Харківський національний університет радіоелектроніки
(повне найменування вищого навчального закладу)

Факультет *інфокомунікацій*
(повна назва)

Кафедра *інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

на тему *Система автоматичної перевірки навичок роботи з AWS. Підтема 2. Робота зі сховищами даних*

Виконав: студент *2* курсу, групи *ІМІм-21-1*
напряму підготовки
172 "Телекомунікації і радіотехніка"
(шифр і назва напряму підготовки)
Лещенко М.Р.
(прізвище та ініціали)
Керівник *Костромицький А.І.*
(прізвище та ініціали)

Допускається до захисту

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

Харків - 2022 рок

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)

"____" _____ 2022 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Лещенку Микиті Руслановичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Система автоматичної перевірки навичок роботи з AWS. Підтема 2. Робота зі сховищами даних*

затверджені наказом ВНЗ від "*21*" *жовтня* 2022 року № *1376 Ст.*

2. Строк подання студентом роботи *10 грудня 2022 року*

3. Вихідні дані до роботи *Розробити концепції завдань для роботи зі сховищами даних для системи автоматичної перевірки навичок на прикладі одного із завдань показати роботу системи*

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

Вступ

1. Сучасні хмарні технології

2. Система автоматичної перевірки навичок роботи з AWS

3. Розробка концепцій завдань для перевірки навичок роботи зі сховищами даних

Висновки

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

Слайди у форматі Power Point (назва та мета роботи, актуальність, система перевірки навичок, типи сховищ, концепції завдань, висновки)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	*доц. Костромицький А.І.*	21.10.22	

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	*21.10.22*	
2	*Підбір літератури за темою роботи.*	*23.10-30.10.22*	
3	*Виконання розділу 1*	*30.10-05.11.22*	
4	*Виконання розділу 2*	*05.11-10.11.22*	
5	*Виконання розділу 3*	*10.11-15.11.22*	
6	*Оформлення презентаційного матеріалу,*	*18.12-19.12.22*	
	підготовка до захисту у ЕК		

Дата видачі завдання *21* *жовтня* *2022 р.*

Студент _____ *Лещенко М.Р.*
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Пояснювальна записка: 45 с., 11 рис., 1 табл., 10 джерел

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

Мета роботи – розробка концепції завдань для перевірки навичок роботи зі сховищами даних.

Розглянута система автоматичної перевірки роботи навичок з AWS. Розроблені концепції завдань роботи з різними сховищами даних (EBS, EFS та S3) і продемонстровано роботу тестового завдання роботи з EBS дисками, яка була створена для демонстрації можливостей системи.

AWS, TERRAFORM, PYTHON, JENKINS, EBS, S3, СИСТЕМА АВТОМАТИЧНОЇ ПЕРЕВІРКИ

ABSTRACT

Explanatory note: 45 p., 11 fig., 1 tab., 10 sources.

The object of study is an automatic AWS skill testing system.

The purpose of this work is to develop tasks for automatic testing system with an overall theme of file storage services of AWS

In this paper, we have examined an automatic AWS skill testing system. We have developed a list of tasks conceptual tasks that help understand how are the AWS file storage services (EBS, EFS and S3) work. The demo task that is used to demonstrate the capabilities of the automatic testing system was also developed.

AWS, TERRAFORM, PYTHON, JENKINS, EBS, S3, AUTOMATIC TESTING SYSTEM

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AWS (Amazon Web Services) – хмарні сервіси компанії Amazon;

IaC (Infrastructure as Code) – підхід до розробки хмарної інфраструктури;

EBS (Elastic Block Storage) – сервіс хмарних дисків;

EC2 (Elastic Compute Cloud) – сервіс віртуальних машин;

S3 (Simple Storage Service) – сервіс зберігання статичних файлів;

IT (Information Technology) – інформаційні технології;

EFS (Elastic File System) – розподілена файлова система;

RDS (Relational Database Service) – реляційна база даних;

CI/CD (Continuous Integration/Continuous Delivery) – підхід до побудування процесу розробки;

CLI (Command-line Interface) – інтерфейс командного рядка

ВСТУП

В полі інформаційних технологій і хмарових обчислень дуже важливо мати механізми збереження і відновлення критичних бізнес даних, бо не завжди можна передбачити можливі збої в роботі системи або вплив зовнішніх факторів [1]. Аналіз аварійних ситуацій сам по собі є важливою частиною організації процесів будь-якого бізнеса. Маючи ці дані, компанія зможе задокументувати загрози, порахувати ризики і вартість різних сценаріїв аварійного копіювання та відповідні варіанти відновлення.

Як сама популярна хмарна платформа, Amazon Web Services має велику кількість інструментів роботи з disaster recovery і забезпеченням відмовостійкості [2].

Також завжди є необхідність відпрацьовувати навички роботи з хмарами і їх інструментами роботи з даними, так як очевидна тенденція міграції більшості європейських бізнесів до інфраструктури, що пропонують хмарні провайдери. Опитування серед європейських компаній (Великобританія, Німеччина, Франція, Нідерланди та ін.) у 2021 році (рис. 1) показало, що одним із найголовніших пріоритетів є міграція у хмару більшого робочого навантаження (70%), оптимізація витрат у хмарних сервісах (59%) і розробка cloud-first підходу, тобто орієнтування на розробку у хмарах (50%).

