Харківський національний університет радіоелектроніки
(повне найменування вищого навчального закладу)

Факультет *інфокомунікацій*
(повна назва)

Кафедра *інформаційно-мережної інженерії*
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

на тему *Розробка концепції комплексних завдань для автоматизованої перевірки навичок роботи з AWS*

Виконав: студент *2* курсу, групи *ІМІзм-21-2*
напряму підготовки
172 "Телекомунікації і радіотехніка"
(шифр і назва напряму підготовки)
Прокопенко Р.О.
(прізвище та ініціали)
Керівник *Костромицький А.І.*
(прізвище та ініціали)

Допускається до захисту

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

Харків - 2023 рік

Не містить відомостей, заборонених до відкритого публікування

Студент _____

Керівник_____

Харківський національний університет радіоелектроніки
(повне найменування вищого навчального закладу)

Факультет *інфокомунікацій*

Кафедра *інформаційно-мережної інженерії*

Рівень вищої освіти *другий (магістерський)*

Напрям підготовки *172 «Телекомунікації і радіотехніка»*
(шифр і назва)

ЗАТВЕРДЖУЮ

Зав.кафедри _____
(Підпис)

"____" _____ 2023 року

З А В Д А Н Н Я
НА АТЕСТАЦІЙНУ РОБОТУ

Прокопенко Родіону Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка концепції комплексних завдань для автоматизованої перевірки навичок роботи з AWS*

затверджені наказом ВНЗ від "*24*" *березня* 2023 року №*59 Ст3*

2. Строк подання студентом роботи *26 травня 2023 року*

3. Вихідні дані до роботи *Провести аналіз та розробити концепцію практичних завдань для роботи з архітектурою AWS та продемонструвати на практиці.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Вступ

1. Введення в хмарні сервіси та основні концепти

2. Автоматизована система практичних завдань та перевірки досвіду користувача з AWS хмарою

3. Розробка практичної задачі для оцінки навичок використання декількох serverless сервісів та event-driven архітектури

Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Презентація Power Point (мета роботи, введення у хмарні технології, платформа для перевірки завданнь, безсерверні сервіси та їх взаємодія, практична робота, висновки)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	*доц. Костромицький А.І.*	27.03.23	

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів атестаційної роботи	Строк виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	*27.03.23*	
2	*Підбір літератури за темою роботи.*	*27.03-30.03.23*	
3	*Виконання розділу 1*	*30.03-05.04.23*	
4	*Виконання розділу 2*	*05.04-10.05.22*	
5	*Виконання розділу 3*	*05.04-10.05.23*	
6	*Оформлення презентаційного матеріалу, підготовка до атестаційної комісії*	*10.05-18.05.23*	

Дата видачі завдання *27* *березня* *2023 р.*

Студент _____ *Прокопенко Р.О..*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Костромицький А.І.*
(підпис) (прізвище та ініціали

РЕФЕРАТ

Пояснювальна записка: 49 стор., 17 рис., 6 джерел

Об'єкт роботи – Система автоматичної перевірки навичок роботи з AWS.

Мета роботи – є розробка масштабованої концепції практичних завдань та лабораторних робіт, а також створення процесу автоматизованої перевірки навичок роботи з AWS хмарою, що дозволить визначити правильність виконання задачі та рівень знань користувача у цій області. У даній роботі розглянута чинна платформа для виконання завдань, наближених до реальних бізнес-сценаріїв, а також інструменти, що використовуються для перевірки практичних навичок з AWS. Були проаналізовані основні підходи до створення завдань, та оптимізації інструментів перевірки знань користувачів, а також методи оцінки результатів тестування. Для створення нового завдання були використані сучасні CI/CD та IaC (Infrastructure as Code) інструменти, такі як Jenkins та Terraform.

Було розроблено завдяння, в якому потрібно налаштувати event-driven архітектуру, де подія, така як додавання файлів в S3 кошик, автоматично надсилає повідомлення в SQS чергу, а черга асинхронно виконує Lambda функцію, яка зменшує розмір завантаженого файлу. Це повністю безсерверне (serverless) рішення, яке широко використовується в різних сценаріях і може масштабуватись під будь-який обсяг навантажень.

AMAZON WEB SERVICES, JENKINS, TERRAFORM, BASH, SQS, S3, LAMBDA.

ABSTRACT

Explanatory note: 49 pp., 17 figures, 6 sources

The object of work is the system of automatic verification of working skills with AWS.

The purpose of the work - The main purpose of this work is to develop a scalable concept of practical tasks and laboratory work, as well as to create a process of automated verification of skills in working with the AWS cloud, which will allow determining the correctness of the task and the level of user knowledge in this area. This paper examines the current platform for performing tasks close to real business scenarios, as well as the tools used to test practical skills with AWS. The main approaches to creating tasks and optimizing tools for testing users' knowledge, as well as methods for evaluating test results, were analyzed. Modern CI/CD and IaC (Infrastructure as Code) tools such as Jenkins and Terraform were used to create the new task.

A task was developed to set up an event-driven architecture, where an event such as adding files to an S3 bucket automatically sends a message to an SQS queue, and the queue asynchronously executes a Lambda function that reduces the size of the downloaded file. It is a completely serverless solution that is widely used in various scenarios and can scale to any volume of workloads.

AMAZON WEB SERVICES, JENKINS, TERRAFORM, BASH, SQS, S3, LAMBDA.

ЗМІСТ

ПРЕЛІК СКОРОЧЕНЬ

AWS (Amazon Web Services) – хмарний провайдер від Amazon

SQS (Simple Queue Service) – сервіс черг в AWS

S3 (Simple Storage Service) – сервіс сховища данних в AWS

Lambda – сервіс безсерверних обчислень в AWS

CI/CD (Continuous Integration/Continuous Delivery) – методологія організації процессу розробки програмного забезбечення

IaC (Infrastructure as Code) – методологія створення хмарної інфраструктури

CLI (Command-line Interface) – інтерфейс командної строки

ВСТУП

У сучасному світі використання хмарних сервісів надає величезний обсяг переваг у порівнянні із традиційним підходом до серверної інфраструктури та розробки програмного забезпечення. Уявіть, що в нас є велика всесвітня компанія, наприклад банк, і ми плануємо створювати під додатки та аплікації IT інфраструктуру. У традиційному підході, так званої on-premise моделі, потрібно закупити сервери, налаштувати їх, адмініструвати та здійснювати технічне обслуговування, забезпечувати безперебійну роботу, відслідковувати та управляти навантаженням, оплачувати оренду тощо. Вказане свідчить про велику кількість пов’язаної операційної роботи і необхідність постійного адміністрування, що як наслідок приводить суб’єкта господарювання до величезних та не завжди оправданих витрат. У випадку використання хмарного сховища все це можна побудувати за декілька хвилин, що суттєво зекономить час та гроші.

«Cloud» змінює правила гри на IT ринку, та створює необхідність для популяризації навчання хмарним технологіям, що в свою чергу ставить перед викладачами завдання з розробки новітніх навчальних програм, створення практичних завдань та лабораторних досліджень (робіт), та автоматизації контролю правильності виконання цих завдань.

Наразі існує багато постачальників IT ресурсів, серед яких виділяються 3 компанії, що займають питому частину ринку (рис.1В). Це Azure від Microsoft, GCP від Google та AWS від Amazon. Кожен «клауд» має свої особливості та відмінності, переваги та недоліки, але загальна концепція схожа між собою – доставляти IT ресурси та сервіси on-demand там де це потрібно. Ми сфокусуємося на лідеру за кількістю користувачів: AWS. Amazon Web Services - це популярний хмарний провайдер, який пропонує широкий набір глобальних IT-продуктів, включаючи обчислювальні потужності, сховища, бази даних, аналітику, мережі, машинне навчання, інтернет речей (IoT), штучний інтелект, корпоративні програми тощо. Ці

послуги допомагають організаціям рухатися швидше, масштабуватись та оптимізувати IT-витрати. AWS стає все більш популярним серед користувачів і вже є стандартом ринку, з огляду на що виникає потреба у якісних спеціалістах хмарних технологій. З цією метою створюються навчальні програми та інструменти для автоматизованої перевірки практичних навичок роботи з AWS, що стає все більш вимогливою задачею.

