальником хмарних послуг, на його частку припадало 32% загальних витрат. Azure збільшила свою частку з 15% до 18%. Хмара Google була третім за величиною постачальником послуг з часткою 6%.

Рисунок 1.1 - Світові витрати на хмарну інфраструктуру та щорічне зростання

Основними перевагами використання сервісів хмарної інфраструктури є:

- гнучкі початкові інвестиційні витрати;
- часті та простіші оновлення продукту;
- скорочення IT-підтримки, що виконується внутрішніми ресурсами
- спільнота користувачів для останніх версій та функцій;
- ефективний для багатокористувацького використання (масштабованість, відновлюваність, виправлення, безпека).

Основними недоліками використання сервісів хмарної інфраструктури є:

- витрати можуть вийти з-під контролю, якщо не впроваджується належне управління;
- доступ до даних може отримати третя сторона;
- витрати на довгострокові проекти повинні бути на місці;
- під час апаратного збою ви не можете контролювати процес відновлення;
- значна крива навчання, адекватно навчений персонал коштує дорого.

Як згадувалося раніше, управління витратами на хмару та хмарне управління є головними проблемами для компаній незалежно від зрілості хмари. Ще однією проблемою є управління ліцензіями на програмне забезпечення, які використовуються в публічних хмарних середовищах. Зокрема, розуміння фінансових наслідків ліцензійного програмного забезпечення, що працює в хмарі, забезпечення дотримання правил і складності ліцензійних правил у загальнодоступній хмарі.

У цій дипломній роботі AWS буде використовуватися через популярність та чудову інтеграцію з Ansible. Правильно налаштувати відповідні сервіси, не маючи попереднього досвіду, було б складним завданням. Також є можливість використовувати Amazon Lightsail для розгортання стека LAMP. Незважаючи на те, що він дуже простий у налаштуванні, він не пропонує жодних послуг моніторингу чи резервного копіювання за замовчуванням.

Обчислювальні екземпляри EC2 будуть використовуватися в цій дипломній роботі для розгортання сервісів. Причина в тому, що він більш економічно ефективний, має контроль над конфігурацією на стороні сервера, можливість запускати ігрові майданчики, де розробники можуть тестувати нові рішення, не втручаючись у тестове та виробниче середовище. Якщо компанія досягне точки, коли виникне потреба перейти на локальне налаштування або гібридне хмарне рішення, вона може використовувати те саме рішення для налаштування інфраструктурних сервісів.

На маркетплейсі AWS вже доступні подібні шаблони CloudFormation. Однак більшість із них виконують те саме розгортання, що й Amazon Lightsail, і та сама проблема зберігається, резервне копіювання та моніторинг не реалізовані за замовчуванням. Крім того, шаблони CloudFormation можна використовувати лише в AWS, тоді як посібник Ansible можна використовувати і в інших місцях.

На рисунку 1.1 представлено Інструмент ANSIBLE.

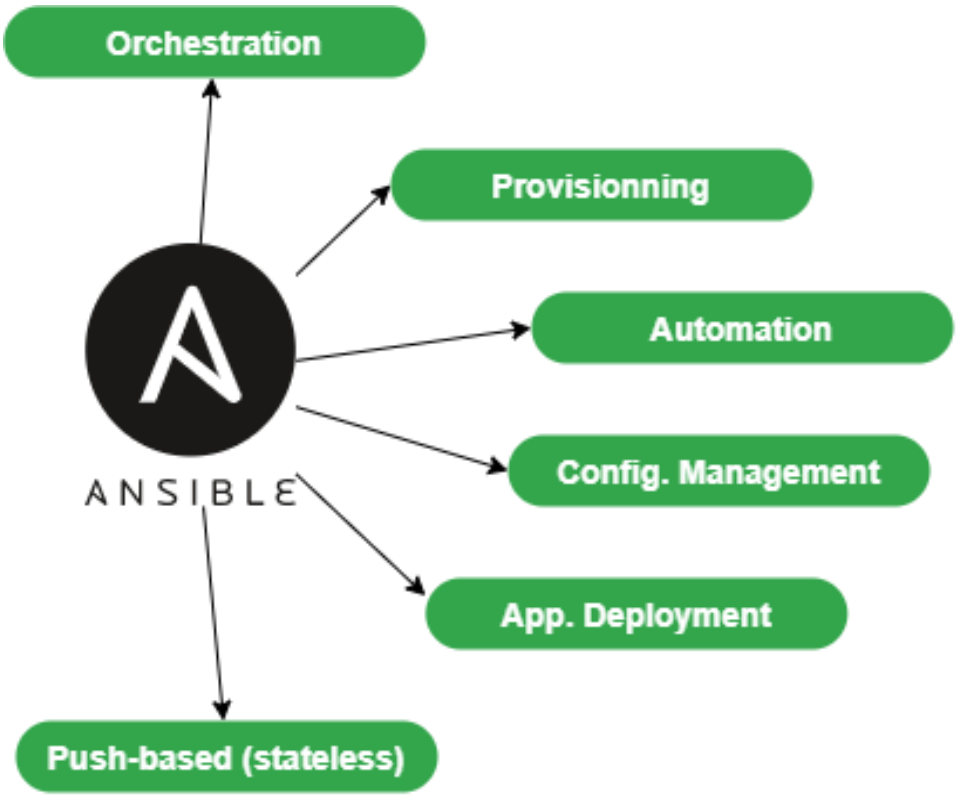

Рисунок 1.2 - Інструмент ANSIBLE

Ansible - це засіб керування конфігураціями серверів (віртуальних та виділених), збереження їх станів, доставки та розгортання на них ПЗ. Якщо перефразувати простіше, Ansible — це інструмент автоматизації роботи з серверами і на них.

Ansible централізує та автоматизує задачі адміністратора. Це є: agentless (не вимагає спеціального розгортання на клієнтах), idempotent (однаковий ефект під час кожного запуску).

Він використовує протокол SSH для віддаленого налаштування клієнтів Linux або протокол WinRM для роботи з клієнтами Windows. Якщо жоден із цих протоколів недоступний, Ansible завжди може використовувати API, що робить Ansible справжнім знахідкою для конфігурації серверів, робочих станцій, служб докерів, мережевого обладнання тощо (фактично, майже всього).

Оскільки Ansible здебільшого базується на push-завантаженні, він не зберігатиме стан своїх цільових серверів між кожним своїм виконанням. Навпаки, він виконуватиме нові перевірки стану під час кожного виконання.

Це допоможе у:

- забезпеченні (розгортанні нової віртуальної машини);
- розгортанні додатків;
- управлінні конфігурацією;
- автоматизації;
- оркестровка (коли використовується більше ніж 1 target).

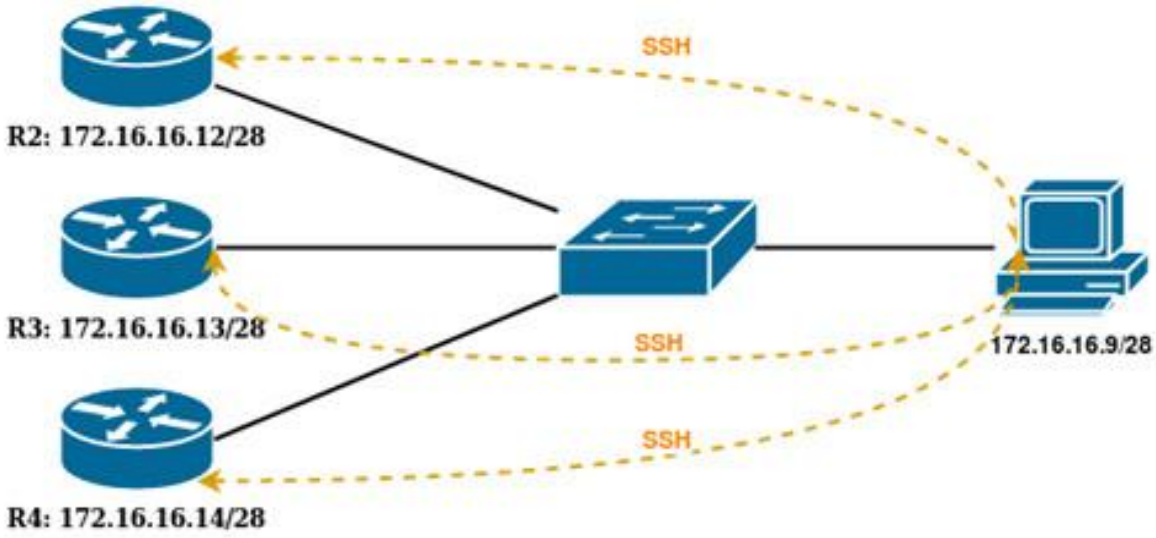

Рисунок 1.3 - Схема роботи Ansible

Алгоритм роботи Ansible (рисунок 1.2) виглядає так:

Адміністратор запускає завдання до Ansible;

Ansible виконує паралельне підключення до списку пристроїв, зазначених у завданні. Число паралельних сесій за умовчанням дорівнює 5, підключення виконується за протоколом SSH, підтримується з'єднання через логін/пароль та сертифікат;

Ansible виконує збирання інформації про пристрій;

Ansible виконує завдання;

Якщо завдань на виконання більше немає, Ansible відключається від пристрою, якщо є - виконуються наступні завдання.

2 ВПРОВАДЖЕННЯ РІШЕНЬ

2.1 Діаграми інфраструктури AWS

Рисунок 2.1 - Схема інфраструктури AWS з балансувальником навантаження

Рішення зможе розгорнути інфраструктуру, яка представлена на рисунку вище. Він складається з 2-х веб-серверів, розташованих у приватній підмережі, які балансуються навантаженням за допомогою Elastic Load Balancer. 2 сервера баз даних з конфігурацією master-slave, один сервер моніторингу та 1 хост Bastion.

Рисунок 2.2 - Схема інфраструктури AWS без балансувальника навантаження

Рішення також зможе розгорнути інфраструктуру, яка представлена на рисунку вище. Ця конкретна конфігурація не розгортає розподілювач навантаження для веб-серверів. Замість цього їх можна використовувати як два різних веб-сервера. Обидва розміщені в загальнодоступній підмережі та мають статичні IP-адреси.

