Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Інфокомунікаційй*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти *другий (магістерський)*

Дослідження підходу "Інфраструктура як код"(IaC) в хмарних технологіях
(тема)

Виконав:
здобувач 2 курсу, групи *ІМІм-23-1*
Береза Є.Д.

Спеціальності *172 Електронні комунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *Освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)
Керівник *доцент каф. ІМІ Іваненко С.А.*
(посада, прізвище, ініціали)

Допускається до захисту

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

2025 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____ *Береза Є.Д.*
(підпис) (прізвище та ініціали)

Керівник _____ *Іваненко С.А.*
(підпис) (прізвище та ініціали)

Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

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

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

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

ЗАТВЕРДЖУЮ:
Зав. кафедри ІМІ_____
(підпис)
“____” _____2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачеві *Березі Євгену Дмитровичу*
(прізвище, ім’я, по батькові)

1. Тема роботи *Дослідження підходу “Інфраструктура як код”(IaC) в хмарних технологіях*

затверджені наказом університету від 2025 року № *1148 Ст*

2. Термін подання студентом роботи до екзаменаційної комісії *13 січня 2025 р.*

3. Вихідні дані до роботи

Дослідити технологію “Інфраструктура як код”(IaC) в хмарних технологіях, розгорнути п’ять сценаріїв можливих варіантів інфраструктур за допомогою інструментів Terraform та AWS CloudFormation, зробити висновки.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ
1. Інформаційна інфраструктура
2. Інструменти IaC
3. Сценарії інфраструктури
4. Аналіз інфраструктури
Висновки

6. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) *Слайди у форматі Power Point (назва, мета і задачі роботи, технологія IaC, розвиток IaC, хмарні обчислення, сценарії інфраструктури, аналіз технології)*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів атестаційної роботи	Строк виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ*	*30.11.24*	*виконано*
2	*Підбір літератури за темою роботи*	*01.12-02.12.24*	*виконано*
3	*Виконання розділу 1*	*04.12-05.12.24*	*виконано*
4	*Виконання розділу 2*	*06.12-07.12.24*	*виконано*
5	*Виконання розділу 3*	*13.12-16.12.24*	*виконано*
6	*Виконання розділу 4*	*18.12-21.12.24*	*виконано*
7	*Оформлення пояснювальної записки*	*02.01-03.01.25*	*виконано*
8	*Оформлення презентаційного матеріалу,*	*12.01-12.01.25*	*виконано*
	підготовка до захисту у ЕК		

Дата видачі завдання *30.11.2024 р.*

Здобувач _____ *Береза Є.Д.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Іваненко С.А.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 99 с., 7 рис., 7 табл., 9 джерел, 7 додатки.

Об'єкт дослідження – технологія “Інфраструктура як код”, хмарна інфраструктура.

Мета роботи – дослідити технологію “Інфраструктура як код” на п’яти сценаріях хмарної інфраструктури.

В роботі розглянуто основні характеристики та функціональні можливості підходу “Інфраструктура як код” двома інструментами Terraform та AWS CloudFormation, розраховано, розгорнуто та проаналізовано п’ять сценаріїв хмарної інфраструктури за допомогою даних інструментів.

Результати роботи можуть бути застосовані при розрахунку та проектуванні хмарної мережі в робочому та навчальному процесах.

ІНФРАСТРУКТУРА ЯК КОД, ХМАРНІ ОБЧИСЛЕННЯ, БАЗИ ДАНИХ, МЕРЕЖНА ОПТИМІЗАЦІЯ

THE ABSTRACT

Explanatory note: 99 p., 7 fig., 7 tabl., 9 sources, 7 app.

The object of the study are the “Infrastructure as Code” technology and cloud infrastructure.

The purpose of the work is to investigate the “Infrastructure as Code” technology in five cloud infrastructure scenarios.

The work considers the main characteristics and functionality of the “Infrastructure as Code” approach using two tools, Terraform and AWS CloudFormation, and calculates, deploys, and analyzes five cloud infrastructure scenarios using these tools.

The results of the work can be applied in the calculation and design of a cloud infrastructure in work and educational processes.

INFRASTRUCTURE AS CODE, CLOUD COMPUTING, DATABASES, NETWORK OPTIMIZATION

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ

AWS – Amazon Web Services;

IaaS – Infrastructure as a Service;

GCP – Google Cloud Platform;

HCL – Hashicorp Configuration Language;

VPC – Virtual Private Cloud;

EC2 – Elastic Compute Cloud;

SG – Security Group;

IaC – Infrastructure as Code.

ВСТУП

Сучасний підхід до управління хмарною інфраструктурою за допомогою концепції "Інфраструктура як Код" (Infrastructure as Code, IaC) стає ключовим елементом DevOps-практик та автоматизації в IT. IaC дозволяє описувати, створювати та керувати інфраструктурою за допомогою текстових файлів конфігураційного коду, що забезпечує повторюваність, масштабованість та контроль над хмарними ресурсами.

Дослідження в даній роботі зосереджене на порівнянні та аналізі використання IaC для автоматизації створення інфраструктури в хмарному середовищі AWS за допомогою двох підходів: власного інструменту AWS CloudFormation та універсального інструменту Terraform. Робота включає практичну реалізацію п'яти сценаріїв розгортання хмарної інфраструктури, серед яких налаштування мережі, баз даних, обчислювальних ресурсів, а також засобів масштабування та балансування навантаження.

Метою дослідження є не лише демонстрація можливостей IaC для автоматизації, але й оцінка ефективності, зручності використання, гнучкості та продуктивності інструментів у різних сценаріях. Результати можуть бути використані як основа для впровадження практик IaC у реальних проєктах, що сприятиме підвищенню ефективності процесів управління інфраструктурою.

1 ОГЛЯД ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ

1.1 Серверна інфраструктура

На початку розвитку інформаційних технологій серверна інфраструктура базувалася виключно на фізичних (або "залізних") серверах. Цей підхід передбачав створення, налаштування та підтримку серверів у виділених серверних кімнатах або дата-центрах. Організації інвестували значні ресурси у придбання обладнання, його встановлення, налаштування мереж, резервування електроживлення та забезпечення безпеки. Процес розгортання, експлуатації та обслуговування фізичних серверів був складним та тривалим.

Фізичний сервер складався з апаратного забезпечення, на якому працювала операційна система, а також програмного забезпечення для обробки запитів, управління даними та виконання бізнес-завдань. В ручну обирались, збирались, обслуговувались та оновлювались основні компоненти: процесор (CPU), Оперативна пам'ять (RAM), файлова **с**истема зберігання даних (HDD/SSD), мережеве обладнання, системи охолодження і так далі.

Серверні кімнати були спеціально обладнаними приміщеннями, у яких розміщувалися фізичні сервери. Їх проєктування та обслуговування вимагали значних зусиль та витрат.

Традиційна серверна інфраструктура мала низку викликів:

1. Високі капітальні витрати: організації витрачали значні кошти на придбання обладнання, побудову серверних кімнат та підтримку їх роботи.

2. Тривалий час розгортання: налаштування серверів займало багато часу, що затримувало запуск нових проєктів.

3. Обмежена масштабованість: додавання нових серверів вимагало додаткових фінансових і часових витрат.

4. Низька ефективність: сервери часто працювали з недостатнім завантаженням, що призводило до неефективного використання ресурсів.

5. Складність управління: адміністратори витрачали багато часу на

моніторинг, оновлення та резервування даних.

Для забезпечення безперебійної роботи критичних систем використовувалося резервування компонентів:

1. Резервування серверів: встановлення додаткових серверів для обробки запитів у разі відмови основних.
2. RAID-масиви: застосовувалися для забезпечення надійного зберігання даних у випадку виходу з ладу одного з накопичувачів.
3. Резервні копії: регулярне створення резервних копій даних для їх відновлення у разі аварії.

Традиційна серверна інфраструктура була ефективною для свого часу, однак її недоліки зумовили перехід до нових підходів, зокрема віртуалізації та хмарних технологій. Саме ці обмеження стали каталізатором розвитку концепції хмарних технологій, яка усуває більшість викликів традиційної інфраструктури.