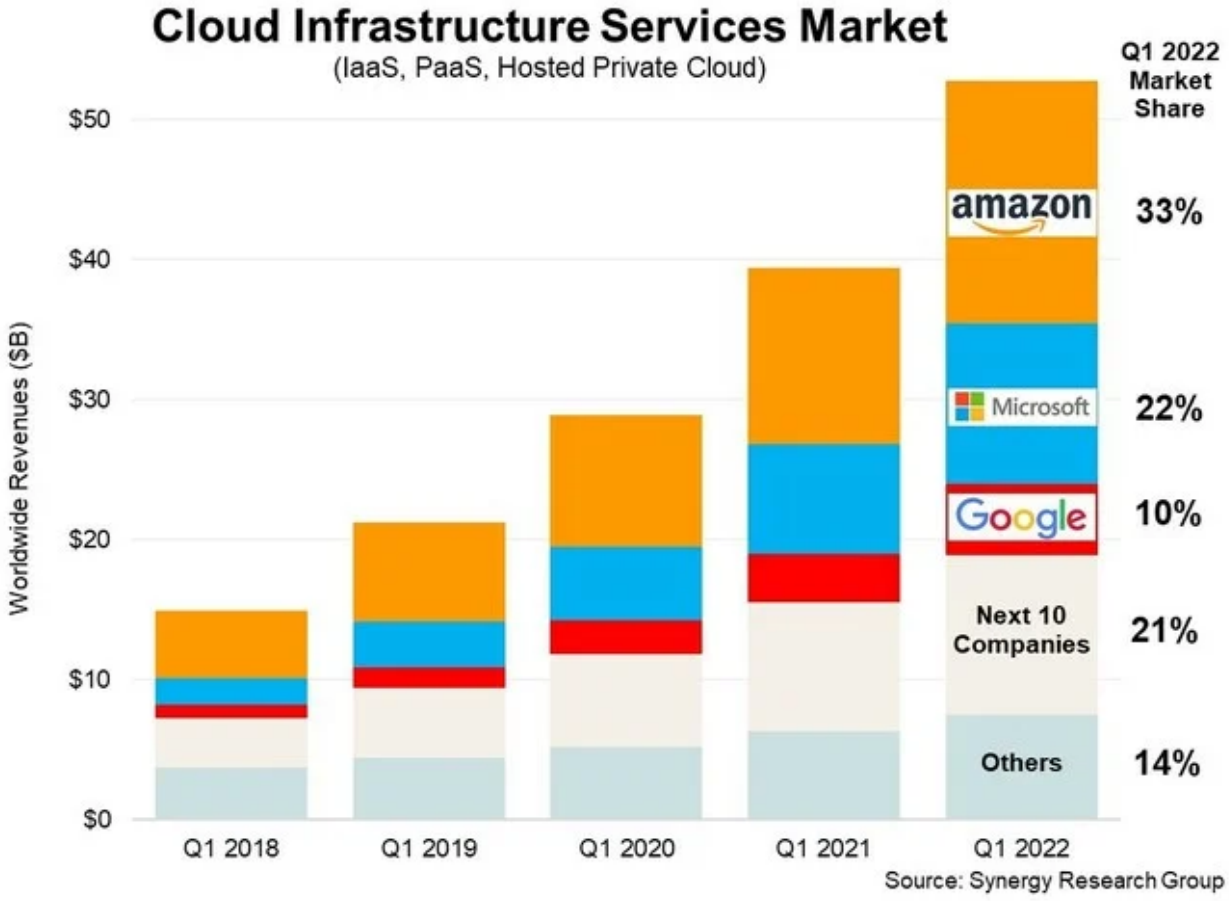

Рисунок 1В – Доля ринку популярних хмарних провайдерів

1 ВВЕДЕННЯ У ХМАРНІ СЕРВІСИ ТА ОСНОВНІ КОНЦЕПЦІЇ

1.1 Що таке хмара і як вона побудована

Глобальна хмарна інфраструктура AWS — це безпечна та надійна хмарна платформа, яка пропонує понад 200 повнофункціональних веб-сервісів із центрів обробки даних по всьому світу. Незалежно від того, чи потрібно вам розгорнути робочі навантаження програми одним кліком миші, чи Ви хочете створити та розгорнути конкретні аплікації ближче до кінцевих користувачів із мінімальною затримкою, AWS надасть Вам хмарну інфраструктуру для будь-яких потреб.

Завдяки мільйонам активних клієнтів і десяткам тисяч партнерів по всьому світу AWS має найбільшу та найдинамічнішу екосистему. Клієнти практично в кожній галузі та будь-якого розміру, включаючи стартапи, підприємства та організації державного сектору, користуються AWS у будь-яких варіантах використання.

Клієнти переходять на AWS, щоб підвищити гнучкість.

Переваги AWS для суб’єктів господарювання:

• Прискорення виходу бізнесу на ринок. Витрачаючи менше часу на придбання інфраструктури та керування нею, Ви можете зосередитися на розробці функцій, які приносять користь Вашим клієнтам.

• Збільшення інновацій – Ви можете пришвидшити свою цифрову трансформацію за допомогою AWS, яка надає інструменти для більш легкого доступу до новітніх технологій і передового досвіду. Наприклад, Ви можете використовувати AWS для розробки автоматизації, застосування контейнеризації та використання машинного навчання.

• Масштабування – AWS надає Вам додаткові ресурси для підтримки нових функцій, можливості збільшити або зменшити наявні ресурси відповідно до Ваших побажань та мети.

• Зменшення складності та ризиків.

• Оптимізація витрат – Ви можете зменшити витрати, сплачуючи лише за ті сервіси, що використовуєте. Замість того, щоб платити за локальне обладнання, яке може не використовуватися на повну потужність, Ви можете платити за обчислювальні ресурси лише під час їх використання.

• Зведіть до мінімуму вразливі місця безпеки – перехід на AWS помістить додатки та дані під розширену фізичну безпеку центрів обробки даних AWS. У AWS існує також багато інструментів для керування доступом до ваших ресурсів.

• Зменшення складності управління – використання служб AWS може зменшити потребу в обслуговуванні фізичних центрів обробки даних, виконувати технічне обслуговування апаратного забезпечення та керувати фізичною інфраструктурою.

• AWS пропонує широкий набір глобальних хмарних продуктів, включаючи обчислення, сховище, базу даних, аналітику, мережу, мобільні пристрої, інструменти розробника, інструменти управління, інтернет речей (IoT), безпеку та корпоративні програми (рис 1.1). Ці послуги допомагають організаціям рухатися швидше, масштабуватися та зменшувати IT-витрати. AWS охоплює послуги інфраструктури, основи та додатків.

Рисунок 1.1 – Категорії сервісів в AWS

1.2 Як організована глобальна інфраструктура AWS

У 2006 році AWS започаткувала хмарні обчислення, щоб забезпечити швидку та безпечну інфраструктуру. AWS постійно впроваджує інновації в дизайн і системи центрів обробки даних, щоб захистити їх від антропогенних і природних ризиків. Сьогодні AWS надає центри обробки даних у великому глобальному масштабі.

AWS впроваджує засоби контролю, створює автоматизовані системи та проводить сторонні аудити для підтвердження безпеки та відповідності. Як наслідок, найбільш регульовані організації у світі щодня довіряють AWS.

Розглянемо основні компоненти, з яких складається хмара AWS.

Зони доступності, або Availability zones

Група з одного або кількох центрів обробки даних називається зоною доступності.

Зона доступності — це один або кілька незалежних центрів обробки даних із резервним живленням, мережею та підключенням у регіоні AWS. Коли ви запускаєте екземпляр VM, ви можете обрати зону доступності або дозволити AWS вибрати її для вас. Якщо ви розподіляєте свої екземпляри між кількома зонами доступності, і один екземпляр виходить з ладу, ви можете створити свою програму так, щоб екземпляр в іншій зоні доступності міг обробляти запити.