2.2 Огляд процесу автоматизованого розгортання в AWS

Автоматизований робочий процес розгортання буде таким:

- створюється віртуальна приватна хмара, під час якої додається таблиця маршрутів за замовчуванням, список контролю доступу до мережі та група безпеки;

- створюється приватна підмережа;

- створюється і налаштовується публічна підмережа. Сервери, підключені до цієї підмережі, автоматично отримають публічну IP-адресу;

- Інтернет-шлюз створюється і підключається до VPC;

- створюється публічна таблиця маршрутів і налаштовуються маршрути;

- публічна підмережа пов'язана з загальнодоступною таблицею маршрутів;

- створено та налаштовано групу безпеки для веб-серверів;

- створено та налаштовано групу безпеки для серверів баз даних;

- створено та налаштовано групу безпеки для моніторингу серверів;

- NAT Gateway створений і налаштований для приватної підмережі;

- маршрут до Інтернету додається до таблиці маршрутів за замовчуванням через NAT Gateway;

- екземпляр Bastion розгортається та налаштовується зі статичною IP-адресою;

- інстанс моніторингу розгортається та налаштовується зі статичною IP-адресою;

- екземпляри баз даних розгортаються та налаштовуються;

- розгортаються та налаштовуються екземпляри веб-серверів;

- налаштовано сегмент S3 для резервного копіювання;

- якщо вказано, розподілювач навантаження буде налаштовано для веб-серверів.

2.3 Налаштування вузла Ansible Control

Усі посібники та команди Ansible запускаються з блоку керування. Вузол, про який йде мова, повинен мати підключення до всіх серверів в інфраструктурі. Вузлом управління може бути ноутбук, загальний робочий стіл або віртуальна машина. Однак немає можливості використовувати машину з Windows як вузол управління. Рішенням у цьому випадку буде встановлення програмного забезпечення віртуалізації на машину та запуск віртуальної машини.

У даній дипломній роботі використовується віртуальну машину, а операційна система, яка використовується Ubuntu 18.04. Розглянута віртуальна машина має 1 vCPU і 1 ГБ оперативної пам'яті. Щоб встановити Ansible і залежності від системи, що використовує ту саму операційну систему, потрібно виконати наступні команди.

```
$ sudo apt update && sudo apt install software-properties-common -y &&
sudo apt-add-repository --yes --update ppa:ansible/ansible && sudo apt
install ansible python-pip python3-pip -y

$ pip install botocore boto boto3 zabbix-api

$ pip3 install botocore boto boto3 zabbix-api
```

Після того, як необхідні пакунки встановлені, посібник має бути завантажений з github.com. Найпростіший спосіб зробити це – скористатися командою git clone.

```
$ git clone https://github.com/karlmjogila/thesis-aws.git

$ cd thesis-aws/
```

Після клонування репозиторію та зміни каталогу користувач повинен створити обліковий запис у консолі AWS. Користувач повинен мати повний доступ до ресурсів, які використовуються під час розгортання. Щоб створити користувачів, увійдіть у консоль AWS і перейдіть до IAM -> Users

-> Додати користувача.

Процес створення користувача:

- створіть ім'я користувача та переконайтеся, що тип доступу "Програмний доступ"

Рисунок 2.3 - AWS IAM

- перейдіть на вкладку "Прикріпити наявні політики напряму" та виберіть такі політики: AmazonEC2FullAccess, AmazonVPCFullAccess, AmazonS3FullAccess, AmazonRoute53FullAccess, AWSCertificateManagerFullAccess

- натисніть Далі -> Далі -> Переконайтеся, що застосовано правильні правила -> Створити користувача.

- обов'язково завантажте файл .csv, що містить обидва ключі доступу, і збережіть облікові дані в менеджері паролів. Вони знадобляться пізніше.

Повернувшись на керуючий вузол, виконайте наступну команду:

$ ansible-vault створити ключі

Користувачеві доведеться створити пароль, який буде використовуватися для шифрування файлу облікових даних. Після створення пароля відкривається редактор, в якому користувач повинен зберігати ключі доступу, створені на попередньому кроці. Репозиторій Github містить приклад файлу credentials, де користувач повинен заповнити значення. Облікові дані зберігаються у форматі ключ-значення.

AWS_SECRET_ACCESS_KEY: <Секретний ключ доступу> AWS_ACCESS_KEY_ID: <Ідентифікатор ключа доступу>

Після того, як файл буде заповнений значеннями, вийдіть з редактора і виконайте наступну команду:

$ ansible-playbook site.yml –-ask-vault-pass

Користувачеві буде запропоновано ввести пароль, який використовувався для шифрування файлу облікових даних. Після цього розпочнеться процес розгортання і після його завершення; інфраструктура розгорнута в AWS.

3 РОЗРОБКА РОЗГОРТАННЯ ВЕБ-СЕРВЕРА

Під час розробки нові віртуальні хости можуть бути додані в групових змінних, після чого настройка і розгортання віртуального хоста автоматично завершується.

Робочий процес розгортання виглядає наступним чином:

- список пакетів оновлюється на сервері;
- встановлено пакет Nginx;
- Nginx налаштований на запуск під час завантаження.

При додаванні нового віртуального хоста до списку створюється каталог webroot, створюється конфігураційний файл, запускається перевірка конфігурації, і якщо перевірка пройдена, Nginx перезавантажується, а віртуальний хост активний (Додаток 7 для розгортання Nginx). Якщо розподілювач навантаження не вказано, буде налаштовано також сертифікат Let's Encrypt (див. Додаток 9 для розгортання Let's Encrypt).

3.1 Розробка розгортання баз даних

Під час розгортання бази даних налаштовується реплікація MySQL (див. Додаток 8 для розгортання бази даних Ansible). Робочий процес розгортання виглядає наступним чином:

- список пакетів оновлюється на серверах;
- на серверах встановлено пакет MySQL;
- служба запускається і налаштовується на запуск при завантаженні;
- оновлено пароль користувача root MySQL;
- користувачі, які не використовуються, і бази даних видаляються;
- створюються бази даних і користувачі баз даних;
-створюються користувачі реплікації;
- Slave налаштований на реплікацію;
- запускається реплікація.

3.2 Розробка розгортання моніторингу

Основною метою рішення для моніторингу є попередження відповідального персоналу про недоступні послуги або високе споживання ресурсів екземпляра. Під час розгортання виконуються такі завдання:

- на сервер додано репозиторій Zabbix;

- список пакетів оновлюється;

- встановлено пакети баз даних Zabbix та MySQL;

- сервіси Zabbix і MySQL налаштовуються і запускаються автоматично при завантаженні;

- на сервері запускаються служби. (див. Додаток 2 для розгортання Ansible Zabbix);

Екземпляри будуть додані в Zabbix за допомогою модуля zabbix_host в Ansible.

3.3 Розробка розгортання резервного копіювання бази даних

Оскільки інфраструктура описана в коді, резервне копіювання конфігураційних файлів не потрібне, оскільки, теоретично, все повинно управлятися Ansible. Єдине, що потрібно часто резервувати, це бази даних. Ця дисертація буде полягати у використанні скрипту mysqldump та duplicity для зберігання резервної копії в сегменті S3 (див. Додаток 10 для розгортання резервних копій Ansible).

Робочий процес розгортання виглядає наступним чином:

- дублювання з необхідними модулями python встановлюються на серверах баз даних;

- сценарій резервного копіювання буде створено з файлу шаблону та завантажено на сервери бази даних;

- завдання Cron налаштовано на резервне копіювання баз даних один раз на день.

ВИСНОВКИ

У роботі проаналізовано різних постачальників хмарних послуг, проаналізував типові помилки, що допускаються при розгортанні інфраструктури та управлінні життєвим циклом у стартапах та малому бізнесі, а також проаналізував інфраструктурні сервіси, щоб знайти оптимальні для використання в рішенні.

У практичному розділі було розроблено посібник Ansible для автоматизації розгортання інфраструктури, який допомагає підтримувати послідовність майбутніх розгортань. У рішенні реалізовано інфраструктурні сервіси, які виявилися оптимальними в розділі аналізу. Також було усунуто типові помилки в управлінні життєвим циклом та розгортанні.

Рішення розгортає такі сервіси:

- 2 Веб-сервери – Nginx використовується як програмне забезпечення веб-сервера;

- 2 Сервери баз даних – MySQL використовується як програмне забезпечення для баз даних;

- 1 Сервер моніторингу – Zabbix використовується як рішення для моніторингу;

- 1 сегмент AWS S3 для резервного копіювання бази даних – для синхронізації даних використовується дублювання;

- 1 хост Bastion для доступу до серверів у VPC.

Рішення було розроблено з використанням найкращих практик безпеки, інфраструктури та кодування. Документація з прикладами також включена для полегшення налаштування. На додаток до цього, включено додатковий посібник для розгортання середовища локально на сервері розробника Робочих станцій.

ПЕРЕЛІК ПОСИЛАНЬ

1. What Is Patch Management? [Online]. Available: https://consoltech.com/blog/patch-management/
2. Evaluation of Infrastructure as a Code for Enterprise Automation. 2018 [Online].
3. Available: https://is.muni.cz/th/liwzz/IaC-Final-el.pdf
4. Cloud Computing Trends: 2019 State of the Cloud Survey. 2019 [Online]. Available: https://www.flexera.com/blog/cloud/2019/02/cloud-computing-trends-2019-state-of-the-cloud-survey/
5. Cloud market share Q4 2022 and full-year 2022. [Online]. Available: https://www.canalys.com/static/press_release/2022/Canalys---Cloud-market-share-Q4-2022-and-full-year-2022.pdf
6. Amazon Lightsail. [Online]. Available: https://aws.amazon.com/lightsail/
7. Common Cloud Configuration Mistakes. [Online]. Available: https://www.darkreading.com/cloud/five-common-cloud-configuration-mistakes/a/did/1335768 [Accessed 29 April 2020].
8. Security Mistakes Organisations Make When Adopting Cloud. [Online]. Available: https://www.cloudreach.com/en/resources/blog/7-security-mistakesorganizations-make-when-adopting-cloud/ [Accessed 29 April 2020].
9. mysql_replication – Manage MySQL replication. [Online]. Available: https://docs.ansible.com/ansible/latest/modules/mysql_replication_module.html

10. 2018 HISCOX Small Business Cyber Risk Report [Online]. Available: https://www.hiscox.com/documents/2018-Hiscox-Small-Business-Cyber-Risk Report.pdf

11. Top 10 AWS Security Mistakes and Solutions. [Online]. Available: https://resources.netskope.com/cloud-security-solution-white-papers/top-10-awssecurity-mistakes-and-solutions zabbix_host – Create/update/delete Zabbix hosts. [Online].