1.2 Хмарна інфраструктура

Хмарна інфраструктура – це набір апаратних і програмних ресурсів, які утворюють обсяг устаткування у віддаленому дата центрі або ж “хмарі”, постачальники котрих надають ці сервіси в оренду своїм користувачам. Хмарні постачальники обслуговують глобальні центри обробки даних із тисячами компонентів IT-інфраструктури, як-от сервери, фізичні пристрої зберігання та мережеве обладнання [5]. Вони налаштовують фізичні пристрої за допомогою всіх типів конфігурацій операційної системи. Вони також встановлюють інші типи програмного забезпечення, необхідні для роботи програми. Ваша організація може взяти хмарну інфраструктуру в оренду за принципом оплати за використання, що допоможе вам значно заощадити на придбанні та обслуговуванні окремих компонентів.

Хмарна інфраструктура стала революційним кроком у розвитку інформаційних технологій, дозволяючи організаціям масштабувати свої

обчислювальні потужності, знижувати витрати та прискорювати розгортання нових сервісів. На відміну від традиційної серверної інфраструктури, хмарна модель базується на використанні ресурсів, які надаються віддаленими центрами обробки даних через інтернет.

Хмарна інфраструктура визначається кількома ключовими особливостями:

1. Віддалене використання ресурсів: обчислювальні потужності, сховища даних та мережеві ресурси знаходяться у віддалених дата-центрах і доступні через інтернет.

2. Масштабованість: хмара дозволяє швидко збільшувати чи зменшувати обсяги використовуваних ресурсів залежно від поточних потреб.

3. Оплата за використання: організації платять лише за фактично використані ресурси, що дозволяє оптимізувати витрати.

4. Автоматизація: багато рутинних процесів, таких як резервне копіювання, моніторинг і оновлення систем, автоматизовані.

5. Глобальна доступність: хмарні провайдери пропонують інфраструктуру в різних регіонах світу, що забезпечує низьку затримку доступу до сервісів для користувачів з різних країн.

Хмарна інфраструктура надається у вигляді кількох моделей послуг:

1. IaaS (Infrastructure as a Service): забезпечує доступ до віртуальних серверів, сховищ даних, мережевих ресурсів. Користувачі керують операційними системами, додатками та даними, а постачальник відповідає за підтримку апаратної інфраструктури.

a. Приклад: Amazon EC2, Google Compute Engine, Microsoft Azure Virtual Machines.

2. PaaS (Platform as a Service): надає платформу для розробки, тестування та розгортання додатків. Адміністрування інфраструктури повністю бере на себе постачальник. Приклад: AWS Elastic Beanstalk, Google App Engine, Microsoft Azure App Service.

3. SaaS (Software as a Service): дозволяє користуватися готовими

програмними рішеннями через інтернет без необхідності їх встановлення чи обслуговування. Приклад: Google Workspace, Microsoft 365, Salesforce.

Залежно від потреб організації та моделі використання, хмарна інфраструктура поділяється на кілька типів:

1. Публічні хмари: надаються великими постачальниками (AWS, Azure, GCP) і доступні для будь-яких організацій. Вони забезпечують гнучкість і масштабованість, але передбачають обмежену варіативність та можливість доналаштування.

2. Приватні хмари: використовуються однією організацією і можуть розгортатися у власному дата-центрі або на інфраструктурі постачальника. Забезпечують максимальний контроль та безпеку.

3. Гібридні хмари: поєднують переваги приватних і публічних хмар, дозволяючи зберігати чутливі дані локально та використовувати публічну хмару для масштабованих обчислень.

4. Багатохмарні рішення: використання кількох хмарних постачальників одночасно для досягнення оптимального результату.

Переваги хмарної інфраструктури:

1. Зменшення витрат: відсутність необхідності закупівлі фізичного обладнання та його обслуговування. Оплата здійснюється лише за використані ресурси.

2. Гнучкість і масштабованість: легке збільшення або зменшення обсягів ресурсів залежно від поточних потреб.

3. Швидкість розгортання: нові сервери або сервіси можуть бути розгорнуті за лічені хвилини, що прискорює запуск проєктів.

4. Висока надійність: хмарні постачальники забезпечують резервування даних, географічне розподілення центрів обробки даних і захист від збоїв.

5. Інновації: постачальники хмарних послуг постійно додають нові сервіси та функціональності, що дозволяє організаціям залишатися конкурентоспроможними.

Недоліки хмарної інфраструктури:

1. Залежність від інтернету: для доступу до хмарних ресурсів потрібне стабільне і швидке інтернет-з'єднання.

2. Обмежений контроль: організації залежать від постачальника хмарних послуг в питаннях обслуговування, безпеки та доступності інфраструктури.

3. Безпека даних: дані, що зберігаються в публічній хмарі, можуть бути вразливими до атак, якщо не дотримуватися належних заходів безпеки.

4. Витрати: хоча хмара зменшує капітальні витрати, операційні витрати можуть зростати при тривалому використанні великих обсягів ресурсів.

Приклади використання хмарної інфраструктури:

1. Розгортання веб-додатків: завдяки гнучкості хмар можна швидко налаштувати веб-сервіси для глобальної аудиторії.

2. Обробка великих даних: хмарні платформи дозволяють обробляти та аналізувати великі обсяги даних, використовуючи сервіси для машинного навчання та аналітики.

3. Тестування і розробка: хмара забезпечує середовище для швидкого тестування нових продуктів і їх інтеграції в бізнес-процеси.

4. Резервування та відновлення: хмарні рішення забезпечують безпечне зберігання резервних копій даних з можливістю їх швидкого відновлення у випадку збоїв.

Розвиток хмарної інфраструктури створив фундамент для концепції Інфраструктури як Код (IaC), яка дозволяє автоматизувати створення, налаштування та управління інфраструктурою. Завдяки IaC процеси, які раніше займали дні чи навіть тижні, тепер виконуються за лічені хвилини, що відкриває нові можливості для бізнесу.

1.3 Порівняння постачальників хмарних послуг AWS, MICROSOFT AZURE TA GCP (Google Cloud Platform)

Особливості хмарного постачальника Amazon Web Services: Створений компанією Amazon у 2006 році, є одним із найстаріших і

найбільших постачальників хмарних послуг у світі. AWS пропонує найбільш повну екосистему хмарних рішень, надаючи понад 200 послуг, включаючи зберігання, обчислювальні потужності, бази даних, аналітику, машинне навчання, Інтернет речей (IoT), DevOps, розробку програмного забезпечення та безпеку особистих та даних користувачів [8]. AWS розроблений з урахуванням високої надійності, і він забезпечує роботу багатьох підприємств, зокрема великих корпорацій та урядових організацій.

Переваги AWS:

1. Широка екосистема послуг: AWS має найбільший набір хмарних сервісів, що дозволяє організаціям обирати рішення відповідно до своїх потреб.

2. Висока доступність: завдяки численним центрам обробки даних (зонам доступності та регіонам) по всьому світу, AWS забезпечує високий рівень доступності та відмовостійкості.

3. Розвинені інструменти DevOps та IaC: AWS пропонує зручні інструменти для автоматизації, як AWS CloudFormation, що спрощує роботу з IaC.

4. Надійна підтримка великої кількості баз даних: AWS пропонує різноманітні сервіси баз даних, як реляційні (RDS), так і NoSQL (DynamoDB, DocumentDB) для різних завдань.

Недоліки AWS:

1. Висока вартість: AWS може бути дорогим, особливо для малих і середніх бізнесів. Зокрема, витрати на передачу даних між різними регіонами можуть бути значними.

2. Складність управління для новачків**:** широкий спектр послуг може створювати певні труднощі для нових користувачів, оскільки налаштування AWS вимагає певних технічних знань.

3. Обмеження на роботу з Microsoft-продуктами: AWS менш інтегрований з продуктами Microsoft, що може бути недоліком для організацій, які використовують Windows Server, Active Directory та інші продукти

Microsoft.