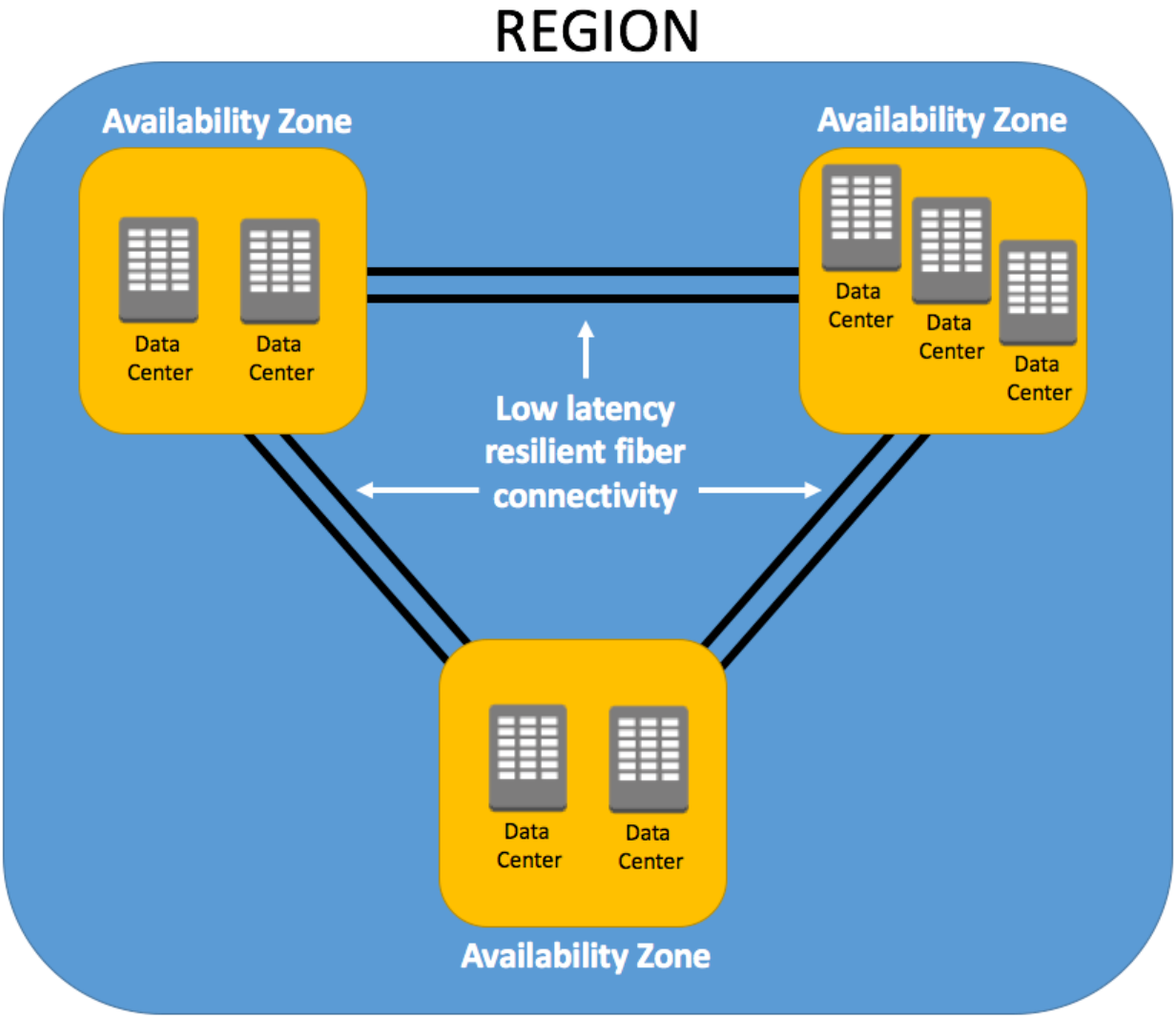

Рисунок 1.2 – Зони доступності та Регіон

Регіони / Regions

Кожен регіон AWS складається з кількох ізольованих і фізично окремих зон доступності в межах географічної області (рис 1.2). Це забезпечує максимально можливу відмовостійкість і стабільність. У своєму обліковому записі ви визначаєте, які регіони вам потрібні.

Коли ви переглядаєте свої ресурси, ви бачите лише ті ресурси, які прив’язані до регіону, який ви вказуєте на консолі. Це пояснюється тим, що регіони ізольовані один від одного, і AWS не реплікує ресурси автоматично між регіонами.

Ви можете запускати програми та робочі навантаження з регіону, щоб зменшити затримку для кінцевих користувачів. Ви можете зробити це,

уникнувши попередніх витрат, довгострокових зобов'язань і проблем масштабування, пов'язаних із підтримкою та експлуатацією глобальної інфраструктури.

Важливо вибрати правильний регіон (рис 1.3). Ви повинні визначити правильний регіон для своїх служб, програм і даних на основі таких факторів:

• Управління та юридичні вимоги – розгляньте будь-які юридичні вимоги, що ґрунтуються на законах про управління даними, суверенітет або конфіденційність.

• Затримка – близькість до клієнтів означає кращу продуктивність.

• Доступність послуги – не всі служби AWS доступні в усіх регіонах.

• Вартість – у різних регіонах різні витрати. Дослідіть ціни на послуги, якими ви плануєте користуватися, і порівняйте витрати, щоб прийняти найкраще рішення для свого робочого навантаження

Edge Locations

Edge locations – це найближча точка до запитувача служби AWS. Локації Edge розташовані у великих містах світу. Вони отримують запити та кешують копії ваших даних для швидшої доставки.

Щоб надавати контент кінцевим користувачам із меншою затримкою, ви використовуєте глобальну мережу периферійних місць, які підтримують служби AWS. CloudFront доставляє клієнтський контент через всесвітню мережу точок присутності (PoP), яка складається з периферійних місць і регіональних серверів периферійного кешу.

Регіональні периферійні кеші, які використовуються за замовчуванням у CloudFront, використовуються, коли у вас є контент, доступ до якого відбувається недостатньо часто, щоб залишатися в крайньому місці. Регіональні периферійні кеші поглинають цей вміст і забезпечують альтернативу необхідності отримувати цей вміст із вихідного сервера.

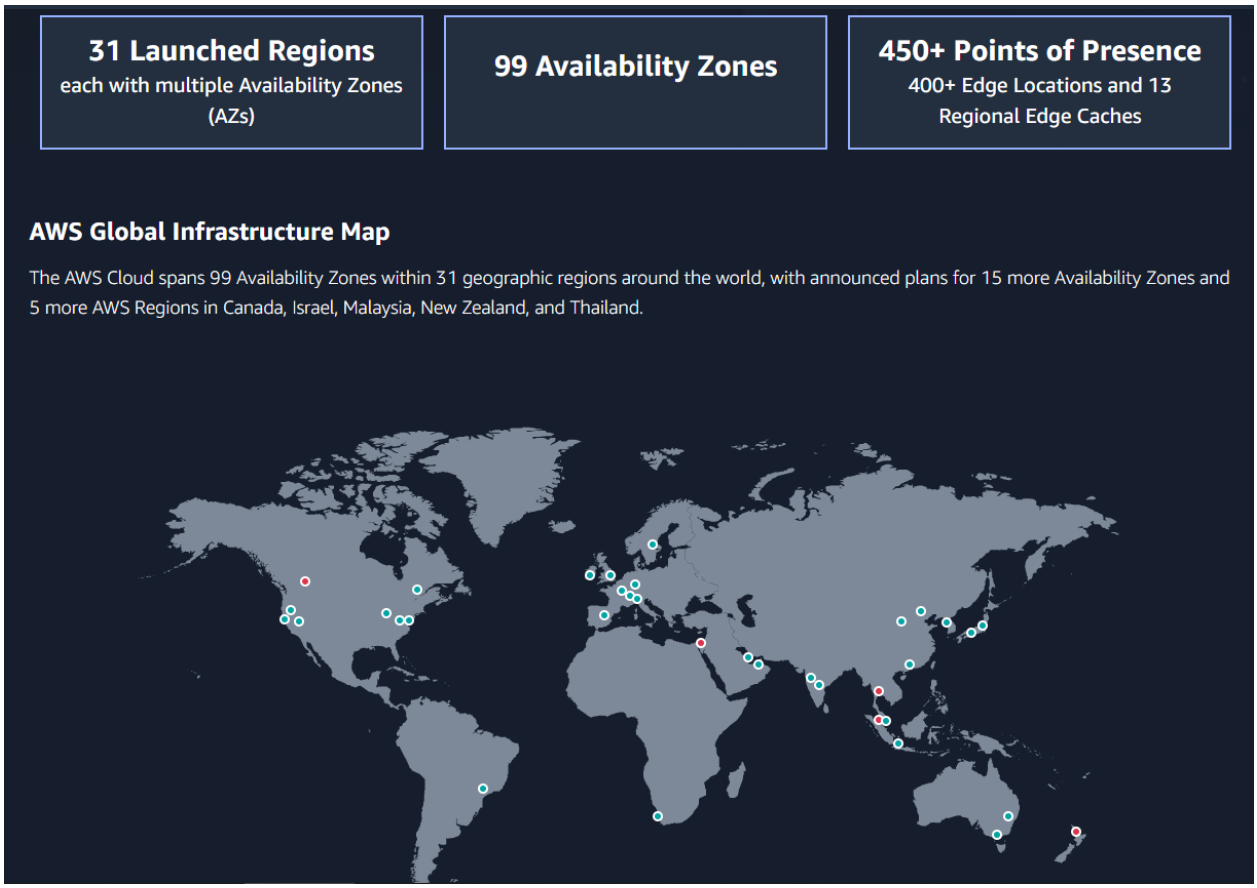

Рисунок 1.3 – Регіони

1.3 Як ми можемо побудувати нашу хмарну інфраструктуру відповідно до найкращих практик

Створення технологічних рішень дуже схоже на будівництво фізичної будівлі. Якщо фундамент не міцний, це може спричинити структурні проблеми, які підірвуть цілісність і функціональність будівлі. AWS Well-Architected Framework допомагає хмарним архітекторам створювати безпечні, високопродуктивні, стійкі та ефективні інфраструктури додатків. Це послідовний підхід для клієнтів і партнерів до оцінки архітектури та реалізації проектів, які можуть масштабуватися з часом.

AWS Well-Architected Framework починався як офіційний документ. Його розширили, включивши предметно-спеціальні об’єктиви, практичні лабораторії та інструмент AWS Well-Architected Tool (AWS WA Tool).

Архітектурні огляди зосереджені на наступному:

• Безпека. Використовуйте найкращі методи безпеки AWS для розробки політик і процесів для захисту даних і активів. Дозволити аудит і відстеження. Відстежуйте, сповіщайте та перевіряйте дії та зміни у вашому середовищі в реальному часі.

• Оптимізація витрат – досягайте ефективності витрат, враховуючи мінливі потреби в ресурсах.

• Надійність – відповідність чітко визначеним робочим порогам для програм. Це включає підтримку для відновлення після збоїв, обробки підвищеного попиту та пом’якшення збоїв.

• Ефективність продуктивності. Забезпечте ефективну продуктивність для набору ресурсів, таких як екземпляри, сховище, бази даних, простір і час.

• Експлуатаційна досконалість – Запуск і моніторинг систем, які забезпечують бізнес-цінність. Постійно вдосконалюйте допоміжні процеси та процедури.