Рисунок 1 – Пріоритети по впровадженню хмар на 2021 рік

Таким чином, на фоні постійно зростаючої популярності хмарних сервісів і бізнес потреб переходу на ці сервіси, кількість спеціалістів, які працюють з AWS і іншими хмарними провайдерами, також зростає. Це можна побачити як і в зростанні залежності використання хмарних сервісів і ефективності європейських бізнесів [4], так і в загальному рості ринка і кількості фінансів в цьому секторі [5]. На рис. 2 зображено ріст американського хмарного ринку з 2018 року з тенденцією росту до 2029 року.

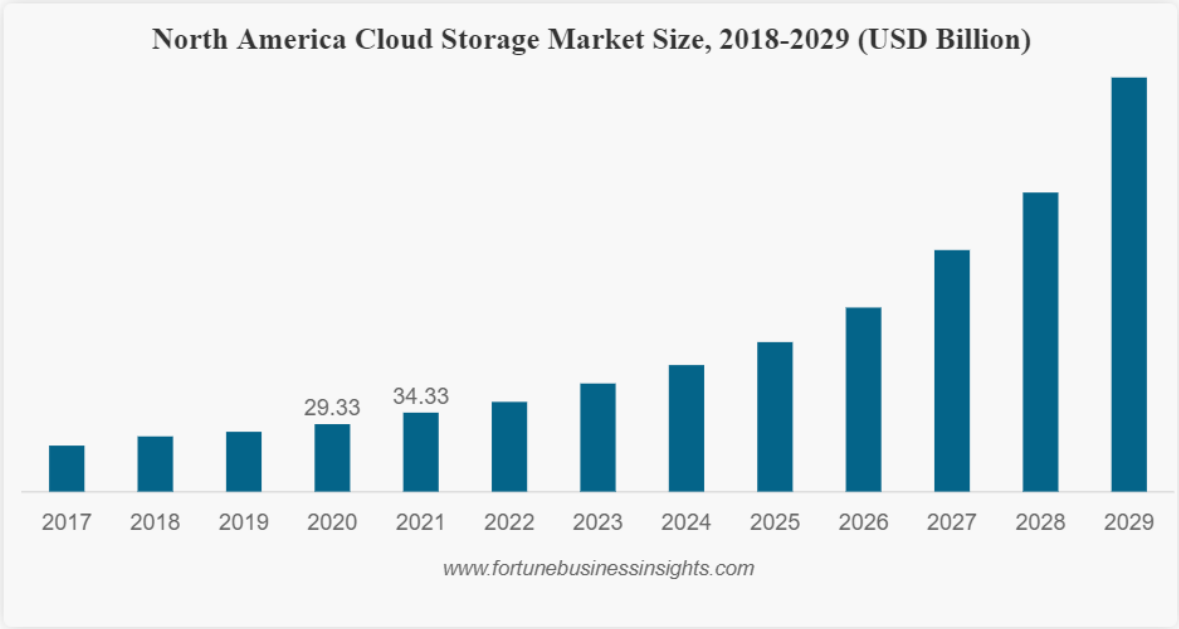


Рисунок 2 – Зріст ринку хмарних сервісів в США

Зі статистичних викладок можна зрозуміти, що існує велика необхідність розробки завдань для навчання інженерів роботі з хмарними сервісами, зокрема роботі з хмарними сховищами даних і дисками віртуальних машин.

1 СУЧАСНІ ХМАРНІ ТЕХНОЛОГІЇ

1.1 Тенденція переходу бізнесів на хмарну інфраструктуру і сервіси

Зростання ринка сучасних хмарних технологій і популяризації інструментів роботи з ними показує, що перехід великих і малих бізнесів до хмари є одним із самих актуальних питань IT-індустрії. Абсолютна більшість компаній в країнах ЄС з 2020 по 2021 рік збільшила процент користування хмарними провайдерами [6]. На рис. 1.1 можна побачити статистику по країнам Європи.