Особливості Microsoft Azure

Microsoft Azure, запущений у 2010 році, є хмарною платформою, створеною з інтеграцією в існуючу екосистему продуктів Microsoft. Azure надає хмарні обчислення, зберігання даних, аналітику, штучний інтелект та безпеку. Він особливо популярний серед організацій, що використовують Microsoft продукти, і пропонує велику кількість рішень для інтеграції з Windows Server, Active Directory та іншими службами Microsoft.

Переваги Azure:

1. Глибока інтеграція з продуктами Microsoft: Azure є оптимальним вибором для організацій, що працюють на базі Microsoft, завдяки інтеграції з такими продуктами, як Windows Server, SQL Server, .NET, Office 365 і Active Directory.

2. Гібридна інфраструктура: Azure підтримує гібридні хмарні рішення, що дозволяє організаціям з легкістю інтегрувати свої локальні системи з хмарними.

3. Широкий вибір продуктів для аналітики та IoT: Azure має потужні інструменти для великих даних та аналітики, такі як Azure Synapse Analytics, а також сервіси IoT, що дозволяють створювати надійні рішення для Інтернету речей.

Недоліки Azure:

1. Складність цінової структури: як і у випадку з AWS, структура ціноутворення Azure може бути складною для розуміння, що ускладнює контроль витрат.

2. Порівняно менша кількість послуг, ніж у AWS: хоча Azure пропонує широкий спектр рішень, він все ж поступається AWS за кількістю доступних сервісів.

3. Складність у масштабуванні для великих обчислювальних навантажень: деякі користувачі відзначають, що Azure може мати обмеження в масштабуванні великих обчислювальних потужностей у порівнянні з AWS.

Особливості GCP

Google Cloud Platform, запущена в 2011 році, відома своїми можливостями для обробки даних, штучного інтелекту, а також рішеннями для великих даних і машинного навчання [9]. Завдяки потужним інструментам, таким як BigQuery і TensorFlow, GCP часто вибирають компанії, які працюють із великими обсягами даних або займаються дослідженнями в галузі машинного навчання та штучного інтелекту.

Переваги GCP:

1. Інноваційні можливості для обробки великих даних: GCP пропонує потужні інструменти для великих даних, як-от BigQuery, що забезпечує високу продуктивність при роботі з великими масивами даних.

2. Сильні рішення для машинного навчання та штучного інтелекту: GCP має розвинені сервіси для штучного інтелекту та машинного навчання, зокрема TensorFlow і AI Platform.

3. Конкурентне ціноутворення: Google Cloud має конкурентні ціни і часто пропонує гнучкіші умови оплати за споживання обчислювальних ресурсів, а також знижки за тривале використання.

Недоліки GCP:

1. Менша присутність на ринку та кількість послуг: GCP поступається AWS і Azure за кількістю послуг та географічною присутністю, що може бути обмеженням для великих міжнародних компаній.

2. Менш насичена екосистема: у порівнянні з AWS та Azure, екосистема GCP є менш розвиненою, і деякі інструменти можуть мати менше можливостей.

3. Менша інтеграція з традиційними корпоративними системами: GCP менш інтегрований з корпоративними рішеннями, такими як Microsoft або SAP, що може створювати певні труднощі для великих підприємств.

Таблиця 1.1 – Порівняння хмарних постачальників та їх послуг

	AWS	Azure	GCP
	Сервіси Найбільший вибір (200+ послуг)	Широкий вибір, але трохи менше ніж AWS	Обмежений вибір порівняно з AWS і Azure
Інтеграція з MS продуктами	Обмежена інтеграція	Повна інтеграція	Обмежена інтеграція
Підтримка великих даних та ШІ	Сильні можливості, але поступається GCP в ШІ	Потужні інструменти, але менше ШІ сервісів, ніж у GCP	Найсильніші можливості для обробки великих даних і ШІ
Ціноутворення	Висока вартість	Складна структура цін	Гнучкіші ціни, знижки за тривале використання
Простота використання	Може бути складним для новачків	Оптимальний для користувачів Microsoft	Найбільш дружній інтерфейс
Присутність на ринку	Найвища	Висока	Середня

Таблиця наведена вище спирається на основні сучасні потреби замовників від хмарних постачальників. Окремим пунктом знедавна виділяється послуги ШІ (штучний інтелект). Це не є частиною дипломної роботи, але варто відзначити у теперішніх реаліях.

2 ІНСТРУМЕНТИ IaC

2.1 Технологія IaC

Інфраструктура як код (IaC) — це підхід до управління інфраструктурою, в якому інфраструктурні компоненти, такі як сервери, мережі, бази даних, лямбда функції, зберігання даних, інші сервіси, надані хмарним постачальником, описуються і налаштовуються декларативно через програмний код. Це дозволяє автоматизувати процеси розгортання, налаштування та масштабування інфраструктури, забезпечуючи повторюваність, відтворюваність та контроль за змінами.

До широкого використання хмарних технологій більшість організацій розгортали та управляли інфраструктурою вручну, використовуючи фізичні сервери та локальні дата-центри. Процеси налаштування серверів, підключення до мереж, розгортання програмного забезпечення та оновлення конфігурацій були тривалими і вимагали великої кількості людських ресурсів. Кожен новий сервер або компонент інфраструктури потрібно було налаштовувати вручну, що підвищувало ризик помилок і ускладнювало масштабування.

З початком ери хмарних обчислень хмарні постачальники Amazon Web Services, Microsoft Azure та Google Cloud, запропонували можливість розгортати інфраструктуру через їх веб-інтерфейси або командний рядок. Але навіть тоді процеси створення та налаштування інфраструктури залишалися трудомісткими і вразливими до помилок. Для того, щоб об'єкти хмарної інфраструктури, такі як віртуальні машини (VM), мережі та сховища, були доступні на запит, адміністратори все одно повинні були вручну вводити команди або натискати кнопки в панелі управління, що часто призводило до неузгодженості в конфігураціях середовищ.

З розвитком концепції DevOps та автоматизації у кінці 2000-х років виникла потреба у швидкому і надійному створенні інфраструктури, яка могла

б легко адаптуватися до постійно змінюваних вимог бізнесу. Це призвело до створення концепції «Інфраструктура як код» (IaC). Вона передбачала декларативний опис інфраструктури за допомогою простих текстових файлів, що зберігали інструкції для автоматичного створення та налаштування серверів, мереж і інших інфраструктурних компонентів.

Одним з перших інструментів, що реалізували концепцію IaC, став CloudFormation від Amazon Web Services (AWS), який дозволяв автоматично створювати ресурси на платформі AWS за допомогою шаблонів JSON або YAML. Кількома роками пізніше, були представлені інші потужні інструменти IaC, такі як Terraform від HashiCorp та Azure Resource Manager від Microsoft Azure. Ці інструменти дозволили організаціям розгортати інфраструктуру не тільки в межах однієї хмарної платформи, але й створювати багатохмарні середовища та інтегрувати їх з іншими інструментами DevOps.

З появою IaC процес створення та управління інфраструктурою став значно більш автоматизованим, передбачуваним та контрольованим. Інфраструктура більше не потребувала ручного налаштування або втручання адміністратора. Замість цього, опис конфігурації інфраструктури виконується у вигляді коду, що може бути збережений у системах контролю версій, протестований і повторно використаний.

Основні переваги використання IaC включають:

1. Автоматизація: всі етапи налаштування, оновлення і видалення інфраструктурних компонентів можна автоматизувати.
2. Масштабованість: легко додавати нові сервери, мережеві компоненти або сховища, без необхідності втручання людини.
3. Контроль версій та відновлення: кожна зміна інфраструктури фіксується в системах контролю версій, що дає можливість відстежувати зміни, відновлювати попередні стани або масштабувати інфраструктуру.
4. Безпека та відповідність вимогам: код можна перевіряти на відповідність вимогам безпеки ще до його застосування, що знижує ризик помилок.