• Екологічність – зведіть до мінімуму та зрозумійте свій вплив на навколишнє середовище під час виконання хмарних робочих навантажень.

За допомогою інструменту ви можете збирати дані та отримувати рекомендації щодо:

• Мінімізуйте системні збої та експлуатаційні витрати.

• Глибоке занурення в бізнес та інфраструктурні процеси.

1.4 Ключові обчислювальні сервіси

Коли мова йде про обчислювальні можливості то потрібно розуміти що таке віртуальні машини та віртуалізація. VM це виділені ресурси такі як оперативна пам’ять, процесор, диск, та операційна система, які ізольовані і виконують роль сервера, та забезпечують наступне:

• Незалежність від апаратного забезпечення

• Швидке надання за хвилини або години

• Розрахункові моделі ціноутворення замість придбання обладнання

• Більший масштаб

• Еластичні ресурси

• Більша маневреність

• Зменшене обслуговування

Amazon EC2 був одним із перших сервісів AWS, випущених у 2006 році, і продовжує залишатися центральним компонентом хмарних обчислень. Нові покоління типів екземплярів EC2 забезпечують більшу ефективність обчислень, що може допомогти зменшити витрати на обчислення.

Контейнерізація забезпечує наступне:

• Незалежність від платформи

• Послідовне середовище виконання

• Більше використання ресурсів

• Простіше та швидше розгортання

• Ізоляція

У 2014 році служба Amazon Elastic Container Service (Amazon ECS) представила можливість запускати розподілені програми на керованому кластері інстансів Amazon EC2 з контейнерами Docker. Підтримка Kubernetes була випущена в 2017 році за допомогою Amazon Elastic Kubernetes Service (Amazon EKS).

Також набирає популярності підхід Serverless. Безсерверні обчислення забезпечують наступне:

• Безперервне масштабування

• Вбудована відмовостійкість

• Платіть коли вікористовуєте

• Нульове обслуговування

AWS Lambda також з’явилася в 2014 році, представивши безсерверні обчислення. За допомогою Lambda ви можете запускати код без підготовки чи керування інстансами EC2. Безсерверні обчислення та контейнеризація були об’єднані в 2017 році з випуском AWS Fargate, механізму безсерверних обчислень для контейнерів, який працює з Amazon ECS і Amazon EKS.

AWS представила спеціалізовані процесори для підтримки впровадження штучного інтелекту (AI) і машинного навчання (ML). Мікросхеми AWS Inferentia були представлені в 2019 році, щоб забезпечити високопродуктивні мікросхеми машинного навчання, розроблені та створені AWS. Вони були створені для підтримки програм машинного навчання. AWS Trainium був представлений у 2021 році як другий чіп машинного навчання, створений AWS. Trainium оптимізовано для високоефективного глибокого навчання.

AWS також створила процесори на замовлення та представила різноманітні процесори AWS Graviton. Процесори Graviton побудовані на основі ядер Arm і широко використовують кремній. У 2018 році AWS Graviton було випущено та створено для масштабованих робочих навантажень, де ви можете розподілити навантаження між групою менших інстанцій. У 2020 році було випущено AWS Graviton2 із використанням 64-розрядних ядер Arm Neoverse, щоб забезпечити найкращу цінову продуктивність для хмарних робочих навантажень, що виконуються в Amazon EC2. У 2022 році було випущено AWS Graviton3, щоб забезпечити найкращу цінову продуктивність для робочих навантажень в Amazon EC2.

Перед початком роботи з інстансами EC2 потрібно продумати параметри віртуальної машини.

Зверніть увагу на наступне:

• Ім’я та теги – як ідентифікувати ваш екземпляр?

• Програма та образ ОС – що ви почнете запускати?

• Тип і розмір екземпляра – які у вас технічні вимоги?

• Автентифікація та пара ключів – як ви плануєте підключитися до примірника?

• Параметри та безпека мережі – яку віртуальну приватну хмару (VPC), підмережу та групи безпеки ви будете використовувати?

• Налаштувати сховище – який тип блокового сховища найкраще підходить для вашого випадку використання?

• Розміщення та оренда – де слід запускати ваші екземпляри EC2?

• Сценарії та метадані – що можна зробити, щоб автоматизувати запуск?

Ви можете вибрати з понад 400 типів екземплярів EC2 для запуску програм, які ви переміщуєте в хмару. Кожен тип екземпляра має різні розміри, з різними розподілами віртуальних ЦП (vCPU) і пам’яті. Вибір правильних розмірів екземплярів має вирішальне значення для їх ефективного використання. Повний тип екземпляра складається з назви сімейства, після якого йде номер покоління, будь-які додаткові властивості, а потім розмір.

Назва типу інстансу складається з наступних компонентів:

• c – перша літера позначає сімейство інстансів. Отже, сімейство c, наприклад, оптимізоване для обчислень. Існують екземпляри EC2 загального призначення, пакетні екземпляри, екземпляри, що потребують інтенсивних обчислень і пам’яті, і це лише деякі з них.

• 6 – наступне число – це покоління, яке поступово збільшується, оскільки AWS оновлює обладнання у своїх центрах обробки даних.

• g – іноді після генерації є одна або кілька літер. Літери позначають додаткові властивості. У цьому прикладі g означає Graviton2, процесор на базі ARM, розроблений AWS. Виберіть властивості відповідно до ваших потреб, як-от оптимізована пропускна здатність мережі або сховище.

• xlarge – остання частина представляє розмір екземпляра. Це включає процесор, пам’ять, сховище та продуктивність мережі.

Цей тип екземпляра читається як екземпляр c-типу шостого покоління з процесором Graviton2 і розгортанням надзвичайно великого розміру.

Виберіть оптимальне сімейство екземплярів для типу робочого навантаження, яке ви плануєте розгорнути. Це економить час і кошти, а також зменшує необхідність змінювати розмір пізніше. Деякі типи екземплярів доступні лише в певних регіонах.

Головна мета

• Баланс обчислення, пам'яті та мережі

• Різноманітні навантаження

• Веб-додатки

Оптимізовано обчислення

• Обчислювальні програми

• Високопродуктивні процесори

• Перекодування медіа

• Наукове моделювання

• Машинне навчання

Оптимізовано пам'ять

• Швидка доставка великих наборів даних у пам'ять

• Сервери баз даних

• Веб-кеші

• Аналіз даних

Прискорені обчислення

• Висока обробка графіки

• Прив’язка GPU

• Машинне навчання

• Високопродуктивне обчислення (HPC)

• Автономні транспортні засоби

Оптимізовано зберігання

• Висока послідовність читання/запису

• Великі набори даних

• Бази даних NoSQL

1.5 IaC (Infrastructure as Code)

Ви можете спростити розгортання своїх ресурсів AWS за допомогою інфраструктури як коду (IaC). З IaC ви використовуєте код для створення, розгортання, налаштування, оновлення та видалення інфраструктури.

Шаблон (template) – це текстовий файл, який описує та визначає ресурси, які потрібно розгорнути у вашому середовищі та їх налаштування, залежності

тощо. Цей шаблон обробляється IaC інструментами, такими як Terraform, Cloudformation, або AWS CDK (cloud development kit), які створюють зазначену інфраструктуру.

У випадку з Cloudformation ви визначаєте весь стек програми (усі ресурси, необхідні для вашої аплікації), та налаштування інфраструктури у файлі шаблону YAML або JSON (рис 1.4).

У CDK шаблон описується на TypeScript, Python та іншими мовами програмування, і на основі цього генерується стек у Cloudformation (рис 1.5).

В Terraform свій синтаксис, який називається HashiCorp Configuration Language (HCL) а файли у форматі tf або json (рис 1.4).

Незалежно від IaC інструменту всі шаблони - це звичайні текстові файли, які можно покласти поруч з кодом аплікації і використовувати усі переваги Git, CI/CD та Agile.