12. Usage statistics of Linux for websites. [Online]. Available: https://w3techs.com/technologies/details/os-linux

13. The Ubuntu lifecycle and release cadence. [Online]. Available: https://ubuntu.com/about/release-cycle

14. DB-Engines Ranking. [Online]. Available: https://db-engines.com/en/ranking

15. Are Developers Running Your Infrastructure? [Online]. Available: https://www.rhythmictech.com/white-papers/are-developers-running-yourinfrastructure/

16. Gitrob: Putting the Open Source in OSINT. [Online]. Available: https://github.com/michenriksen/gitrob

17. Redis vs MySQL - A quick database comparison. [Online]. Available: https://tableplus.com/blog/2018/10/redis-vs-mysql-database-comparison.html

18. Five common backup mistakes start-ups make. [Online]. Available: https://www.startupdonut.co.uk/blog/18/06/five-common-backup-mistakes-start-upsmake

19. BJC Healthcare data breach, 33,000 affected. [Online]. Available: https://www.scmagazine.com/home/security-news/data-breach/bjc-healthcare-databreach-Accentuate the negative: Accenture exposes data related to its enterprise cloud platform. [Online]. Available: https://www.scmagazine.com/home/security-news/databreach/accentuate-the-negative-accenture-exposes-data-related-to-its-enterprise-cloudplatform/

20. 10 Data Security Mistakes Startups Can't Afford to Make. [Online]. Available: https://www.stamfordadvocate.com/news/article/10-Data-Security-Mistakes-Startups-Can-t-Afford-8318455.php

21. Information Technology Services Issues and Challenges with A Case Study in Small Medium Enterprises. [Online]. Available: https://www.researchgate.net/publication/308009820_Information_Technology_Services_Issues_and_Challenges_with_A_Case_Study_in_Small_Medium_Enterprises

22. Data Breaches Caused by Misconfigured Servers. [Online]. Available:https://www.scmagazine.com/home/opinions/data-breaches-caused-by-misconfiguredservers/

23. The Importance of Backup and Recovery. [Online]. Available: https://www.grayanalytics.com/blog/the-importance-of-backup-and-recovery

Додаток 1 – Роль розгортання та управління Ansible VPC

vpc.yml:

```
---
- hosts: localhost connection: local
  gather_facts: no environment:
    AWS_ACCESS_KEY_ID: "{{ aws_access_key_id }}"
    AWS_SECRET_ACCESS_KEY: "{{ aws_secret_access_key }}"
    AWS_REGION: "{{ AWS_REGION }}"
  vars_files:
    - keys tasks:
    - name: Include VPC role include_role:
        name: "{{ role_name }}" vars:
        create_initial_vpc: True loop:
        - vpc
        - bastion
        - le
        - zabbix
        - nginx
        -  mysql loop_control:
        loop_var: role_name
    - name: Set create_initial_vpc to fact set_fact:
        create_initial_vpc: True
```

main.yml:

```
---
- name: Deploy initial VPC include_tasks:
  deploy_vpc.yml
  when: create_initial_vpc is defined and create_initial_vpc ==
True
```

```
- name: Gather VPC information include_tasks:
  get_vpc_information.yml
  when: fetch_vpc_info is defined and fetch_vpc_info == True
```

deploy_vpc.yml:

```
---
# Створити VPC
-     include: create_vpc.yml
# Створити приватну підмережу
-     include: create_private_subnet.yml
# Створити публічну підмережу
-     include: create_public_subnet.yml
# Створити інтернет-шлюз
-     include: create_igw.yml #
Створити публічну таблицю
маршрутів
- включати: create_public_rt.yml
# Створити охоронну групу для бастіону
- включати: create_bastion_sg.yml
# Створення групи безпеки для моніторингу
- включати: create_monitoring_sg.yml
# Створення групи безпеки для веб-серверів
- включати: create_webserver_sg.yml
# Створення групи безпеки для серверів баз даних
- включати: create_db_sg.yml
# Моніторинг оновлення групи безпеки
-     include: update_monitoring_sg.yml
# Оновити групу безпеки бази даних
- включати: update_db_sg.yml
# Створення шлюзу NAT для приватної підмережі
- включати: create_nat_gw.yml
# Створення таблиці маршрутів для приватної підмережі
- включати: create_private_rt.yml
```

create_vpc.yml:

```
- name: Create VPC ec2_vpc_net:
    name: "{{ vpc_name }}" cidr_block: "{{
    vpc_cidr_block }}"
    region: "{{ AWS_REGION }}"
```

```
    state: present tags:
      Name: "{{ vpc_name }}" Environment: "{{
      project_env }}"
  register: deployed_vpc

- name: Assign VPC ID to a variable set_fact:
    vpc_id: "{{ deployed_vpc.vpc.id }}"
```

create_private_subnet.yml

```
---
- name: Create private subnet for database servers ec2_vpc_subnet:
    state: present vpc_id: "{{ vpc_id }}"
    cidr: "{{ private_subnet_cidr_block }}" tags:
      Name: "{{ vpc_name }}_private_sn" register:
  private_subnet

- name: Assign private subnet ID to variable set_fact:
    private_subnet_id: "{{ private_subnet.subnet.id }}"
```

create_public_subnet.yml

```
---
- name: Create public subnet for webservers servers ec2_vpc_subnet:
    state: present vpc_id: "{{ vpc_id }}"
    cidr: "{{ public_subnet_cidr_block }}" map_public: yes
    tags:
      Name: "{{ vpc_name }}_public_sn" register:
  public_subnet

- name: Assign public subnet ID to variable set_fact:
    public_subnet_id: "{{ public_subnet.subnet.id }}"
```

create_igw.yml

```
---
- name: Create internet gateway ec2_vpc_igw:
    vpc_id: "{{ vpc_id }}" state: present
    tags:
      VPC: "{{ vpc_name }}" register: igw

- name: Set internet gateway ID to a variable set_fact:
    igw_id: "{{ igw.gateway_id }}"
```

create_public_rt.yml

```
---
- name: Create route table for public subnet ec2_vpc_route_table:
    vpc_id: "{{ vpc_id }}"
    subnets: "{{ public_subnet_id }}" routes:
      - dest: "0.0.0.0/0"
        gateway_id: "{{ igw_id }}" tags:
      Name: "{{ vpc_id }}_public_rt"
```

create_bastion_sg.yml

```
---
- name: Create security group for bastion hosts ec2_group:
    name: "{{ vpc_name }}_bastion_sg" description: "SG
    for bastion hosts" vpc_id: "{{ vpc_id }}"
    rules:
      - proto: tcp from_port: 22
        to_port: 22 cidr_ip: 0.0.0.0/0
    tags:
      Name: "{{ vpc_name }}_bastion_sg"
      Environment: "{{ project_env }}"
```

```
    register: bastion_sg

- name: Assign bastion security group properties to facts set_fact:
      bastion_sg_id: "{{ bastion_sg.group_id }}" bastion_sg_owner: "{{
      bastion_sg.owner_id }}"
      bastion_sg_name: "{{ bastion_sg.group_name }}"
```

create_monitoring_sg.yml

```
---
- name: Create security group for monitoring servers ec2_group:
      name: "{{ vpc_name }}_monitoring_sg" description: "SG
      for monitoring" vpc_id: "{{ vpc_id }}"
      tags:
        Name: "{{ vpc_name }}_monitoring_sg" Environment:
        "{{ project_env }}"
    register: monitoring_sg

- name: Assign monitoring security group properties to facts set_fact:
      monitoring_sg_id: "{{ monitoring_sg.group_id }}" monitoring_sg_owner:
      "{{ monitoring_sg.owner_id }}"
      monitoring_sg_name: "{{ monitoring_sg.group_name }}"
```

create_webserver_sg.yml

```
---
- name: Create security group for webservers ec2_group:
      name: "{{ vpc_name }}_webserver_sg" description:
      "SG for webservers" vpc_id: "{{ vpc_id }}"
      rules:
        - proto: tcp from_port: 80
          to_port: 80 cidr_ip: 0.0.0.0/0
        - proto: tcp
```

```
            from_port: 443
            to_port: 443 cidr_ip: 0.0.0.0/0
          - proto: tcp from_port: 10050
            to_port: 10050
            group_id: "{{ monitoring_sg_owner }}/{{ monitoring_sg_id
}}/{{ monitoring_sg_name }}"
          - proto: tcp from_port: 22
            to_port: 22
            group_id: "{{ bastion_sg_owner }}/{{ bastion_sg_id }}/{{ bastion_sg_name }}"
      tags:
        Name: "{{ vpc_name }}_webserver_sg" Environment:
        "{{ project_env }}"
    register: webserver_sg

- name: Assign webserver security group properties to facts set_fact:
      webserver_sg_id: "{{ webserver_sg.group_id }}" webserver_sg_owner:
      "{{ webserver_sg.owner_id }}"
      webserver_sg_name: "{{ webserver_sg.group_name }}"
```

create_db_sg.yml

```
---
- name: Create security group for db servers ec2_group:
      name: "{{ vpc_name }}_database_sg" description:
      "SG for db servers" vpc_id: "{{ vpc_id }}"
      tags:
        Name: "{{ vpc_name }}_database_sg" Environment:
        "{{ project_env }}"
    register: database_sg

- name: Assign db servers security group ID to a variable set_fact:
      database_sg_id: "{{ database_sg.group_id }}"
      database_sg_owner: "{{ database_sg.owner_id }}" database_sg_name:
      "{{ database_sg.group_name }}"
```