Інструменти IaC можуть бути поділені на два типи:

1. Декларативні інструменти (наприклад, Terraform, CloudFormation):
    1.1. У цих інструментах описується бажаний стан інфраструктури, а сам інструмент автоматично виводить поточний стан в цей бажаний.
    1.2. Зазвичай використовуються для більш складних та довготривалих проектів.
2. Імперативні інструменти (наприклад, Ansible, Chef, Puppet):
    1.1. Ці інструменти описують інструкції для послідовних дій, які повинні бути виконані для досягнення бажаного результату.
    1.2. Зазвичай вони використовуються для конфігурації серверів і програмного забезпечення на них, а також для оновлення налаштувань.

Технологія IaC продовжує розвиватися, і з кожним роком з’являються нові функціональності та поліпшення у вже існуючих інструментах. Вони дозволяють не тільки спрощувати процеси розгортання інфраструктури, а й інтегрувати інфраструктуру з іншими інструментами для тестування, моніторингу та безпеки.

2.2 Terraform

Terraform — це один із найпопулярніших інструментів для автоматизації управління інфраструктурою, що реалізує підхід Інфраструктура як код (IaC). Він дозволяє описувати інфраструктуру у вигляді коду, використовуючи декларативний синтаксис мовою HCL (HashiCorp Configuration Language), і автоматично створювати, змінювати та видаляти ресурси у хмарних постачальниках, а також на локальних середовищах чи в гібридних інфраструктурах. Terraform був розроблений компанією HashiCorp і підтримує великий набір хмарних постачальників та інших технологій [1].

1. Декларативний підхід**:** у Terraform опис інфраструктури відбувається

декларативно, що означає вказівку бажаного кінцевого стану інфраструктури, а не кроків, як цього досягти. Наприклад, замість того, щоб вручну створювати інстанси серверів або налаштовувати мережу, ви описуєте, які ресурси і з якими параметрами повинні бути присутніми в системі. Terraform самостійно визначає, як досягти цього стану, автоматично виконуючи необхідні кроки.

2. Підтримка мультихмарних середовищ: однією з основних переваг Terraform є його здатність працювати з кількома хмарними постачальниками одночасно. За допомогою єдиного конфігураційного файлу можна створювати ресурси в різних хмарних середовищах - AWS, Azure, Google Cloud, а також інтегрувати інші інструменти та платформи (наприклад, Kubernetes, VMware, та інші).

3. Модулі та повторне використання коду: Terraform підтримує використання модулів — це набір ресурсів і їх налаштувань, які можна використовувати багато разів у різних частинах конфігурації або навіть у різних проектах. Це дозволяє значно зменшити обсяг повторюваного коду, що знижує ймовірність помилок і полегшує підтримку інфраструктури.

4. Планування змін (Terraform Plan): однією з ключових особливостей Terraform є можливість попереднього перегляду змін перед їх застосуванням через команду terraform plan. Це дозволяє користувачеві побачити, які ресурси будуть створені, змінені чи видалені, і дає можливість уникнути неочікуваних змін у системі. Це забезпечує додатковий рівень безпеки перед застосуванням змін.

5. Робота з конфігураціями у системах контролю версій: конфігурації Terraform — це текстові файли, зазвичай з розширенням .tf. Це дозволяє зберігати та відслідковувати зміни в інфраструктурі за допомогою систем контролю версій, таких як Git. Усі зміни до інфраструктури можна перевіряти, фіксувати та надавати відповідний доступ до коду.

Архітектура Terraform:

1. Конфігураційні файли**:** Terraform використовує конфігураційні файли для опису інфраструктури, зазвичай в синтаксисі HCL (HashiCorp Configuration

Language) або JSON. Конфігураційний файл містить ресурси, які повинні бути створені або змінені, а також їх параметри.

2. План виконання: **к**оли користувач застосовує Terraform до конфігурації, він спочатку генерує план виконання через команду terraform plan, який показує, які зміни будуть внесені до інфраструктури. Це дає змогу побачити можливі конфлікти або помилки до того, як зміни будуть застосовані.

3. Застосування змін: після перегляду плану користувач може застосувати зміни до інфраструктури через команду terraform apply. Terraform автоматично здійснює необхідні дії (створює, оновлює або видаляє ресурси) для того, щоб інфраструктура відповідала описаному стану.

4. Стан інфраструктури: під час виконання Terraform зберігає інформацію про поточний стан інфраструктури в так званому файлі стану (terraform.tfstate). Цей файл містить відомості про поточні ресурси, що належать до проекту, та необхідний для коректної роботи Terraform. Файл стану можна зберігати локально або у віддаленому сховищі, щоб підтримувати командну роботу.

Переваги використання Terraform:

1. Багатохмарність і гнучкість: Terraform підтримує понад 200 постачальників інфраструктури, таких як AWS, Azure, Google Cloud, а також інші платформи та сервіси. Це дозволяє організаціям будувати та управляти гібридними чи мультихмарними інфраструктурами без прив'язки до конкретного постачальника.

2. Відтворюваність та масштабованість: завдяки декларативному підходу інфраструктуру можна автоматично відтворювати у будь-яких середовищах: на тестових, розробницьких чи продуктивних серверах. Це дозволяє швидко масштабувати інфраструктуру в залежності від потреб бізнесу.

3. Зменшення помилок: автоматизація процесів налаштування і масштабування інфраструктури знижує ймовірність помилок, що виникають при ручному налаштуванні, і забезпечує більшу стабільність середовища.

4. Контроль версій та безпека: використання системи контролю версій для збереження конфігурацій Terraform дозволяє відслідковувати зміни в

інфраструктурі, що є важливим для забезпечення безпеки, а також для відновлення попередніх станів при необхідності.

5. Можливість командної розробки інфраструктури: Terraform підтримує роботу в команді через віддалене збереження файлів стану та можливість автоматизованого процесу розгортання через CI/CD. Це дозволяє багатьом користувачам працювати над одним проектом без перешкод.

Недоліки Terraform:

1. Важкість в освоєнні: для новачків, які тільки починають працювати з IaC, Terraform може здатися складним для розуміння через необхідність працювати з великою кількістю конфігурацій та параметрів.

2. Обмежена підтримка для деяких постачальників хмарних послуг: хоча Terraform підтримує багато постачальників, інколи можливості або підтримка певних сервісів для менш популярних платформ може бути обмежена.

3. Масштабованість в дуже великих середовищах: у великих проектах з великою кількістю ресурсів може виникнути потреба в додаткових налаштуваннях для оптимізації продуктивності, таких як використання віддаленого стану або кращої організації модулів.

4. Відсталість із оновленнями: в разі виходу нового сервісу чи оновлення старого у постачальника хмарних послуг, Terraform може не підтримувати нове оновлення, т.я. у компанії-розробника Hashicorp зазвичай займає кілька днів на оновлення.

Загалом, Terraform є потужним і гнучким інструментом для автоматизації створення та управління інфраструктурою. Його популярність зростає завдяки простоті у використанні, підтримці багатохмарних середовищ та широкому спектру можливостей для автоматизації й інтеграції з іншими інструментами DevOps.

2.3 AWS CloudFormation

AWS CloudFormation — це сервіс від Amazon Web Services, який дозволяє

описувати та автоматизувати процес створення та управління ресурсами в AWS [2]. CloudFormation є основним інструментом для реалізації концепції "Інфраструктура як код" (IaC) в екосистемі AWS, дозволяючи користувачам описувати бажану інфраструктуру у вигляді шаблонів (Templates), що потім можуть бути автоматично застосовані для створення, зміни чи видалення ресурсів за аналогією із Terraform, тільки має власні особливості.

Основні можливості та особливості

1. Шаблони для опису інфраструктури: CloudFormation інфраструктура описується через шаблони, що представляють собою текстові файли, написані у форматі JSON або YAML. Ці шаблони містять визначення ресурсів, їх властивостей та залежностей, а також можуть включати умови, параметри та вихідні дані, що забезпечують динамічну настройку та гнучкість конфігурацій.

2. Автоматизація створення та керування ресурсами: CloudFormation автоматично розгортає інфраструктуру, не вимагаючи від користувачів вручну створювати кожен ресурс. Він керує залежностями між ресурсами, створюючи або змінюючи їх у правильному порядку. Наприклад, якщо для запуску сервера EC2 потрібен певний VPC або підмережа, CloudFormation автоматично створить їх у потрібній послідовності.