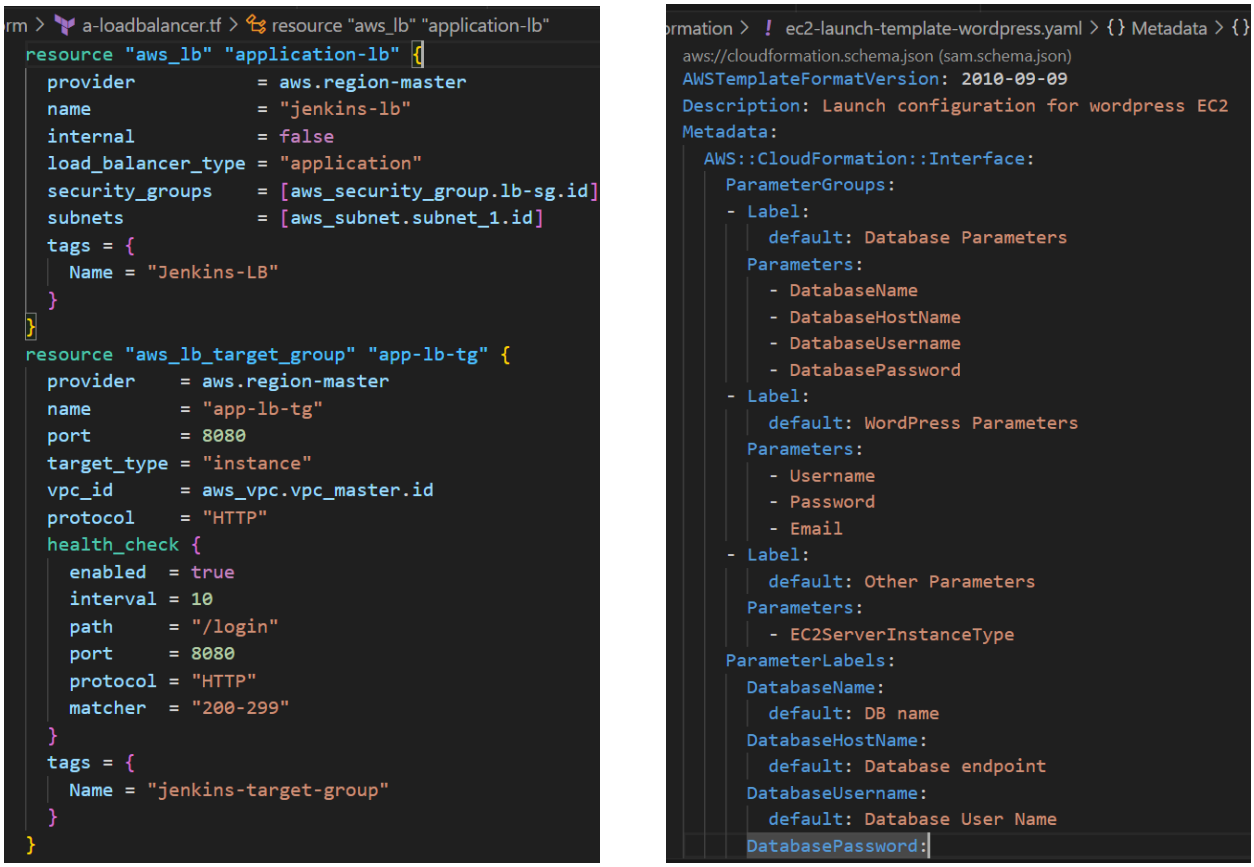

Рисунок 1.4 – Зліва приклад Terraform template, справа – Cloudformation

Рисунок 1.5 – Код для Aws Cdk на TypeScript

IaC має такі переваги:

• Швидкість і безпека. Ваша інфраструктура побудована програмно, що робить її швидшою, ніж розгортання вручну, і зменшує ймовірність помилок.

• Можливість багаторазового використання – ви можете організувати свою інфраструктуру в модулі для повторного використання (рис 1.6).

• Документація та контроль версій – ваші шаблони документують ваші розгорнуті ресурси, а контроль версій надає історію вашої інфраструктури за певний час. Ви також можете повернутися до попередньої робочої версії вашої інфраструктури в разі помилки.

• Перевірка – ви виконуєте перевірку та автотестування коду своїх шаблонів, що зменшує ймовірність помилок.

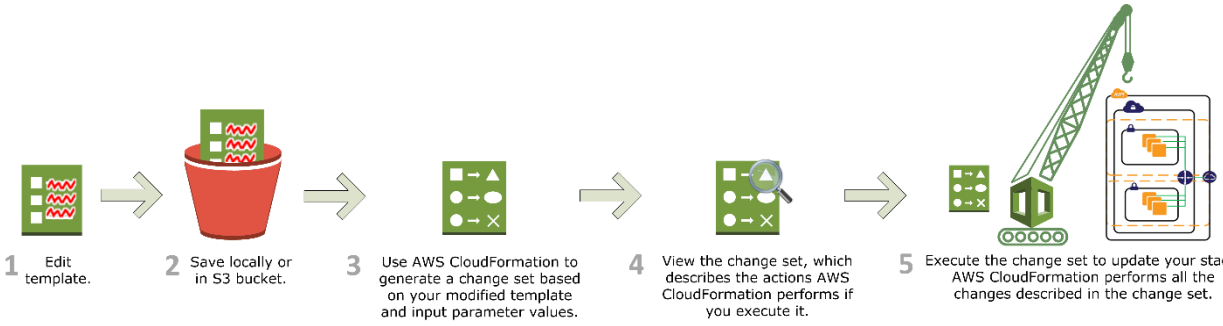

Рисунок 1.6 – Приклад делпою Cloudformation template

1.6 DevOps Continuous integration & Continuous delivery (CI/CD)

Одна з ключових ДевОпс практик це безперервна інтеграція та безперервна доставка. Конвеєр CI/CD використовує інструменти для автоматизації робочих процесів команд розробки, щоб допомогти їм у кодуванні, створенні, тестуванні та розгортанні їхніх програм швидше, якісніше та надійніше.

Завдяки безперервній інтеграції (CI) розробники використовують спільний репозиторій за допомогою системи контролю версій, такої як GitHub, Bitbucket або AWS CodeCommit. Перед кожним комітом розробники можуть запускати локальні модульні тести свого коду як додатковий рівень перевірки перед інтеграцією. Служба безперервної інтеграції автоматично створює та запускає модульні тести на основі нових змін коду, щоб негайно виявити будь-які помилки та проаналізувати зміни.

Безперервна доставка та безперервне розгортання або просто Continuous delivery/deployment (CD) розширюють постійну інтеграцію для автоматичного випуску програми. Різниця між безперервною доставкою та безперервним розгортанням полягає в тому, що безперервна доставка призупиняє конвеєр і вимагає втручання людини перед розгортанням у production, в той час коли безперервне розгортання все робить автоматично.

CD автоматизує більшу частину процесу випуску програмного забезпечення. Кожна зафіксована версія запускає автоматичний потік, який збирає, тестує, а потім виконує оновлення. Однак розробник повинен ініціювати остаточне розгортання в живому робочому середовищі, яке не є автоматизованим. У безперервному розгортанні повний випуск відбувається автоматично без початкової перевірки вихідного коду.

CD розширює безперервну інтеграцію, розгортаючи всі зміни коду в середовищі тестування, робочому середовищі або в обох після етапу

складання (рис 1.6). Якщо безперервне постачання реалізовано належним чином, розробники матимуть перевірену збірку, готову до розгортання.

Завдяки безперервній доставці Ви можете:

• Автоматизувати процеси випуску програмного забезпечення.

• Підвищувати продуктивність розробників.

• Швидко знаходити та усувати помилки.

• Доставляти оновлення швидше та надійніше.

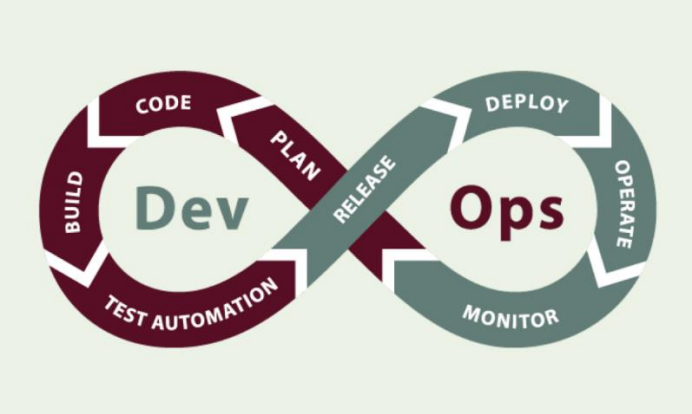

Рисунок 1.6 – DevOps методологія

Команди, які використовують DevOps CI/CD, отримують переваги від таких найкращих практик:

• Частота випусків зменшує труднощі та виклики та дозволяє швидше окупити інвестиції.

• Нові ітерації мають бути швидкими, а тому вимагають ефективної розробки програмного забезпечення з чітко визначеним обсягом і комплексним тестуванням.

• Послідовність підвищує впевненість.

• Автоматизуйте все для повторюваності, ефективності та послідовності.

• Автоматизація тестування для швидшого циклу зворотного зв’язку.б

Швидша доставка покращує практику розробки програмного забезпечення.

Постійний розвиток є важливим аспектом постійної інтеграції. У практиках безперервної розробки над кодом працюють кілька людей. Дуже важливо, щоб усі вони використовували останню робочу збірку для своїх зусиль. Репозиторії коду підтримують різні версії коду, а також роблять код доступним для команди. Ви перевіряєте код зі сховища, вносите зміни або пишете новий код у локальній копії, компілюєте та тестуєте свій код, а потім часто надсилаєте код назад у сховище.

2 АВТОМАТИЗОВАНА СИСТЕМА ПРАКТИЧНИХ ЗАВДАНЬ ТА ПЕРЕВІРКИ ДОСВІДУ КОРИСТУВАЧА З AWS ХМАРОЮ

2.1 Платформа для завдань

Фундаментом для цієї роботи стала існуюча навчальна платформа Cloud Mentor, яка містить багато практичних завдань та дозволяє перевіряти знання користувачів та правильність виконання робіт з AWS різного рівня складності. Основа платформи – це CI/CD конвеєр Jenkins, який керує усіма процесами аплікації, та взаємодіє з IaC сервісом Terraform для створення ресурсів в AWS, а також виконує роль веб інтерфейсу для кінцевого користувача. При виконанні кожного завдання автоматично створюється окреме оточення та інфраструктура необхідна для проходження. Інтерфейс користувача показано на рис. 2.1.