update_monitoring_sg.yml

```
---
- name: Update monitoring security group ec2_group:
    name: "{{ vpc_name }}_monitoring_sg" description: "SG
    for monitoring" group_id: "{{ monitoring_sg_id }}"
    vpc_id: "{{ vpc_id }}"
    state: present rules:
      - proto: tcp from_port: 10051
        to_port: 10051
        group_id: "{{ webserver_sg_owner }}/{{ webserver_sg_id
}}/{{ webserver_sg_name }}"
      - proto: tcp from_port: 10051
        to_port: 10051
        group_id: "{{ bastion_sg_owner }}/{{ bastion_sg_id }}/{{ bastion_sg_name }}"
      - proto: tcp from_port: 10051
        to_port: 10051
        group_id: "{{ database_sg_owner }}/{{ database_sg_id }}/{{ database_sg_name }}"
      - proto: tcp from_port: 22
        to_port: 22
        group_id: "{{ bastion_sg_owner }}/{{ bastion_sg_id }}/{{ bastion_sg_name }}"
      - proto: tcp from_port: 80
        to_port: 80 cidr_ip: 0.0.0.0/0
      - proto: tcp from_port: 443
        to_port: 443 cidr_ip: 0.0.0.0/0
```

update_dg_sg.yml

```
- name: Create security group for db servers ec2_group:
    name: "{{ vpc_name }}_database_sg"
```

```
    description: "SG for db servers" vpc_id: "{{ vpc_id
    }}"
    rules:
      - proto: tcp from_port: 3306
        to_port: 3306
        group_id: "{{ webserver_sg_owner }}/{{ webserver_sg_id
}}/{{ webserver_sg_name }}"
      - proto: tcp from_port: 10050
        to_port: 10050
        group_id: "{{ monitoring_sg_owner }}/{{ monitoring_sg_id
}}/{{ monitoring_sg_name }}"
      - proto: tcp from_port: 22
        to_port: 22
        group_id: "{{ bastion_sg_owner }}/{{ bastion_sg_id }}/{{ bastion_sg_name }}"
      - proto: tcp from_port: 3306
        to_port: 3306
        group_id: "{{ database_sg_owner }}/{{ database_sg_id }}/{{ database_sg_name }}"
```

create_nat_gw.yml

```
---
- name: Create NAT gateway for servers in private subnet ec2_vpc_nat_gateway:
    state: present
    subnet_id: "{{ private_subnet_id }}" wait: true
    if_exist_do_not_create: true register: nat_gw

- name: Assign NAT GW id to a variable set_fact:
    nat_gateway_id: "{{ nat_gw.nat_gateway_id }}"
```

create_private_rt.yml

```
---
- name: Create private subnet for database servers ec2_vpc_subnet:
    state: present vpc_id: "{{ vpc_id }}"
    cidr: "{{ private_subnet_cidr_block }}" tags:
      Name: "{{ vpc_name }}_private_sn" register:
  private_subnet

- name: Assign private subnet ID to variable set_fact:
    private_subnet_id: "{{ private_subnet.subnet.id }}"
```

get_vpc_information.yml:

```
- name: Get VPC information for deployment
  ec2_vpc_net_info:
    filters:
      "tag:Name": "{{ vpc_name }}" register:
  vpc_info

- name: Set VPC id to a variable set_fact:
    vpc_id: "{{ vpc_info.vpcs | map(attribute='vpc_id') | join }}"

- name: Get public subnet information for deployment
  ec2_vpc_subnet_info:
    filters:
      vpc-id: "{{ vpc_id }}"
      "tag:Name": "{{ vpc_name }}_public_sn" register:
  public_subnet_info

- name: Set public subnet id to a variable set_fact:
    public_subnet_id: "{{ public_subnet_info.subnets |
map(attribute='subnet_id') | join }}"

- name: Get private subnet information for deployment ec2_vpc_subnet_info:
    filters:
      vpc-id: "{{ vpc_id }}"
      "tag:Name": "{{ vpc_name }}_private_sn"
```

```
    Зареєструватися: private_subnet_info

- name: встановити ідентифікатор приватної
  підмережі на змінну set_fact:
    private_subnet_id: "{{ private_subnet_info.subnets |
map(attribute='subnet_id') | Приєднуйтесь }}"

- name: Отримання відомостей про групу безпеки
  моніторингу ec2_group_info:
    Фільтри:
      "tag:Name": "{{ vpc_name }}_monitoring_sg" register:
  monitoring_sg_info

- name: установіть для інформації про групу безпеки
  моніторингу змінні set_fact:
    monitoring_sg_id: "{{ monitoring_sg_info.security_groups |
map(attribute='group_id') | Приєднуйтесь }}"
    monitoring_sg_owner: "{{ monitoring_sg_info.security_groups |
map(attribute='owner_id') | Приєднуйтесь }}"
    monitoring_sg_name: "{{ monitoring_sg_info.security_groups |
map(attribute='group_name') | Приєднуйтесь }}"

- name: Отримати інформацію про групу безпеки
  веб-сервера ec2_group_info:
    Фільтри:
      "tag:Name": "{{ vpc_name }}_webserver_sg" register:
  webserver_sg_info

- name: установіть для інформації про групу безпеки
  веб-сервера значення змінних set_fact:
    webserver_sg_id: "{{ webserver_sg_info.security_groups |
map(attribute='group_id') | Приєднуйтесь }}"
    webserver_sg_owner: "{{ webserver_sg_info.security_groups | map(attribute='owner_id')
| Приєднуйтесь }}"
    webserver_sg_name: "{{ webserver_sg_info.security_groups |
map(attribute='group_name') | Приєднуйтесь }}"

- name: Отримати інформацію про групу безпеки
  бази даних ec2_group_info:
```

Фільтри:

```
    set_fact:
      database_sg_id: "{{ database_sg_info.security_groups |
map(attribute='group_id') | Приєднуйтесь }}"

- name: Отримати інформацію про групу
  безпеки бастіону ec2_group_info:
    Фільтри:
      "tag:Name": "{{ vpc_name }}_bastion_sg" register:
  bastion_sg_info

- name: Встановіть для інформації про групу безпеки
  bastion змінні set_fact:
    bastion_sg_id: "{{ bastion_sg_info.security_groups |
map(attribute='group_id') | Приєднуйтесь }}"
    bastion_sg_owner: "{{ bastion_sg_info.security_groups |
map(attribute='owner_id') | Приєднуйтесь }}"
    bastion_sg_name: "{{ bastion_sg_info.security_groups |
map(attribute='group_name') | Приєднуйтесь }}"

- name: Отримати інформацію про
  екземпляр бастіону
  ec2_instance_info:
    Фільтри:
      "tag:Name": "{{ bastion_host_name }}"
      instance-state-name: запуск
  Реєстрація: bastion_instance_info

- name: Встановіть IP-адресу хоста bastion на
  змінну set_fact:
    bastion_ip: "{{ bastion_instance_info.instances[0].public_ip_address
}}"

- name: Отримати інформацію про екземпляр
  моніторингу ec2_instance_info:
    Фільтри:
      "tag:Name": "{{ monitoring_host_name }}"
      instance-state-name: запуск
  Реєстрація: monitoring_instance_info
```

```
- name: встановити IP-адреси вузлів
```

```
- name: Get first web server instance info ec2_instance_info:
    filters:
      "tag:Name":          "{{          webserver_host_name[0]          }}"
      instance-state-name: running
  register: webserver1_instance_info

- name: Get second web server instance info ec2_instance_info:
    filters:
      "tag:Name":          "{{          webserver_host_name[1]          }}"
      instance-state-name: running
  register: webserver2_instance_info

- name: Set web server private ip's to fact set_fact:
    web1_private_ip: "{{ webserver1_instance_info.instances[0].private_ip_address }}"
    web2_private_ip: "{{ webserver2_instance_info.instances[0].private_ip_address }}"
```

Додаток 2 – Роль розгортання та керування Zabbix

monitoring.yml:

```
---
- hosts: localhost connection: local
  gather_facts: no environment:
    AWS_ACCESS_KEY_ID:          "{{          aws_access_key_id          }}"
    AWS_SECRET_ACCESS_KEY:      "{{      aws_secret_access_key      }}"
    AWS_REGION: "{{ AWS_REGION }}"
  vars_files:
    - keys tasks:
    - name: Get VPC information include_role:
        name: vpc vars:
        fetch_vpc_info: True
      when: hostvars['localhost']['create_initial_vpc'] is not defined

- hosts: monitoring become: true
  vars:
    ansible_ssh_common_args: "-o ProxyCommand='ssh -o StrictHostKeyChecking=no -i {{
private_key_location }} -W %h:%p -q ubuntu@{{ hostvars['localhost']['bastion_ip'] }}'"
  vars_files:
    - keys tasks:
    - name: Patch monitoring host include_role:
        name: patching
      when: hostvars['localhost']['create_initial_vpc'] is defined
    - name: Install common packages include_role:
        name: common
      when: hostvars['localhost']['create_initial_vpc'] is defined
    - name: Configure monitoring include_role:
        name: zabbix
      vars:
```

```
        configure_monitoring_host: True
      when: hostvars['localhost']['create_initial_vpc'] is defined

- hosts: monitoring become: true
  vars_files:
    - keys tasks:
    - name: Configure monitoring host include_role:
        name: zabbix vars:
        configure_monitoring_host: True
      when: hostvars['localhost']['create_initial_vpc'] is not
defined
```

tasks/main.yml:

```
---
- name: Deploy monitoring instance during VPC deploy include_tasks:
  create_instance.yml
  when: create_initial_vpc is defined and create_initial_vpc == True

- name: Configure monitoring host include_tasks:
  configure_monitoring.yml
  when: configure_monitoring_host is defined and
configure_monitoring_host == True

- name: Configure agent include_tasks:
  configure_agent.yml
  when: zabbix_agent is defined and zabbix_agent == True
```