3. Ідемпотентність: однією з ключових особливостей CloudFormation є ідемпотентність. Це означає, що навіть якщо шаблон буде застосований кілька разів, він не змінить існуючу інфраструктуру, якщо не буде виявлено необхідних змін. Це дозволяє забезпечити консистентність і передбачуваність у процесах оновлення та підтримки інфраструктури.

4. Підтримка ресурсів AWS та сторонніх провайдерів: CloudFormation підтримує всі основні сервіси AWS, такі як EC2, S3, VPC, IAM, Lambda, RDS та багато інших, що дозволяє користувачам створювати повноцінну інфраструктуру без необхідності вручну налаштовувати кожен ресурс. Окрім того, за допомогою розширень (так званих "Custom Resources"), можна використовувати CloudFormation для інтеграції з ресурсами сторонніх постачальників та систем, але AWS стягає немалий додатковий кошт за кожний

виклик шаблона від стороннього постачальника.

5. Статус та управління стеком: CloudFormation працює з концепцією стеків (stacks), які представляють собою набір ресурсів, що створюються або керуються разом як єдине ціле. Стек може бути створений, оновлений або видалений, а його статус і зміни можна відслідковувати через AWS Management Console, AWS CLI чи API. Це дозволяє зручно управляти великою кількістю ресурсів у межах одного середовища.

6. Підтримка параметризації: шаблони CloudFormation можуть бути параметризованими, що дозволяє адаптувати інфраструктуру до різних середовищ (наприклад, розробка, тестування, продуктивне середовище) без необхідності змінювати сам шаблон. Користувач може вказувати значення параметрів під час запуску шаблону, таким чином дозволяючи надавати різні налаштування в залежності від контексту.

7. Інтеграція з іншими сервісами AWS: CloudFormation інтегрується з численними іншими сервісами AWS, що дозволяє здійснювати моніторинг, автоматизацію та управління інфраструктурою без зайвих зусиль. Наприклад, можна автоматично запускати шаблони CloudFormation через AWS Lambda або AWS Step Functions, або інтегрувати їх з процесами для автоматичного розгортання.

Архітектура AWS CloudFormation

1. Шаблони CloudFormation: шаблони CloudFormation описують інфраструктуру за допомогою JSON або YAML форматів. Вони містять визначення ресурсів, їхні атрибути та залежності. Шаблони можуть бути збережені локально або в Amazon S3, щоб бути використаними для розгортання стеків.

2. Стек CloudFormation. Стек — це набір ресурсів, які керуються одним шаблоном. Стек може бути створений, оновлений або видалений, а зміни у ньому здійснюються автоматично. Всі ресурси в межах одного стеку мають спільну конфігурацію та залежності, що дозволяє зручно керувати ними як єдиним цілим.

3. Механізм оновлення: оновлення стеку в CloudFormation здійснюється шляхом зміни шаблону та застосування змін до вже існуючих ресурсів. CloudFormation автоматично розраховує, які ресурси необхідно оновити, а які не потребують змін. У разі виникнення проблем оновлення можна відкотити стек до попереднього стану.

4. Моніторинг та зворотний зв'язок: після застосування шаблону CloudFormation надає повний звіт про стан розгорнутих ресурсів, включаючи інформацію про успіх або невдачу кожної операції. Це дозволяє швидко отримувати інформацію про можливі помилки та вирішувати їх безпосередньо в процесі розгортання.

Переваги використання AWS CloudFormation

1. Повна автоматизація: CloudFormation дозволяє автоматизувати весь процес розгортання інфраструктури в AWS, від створення до зміни та видалення ресурсів, що значно знижує можливість помилок та зменшує людський фактор.

2. Ідемпотентність та передбачуваність: завдяки ідемпотентності шаблонів CloudFormation можна бути впевненим у тому, що повторний запуск шаблону не призведе до непередбачуваних змін, що забезпечує стабільність та передбачуваність інфраструктури.

3. Інтеграція з іншими сервісами AWS: CloudFormation є частиною екосистеми AWS, що забезпечує повну інтеграцію з іншими сервісами AWS, такими як AWS Lambda, AWS Systems Manager, AWS CodePipeline тощо. Це дозволяє автоматизувати весь цикл життя інфраструктури, включаючи її оновлення та моніторинг.

4. Керовані стеки: працюючи з стеком CloudFormation, можна легко відслідковувати стан інфраструктури та здійснювати необхідні операції на рівні всіх ресурсів у межах одного середовища. Це спрощує управління складними інфраструктурами.

5. Гнучкість і масштабованість: завдяки можливості параметризації шаблонів та інтеграції з іншими AWS сервісами, CloudFormation дозволяє

створювати складні й масштабовані інфраструктури, налаштовуючи їх під різні потреби та бізнес-стратегії.

Недоліки AWS CloudFormation:

1. Складність у використанні для новачків: хоча CloudFormation надає великий функціонал для автоматизації інфраструктури, для новачків може бути складним у освоєнні через багатий синтаксис і необхідність глибокого розуміння AWS ресурсів і залежностей.

2. Час виконання: оскільки CloudFormation керує інфраструктурою на рівні всіх ресурсів у стеку, час створення чи оновлення інфраструктури може бути значним для великих проектів, особливо при наявності багатьох ресурсів.

3. Обмеження шаблонів: хоча CloudFormation підтримує більшість сервісів AWS, інколи можливості шаблонів можуть бути обмежені, і для деяких специфічних випадків може знадобитися додаткове налаштування або інтеграція з іншими інструментами.

У підсумку, AWS CloudFormation є потужним інструментом для автоматизації управління інфраструктурою в AWS, який дозволяє користувачам створювати, змінювати та видаляти ресурси в хмарі на основі декларативних шаблонів. Він ідеально підходить для великих проектів, де необхідна стабільність, масштабованість і автоматизація процесів.

3 РОЗГОРТАННЯ СЦЕНАРІЇВ ХМАРНОЇ ІНФРАСТУКТУРИ

В якості об’єктів дослідження інструментів Terraform та AWS CloudFormation будуть використані п’ять реальних сценаріїв хмарної інфраструктури, кожний своєї комплексності та вартості, та підняті в хмарі постачальника AWS.

3.1 Простий веб-застосунок

Перший сценарій хмарної інфраструктури передбачає створення простого сховища у вигляді сервісу простого зберігання файлів S3-бакета з публічним доступом, налаштованого для зберігання веб-сторінки (або іншого медіаконтенту). Також метою такого підходу може бути забезпечення середовища для резервного копіювання, збереження архівів, файлів або інших типів інформації.

3.1.2 Розрахунок та розгортання 1-ого сценарію

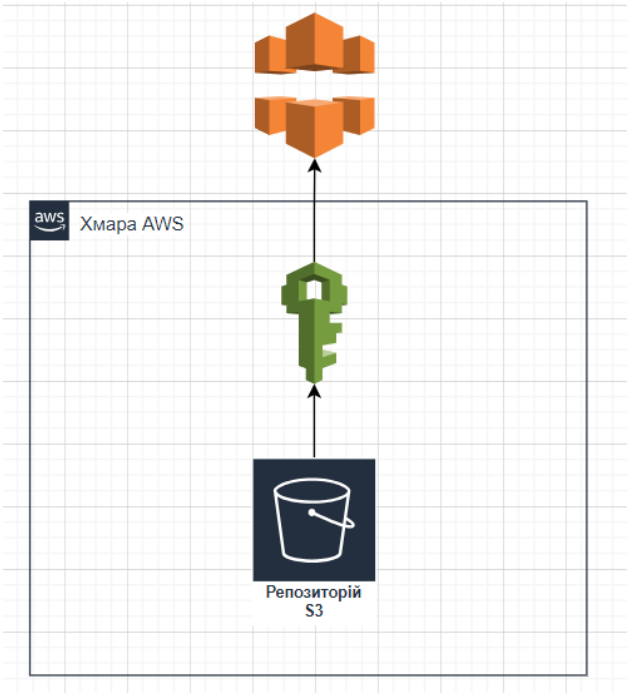

Рисунок 3.1 - Схема інфраструктури 1-ого сценарію

Код, котрим підіймався 1-ий сценарій інфраструктури Див. Додаток А

Лістинг 1 та Лістинг 2.