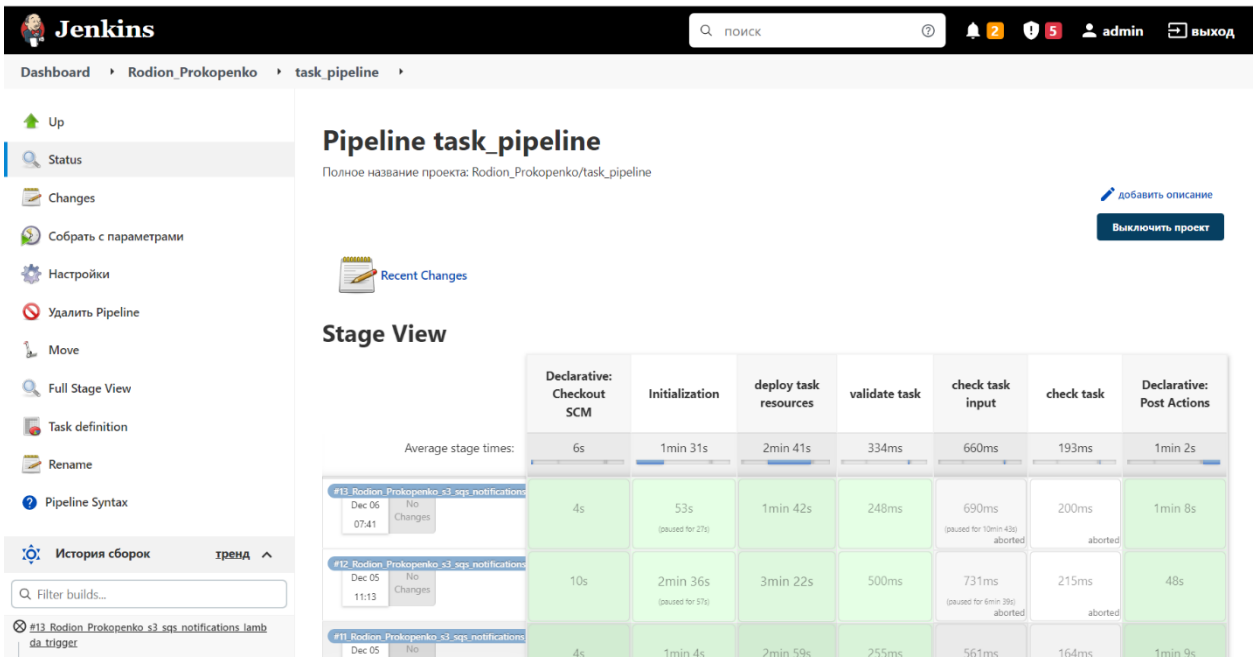

Рисунок 2.1. Інтерфейс користувача

Для розробки завдань були використані реальні бізнес-сценарії, а також рекомендації від фахівців з AWS.

Було встановлено, що виконання запропонованих робіт допомагає студентам поглиблювати свої знання та практичні навички роботи з AWS.

Оскільки платформа орієнтована на користувачів з початковими знаннями AWS, то завдання в першу чергу націлені на отримання практичних навичок з найпопулярнішими ресурсами, що входять до рівня безкоштовного використання (free tier).

Проте після отримання навичок роботи з базовими сервісами, наступним етапом є робота із сукупністю сервісів, їх інтеграція та спільна взаємодія в більш складних архітектурах.

Приклад комплексного завдання показано на рис.2.2. В цьому завданні потрібно налаштувати event-driven архітектуру, де подія, така як додавання файлів в S3 кошик, автоматично надсилає повідомлення в SQS чергу, а черга асинхронно виконує Lambda функцію, яка зменшує розмір завантаженого файлу. Це повністю безсерверне (serverless) рішення, яке широко використовується в різних сценаріях і може масштабуватись під будь-який обсяг навантажень. Ми ще повернемося до цієї задачі і більш детально її розглянемо.

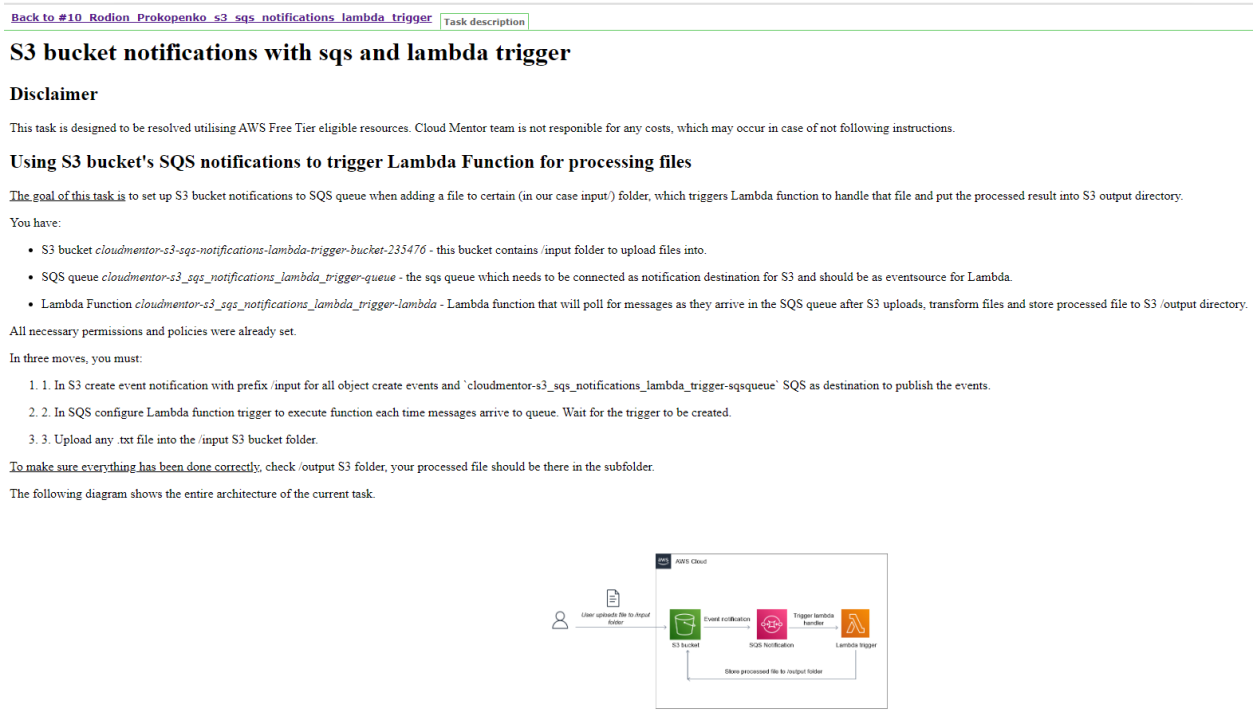

Рисунок 2.2. Приклад практичного завдання

Система добре масштабується і дозволяє легко створювати та додавати нові задачі та функціонал.

У якості прикладу давайте розглянемо локальний запуск та перше знайомство розробника з платформою.

Після завантаження Git репозиторію з кодом аплікації, доступно два способи локального розгортання. Перший: використовуючи Docker-compose файл, який запускає Jenkins, базу данних та api додатки у контейнерах, при виконанні команди *docker-compose up -–build* за хвилину всі сервіси будуть активовані. Та другий спосіб – використовуючи Oracle Virtual Box VM та Vagrant, що створює локальну віртуальну Linux машину з Jenkins та усіма налаштуваннями, в цьому випадку у відповідному каталозі репозиторію нам потрібно виконати команду *vagrant up* що створить та налаштує нам готову систему з аплікацією. Користувачу потрібно мати власний Aws аккаунт і вказати secret_key та access_key при старті платформи, а також змінні оточення, такі як регіон. Як результат обох способів в нас стає доступен UI інтерфейс Дженкінсу, де потрібно запустити стартовий Pipeline для створення

і налаштування службових ресурсів в нашому Aws аккаунті, таких як IAM user, role, s3 bucket для зберігання Terraform state тощо.

2.1 Структура навчальної платформи

Код навчальної платформи добре стандартизований. В головному каталозі ми маємо службові файли, документацію, змінні, та підкаталог із завданнями. В цьому каталозі лежать усі доступні задачі (рис 2.3).