tasks/create_instance.yml

```
---
- name: Create monitoring host include_role:
    name: provisioning vars:
    instance_name: "{{ item }}"
    instance_role: "monitoring" subnet_id: "{{
    public_subnet_id }}"
```

```
    security_group: "{{ monitoring_sg_id }}" with_items:
    - "{{ monitoring_host_name }}"

- name: Встановити ідентифікатор екземпляра
  моніторингу на змінні факту:
    fact_string: "{{ item }}_instance_info" set_fact:
    "{{ item }}_host_id": "{{ hostvars['localhost'][fact_string].instance_ids |
Приєднуйтесь }}"
    "{{ item }}_host_ip": "{{ hostvars['localhost'][fact_string].instances[0].private_ip_address
}}"
  with_items:
    - "{{ monitoring_host_name }}"

- name: Призначте еластичну IP-адресу для
  моніторингу змінних хоста:
    fact_string: "{{ item }}_host_id" ec2_eip:
    device_id: "{{ hostvars['localhost'][fact_string] }}" in_vpc: true
    Держава: теперішній
  реєстр: elastic_ip with_items:
    - "{{ monitoring_host_name }}"

- name: встановити IP-адресу вузла моніторингу на
  змінну vars:
    ip_string: "{{ item }}_host_ip" set_fact:
    monitoring_private_ip: "{{ hostvars['localhost'][ip_string] }}" "{{ item }}_ip": "{{
    elastic_ip.results |
map(attribute='public_ip') | join }}" with_items:
    - "{{ monitoring_host_name }}"

- name: Додати приватну IP-адресу до групи
  in-memory add_host:
    hostname: "{{ item }}"
    ansible_host: "{{ monitoring_private_ip }}" групи:
      -   with_items
  моніторингу:
```

```
    - "{{ monitoring_host_name }}"
```

tasks/configure_zabbix.yml

```
---
- name: Add GPG key and repository include_tasks:
  configure_repo.yml

- name: Install Zabbix server packages apt:
    name: ['zabbix-server-mysql', 'zabbix-frontend-php', 'zabbix- nginx-conf']
    state:    present    update_cache:    yes
    install_recommends: yes

- name: Install Ansible module dependencies apt:
    name: ['python3-psycopg2', 'python3-mysqldb', 'mariadb-client'] state: present
    update_cache: yes

- name: Remove anonymous users mysql_user:
    name: ''
    host: "{{ item }}"
    login_unix_socket: /var/run/mysqld/mysqld.sock state: absent
  with_items:
    - localhost
    - 127.0.0.1
    - ::1

- name: Remove test database if present mysql_db:
    name: test state: absent
    login_unix_socket: /var/run/mysqld/mysqld.sock

- name: Create MySQL database mysql_db:
    name: "{{ zabbix_database_name }}"
    encoding: utf8
```

```
    Порівняння: utf8_bin Стан:
    присутній
    login_unix_socket: /var/ run/mysqld/mysqld.sock реєстр:
  zabbix_db_creation

- name: Створити mysql_user
  користувача MySQL:
    name: "{{ zabbix_database_user }}" host:
    localhost
    password: "{{ zabbix_database_password }}" priv: "{{
    zabbix_database_name }}.*:ALL,GRANT" стан: присутній

- name: Імпорт бази даних
  mysql_db:
    name: "{{ zabbix_database_name }}"
    кодування: utf8
    Порівняння: utf8_bin Стан:
    імпорт
    target: '/usr/share/doc/zabbix-server-mysql/create.sql.gz коли:
  zabbix_db_creation.changed

- name: Оновити змінні пароля
  користувача адміністратора:
    query: "UPDATE zabbix.users SET passwd=md5('{{
zabbix_admin_password }}') де alias='Admin'"
  команда: mysql -u root zabbix --execute "{{ query }}"; Коли:
  zabbix_db_creation.змінено

- name: Налаштувати шаблон
  zabbix-server:
    джерело: zabbix_server.conf.j2
    dest: /etc/zabbix/zabbix_server.conf Власник:
    Zabbix
    Група: Zabbix
    Режим: 0640

- name: Налаштування шаблону PHP-файлу
  Zabbix:
```

```
    повідомити:
      - Перезапустіть сервер

  - name: Завантажити сертифікати
    include_role:
      Назва: Le vars:
      upload_certificate: Правда
    коли: issue_le_certificate == True

  - name: Налаштування шаблону
    nginx:
      джерело: nginx.conf.j2
      dest: /etc/zabbix/nginx.conf власник:
      root
      Група: Кореневий
      режим: 0640
    повідомити:
      - Перезапустіть nginx

  - name: Вилучити файл файлу nginx за
    замовчуванням:
      шлях: /etc/nginx/sites-enabled/ default стан:
      відсутній


  - name: Налаштування шаблону
    PHP-FPM:
      Джерело: php-fpm.conf.j2
      dest: /etc/zabbix/php-fpm.conf власник:
      root
      Група: Кореневий
      режим: 0640
    повідомити:
      - Перезапустіть php-fpm

  - name: Увімкнути
```

```
- name: Set fact set_fact:
    dns_ip_value: "{{ monitoring_host_name }}_ip"

- name: Manage DNS entry include_tasks:
  manage_dns_entry.yml vars:
    dns_name: "{{ zabbix_webui_url }}"
    dns_ip: "{% if create_initial_vpc is defined and domain_name != "
%}{{hostvars['localhost'][dns_ip_value]}}{% else %}{{
hostvars['localhost']['monitoring_public_ip'] }}{% endif %}"
  when: load_balancer_setup is not defined or load_balancer_setup
!= True and manage_dns is defined and manage_dns == True
```

tasks/configure_repo.yml

```
---
- name: Set Zabbix repository URL to a fact set_fact:
    zabbix_repo: "http://repo.zabbix.com/zabbix/{{ zabbix_version
}}/{{ ansible_distribution.lower() }} {{
ansible_distribution_release }} main"

- name: Install Zabbix GPG key apt_key:
    id: "{{ zabbix_gpg_key_id }}"
    url: http://repo.zabbix.com/zabbix-official-repo.key

- name: Install Zabbix repository apt_repository:
    repo: "{{ item }} {{ zabbix_repo }}" state: present
                with_items:
            - deb-src
    - deb
```

tasks/configure_monitoring.yml

```
---
# Configure SSH options
- name: Configure SSH include_role:
    name: ssh
```

```
# Configure users
- name: Configure users include_role:
    name: users

- name: Install and configure Zabbix include_tasks:
  configure_zabbix.yml
```

tasks/configure_agent.yml

```
---
- name: Add GPG key and repository include_tasks:
  configure_repo.yml

- name: Install Zabbix agent apt:
    name:  zabbix-agent  update_cache:
    yes state: latest

- name: Install Zabbix API PIP module pip:
    name: zabbix-api

- name: Configure Zabbix agent template:
    src: zabbix_agentd.conf.j2
    dest: /etc/zabbix/zabbix_agentd.conf owner: root
    group: root mode: '0644'
  notify:
    - Restart agent

- name: Upload PSK file template:
    src: zabbix_agentd.psk.j2
    dest: /etc/zabbix/zabbix_agentd.psk owner: root
    group: zabbix mode: '0640'
  notify:
    - Restart agent
```

tasks/manage_dns_entry.yml

```
---
- name: Create DNS entry for "{{ dns_name }}" route53:
      zone: "{{ domain_name }}"
      record: "{{ dns_name }}" type: A
      value: "{{ dns_ip }}" ttl: 300
      state: "{{ dns_state | default('present') }}" wait: yes
      overwrite:  yes  delegate_to:  localhost
  become: no
```

handlers/main.yml

```
- name: Restart agent
  service: name=zabbix-agent state=restarted

- name: Restart server
  service: name=zabbix-server state=restarted

- name: Restart nginx
  service: name=nginx state=restarted

- name: Restart php-fpm
  service: name=php7.4-fpm state=restarted
```

Додаток 3 – Роль створення та керування користувачами

tasks/main.yml

```
---
# Create groups
- include: groups.yml
# Create users on systems
- include: users.yml
```

tasks/groups.yml

```
---
- name: Create groups group:
      name: "{{ item.name }}"
      state: "{{ item.state | default('present') }}" with_items:
      - "{{ user_groups }}"
```

tasks/users.yml

```
---
# https://docs.ansible.com/ansible/latest/modules/user_module.html
- name: Create user user:
      name: "{{ item.name }}"
      state: "{{ item.state | default('present') }}"
      group: "{% if item.admin_user is defined and item.admin_user == True %}sudo{% else
%}developers{% endif %}"
      home: "{{ item.home | default('/home/'+item.name) }}" shell: "/bin/bash"
      password: "{{ item.password }}"
      comment: "{{ item.comment | default('') }}" append: yes
   with_items:
      - "{{ users }}"
   when: item.state != 'absent'

- name: Add SSH key to user
   authorized_key:
      user: "{{ item.name }}"
      key: "{{ item.ssh_key }}"
      state: "{{ item.state | default('present') }}"
```

```
    with_items:
      - "{{ users }}"
    when: item.state != 'absent'

- name: Delete user user:
    name: "{{ item.name }}" state: absent
    with_items:
      - "{{ users }}"
    when: item.state == 'absent'
      - "{{ users }}"
    when: item.state != 'absent'

- name: Delete user user:
    name: "{{ item.name }}" state: absent
    with_items:
      - "{{ users }}"
    when: item.state == 'absent'
```

Додаток 4 – Конфігурація та роль керування SSH

tasks/main.yml

```
---
- name: Replace default SSH config template:
    src: sshd_config.j2
    dest: "/etc/ssh/sshd_config" owner: "root"
    group: "root" mode: "0644"
  notify:
    - Test SSHD config
    - Restart SSHD
```

handlers/main.yml

```
---
- name: Test SSHD config command:
  sshd -t

- name: Restart SSHD
  service: name="ssh" state="restarted"
```

Додаток 5 – Роль ініціалізації інстансу

tasks/main.yml

```
---
- name: Create {{ instance_name }} instance ec2_instance:
    name: "{{ instance_name }}" key_name: "{{
    key_name }}" vpc_subnet_id: "{{ subnet_id }}"
    security_group: "{{ security_group }}" instance_type: "{{
    instance_type }}" image_id: "{{ image_id }}"
    state: running wait: yes
    wait_timeout: 60 tags:
      Role: "{{ instance_role }}" Environment: "{{
      project_env }}"
  register: instance_info

- name: Set instance information to a fact set_fact:
    "{{ instance_name }}_instance_info": "{{ instance_info }}"
```

Додаток 6 – Роль виправлення системи

tasks/main.yml

```
--- #
https://encoretechnologies.github.io/blog/2018/06/ansiblepatchingau tomation/
- name: Wait for any possibly running unattended upgrade to finish raw: systemd-run
  --property="After=apt-daily.service apt-daily-
upgrade.service" --wait /bin/true

- name: Update packages... apt:
    upgrade: dist update_cache: yes

-       name: Reboot if kernel was updated and reboot is requested by the system
  shell: 'sleep 5 && /sbin/shutdown -r now' args:
    removes: /var/run/reboot-required async: 1
  poll: 0 ignore_errors: true

- name: Waiting for system to come back from reboot... wait_for_connection:
    connect_timeout: 20
    sleep: 5
    delay: 5
    timeout: 180
```