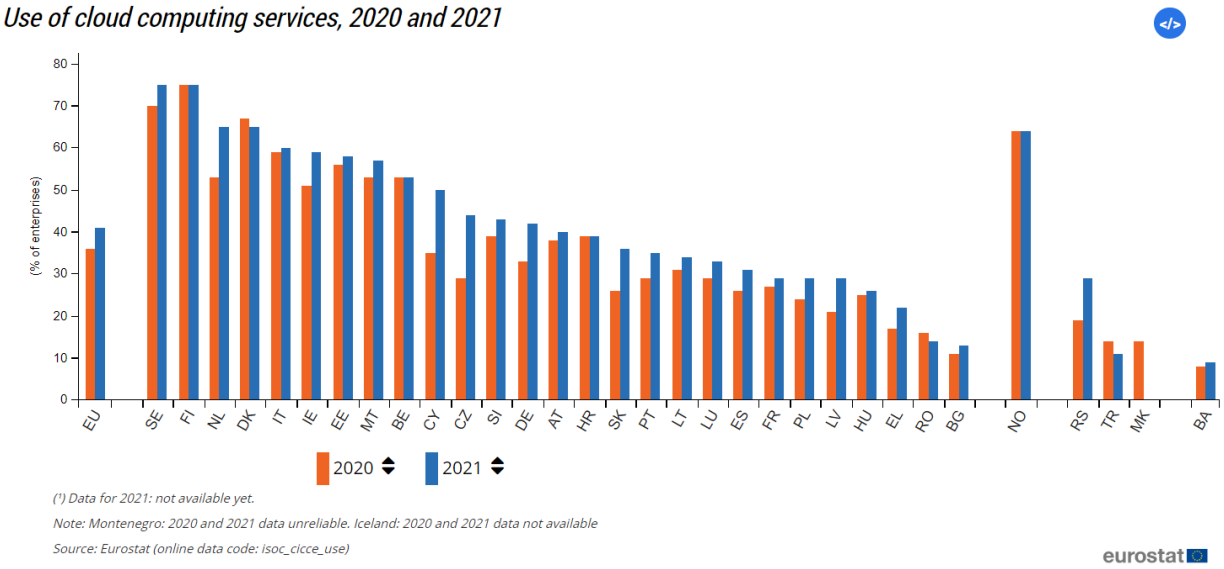


Рисунок 1.1 – Використання хмарних сервісів в Європі

Amazon Web Services, що є найпопулярнішим хмарним провайдером (32% від всього ринку хмарних обчислень), надає сервіси і платформу багатьом крупним, середнім і малим бізнесам, що, в свою чергу, дає можливість компаніям збільшити ефективність своєї роботи і зменшити витрати. Як приклад, компанія DCI (Digital Commerce Intelligence), яка

базується в Сінгапурі і зайнята сбіром та аналізом бізнес статистики, за допомогою міграції до середовища AWS і користування дата сервісом S3, зменшила витрати на хмарну інфраструктуру в місяць на 27 відсотків [7].

Використання сервісів Amazon не тільки допомагає зменшити трати на інфраструктуру, а і робить можливим імплементацію нових підходів до створення і аналізу бізнес продуктів. Так як робота фокусується на розгляді сховищ даних для AWS, приведемо приклади того, як ці технології допомогли реалізувати ідеї в компанії ŠKODA, а також розглянемо сферу навчання на прикладі Каліфорнійського технологічного інституту.

Для ŠKODA була актуальна задача забезпечення безперебійного моніторинга за конвеєром, що збирав автомобілі. Це було зроблено для того, щоб забезпечити своєчасне виявлення збоїв і їх ліквідацію. Для цього було розроблена система моніторингу Magic Eye, основою якою було використання сервісів AWS. Magic Eye використовує шість камер, встановлених на конвеєрній рамі, для моніторингу обладнання та доступу до місць, куди оператори не можуть легко потрапити. За той час, який потрібен автомобілю для проходження виробничої лінії ŠKODA, ці камери збирають майже 450 000 фотографій. Камери підключаються до потужного комп’ютера на конвеєрі, де 10 штучних нейронних мереж збирають і аналізують фотографії. Результати надсилаються безпосередньо в хмару та зберігаються за допомогою Amazon Simple Storage Service (Amazon S3), об’єктного сховища, створеного для отримання будь-якої кількості даних з будь-якого місця.

Для обчислень і масштабування Magic Eye використовує Amazon Elastic Compute Cloud (Amazon EC2), яка пропонує безпечну обчислювальну потужність зі змінним розміром практично для будь-якого робочого навантаження. Рішення ŠKODA також використовує додаткові сервіси AWS, такі як Amazon Relational Database Service (Amazon RDS), який надає користувачам можливість налаштувати, керувати та масштабувати реляційну базу даних [8].

Каліфорнійський технологічний інститут вже користувався послугами Amazon і проводив імплементацію системи сховища даних EFS для зберігання конфігурації і образів своїх контейнерних додатків. Amazon EFS надав команді Caltech централізовану файлову систему, яка є надійною та легкою в управлінні, що дозволяло команді університета швидше реагувати на потреби своїх клієнтів. Команда могла налаштувати нове середовище додатків за дві години замість двох днів. Наприклад, команда швидко відреагувала, коли дослідникам потрібно було створити новий веб-сайт, який оголошував про збір пожертв для університета Резніка [9].