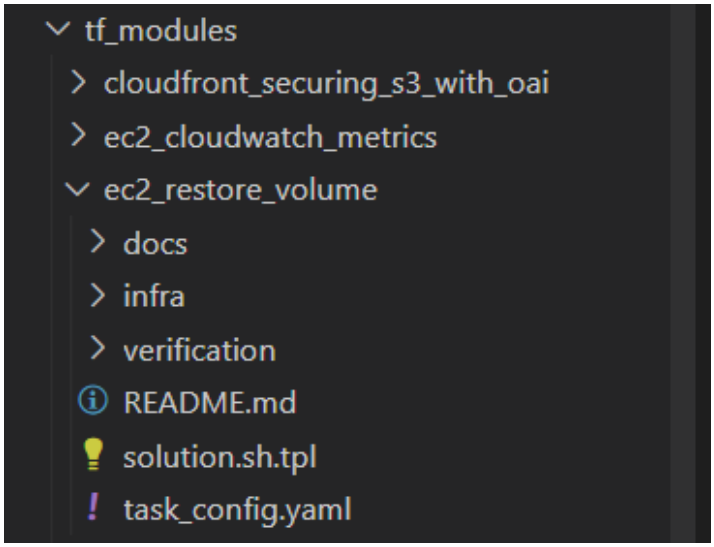

Рисунок 2.3 – Приклад каталогу завдань

По суті кожне завдання - це Terraform модуль, який виконується під час Jenkins пайплайну і виконують наступні функції:

1. Створює необхідну інфраструктуру та налаштування в AWS
2. Після виконання задачі студентом запускає скріпти для валідації правильності виконання.
3. Видає результат проходження.
4. Видаляє створені ресурси в AWS.

Процес проходження типового завдання виглядає наступним чином (рис 2.4):

1. Студент у Jenkins запускає пайплайн, та обирає необхідне завдання.

2. Починається створення інфраструктури та всіх ресурсів для вирішення завдання. Після цього пайплайн призупиняється і чекає не підтвердження користувача.

3. Студент приступає к вирішенню задачі.

4. Після вирішення учень продовжує пайплайн, що в свою чергу запускає валідаційні скрипти для перевірки правильності виконання. Зазвичай це або Bash скріпти, або Aws Cli команди, або парсинг CloudTrail логів в залежності від завдання.

5. Після валідації приходить результат.

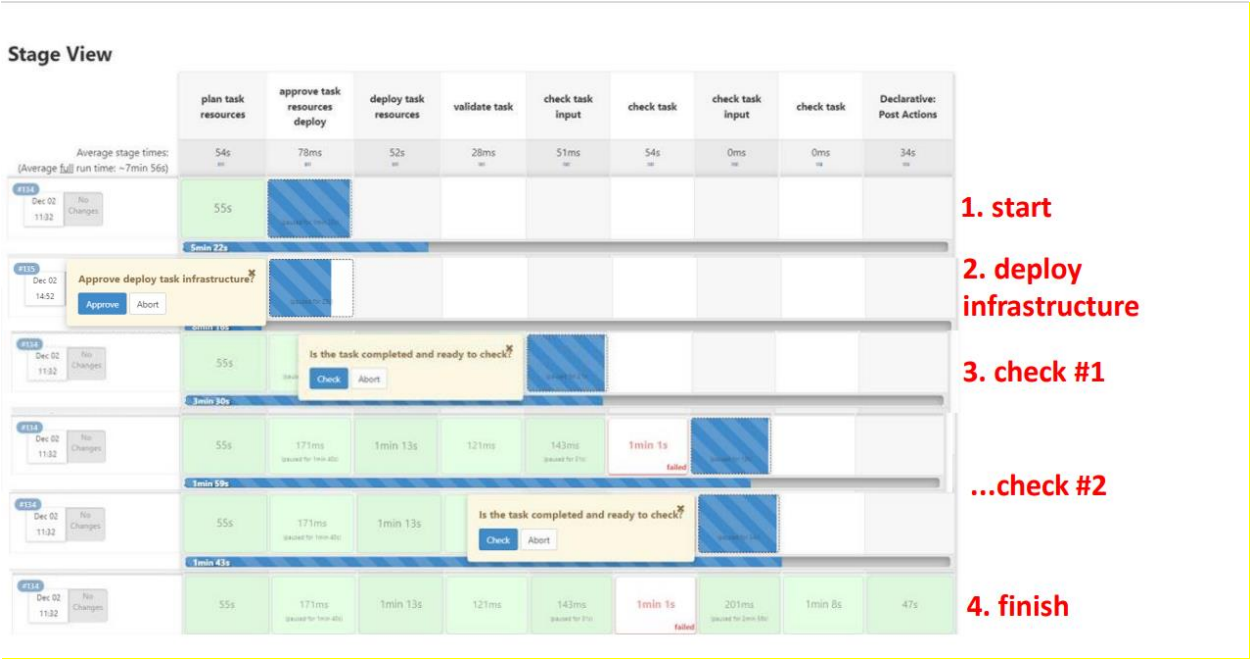

Рисунок 2.4 – Pipeline steps

3 РОЗРОБКА КОНЦЕПЦІЇ ПРАКТИЧНИХ ЗАВДАНЬ ДЛЯ ОЦІНКИ НАВИЧОК ВИКОРИСТАННЯ ДЕКІЛЬКОХ СЕРВІСІВ

3.1 Архітектура задачі

Щоб краще зрозуміти суть завдання треба познайомитись з безсерверними обчисленнями або serverless.

Завдяки безсерверним обчисленням ви можете створювати та запускати програми та служби, не думаючи про сервери. Безсерверні програми не потребують надання, масштабування та керування будь-якою інфраструктурою – все це робить хмарний провайдер. Ви можете створювати їх майже для будь-якого типу додатків або серверних служб, і все, що потрібно для запуску та масштабування вашої програми з високою доступністю, буде виконано за вас.

Навіщо використовувати безсерверні обчислення?

Завдяки безсерверним програмам ваші розробники можуть зосередитися на своєму основному продукті замість того, щоб турбуватися про керування та експлуатацію серверів або середовища виконання. Завдяки цьому зменшені накладні витрати розробники можуть повернути час і енергію, які можна витратити на розробку чудових продуктів, які масштабуються та є надійними.

За допомогою Lambda ви можете запускати код без підготовки та керування серверами. Служба запускає ваш код у обчислювальній інфраструктурі високої доступності та виконує адміністрування ресурсів. Це включає:

- Обслуговування сервера та ОС
- Надання ємності та автоматичне масштабування
- Моніторинг і журналювання коду

Усе, що вам потрібно зробити, це надати свій код на одній із мов, які підтримує Lambda — наразі це Node.js, Java, C#, Python, Go, Ruby та PowerShell.

Основними компонентами Lambda є джерело подій і функція Lambda. Джерела подій публікують події. Лямбда-функція — це спеціальний код, який ви пишете для обробки подій. Lambda запускає вашу функцію.

Функція Lambda складається з вашого коду, пов’язаних залежностей і конфігурації. Конфігурація містить таку інформацію, як:

• Обробник, який отримає подію

• Роль AWS Identity and Access Management (IAM), яку Lambda може виконувати для запуску функції Lambda від вашого імені

• Обчислювальний ресурс, який ви бажаєте виділити

• Час очікування доставки

Немає додаткової плати за створення лямбда-функцій. За роботу функції та передачу даних між Lambda та іншими службами AWS стягується плата. За деякі додаткові функції Lambda, наприклад, передбачений паралелізм, також стягується плата.

Функцію Lambda можна викликати вручну або за допомогою служби AWS.

3.2 Завдання event-driven file processing

Суть цього завдання налаштувати за допомогою безсерверних сервісів архітектуру, яка буде автоматично обробляти файли, завантажені користувачами у централізоване сховище. Це рішення повинно мати можливість працювати з величезною кількістю одночасних завантажень, та мінімізувати імовірність втрати даних. Для успішного проходження користувач повинен вже бути знайомий з декількома сервісами AWS, та розуміти як їх можна інтегрувати між собою (рис 3.1).

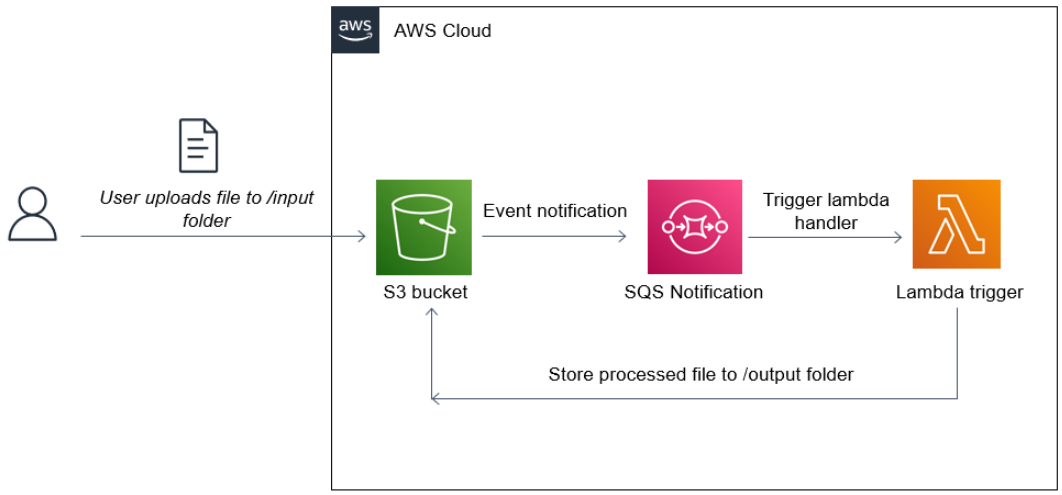

Рисунок 3.1 – Архітектура рішення задачі

Основні ресурси, які задіяні у цьому завданні:

S3 (Simple Storage Service) – це сервіс об’єктного сховища даних, в даному випадку це буде наш файлообмінник. Тут користувач, або додаток буде завантажувати файли, та мати доступ до оброблених даних.