Розгорнуті ресурси:

1. S3 bucket: сховище, де знаходить безпосередньо сама веб-сторінка із публічним доступом.

2. CloudFront Origin Access Control: використовується для надання CloudFront доступу до приватного S3-бакета.

3. CloudFront Distribution**:** створюється для дистрибуції кешування в регіонах відмінних від регіону, де знаходиться веб-сторінка, та публічного доступу до об'єктів у S3.

Даний сценарій дозволяє розміщати веб-сторінку на сервісі S3, надати їй публічного доступу та за допомогою сервісу CloudFront зробити кешування в інших регіонах для зменшення затримки доступу до контенту. Ідеальне малобюджетне рішення для немасштабних проектів.

3.2 Тестове середовище

Другий сценарій надає можливість розгорнути тестове оточення для розробки та/або проведення тестування різноманітних програмних продуктів, на кшталт мобільного застосунку, веб-блогу чи будь-чого іншого.

3.2.1 Розрахунок та розгортання 2-ого сценарію

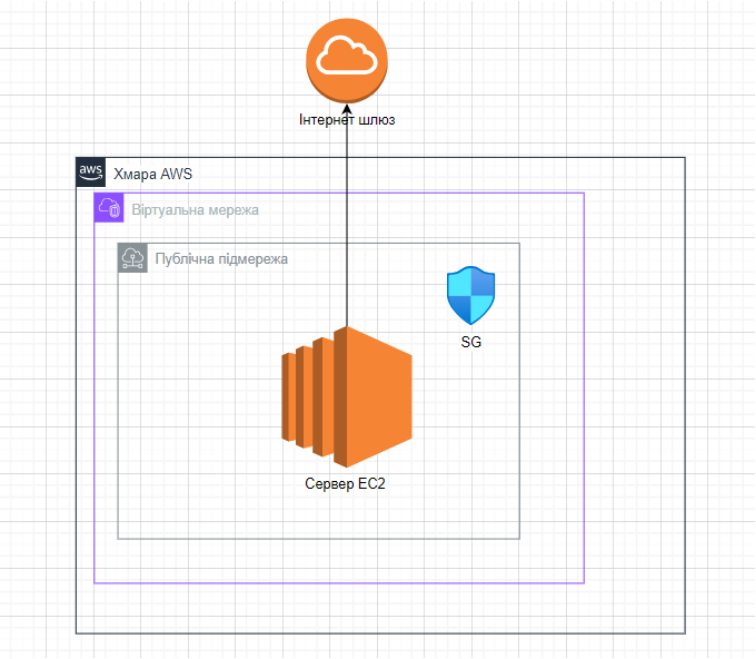

Рисунок 3.2 – Схема інфраструктури 2-ого сценарію

Код для 2-ого сценарію інфраструктури Див. Додаток А Лістинг 3 та Лістинг 4.

Розгорнути ресурси:

1. VPC: створюється основна мережа VPC з адресою 10.0.0.0/16

2. Публічна підмережа з адресою 10.0.1.0/24

3. **І**нтернет-шлюз (InternetGateway) забезпечує доступ до Інтернету

4. Таблиця маршрутизації (PublicRouteTable) з маршрутом 0.0.0.0/0 (що в реальній ситуації варто було б уникнути та гранульовано обмежити доступ тільки до тих адрес, з яких буде доступ у групи розробки/тестування), котра і дає доступу до Інтернету для Інтернет-шлюзу.

5. Групи доступу (Security Groups): SSH (порт 22) для доступу адміністратора, HTTP (порт 80) для веб-доступу.

6. Сервер EC2: сервер створюється з операційною системою Amazon Linux 2 AMI, розташований у публічній підмережі та з публічною IP-адресою.

Такий сценарій дозволяє розгорнути тестовий сервер із веб-доступом по

80 порту та обмеженим ssh-доступом по порту 22 для обслуговування. Спектр застосувань обмежує групу розробки лише їх уявою та потужностями самого сервера.

3.3 Стримінговий сервіс

Третій сценарій підходить для сервісів із високим користувацьким навантаженням, рішенням для котрого є два EC2-сервери, навантаження на котрі контролює балансувальник навантаження ALB (Application Load Balancer).

3.3.2 Розрахунок та розгортання 3-ого сценарію

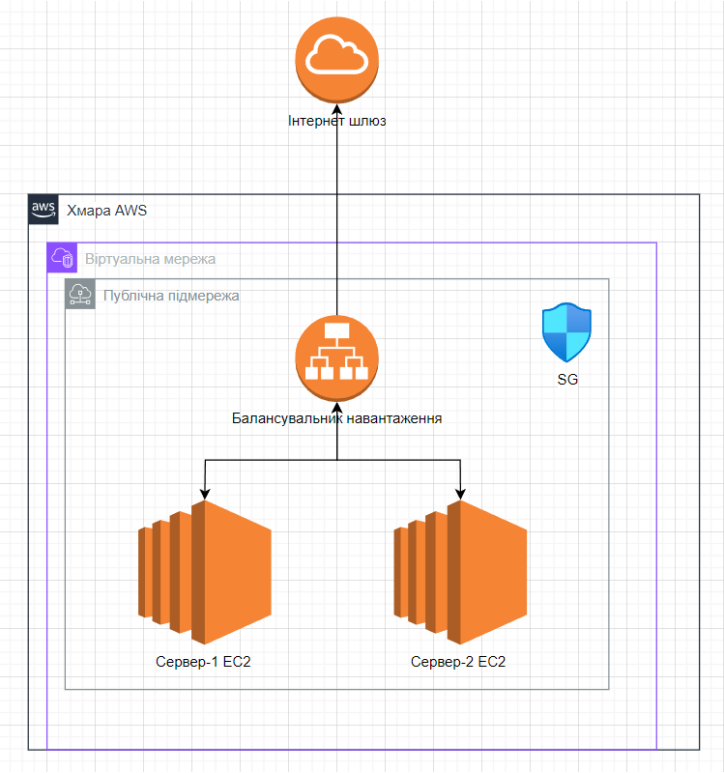

Рисунок 3 – Схема інфраструктури 3-ого сценарію

Код для 3-ого сценарію інфраструктури Див. Додаток А Лістинг 5 та Лістинг 6.

Розгорнути ресурси:

1. VPC: створюється основна мережа VPC з адресою 10.0.0.0/16

2. Публічна підмережа з адресою 10.0.1.0/24

3. Інтернет-шлюз (InternetGateway) забезпечує доступ до Інтернету

4. Таблиця маршрутизації (PublicRouteTable) з маршрутом 0.0.0.0/0, котра і дає доступу до Інтернету для Інтернет-шлюзу.

5. Групи доступу (Security Groups): SSH (порт 22) для доступу адміністратора, HTTP (порт 80) для веб-доступу для обох серверів. Лише для балансувальника навантаження HTTP (порт 80) для веб-доступу.

6. Балансувальник навантаження (ALB): для розподілу трафіку між двома EC2-серверами.

7. Два сервери EC2: на кожний із серверів балансувальник розподіляє користувацький трафік.

Сценарій підходить для відео-хостингу, стримінг сервісів та інших застосунків, котрі очікують велике навантаження запитів. Дану інфраструктуру можна легко горизонтально масштабувати, додавши в підмережу третій EC2-сервер та додати його в цільову групу балансувальника, таким чином збільшити пропускну здатність всього сервісу при тому не зупиняючи його роботу.

3.4 Корпоративний CRM-застосунок

Сценарій номер чотири задовольнить інфраструктурну потребу для CRM-систем (Customer Relationship Management), або ж застосунків для роботи із особистими даними користувачів. Перевагою даного варіанту є те, що доки сервер із відкритим доступом знаходиться в публічній підмережі, база даних, до якої звертається сервер, знаходить в приватній підмережі.