1.2 Підхід інфраструктури як коду для створення хмарної інфраструктури

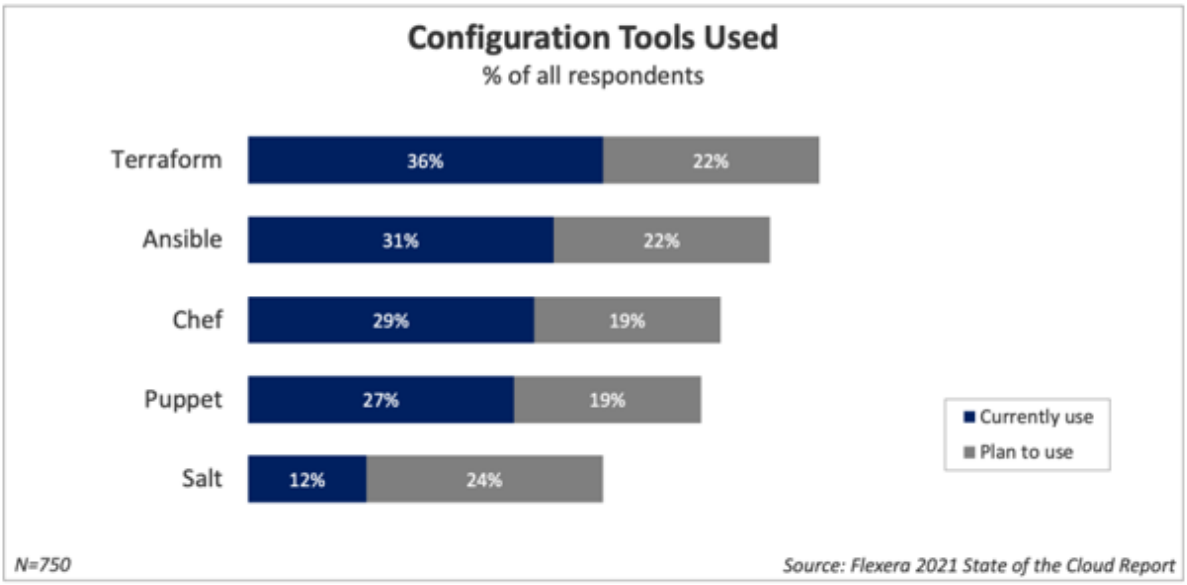
Зі збільшенням кількості бізнесів, які користуються послугами хмарних провайдерів, також з’явилася потреба в інструментах простої і надійної конфігурації хмарної інфраструктури. В результаті був розроблений підхід IaC (Інфраструктура як код), на основі якого створювалися інструменти конфігурації хмарної інфраструктури. Мета підходу – зберігати архітектуру хмарного проекта у вигляді написаної конфігурації, яка працює по принципу «ідемпотентності» – тобто не важливо скільки буде ця конфігурація застосована, вихідний результат завжди буде однаковий, що в свою чергу означає, що неважливо, з якого пристрою запускається конфігурація чи з якої операційної системи, що надає стабільність і універсальність конфігурації.

Без IaC ручне керування інфраструктурою або її розгортання є повільним процесом. Якщо є необхідність внести зміну інфраструктури визначено через якусь проблему, системному інженеру може знадобитися велика кількість часу, щоб відреагувати та застосувати зміни. Це призводить до збоїв і розчарованих клієнтів. Завдяки IaC інфраструктура може автоматично адаптуватися до змін у конфігурації (в парній роботі з CI/CD) та

реагувати на стрибки трафіку за допомогою функцій автоматичного масштабування.

Інфраструктура як код дає більше контролю та видимості адмініструванню систем. З файлами конфігурації інфраструктури, закріпленими в центральному репозиторії контролю версій, усі члени команди можуть переглядати та редагувати дані інфраструктури. Це забезпечує потужні можливості аудиту.

Згідно статистики, застосування таких інструментів збільшилося, і самим популярним є Terraform від HashiCorp, який є клауд-агностичним, гнучким інструментам для роботи з усіма популярними хмарними



оточеннями.

Рисунок 1.2 – Інструменти конфігурації

Саме через універсальність Terraform в цій роботі використовується саме він, не зважаючи на те, що Amazon має свій інструмент Amazon CloudFormation.

1.3 Швидка розробка за допомогою CI/CD

Ще більшої ефективності і гнучкості розробці сучаних хмарних додатків і сервісів дає підхід CI/CD, що означає неперервну інтеграцію коду за допомогою частих змін до репозиторія проекту і їх автоматичне тестування, після чого по неперервну доставку змін до середовища розробки і продакшена.

З основних переваг такого підходу можна виділити:

1) Організації, які створюють пайплайни CI/CD, можуть швидше випускати код. Завдяки стандартизації збірок, розробці тестів і автоматизації розгортань команди можуть приділяти більше часу вдосконаленню програм і менше часу на технічні процеси доставки коду в різні середовища;

2) Оскільки розробники, які частіше приймають код CI/CD, команди можуть швидко виявити проблеми з якістю за допомогою перевірки менших пакетів коду, а не більших, створених пізніше за графіком проекту;

3) CI/CD допомагає контролювати використання програм. Це також дозволяє командам постійно вдосконалюватись, наприклад змінювати досвід користувача та додавати посібники в програмі, щоб заохочувати користувачів використовувати програму та її функції.

4) Можливість інтегрувати розробників і процеси за допомогою CI/CD може підвищити продуктивність і співпрацю між командами, що працюють над проектом, особливо якщо ці команди розкидані територіально.

CI/CD – одна із самих широко застосованих практик серед розробки сучасного програмного забезпечення. За репортом за перший квартал 2022 рок від Continuous Delivery Foundation побачили значне зростання впровадження технологій CI/CD (зростання на 26% прийняття) та «інфраструктури як код» (зростання на 27% прийняття). Також було відзначено, що користувачі інструментів CI/CD з більшою ймовірністю будуть найефективнішими за характеристиками частоти виконання змін в кодовій базі (користувачі інструментів CI/CD є більш ефективними ніж користувачі інших технології на 22%) і часу для відновлення сервісів

(користувачі інструментів CI/CD є більш ефективними ніж користувачі інших технології на 21%) [10].