Додаток 7 – Конфігурація та роль керування Nginx

webservers.yml

```
---
- hosts: localhost connection: local
  gather_facts: no environment:
    AWS_ACCESS_KEY_ID: "{{ aws_access_key_id }}"
    AWS_SECRET_ACCESS_KEY: "{{ aws_secret_access_key }}"
    AWS_REGION: "{{ AWS_REGION }}"
  vars_files:
    - keys tasks:
    - name: Get VPC information include_role:
        name: vpc vars:
        fetch_vpc_info: True
      when: hostvars['localhost']['create_initial_vpc'] is not defined

- hosts: webserver become: true
  vars_files:
    - keys vars:
    ansible_ssh_common_args: "-o ProxyCommand='ssh -o StrictHostKeyChecking=no -i {{
private_key_location }} -W %h:%p -q ubuntu@{{ hostvars['localhost']['bastion_ip'] }}'"
  tasks:
    - name: Patch webserver hosts include_role:
        name: patching
      when: hostvars['localhost']['create_initial_vpc'] is defined
    - name: Install common packages include_role:
        name: common
      when: hostvars['localhost']['create_initial_vpc'] is defined
    - name: Configure webservers include_role:
        name: nginx
      vars:
```

```
        configure_nginx_host: True
      when: hostvars['localhost']['create_initial_vpc'] is defined

- hosts: webserver become: true
  environment:
    AWS_ACCESS_KEY_ID: "{{ aws_access_key_id }}"
    AWS_SECRET_ACCESS_KEY: "{{ aws_secret_access_key }}"
    AWS_REGION: "{{ AWS_REGION }}"
  vars_files:
    - keys tasks:
    - name: Configure webservers include_role:
        name: nginx vars:
        configure_nginx_host: True
      when: hostvars['localhost']['create_initial_vpc'] is not defined
```

tasks/main.yml

```
---
- name: Deploy nginx instances during VPC deploy include_tasks:
  create_instance.yml
  when: create_initial_vpc is defined and create_initial_vpc == True

- name: Configure nginx host include_tasks:
  configure_nginx.yml
  when: configure_nginx_host is defined and configure_nginx_host == True
```

tasks/create_instance.yml

```
---
- name: Set subnet to a fact set_fact:
                subnet_group: "{% if load_balancer_setup is not defined or
            load_balancer_setup != True %}public{% elif load_balancer_setup is
```

```
визначено і load_balancer_setup == True %}private{% else %}public{% endif %}"
- name: Створіть хости веб-сервера
  include_role:
    Назва: Змінні ініціалізації:
    instance_name:     "{{     item     }}"
    instance_role:  "веб-сервер"
    subnet_id:  "{% if subnet_group == 'public' %}{{ public_subnet_id }}{% else %}{{
private_subnet_id }}{% endif %}"
    security_group: "{{ webserver_sg_id }}" with_items:
    - "{{ webserver_host_name }}"

- name: Встановити ідентифікатори екземплярів
  веб-сервера на змінні факту:
    fact_string: "{{ item }}_instance_info" set_fact:
    "{{ item }}_host_id": "{{ hostvars['localhost'][fact_string].instance_ids |
Приєднуйтесь }}"
    "{{ item }}_host_ip": "{{ hostvars['localhost'][fact_string].instances[0].private_ip_address
}}"
  with_items:
    - "{{ webserver_host_name }}"

- name: Призначте еластичну IP-адресу змінним
  хоста веб-сервера:
    fact_string: "{{ item }}_host_id" ec2_eip:
    device_id: "{{ hostvars['localhost'][fact_string] }}" in_vpc: true
    Держава:       теперішній
  реєстр: elastic_ip with_items:
    -     "{{ webserver_host_name }}"
  коли: subnet_group == 'public'

- name: Встановити публічну ip-адресу веб-сервера як факт,
  якщо присутні змінні:
    fact_string: "{{ item }}_instance_info" set_fact:
```

```
    "{{ item }}_public_ip": "{{ hostvars['localhost'][fact_string].instances[0].public_ip_address
}}"
  with_items:
    -    "{{    webserver_host_name    }}"    when:
  subnet_group == 'public'

- name: Get first web server instance info ec2_instance_info:
    filters:
      "tag:Name":         "{{         webserver_host_name[0]         }}"
      instance-state-name: running
  register: webserver1_instance_info

- name: Get second web server instance info ec2_instance_info:
    filters:
      "tag:Name":         "{{         webserver_host_name[1]         }}"
      instance-state-name: running
  register: webserver2_instance_info

- name: Set web server private ip's to fact set_fact:
    web1_private_ip: "{{ webserver1_instance_info.instances[0].private_ip_address }}"
    web2_private_ip: "{{ webserver2_instance_info.instances[0].private_ip_address }}"

- name: Add private ip to in-memory group vars:
    ip_string: "{{ item }}_host_ip" add_host:
    hostname: "{{ item }}"
    ansible_host: "{{ hostvars['localhost'][ip_string] }}" groups:
      - webserver with_items:
    - "{{ webserver_host_name }}"
```

tasks/configure_nginx.yml

```
---
```

```
#
https://docs.ansible.com/ansible/latest/modules/zabbix_host_module. html
# Налаштування SSH
- name: Налаштувати SSH
  include_role:
    Ім'я: SSH

# Налаштування користувачів
- name: Налаштуйте
  користувачів include_role:
    Ім'я: Користувачі

# Налаштування агента Zabbix
- name: Налаштування агента Zabbix
  include_role:
    Назва: Zabbix vars:
    zabbix_agent: Правда

- name: Додати хост до Zabbix
  zabbix_host:
    server_url: "https://{{ zabbix_webui_url }}" login_user:
    Адміністратор
    login_password: "{{ zabbix_admin_password }}" host_name:
    "{{ inventory_hostname }}" статус: увімкнено
    Стан: теперішній час
    tls_psk_identity: "{{ zabbix_tls_identity_name }}" tls_connect: 2
    tls_psk: "{{ zabbix_tls_identity_password }}" tls_accept: 2
    host_groups:
      - Сервери Linux
    link_templates:
      - Шаблон ОС Linux від агента Zabbix
      - Шаблон App Nginx від інтерфейсів
    агента Zabbix:
      - Тип: 1
        Основна: 1
        Використання: 1
```

```
# - ім'я: Встановлення nginx пакетів
- name: Встановіть пакет nginx
  apt:
    Ім'я: Nginx Держава:
    остання update_cache:
    Так

- name: Встановіть пакети PHP
  apt:
    name: ['php-fpm', 'php-mysql'] Стан:
    останній
    update_cache: Так

- name: Налаштування шаблону
  nginx:
    джерело: nginx.conf.j2
    dest: /etc/nginx/nginx.conf власник:
    root
    Група: Кореневий
    режим: '0644'
  повідомити:
    - Перезапустіть nginx

- name: Налаштувати сайт за замовчуванням для шаблону
  перевірок zabbix:
    src: default.conf.j2
    dest: /etc/nginx/sites-enabled/default власник: root
    Група: Кореневий
    режим: '0644'
  повідомити:
    - Перезавантажити nginx

  коли: vhosts визначено, а vhosts != ''
                                    tasks/manage_dns_entry.yml
---
- name: Create DNS entry for "{{ dns_name }}"
```

```
    route53:
      zone: "{{ domain_name }}"
      record: "{{ dns_name }}" type: A
      value: "{{ dns_ip }}" ttl: 300
      state: "{{ dns_state }}" wait: yes
      overwrite: yes delegate_to: localhost
    become: no
```

tasks/manage_elb.yml

```
---
# Let's Encrypt certificates does not work with ELB # - name: Import
certificate into aws ACM
#     vars:
#       full_cert_file: "{{ certificate_directory }}/{{ domain_name
}}-fullchain.crt"
#       cert_file: "{{ certificate_directory }}/{{ domain_name
}}.crt"
#       key_file: "{{ certificate_directory }}/{{ domain_name }}.key" #shell: >
#       aws acm import-certificate
#       --certificate "file://{{ cert_file }}" #         --private-key
"file://{{ key_file }}"
#       --certificate-chain "file://{{ full_cert_file }}" #    --region                  "{{
AWS_REGION }}"
#     register: certificate_id #       delegate_to:
localhost #    become: no

- name: Set webserver instance-id's to variables vars:
      app1_string: "{{ webserver_host_name[0] }}_host_id" app2_string: "{{
      webserver_host_name[1] }}_host_id"
    set_fact:
      app1_id: "{{ app1_string }}" app2_id: "{{
      app2_string }}"
    delegate_to: localhost become: no
```

```
- name: Create ELB ec2_elb_lb:
    name: 'ELB' state: present
    region: "{{ AWS_REGION }}" instance_ids: ['app1_id',
    'app2_id'] security_group_ids: "{{
hostvars['localhost']['webserver_sg_id'] }}"
    subnets: "{{ hostvars['localhost']['public_subnet_id'] }}" listeners:
      - protocol: http load_balancer_port: 80
        instance_port: 80 proxy_protocol: True
      - protocol: https load_balancer_port: 443
        instance_protocol: http instance_port: 80
        ssl_certificate_id: "{{ ssl_certificate_location }}" delegate_to: localhost
  become: no
```