3.4.2 Розрахунок та розгортання 4-ого сценарію

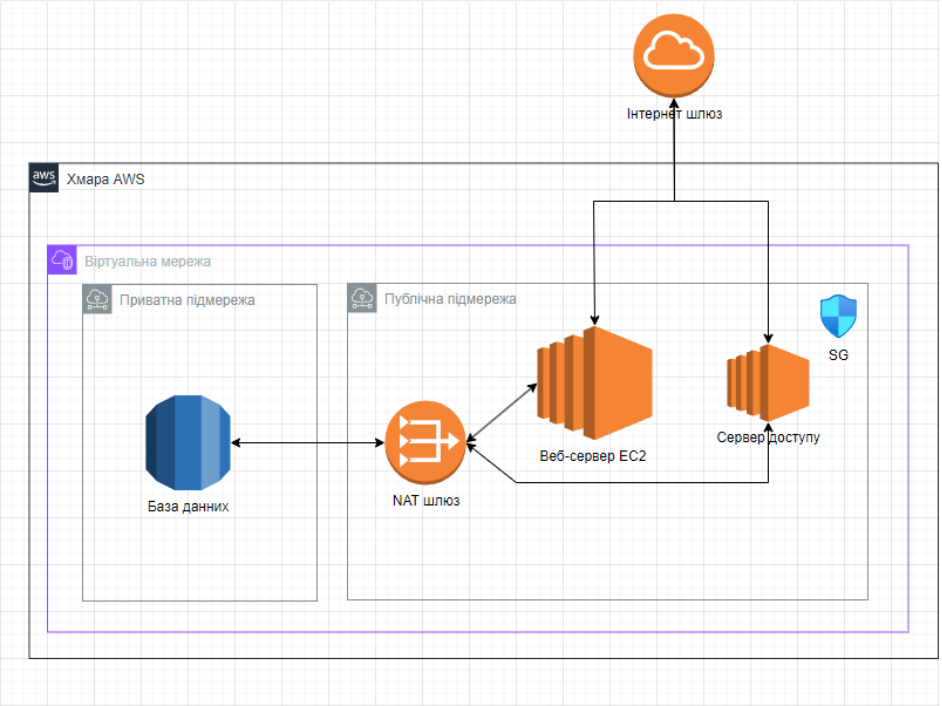

Рисунок 4 – Схема інфраструктури 4-огосценарію

Код для 4-ого сценарію інфраструктури Див. Додаток А Лістинг 7 та Лістинг 8.

1. VPC: створюється основна мережа VPC з адресою 10.0.0.0/16

2. Публічна та приватна підмережі: публічна підмережа 10.0.1.0/24 і приватна підмережа 10.0.2.0/24.

3. Інтернет-шлюз та NAT-шлюз: інтернет-шлюз для публічної підмережі та NAT-шлюз для приватної підмережі.

4. Таблиця маршрутизації (PublicRouteTable) з маршрутом 0.0.0.0/0, котра і дає доступу до Інтернету для Інтернет-шлюзу.

5. Bastion-сервер доступу: EC2-сервер в публічній підмережі, через який здійснюється доступ до веб-серверу EC2 та бази даних RDS для їх технічного обслуговування.

6. EC2-веб-сервер: сервер, на якому знаходить застосунок.

7. RDS: База даних MySQL в приватній підмережі.

Сценарій підходить для корпоративних задач та вирішує проблему із захистом особистих даних користувачів розміщенням бази даних в приватній

підмережі, із доступом до неї лише через Bastion-сервер доступу в публічній підмережі.

3.5 Мікросервісна інфраструктура

Інфраструктура п’ятого сценарію розгортає устаткування, котре забезпечує потреби мікросервісів та оркестрації. В даному випадку технології Kubernetes. Обсяг застосування мікросервісної архітектури охоплює багато можливостей, один із подібних - розбиття одного великого сервісу на кілька менших для економії ресурсів та вирішення проблеми із захищеністю даних.

3.5.2 Розрахунок та розгортання 5-ого сценарію

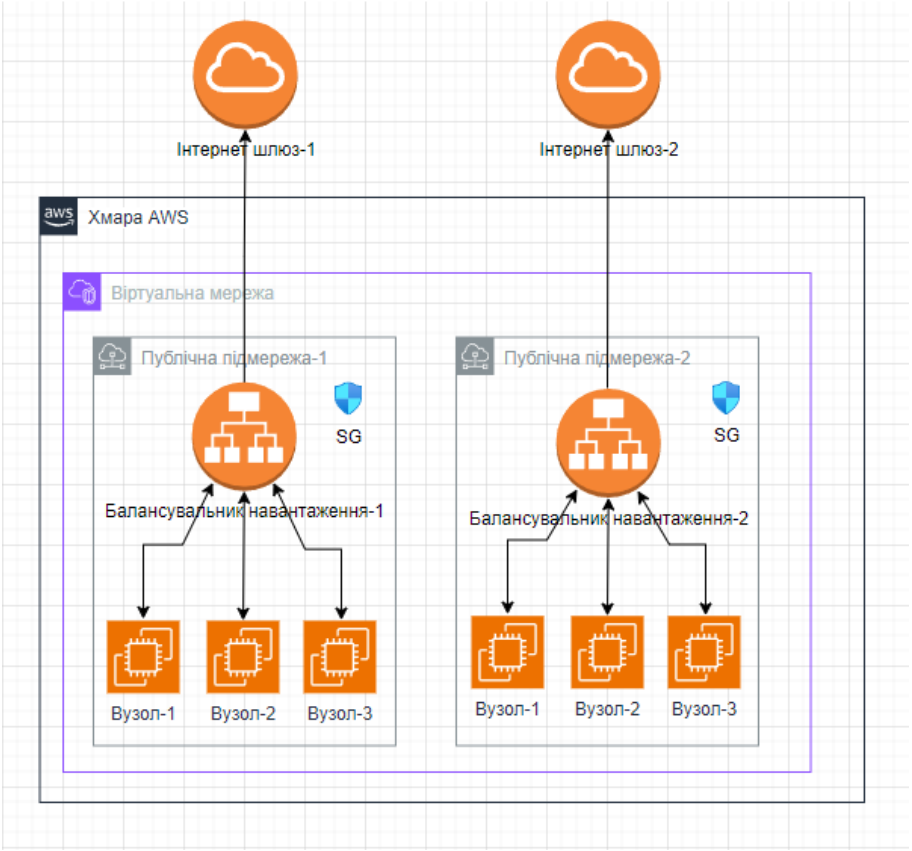

Рисунок 3.5 — Схема інфраструктури 5-огосценарію

Код для 5-ого сценарію інфраструктури Див. Додаток А Лістинг 9 та Лістинг 10.

1. VPC: створюється основна мережа VPC з адресою 10.0.0.0/16.

2. Дві публічні підмережі з адресами 10.0.1.0/24 та 10.0.2.0/24 відповідно.

3. Два інтернет-шлюзи (InternetGateway): забезпечують балансувальникам доступ до Інтернету.

4. EKS Кластер: кластер з двома групами вузлів (по три ноди в кожній підмережі), які підключаються до публічних підмереж.

5. Балансувальники навантаження: два ALB, по одному для кожної підмережі, для обробки HTTP трафіку.

4 АНАЛІЗ ІНФРАСТРУКТУРИ

4.1 Перевірка працездатності

Перевірка працездатності проводитиметься за допомогою скрипту штучного навантаження (Див. Додаток А Лістинг 11). За його допомогою будуть створюватись випадкові заклики за протоколом HTTP на порт 80 із частотою як би це був максимально правдоподібний людський трафік. Навантаження триватиме приблизно годину на кожним зі сценаріїв зі змінною потужністю, що дозволить, по-перше, перевірити стан завантаження CPU серверів, по-друге, на основі годинних витрат на інфраструктуру спрогнозувати можливі витрати на місяць при такому ж очікуваному трафіку.

Рисунок 4.6 – Графік навантаження процесора сервера під час стрес-тесту

Метрики навантаження були зроблені в сервісі AWS CloudWatch, котрий дозволяє встановити моніторинг на будь-який піднятий ресурс.