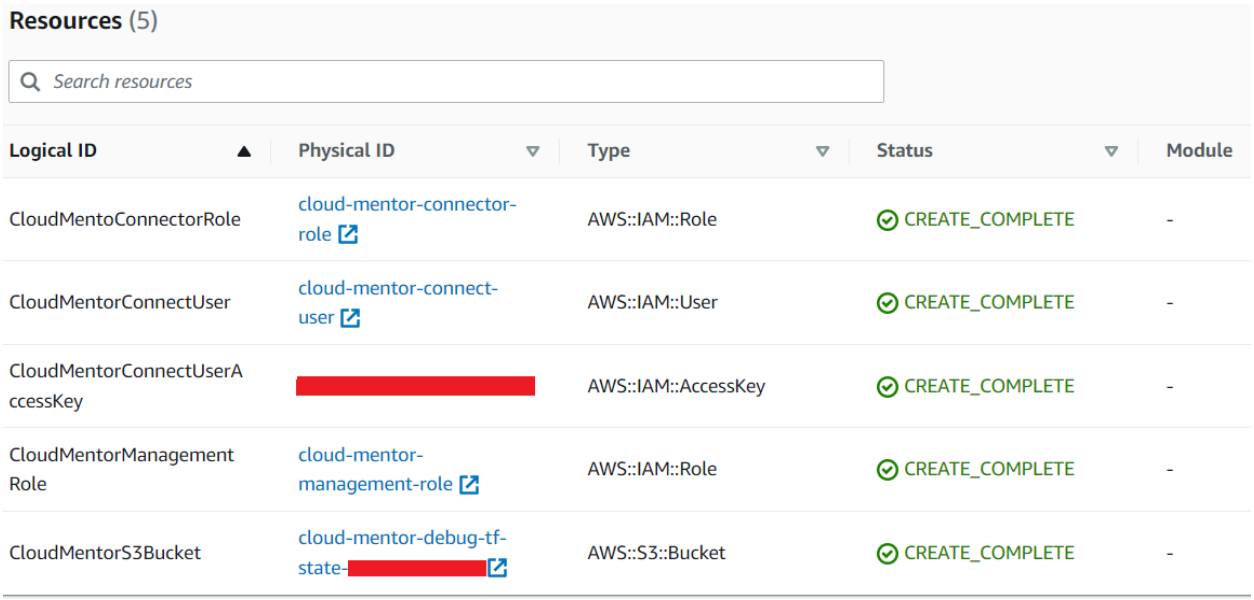
2 СИСТЕМА АВТОМАТИЧНОЇ ПЕРЕВІРКИ НАВИЧОК РОБОТИ З AWS

2.1 Концепція платформи

Враховуючи всю актуальність хмарних технологій і потребу в навчанні спеціалістів необхідним навичкам, за основу роботи взято систему автоматичної перевірки навичок роботи з самим популярним хмарним провайдером AWS [11]. На платформі системи, яка сама побудована за принципами CI/CD (імплементована за допомогою Jenkins) та IaC (використовується Terraform), розробляються завдання для студентів. Завдання фокусуються на різних темах, таких як: налаштування віртуальних машин EC2, робота зі сховищами даних EBS, EFS та S3, налаштування доступу до AWS аккаунта за допомогою IAM та ін. Головні особливості системи: легкість в користуванні, гнучкий підхід до створення своїх завдань і можливість як ручної (в версії для розробки), так і автоматичної перевірки завдань.

Принцип роботи системи буде описаний на прикладі локальної версії, яка встановлена для розробки завдань. Це дасть можливість більш детально розібрати архітектуру проекту, описати процес встановлення і налаштування середовища і показати структуру кодової бази завдань.

Встановити систему для розробки можна двома способами – використати Docker Compose або Vagrant від Hashicorp. Перш за все створюються ресурси CloudFormation, які зв’язують аккаунт розробника з локальною системою через створені IAM ролі, які забезпечують доступ до AWS аккаунта, а також створюється S3 бакет, що зберігає Terraform State (список всіх ресурсів, за які відповідає Terraform). Також створюється пара ключів, які дають можливість підключатися до аккаунта напряму.



Resources (5)

Search resources

Logical ID	Physical ID	Type	Status	Module
CloudMentoConnectorRole	cloud-mentor-connector-role	AWS::IAM::Role	CREATE_COMPLETE	-
CloudMentorConnectUser	cloud-mentor-connect-user	AWS::IAM::User	CREATE_COMPLETE	-
CloudMentorConnectUserAccessKey	[redacted]	AWS::IAM::AccessKey	CREATE_COMPLETE	-
CloudMentorManagementRole	cloud-mentor-management-role	AWS::IAM::Role	CREATE_COMPLETE	-
CloudMentorS3Bucket	cloud-mentor-debug-tf-state-[redacted]	AWS::S3::Bucket	CREATE_COMPLETE	-

Рисунок 2.1 – Ресурси CloudFormation

Далі встановлення для двох основних способів відрізняється. Для роботи в якості приклада був обраний спосіб встановлення через Docker Compose, що має один центральний файл конфігурації, який описує, які контейнери потрібно створити. В нашому випадку контейнера три: Jenkins, локальна база даних DynamoDB і API для роботи з базою даних, яке потрібно для створення користувача системи.

Перед створення контейнерів також необхідно локально загрузити Git репозиторій з кодом проекта і налаштувати AWS Credentials для свого пристрою. До AWS Credentials додаються ключи, які були створення в стеку CloudFormation:

[cloud-mentor-connect-user]

aws_access_key_id = <aws_access_key_id>

aws_secret_access_key = <aws_secret_access_key>

[profile cloud_mentor_debug]

role_arn = arn:aws:iam::<your AWS account ID>:role/cloud-mentor-management-role

source_profile = cloud-mentor-connect-user

Після цього потрібно експортувати перелік змінних оточення, серед яких: регіон AWS, IAM профіль системи, ім’я S3 бакета, де буде зберігатися Terraform State, і група Docker, в якій знаходиться користувач Jenkins. Після цього виконуються команди по запуску Jenkins:

docker-compose -f docker-compose-local.yaml build

docker-compose -f docker-compose-local.yaml up

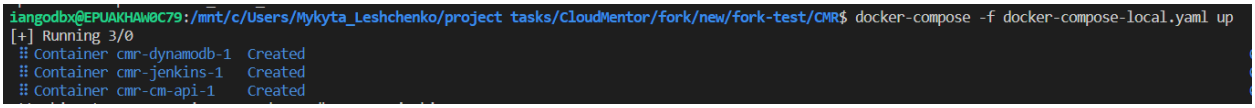


Рисунок 2.2 – Створення контейнерів системи

Після створення оточення для розробки необхідно створити юзера студента, від імені якого будуть запускатися завдання. Це можна зробити за допомогою API, яку було створено раніше. Створюється група і юзер.