SQS (Simple Queue Service) – сервіс черг повідомлень. Додавання файлів буде створювати повідомлення з даними об’єкту та кидати їх у чергу. Це дозолить синхронно виконувати Lambda функцію, що більш надійно, та налаштувати DLQ (Dead letter queue), щоб уникнути помилок та втрат даних.

Lambda – сервіс безсерверних обчислень. Функція на Python яка зменшує розмір файлу (зображення).

Коли студент запускає завдання, в його акаунті створюються s3 bucket, sqs черга та Lambda функція із Python кодом для обробки файлів.

Задача – налаштувати взаємодію між цими сервісами для задоволення бізнес сценарію.

Щоб налаштувати взаємодію між цими сервісами для задоволення бізнес сценарію потрібно:

• Створити в s3 bucket так звані event notifications - повідомлення про події в бакеті, такі як додавання файлів. Ці повідомлення ми будемо надсилати до SQS черги, і звідти обробляти їх далі (рис 3.2).

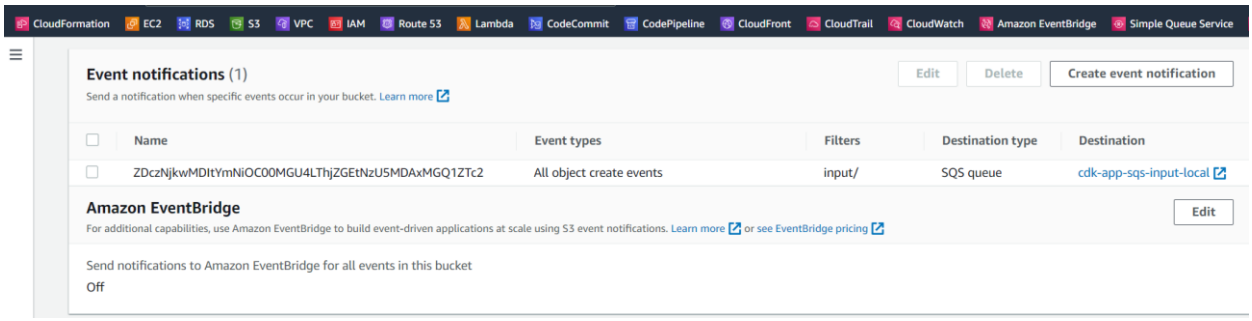

Рисунок 3.2 – S3 event notification

• Налаштувати Lambda функцію, щоб вона виконувалась кожен раз коли у черзі з’являються повідомлення (після додавання файлів у бакет). При спрацьовуванні функція зменшить розмір доданого файлу і перемістить його в інший каталог (рис 3.3).

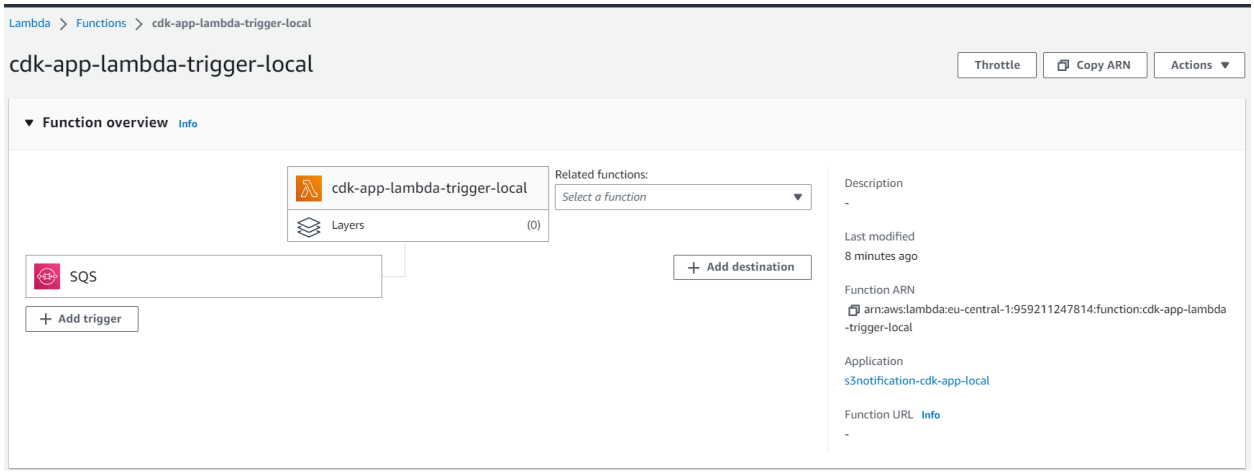

Рисунок 3.3 – Lambda as SQS Trigger

Якщо все зроблено вірно – то при завантаженні файлу у бакет – запуститься увесь процес обробки і за декілька секунд ми побачимо оброблений файл в output/ каталозі.

Валідація виконується за допомогою aws cli команд, які перевіряють, чи є в s3 бакета налаштована event notifications, далі перевіряє лямбда функцію чи є в неї тригер, і на останнє перевіряється наявність оброблених файлів в /output каталозі (рис 3.4).

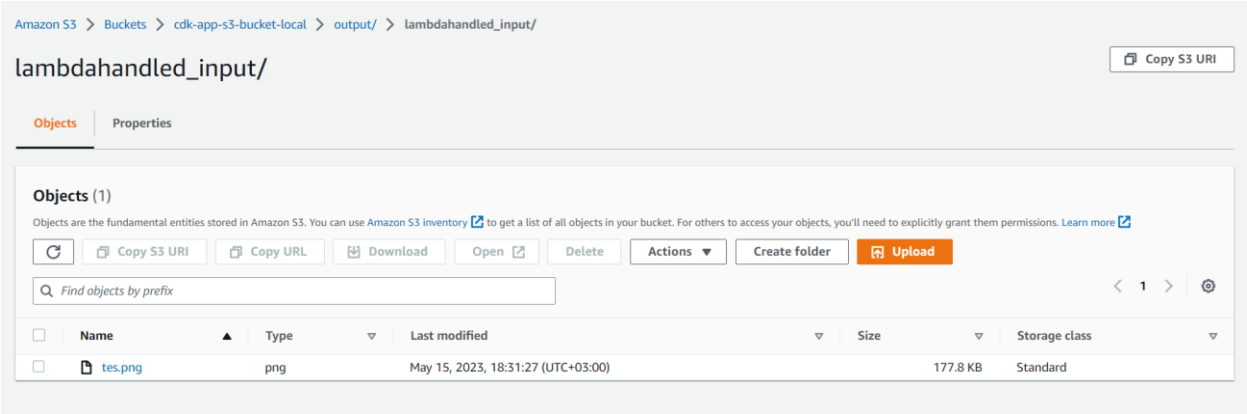

Рисунок 3.4 – Оброблений файл

ВИСНОВКИ

У цьому докладі ми розглянули ключові концепції хмарних технологій, та основні веб сервіси провайдера AWS. Також зазирнули у технологію serverless і описали типовий бізнес сценарій її використання. Оскільки хмарні провайдери стають все популярніше, та мають багато переваг з традиційним підходом, існує необхідність у якісних спеціалістах та навчальних програмах, таких як Cloud Mentor.

Була розроблена задача для студентів на налаштування serverless архітектури та взаємодії з декількома ключовими сервісами.

Можна зробити висновок, що розроблена платформа та її практичні завдання можуть бути використані для навчання студентів та IT фахівців різного рівня досвіду, система добре себе показала та значно покращила процес навчання, спрямований на розвиток практичних навичок студентів. Результати дослідження можуть бути корисними для педагогів, які займаються підготовкою фахівців з хмарних технологій. Також на платформі є можливість додавати нові завдання.

Виконання послідовності таких завдань з використанням розробленої платформи дає можливість студенту опанувати необхідний спектр хмарних сервісів, а викладачу зменшити час на контроль проходження студентом необхідних завдань. Гнучкість платформи дозволяє додавати нові завдання та створювати набори завдань різної складності та різної спеціалізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Architecting on AWS - Online Course Supplement [вебсайт]: https://explore.skillbuilder.aws/learn/course/external/view/elearning/8319/architecting-on-aws-online-course-supplement
2. Aws documentation [вебсайт]: https://docs.aws.amazon.com/
3. Aws Whitepapers [вебсайт]: https://aws.amazon.com/whitepapers/
4. Aws Faqs [вебсайт]: https://aws.amazon.com/faqs/
5. Well-Architected Framework [вебсайт]: https://aws.amazon.com/architecture/well-architected/
6. Прокопенко Р. О. “Розробка концепції комплексних завдань для автоматизованої перевірки практичних навичок роботи з AWS” / Р. О. Прокопенко, А. І. Костромицький. // Тези доповідей V Международная научно-практическая конференция “INNOVATIONS AND PROSPECTS IN MODERN SCIENCE”, 8-10 мая 2023 года Стокгольм, Швеция 2023. – P. 33 Стр. 157. https://sci-conf.com.ua/wp-content/uploads/2023/05/INNOVATIONS-AND-PROSPECTS-IN-MODERN-SCIENCE-8-10.05.23.pdf