tasks/manage_vhosts.yml

```
---
- name: Create vhost directory file:
    path: "/srv/vhosts/{{ item.name }}"
    state: "{{ item.state | default('directory') }}" owner: root
    group: developers mode: '0775'
  with_items: "{{ vhosts }}"

- name: Create html directory file:
    path: "/srv/vhosts/{{ item.name }}/html"
    state: "{{ item.state | default('directory') }}" owner: root
    group: developers mode: '0775'
  with_items: "{{ vhosts }}"
```

```
- name: Створити файл каталогу
  журналу:
    шлях: "/srv/vhosts/{{ item.name }}/log"
    state: "{{ item.state | default('directory') }}" власник: root
    Група: Розробники
    Режим: '0640'
  with_items: "{{ vhosts }}"

- name: Створити файл каталогу
  tmp:
    шлях: "/srv/vhosts/{{ item.name }}/tmp"
    state: "{{ item.state | default('directory') }}" власник: root
    Група: Кореневий
    режим: '0777'
  with_items: "{{ vhosts }}"

- name: Завантажити сертифікати
  include_role:
    Назва: Le vars:
    upload_certificate: Правда
  коли: issue_le_certificate == True

- name: Створити шаблон конфігурації
  nginx:
    src: vhost.conf.j2
    dest: "/etc/nginx/sites-available/{{ item.name }}.conf" власник: root
    Група: Кореневий
    режим: '0644'
  with_items: "{{ vhosts }}"

- name: Увімкнути файл конфігурації
  nginx:
    src: "/etc/nginx/sites-available/{{ item.name }}.conf" dest:
    "/etc/nginx/sites-enabled/{{ item.name }}.conf" state : "{{
```

```
    vars:
      ip_string: "{{ item }}_public_ip" set_fact:
      dns_ip_value: "{{ hostvars['localhost'][ip_string] }}"
    when: load_balancer_setup is not defined or load_balancer_setup
!= True with_items:
      - "{{ webserver_host_name }}"

-    name:    Manage    DNS    entry    include_tasks:
    manage_dns_entry.yml vars:
      dns_name:  "{{  item.name  }}"  dns_ip:  "{{
      dns_ip_value }}"
      dns_state: "{{ item.state | default('present') }}" with_items: "{{ vhosts }}"
    when: load_balancer_setup is not defined or load_balancer_setup
!= True and manage_dns is defined and manage_dns == True
```

handlers/main.yml

```
---
- name: Test nginx config shell: nginx -t
  notify:
    - Reload nginx

- name: Reload nginx
  service: name=nginx state=reloaded

- name: Restart nginx
  service: name=nginx state=restarted
```

Додаток 8 – Роль конфігурації та керування MySQL

dbservers.yml

```
---
- hosts: localhost connection: local
  gather_facts: no environment:
    AWS_ACCESS_KEY_ID: "{{ aws_access_key_id }}"
    AWS_SECRET_ACCESS_KEY: "{{ aws_secret_access_key }}"
    AWS_REGION: "{{ AWS_REGION }}"
  vars_files:
    - keys tasks:
    - name: Get VPC information include_role:
        name: vpc vars:
        fetch_vpc_info: True
      when: hostvars['localhost']['create_initial_vpc'] is not defined

- hosts: database become: true
  vars_files:
    - keys vars:
    ansible_ssh_common_args: "-o ProxyCommand='ssh -o StrictHostKeyChecking=no -i {{
private_key_location }} -W %h:%p -q ubuntu@{{ hostvars['localhost']['bastion_ip'] }}'"
  tasks:
    - name: Patch database hosts include_role:
        name: patching
      when: hostvars['localhost']['create_initial_vpc'] is defined
    - name: Install common packages include_role:
        name: common
      when: hostvars['localhost']['create_initial_vpc'] is defined
    - name: Configure database servers include_role:
        name: mysql
      vars:
```

```
        configure_database_host: True
      when: hostvars['localhost']['create_initial_vpc'] is defined

- hosts: database become: true
  environment:
    AWS_ACCESS_KEY_ID: "{{ aws_access_key_id }}"
    AWS_SECRET_ACCESS_KEY: "{{ aws_secret_access_key }}"
    AWS_REGION: "{{ AWS_REGION }}"
  vars_files:
    - keys tasks:
    - name: Make sure common packages are installed include_role:
        name: common
    - name: Configure database include_role:
        name: mysql vars:
        configure_database_host: True
      when: hostvars['localhost']['create_initial_vpc'] is not defined
```

tasks/main.yml

```
---
- name: Deploy database instances during VPC deploy include_tasks:
  create_instance.yml
  when: create_initial_vpc is defined and create_initial_vpc == True

- name: Configure database host include_tasks:
  configure_database.yml
  when: configure_database_host is defined and
configure_database_host == True
```

tasks/create_instance.yml

```
---
- name: Create database hosts include_role:
    name: provisioning
```

```
    vars:
      instance_name: "{{ item }}" instance_role: "database"
      subnet_id: "{{ private_subnet_id }}"
      security_group: "{{ database_sg_id }}" with_items:
      - "{{ database_host_name }}"

- name: Set database instance ids to a fact vars:
      fact_string: "{{ item }}_instance_info" set_fact:
      "{{ item }}_host_id": "{{ hostvars['localhost'][fact_string].instance_ids | join }}"
      "{{ item }}_host_ip": "{{ hostvars['localhost'][fact_string].instances[0].private_ip_address
}}"
    with_items:
      - "{{ database_host_name }}"

- name: Add private ip to in-memory group vars:
      ip_string: "{{ item }}_host_ip" add_host:
      hostname: "{{ item }}"
      ansible_host: "{{ hostvars['localhost'][ip_string] }}" groups:
        - database with_items:
      - "{{ database_host_name }}"
```

tasks/configure_database.yml

```
--- #
https://docs.ansible.com/ansible/latest/modules/zabbix_host_module. html
# Configure SSH
- name: Configure SSH include_role:
      name: ssh

# Configure users
- name: Configure users
```

```
    include_role:      ім'я:
      користувачі

# Налаштування агента Zabbix
- name: Налаштування агента Zabbix
  include_role:
    Назва: Zabbix vars:
    zabbix_agent: Правда

- name: Додати хост до Zabbix
  zabbix_host:
    server_url:  "https://{{  zabbix_webui_url  }}"  login_user:
    Адміністратор
    login_password: "{{ zabbix_admin_password }}" host_name:
    "{{ inventory_hostname  }}" статус: увімкнено
    Стан: теперішній час
    tls_psk_identity: "{{ zabbix_tls_identity_name }}" tls_connect: 2
    tls_psk: "{{ zabbix_tls_identity_password }}" tls_accept: 2
    host_groups:
      -    Сервери Linux
    link_templates:
      - Шаблон ОС Linux від агента Zabbix
      -    Шаблон БД MySQL від інтерфейсів
    агента Zabbix:
     -         Тип: 1
           Основна: 1
        Використання: 1
        IP:          "{{      ansible_default_ipv4.address      }}"
    validate_certs: ні

- name: Встановіть пакет MySQL
  apt:
    Ім'я:  MySQL-Server  Стан:
    останній       update_cache:
    Так
```

```
- name: Configure MySQL server template:
    src: my.cnf.j2
    dest: /etc/mysql/mysql.conf.d/mysqld.cnf owner: root
    group: root mode: '0644'
  notify:
    - Restart mysql

- name: Start and enable MySQL database service service:
    name:   mysql   state:   started
    enabled: yes

- name: Remove anonymous users mysql_user:
    name: ''
    host: "{{ item }}"
    login_unix_socket: /var/run/mysqld/mysqld.sock state: absent
  with_items:
    - localhost
    - 127.0.0.1
    - ::1

- name: Remove test database if present mysql_db:
    name: test state: absent
    login_unix_socket: /var/run/mysqld/mysqld.sock

- name: Manage databases include_tasks:
  manage_databases.yml
  when: databases is defined and databases != ''
```

tasks/manage_database.yml

```
---
# https://topic.alibabacloud.com/a/managing-mysql-replication-with-ansible_1_41_30026734.html
- name: Create database
```

```
    mysql_db:
      name: "{{ item.name }}"
      state: "{{ item.state | default('present') }}" login_unix_socket:
      /var/run/mysqld/mysqld.sock
    with_items:
      - "{{ бази даних }}"

- name: Створити користувача бази даних з першого
  веб-сервера mysql_user:
      name: "{{ item.user | default(item.name) }}" password: "{{
      item.password }}"
      host: "{{ hostvars['localhost']['web1_private_ip'] }}" priv: "{{
      item.name }}.*:ALL,GRANT"
      state: "{{ item.state | default('present') }}" login_unix_socket:
      /var/run/mysqld/mysqld.sock
    with_items:
      - "{{ бази даних }}"

- name: Створити користувача бази даних з другого
  веб-сервера mysql_user:
      name: "{{ item.user | default(item.name) }}" password: "{{
      item.password }}"
      host: "{{ hostvars['localhost']['web2_private_ip'] }}" priv: "{{
      item.name }}.*:ALL,GRANT"
      state: "{{ item.state | default('present') }}" login_unix_socket:
      /var/run/mysqld/mysqld.sock
    with_items:
      - "{{ бази даних }}"

- name: Створити реплікацію
  користувача mysql_user:
      name: "{{ item.user_name }}" host:
      "%"
      password: "{{ item.password }}" priv:
      "*.*:REPLICATION SLAVE"
      state: "{{ item.state | default('present') }}" login_unix_socket:
      /var/run/mysqld/mysqld.sock
    with_items:
      - "{{ replication_setup }}"
    коли: mysql_replication_role == 'master'
```

```
    query: "ALTER USER {{ item.user_name }} ІДЕНТИФІКОВАНИЙ ЗА
mysql_native_password '{{ item.password }}'"
  команда: mysql -u корінь --execute "{{ query }}"; Коли:
  create_replication_user.змінив with_items:
    - "{{ replication_setup }}"

- name: Налаштування підлеглого
  вузла mysql_replication:
    Режим: getslave
    login_unix_socket: /var/run/mysqld/mysqld.sock
  ignore_errors: true
  Реєстрація: slave
  коли: mysql_replication_role == 'slave'

- name: встановити IP-адресу та ім'я хоста для змінної
  vars:
    master_hostname: "{{ database_host_name[0] }}" set_fact:
    master_ip: "{{ hostvars[master_hostname]['ansible_default_ipv4']['address'] }}"
    master_hostname: "{{ master_hostname }}"

- name: Переконайтеся, що записи hosts на місці
  vars:
    mysql_replication_master_ip: lineinfile:
    dest: /etc/hosts
    line: "{{ master_ip }} {{ master_hostname }}" коли:
  mysql_replication_role == 'slave'

- name: Отримати статус реплікації майстрів
  mysql_replication:
    Режим роботи: GetMaster
    login_unix_socket: /var/run/mysqld/mysqld.sock delegate_to:
  "{{ master_hostname }}"
  Коли: mysql_replication_role == 'slave' регістр:
  replication_status

- name: Зміна ведучого в slave
```

```
    master_password: "{{ replication_setup[0].password }}" master_log_file: "{{
    replication_status.File }}" master_log_pos: "{{ replication_status.Position }}"
    login_unix_socket: /var/run/mysqld/mysqld.sock
  when: mysql_replication_role == 'slave'

- name: Start slave in slave and start replication mysql_replication:
    mode: startslave
    login_unix_socket: /var/run/mysqld/mysqld.sock when:
  mysql_replication_role == 'slave'
```

handlers/main.yml

```
---
- name: Restart mysql
  service: name=mysql state=restarted
```