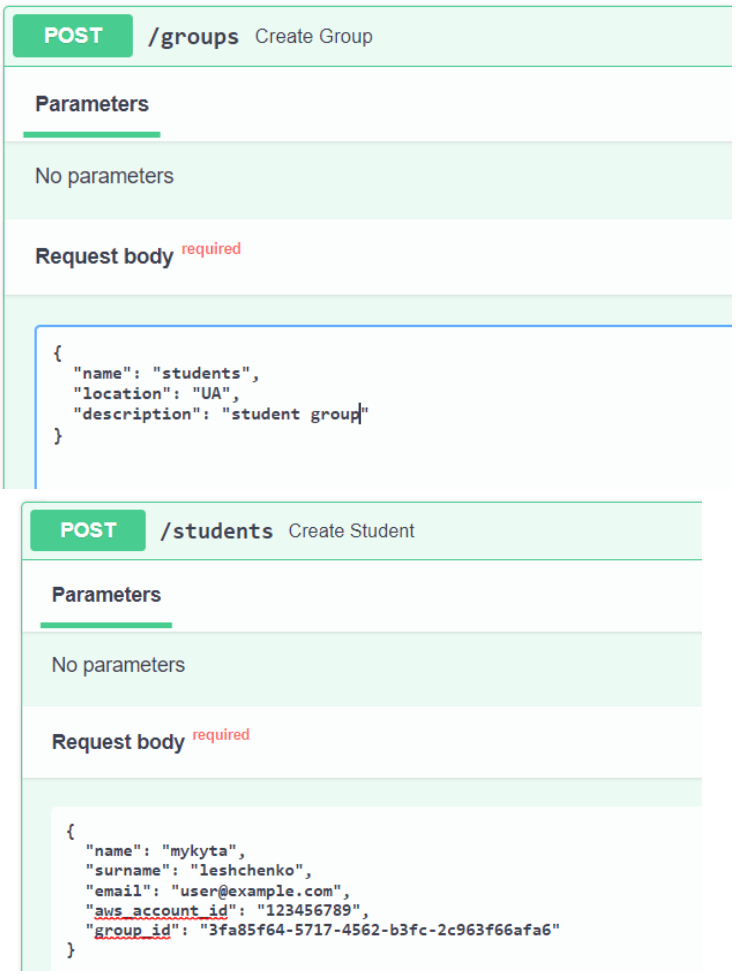


Рисунок 2.3 – Створення групи і юзера

Встановлення завершується створенням папки користувача в Jenkins шляхом вказання ID студента в службовому пайплайні *create_user_workspace*. В пайплайні *task_pipeline* проходить процес виконання завдань.

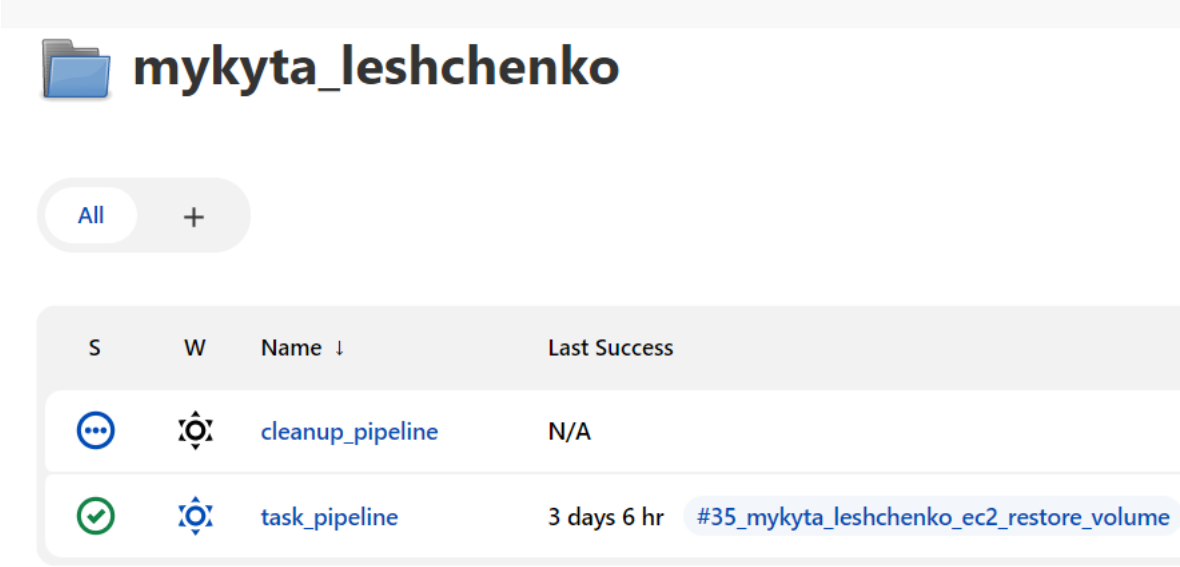


Рисунок 2.4 – Папка юзера

2.2 Загальний підхід до розгортання інфраструктури і перевірки завдань

Кожне завдання в системі має загальну структуру кодової бази, яку можна розділити на три частини:

1) Загальні модулі для роботи з системою
2) Інфраструктура
3) Код для перевірки завдання

Загальні модулі використовуються для двох речей: стандартизація в іменуванні завдань і ресурсів, які будуть створюватися при виконанні завдання, і запис логів для подальшої валідації завдання.

Інфраструктура і код для перевірки завдань знаходиться в спільній директорії самого завдання у наступному вигляді: в директорії *tf_modules* знаходиться список всіх доступних для виконання завдань, в самій папці завдання – директорії *infra* і *verification*, які відповідно створені для кода інфраструктури і валідації завдання.

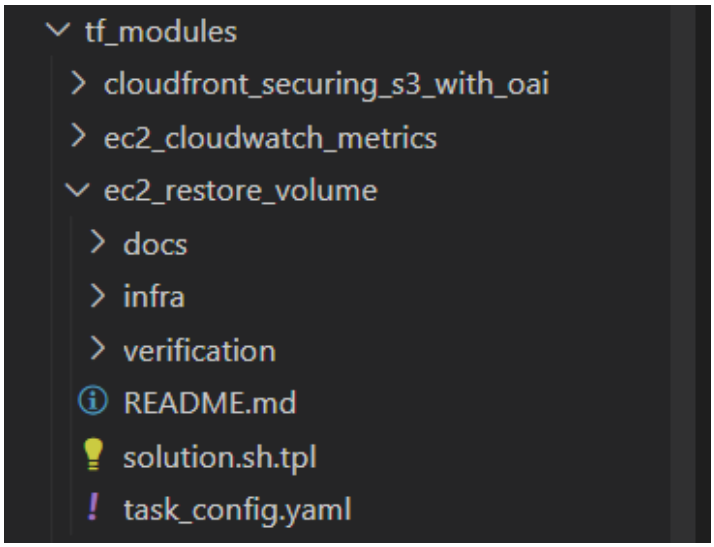


Рисунок 2.5 – Структура завдання

Створення ресурсів завдання відбувається шляхом ініціалізації Terraform директорії в папці з самим завданням, яке є набором Terraform файлів, логічно розбиті за функціоналом. Вони включають в себе головний файл з ресурсами завдання, конфігурація мережі, налаштування способу зберігання Terraform State файлу, змінні і вихідні дані. Файли валідації містять Bash-скрипт перевірки завдання, конфігурацію взаємодії зі спільним модулем верефікації, змінні і вихідні данні у вигляди логів, що відображають стан завдання (правильно виконане, чи неправильно).

Таким чином, процес проходження завдання має наступний вигляд:

1) Студент обирає завдання із списку доступних завдань в Jenkins;

2) Відбувається первинне розгортання інфраструктури, кінцевий результат якої є повністю робоче завдання (ці логи студент не бачить);

3) На другому етапі розгортання інфраструктури, виконується скрипт, який приводить частину сервісів до неробочого стану і призупиняє пайплайн з вказівками щодо суті завдання;

4) Студент має встановити причину збою в роботі сервісів і відновити їх працездатність;

5) Після виконання роботи система по логам CloudTrail перевіряє, чи були виконані всі вимогу завдання і чи сервіси працюють як треба (в тому числі перевіряється, працював студент із AWS CLI або ні);

6) Після успішного виконання завдання всі ресурси автоматично видаляються.

3 РОЗРОБКА КОНЦЕПЦІЙ ЗАВДАНЬ ДЛЯ ПЕРЕВІРКИ НАВИЧОК РОБОТИ ЗІ СХОВИЩАМИ ДАНИХ

Основними сервісами для сховища даних, що розглянуті в роботі є: EBS, EFS і S3. EBS є основним сервісом AWS для роботи з дисками віртуальних машин, EFS – розподілена файлова система, а S3 – середовище зберігання статичних файлів, медіа-контенту, статичних веб-сайтів і логів.

Всі сервіси мають своє призначення і особливості, які відображені в таблиці 3.1.

Таблиця 3.1 – Порівняння сховищ даних

Сервіс	Швидкість	Доступність	Контроль доступу	Ліміти по файлам	Ціна	Найкраще використання
EBS	HDD диски – 250/500 IOPS SDD – 16/64тис. IOPS	99.9%, доступний для однієї віртуальної машини	- Security groups - IAM	- 16 ТБ - Немає ліміта на розмір файлів	- 30 ГБ безкоштовно - $0.045 за ГБ/міс. HDD - $0.125 за ГБ/міс. SDD	Швидкісне сховище даних для EC2
EFS	Базова швидкість 3 ГБ/с Швидкість до 10 ГБ/с - До 7 тис. IOPS	- Доступність із будь-яких регіонів - До 1000 підключених EC2	- Security groups - IAM	- 16 ТБ на диск - 52 ГБ ліміт на розмір файлів	$0.30-$0.39 за ГБ в залежності від регіону	Спільне файлове середовище для EC2 з автоматичним скейлінгом
S3	- 3500 GET/PUT/запитів в секунду - Можливість розширити до 5000 GET запитів	99.9%, якщо нижче – повертається 10-90% від ціни - Доступне через API запити	- IAM - Bucket policy	- без лімітів по кількості файлів - розмір одного файла не більше 5 ТБ	- 5 ГБ безкоштовно - Перші 50 ТБ/міс - $0.023 - Більше 50 ГБ - $0.021 за ГБ	Система зберігання логів і архівних файлів, статичні веб-сайти