4.2 Прогноз коштовності інфраструктури

Сервіс AWS Billing дозволяє продивитись рахунки за роботу піднятої інфраструктури та приблизно спрогнозувати її вартість на поточний місяць. Дуже добре працюють сервіси AWS Budgets та AWS Pricing Calculator [6] у зв'язці для точнішого контролю коштів на ресурси.

Прогнози по п'яти сценаріях на місяць роботи при такому ж імовірному навантажені буде:

Таблиця 4.2 — Місячний прогноз для 1-ого сценарію

S3	0,06$
CloudFront	5,28$
Загалом:	5,34$

Таблиця 4.3 – Місячний прогноз для 2-ого сценарію

VPC	1,53$
Internet Gateway	10,73$
EC2	12,64$
EBS	0,61$
Загалом:	24,9$

Таблиця 4.4 – Місячний прогноз для 3-ого сценарію

VPC	1,53$
Internet Gateway	10,73$
EC2 (x2)	25,28$
App Load Balancer	24,38$
Загалом:	61,92$

Таблиця 4.5 – Місячний прогноз для 4-ого сценарію

VPC	1,53$

Internet Gateway	10,73$
EC2	13,64$
EC2 (bastion)	4,91$
RDS	34,66$
NAT Gateway	10,73$
Загалом:	76,2$

Таблиця 4.6 — Місячний прогноз для 5-ого сценарію

VPC	1,53$
Internet Gateway (x2)	21,46$
App Load Balancer (x2)	48,76$
EKS	73$
EC2 nodes (x6)	75,84$
Загалом:	220,59$

4.3 Аналіз підходу ручного розгортання хмарної інфраструктури та підходу IaC

Основна перевага використання підходу “Інфраструктура як код” — швидкість розгортання. Під час ручного розгортання DevOps інженеру необхідно в консолі сайту постачальника хмарних послуг “вручну” по черзі обирати кожний ресурс та параметри для його підняття. Для підходу із використанням інструментів Terraform або CloudFormation працівник витрачає значно менше часу на написання коду та підняття інфраструктури. Звісно час підняття/обслуговування інфраструктури залежить від інтернет з’єднання, досвіду спеціаліста, що виконує роботу, та комплексності інфраструктури.

Нижче наведено порівняння приблизного часу підняття кожного із сценаріїв за допомогою інструментів IaC відносно до ручного методу:

Таблиця 4.7 – Середній час розгортання сценаріїв

Номер сценарію	Час ручного розгортання	Час розгортання IaC	Співвідношення
1	8 хв	3 хв	2,6
2	12 хв	4 хв	3
3	21 хв	7 хв	3
4	27 хв	8 хв	3,3
5	38 хв	12 хв	3,1

Середня заробітна плата DevOps інженера на момент січня 2025 року становить приблизно 50000 грн [7]. Середня кількість робочих днів в місяць — 20. Відповідно, приблизна середня оплата праці є 312,5 грн/годину. Завдяки технології IaC, спеціаліст виконуючи підняття/обслуговування хмарної інфраструктури, витрачає приблизно втричі менше часу.

Якщо брати до уваги, що обсяг запланованих робіт міг ставити три дні, то витрати бізнесу тільки на оплату роботи DevOps інженера ставитимуть 7500 грн. За нового підходу ці витрати складатимуть приблизно в три рази нижче — 2500 грн.

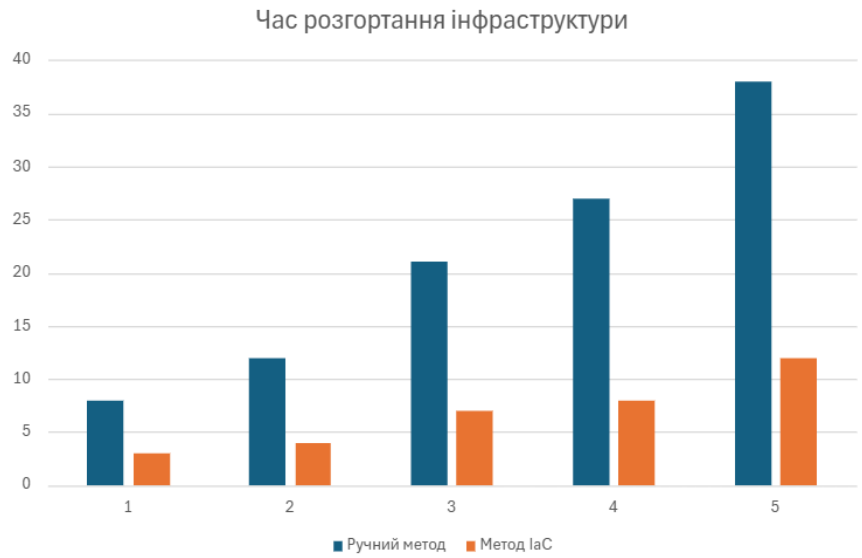

Рисунок 4.7 — Порівняння часу розгортання інфраструктури

Проаналізувавши графік на Рисунку 4.7 можна зробити висновок, що чим

комплексніша інфраструктура, тим більша економія часу для підняття методом IaC. Хоча цей результат може відрізнятись залежно від складності самого ресурсу, що підіймаються, на стороні дата центру хмарного постачальника.

ВИСНОВКИ

Було проведено аналіз технології “Інфраструктура як код”(IaC) в хмарних технологіях, постачальника хмарних послуг AWS. Використані інструменти IaC — універсальних багатохмарний Terraform та вузьконаправлений AWS CloudFormation. Інструмент Terraform є значно функціональнішим та гнучкішим знаряддям, оскільки є підтримка багатьох постачальників хмірних послуг, багато програмних розширень та доповнень. AWS CloudFormation є вузькопрофільний інструмент, розрахований тільки на використання в екосистемі хмари AWS. Маю значно плавнішу та повну інтеграцію з майже усіма сервісами Amazon Web Services, однак має трішки заплутаніший з візуальної точки зору синтаксис.

Технологія IaC загалом пришвидшує та полегшує обслуговування хмарної інфраструктури можливостями швидкого розгортання/згортання, ідемпотентності, швидкого відновлення та перевикористання шаблонів інфраструктури. Як показав аналіз вище, розгортання типових сценаріїв методом IaC в середньому втричі швидший за ручне підняття хмарних ресурсів. Де є швидкість виконання роботи, там є економія бюджету. З іншої сторони, для обох інструментів потрібна первинна технічна підготовка.

ПЕРЕЛІК ПОСИЛАНЬ

1. Hashicorp Terrafrom Documentation: Terraform Documentation Language [Електронний ресурс] — Режим доступу до ресурсу: https://developer.hashicorp.com/terraform/language.

2. AWS Official Documentation: What is AWS Cloudformation? [Електронний ресурс] —
Режим доступу до ресурсу: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html.

3. Microsoft Ignite: What is Azure ARM? [Електронний ресурс] — Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/azure-resource manager/management/overview.

4. Google Cloud Platform documentation: GCP Deployment Manager fundamentals [Електронний ресурс] —
Режим доступу: https://cloud.google.com/deployment-manager/docs/fundamentals.

5. AWS What-Is Documentaion: What is Cloud Infrastructure? [Електронний ресурс] — Режим доступу: https://aws.amazon.com/what-is/cloud-infrastructure/.

6. AWS Calculator: Калькулятор оцінки інфраструктури [Електронний ресурс] — Режим доступу: https://calculator.aws/#/estimate.

7. Платформа вакансій WorkUa: Статистика зарплатних виплат за спеціальністю DevOps інженер [Електронний ресурс] — Режим доступу: https://www.work.ua/salary-devops+engineer/?salaryTrendsPeriod=3&jobsTrendsPeriod=3.

8. AWS офіціальний портал: What is AWS? [Електронний ресурс] — Режим доступу: https://aws.amazon.com/what-is-aws/.

9. Google Cloud documentation: Google Cloud overview [Електронний ресурс] — Режим доступу: https://cloud.google.com/docs/overview.