Додаток 9 – Автоматизована роль розгортання сертифіката Let's Encrypt

tasks/main.yml

```
---
- name: Issue wildcard during VPC deploy include_tasks:
  issue_certificate.yml
  when: create_initial_vpc is defined and create_initial_vpc == True and issue_le_certificate ==
True and manage_dns is defined and manage_dns == True

- name: Upload certificate to other instances include_tasks:
  upload_certificate.yml
  when: issue_le_certificate is defined and issue_le_certificate == True and upload_certificate is
defined and upload_certificate == True and manage_dns is defined and manage_dns == True
```

tasks/issue_certificate.yml

```
--- #
https://docs.ansible.com/ansible/latest/modules/acme_account_module
.html #
https://docs.ansible.com/ansible/latest/modules/acme_certificate_mo
dule.html#acme-certificate-module
#
https://docs.ansible.com/ansible/latest/modules/route53_module.html

- name: Set domain name to a fact set_fact:
    le_cert_domain: "{{ domain_name }}" wildcard: "*.{{
    domain_name }}"

- name: Create private key directory file:
    path: '{{ certificate_directory }}' state: directory
    owner: ansible group: ansible
```

```
- name: Створіть приватний ключ для облікового
  запису ACME openssl_privatekey:
    path: "{{ certificate_directory }}/account.key" розмір: 4096

- name: Створити закритий ключ для облікового запису
  сертифіката openssl_privatekey:
    path: "{{ certificate_directory }}/{{ le_cert_domain }}.key" розмір: 4096

- name: Згенерувати КСВ для {{ le_cert_domain }}
  openssl_csr:
    path: "{{ certificate_directory }}/{{ le_cert_domain }}.csr" privatekey_path: "{{
    certificate_directory }}/{{ le_cert_domain
}}.ключ"
    common_name: "{{ wildcard }}"

- name: Переконайтеся, що обліковий запис
  ACME існує acme_account:
    account_key_src: "{{ certificate_directory }}//account.key" acme_version: 2
    acme_directory: «https://acme-
v02.api.letsencrypt.org/directory»
    Стан: теперішній
    terms_agreed: Так
    Контакти:
      - "mailto:{{ acme_account_email }}"

- name: Створити виклик для {{ le_cert_domain }}
  acme_certificate:
    modify_account: No acme_directory:
    «https://acme-
v02.api.letsencrypt.org/directory' acme_version:
    2
    account_key_src: "{{ certificate_directory }}/account.key" src: "{{
    certificate_directory }}/{{ le_cert_domain }}.csr" cert: "{{ certificate_directory
    }}/{{ le_cert_domain }}.crt" Виклик : DNS-01
    remaining_days: 60
  Реєстрація: dns_challenge

- name: Створіть запис DNS
```

```
      record: '{{ dns_challenge.challenge_data[wildcard]["dns- 01"]["record"] }}'
      type:  TXT  ttl:  60  state:
      present wait: yes
      value: '"{{ dns_challenge.challenge_data[wildcard]["dns- 01"]["resource_value"] }}"'
      overwrite: yes
  when: dns_challenge.changed

- name: Wait for DNS record to expire pause:
    seconds: 120
  when: dns_challenge.changed

-      name: Let the challenge be validated and retrieve the cert and intermediate
certificate
  acme_certificate: modify_account: no
    account_key_src: "{{ certificate_directory }}/account.key" src: "{{ certificate_directory
    }}/{{ le_cert_domain }}.csr" cert: "{{ certificate_directory }}/{{ le_cert_domain }}.crt"
    fullchain: "{{ certificate_directory }}/{{ le_cert_domain }}-
fullchain.crt"
    chain: "{{ certificate_directory }}/{{ le_cert_domain }}- intermediate.crt"
    challenge:  dns-01  acme_version:  2
    acme_directory: 'https://acme-
v02.api.letsencrypt.org/directory' remaining_days: 60
    data: "{{ dns_challenge }}"
  when: dns_challenge is changed
```

tasks/upload_certificate.yml

```
---
- name: Create directory file:
    path: "/etc/ssl/{{ domain_name }}" state: directory
    owner: root
    group: root
```

```
- name: Upload {{ domain_name }} certificate copy:
    src: "{{ certificate_directory }}/{{ domain_name }}.crt" dest: "/etc/ssl/{{
    domain_name }}/{{ domain_name }}.crt" owner: root
    group: root

- name: Upload {{ domain_name }} private key copy:
    src: "{{ certificate_directory }}/{{ domain_name }}.key" dest: "/etc/ssl/{{
    domain_name }}/{{ domain_name }}.key" owner: root
    group: root mode: '0600'
```

Додаток 10 – Розгортання резервного копіювання бази даних Duplicity

backups.yml

```
---
- hosts: database become: true
  environment:
    AWS_ACCESS_KEY_ID: "{{ aws_access_key_id }}"
    AWS_SECRET_ACCESS_KEY: "{{ aws_secret_access_key }}"
    AWS_REGION: "{{ AWS_REGION }}"
  vars_files:
    - keys tasks:
    - name: Configure backups include_role:
        name: duplicity vars:
        configure_duplicity_backups: True
```

tasks/main.yml

```
---
- name: Configure duplicity backup on database nodes include_tasks:
  configure_duplicity.yml
  when: configure_duplicity_backups is defined and
configure_duplicity_backups == True
```

tasks/configure_duplicity.yml

```
---
- name: Install Duplicity apt:
    name: duplicity state: latest
    update_cache: yes

- name: Install boto module pip:
    name: boto

- name: Upload gpg key generation script
```

```
    Шаблон:
      src: gen_gpg.j2
      dest:  /root/duplicity_gpg  власник:
      root
      група:    кореневий
      режим:  '0700'

- назва: Створити ключ gpg
  Команда: 'gpg --batch --generate-key /root/duplicity_gpg' аргументи:
      chdir:  /корінь  ставати:
    так become_user: корінь

- name: Створити шаблон файлу подвійних
  облікових даних:
      src:       duplicity_credentials.j2       dest:
      /root/.duplicity власник: root
      Група:    кореневий
      режим:  '0600'
    ставати:                  так
    become_user: корінь

- name: Створити файл каталогу
  резервної копії:
      шлях:   /srv/backup/mysql  Стан:
      Власник каталогу  : Root
      група:    кореневий
      режим:  '0700'
```

```
hour: "23"
minute: "0" user: root
job: "/srv/backup/backup_script.sh {{ s3_bucket_name }}" cron_file:
ansible_auto_backups
```

Додаток 11 – Загальна роль встановлення пакунків

tasks/main.yml

```
---
- name: Install packages apt:
      name: ['python-apt', 'python3-apt', 'python3-pip', 'vim-nox', 'net-tools', 'mc']
      state: latest update_cache: yes
```

Додаток 12 – Розгортання та налаштування хоста Bastion

bastion.yml

```
---
- hosts: localhost connection: local
  gather_facts: no environment:
    AWS_ACCESS_KEY_ID: "{{ aws_access_key_id }}"
    AWS_SECRET_ACCESS_KEY: "{{ aws_secret_access_key }}"
    AWS_REGION: "{{ AWS_REGION }}"
  vars_files:
    - keys tasks:
    - name: Get VPC information include_role:
        name: vpc vars:
        fetch_vpc_info: True
      when: hostvars['localhost']['create_initial_vpc'] is not defined

- hosts: bastion gather_facts: no
  become: true tasks:
    - name: Patch bastion host include_role:
        name: patching
      when: hostvars['localhost']['create_initial_vpc'] is defined
    - name: Install common packages include_role:
        name: common
      when: hostvars['localhost']['create_initial_vpc'] is defined
    - name: Configure bastion host include_role:
        name: bastion vars:
        configure_bastion_host: True
      when: hostvars['localhost']['create_initial_vpc'] is defined
```

tasks/main.yml

```
---
- name: Create bastion instance during VPC deployment include_tasks:
  create_instance.yml
  when: create_initial_vpc is defined and create_initial_vpc == True

- name: Configure bastion host include_tasks:
  configure_bastion.yml
  when: configure_bastion_host is defined and
configure_bastion_host == True
```

tasks/create_instance.yml

```
---
- name: Create bastion host include_role:
    name: provisioning vars:
    instance_name: "{{ item }}" instance_role: "bastion"
    subnet_id: "{{ public_subnet_id }}"
    security_group: "{{ bastion_sg_id }}" with_items:
    - "{{ bastion_host_name }}"

- name: Set bastion instance id to a fact vars:
    fact_string: "{{ item }}_instance_info" set_fact:
    "{{ item }}_host_id": "{{ hostvars['localhost'][fact_string].instance_ids | join }}"
  with_items:
    - "{{ bastion_host_name }}"

- name: Assign elastic IP to bastion host ec2_eip:
    device_id: "{{ item }}" in_vpc: true
    state: present register: elastic_ip
  with_items:
    - "{{ bastion_host_id }}"
```

```
- name: Set bastion host IP to a variable set_fact:
    "{{ item }}_ip": "{{ elastic_ip.results | map(attribute='public_ip') | join }}"
  with_items:
    - "{{ bastion_host_name }}"

- name: Add public_ip to in-memory group vars:
    ip_string: "{{ item }}_ip" add_host:
    hostname: "{{ item }}"
    ansible_host: "{{ hostvars['localhost'][ip_string] }}" groups:
      - bastion with_items:
    - "{{ bastion_host_name }}"

- name: Create dynamic inventory file template:
    src: aws_ec2.j2 dest: ./aws_ec2.yml
```

tasks/configure_bastion.yml

```
---
# Configure SSH
- name: Configure SSH include_role:
    name: ssh

# Configure users and groups
- name: Configure users include_role:
    name: users
```