3.1 Типи сховищ даних

Amazon EBS використовується як диски віртуальних машин. Сервіс зберігає дані в блоках однакового розміру та організовує їх в ієрархію, подібну до традиційної файлової системи. Розмір томів надається за розміром і приєднується до EC2 у спосіб, подібний до локального диска на фізичній машині.

EFS є найкращим вибором для запуску будь-якого продукту, який має високе робоче навантаження, потребує масштабованого сховища та має швидко створювати вихідні дані. Він масштабується автоматично, це залежить від потреби в сховищі. Після закінчення періоду потреби в сховищі великого обсягу EFS автоматично зменшиться. EFS можна підключити до різних служб AWS і отримати доступ з віртуальних машин. Зазвичай, EFS використовується для запуску спільних томів або для аналізу великих даних.

Amazon S3 (Amazon Simple Storage Service) зберігає дані як об’єкти в плоскому середовищі (без ієрархії). Кожен об'єкт (файл) у сховищі містить заголовок із відповідною послідовністю байтів (від 0 байт до 5 ТБ). Об’єкти цього типу зберігання пов’язані з унікальним ідентифікатором (ключем), тому до них можна отримати доступ через веб-запити з будь-якого місця. Наприклад, будь-який авторизований вузол у вашому власному центрі обробки даних або зовнішній користувач може отримати доступ до будь-якого об’єкта у вашому сегменті.

3.2 Розробка завдання по роботі з EBS

Суть цього завдання полягає в відновленні доступа до веб-сервера шляхов відновлення диска, критичні бізнес дані якого були втрачені під час збою. Кінцевим результатом буде доступна веб-сторінка з повідомленням про роботу бізнес оточення.

Сам процес виконання завдання передбачає використання root restore task, який можна зробити як через графічний інтерфейс консолі AWS, так і через CLI консоль.

Розглянемо чотири основних ресурса, які беруть участь у створенні інфраструктури, а саме: віртуальна машина, скрипт, який чекає на доступність віртуальної машини, EBS снепшот робочого диска, скрипт, який симулює втрату важливих даних.

Віртуальна машина на базі Amazon Linux 2 – це наш основний веб-сервер. За основу веб-сервера взято Nginx:

```
resource "aws_instance" "root_ec2" {
  ami           = data.aws_ami.this.id
  instance_type = "t2.micro"
  key_name      = aws_key_pair.key_to_connect.key_name

  network_interface {
    network_interface_id = aws_network_interface.client.id
    device_index         = 0
  }

  user_data = templatefile("userdata/create_important_data.sh.tftpl", { folder_path = var.folder_path })

  provisioner "file" {
    source      = "userdata/rm_files.sh"
    destination = "/tmp/rm_files.sh"

    connection {
    host        = aws_instance.root_ec2.public_ip
    type        = "ssh"
```

```
    user        = "ec2-user"
    private_key = tls_private_key.exec_key.private_key_pem
    }
  }

  tags = {
    Name = "${module.naming.resource_prefix.ec2_instance}-restore"
  }
}
```

Як можна бачити з приведеного коду, віртуальна машина створюється і скрипт *create_important_data.sh.tftpl* встановлює веб-сервер і створює html сторінку, також в файлову систему машини загружається скрипт *rm_files.sh*, який у майбутньому буде використаний для видалення html сторінки.

Наступний ресурс – скрипт, який чекає на повну готовність віртуальної машини, щоб зробити повністю робочий снепшот root диска.

```
resource "null_resource" "wait_for_availability" {
  depends_on = [
    aws_instance.root_ec2
  ]
    provisioner "local-exec" {
    interpreter = ["/bin/bash", "-c"]
    command = <<-EOT
aws ec2 wait instance-status-ok --instance-ids ${aws_instance.root_ec2.id} --filters "Name=instance-status.reachability,Values=passed" && \
sleep 120
    EOT
  }
}
```

Після того, як віртуальна машина створилася і ми дочекалися встановлення веб-сервера, ресурс *aws_ebs_snapshot.backup_root* робить копію root диска і зберігає її у вигляді снепшота.

```
resource "aws_ebs_snapshot" "backup_root" {
  depends_on = [
    aws_instance.root_ec2,
    null_resource.wait_for_availability
  ]
  volume_id = data.aws_ebs_volume.ebs_volume.id

  tags = {
    Name = "${module.naming.resource_prefix.ebs_snapshot}-restore"
  }
}
```

Останній основний ресурс, що використовується при створенні інфраструктури виконується після того, як всі ресурси для отримання робочого завдання готові, і модулює збій в системі, який призводить до втрати бізнес даних.

```
resource "null_resource" "remove_important_files" {
  count = var.deploy_task_resources ? 0 : 1
  depends_on = [
    aws_ebs_snapshot.backup_root,
    aws_instance.root_ec2,
    null_resource.wait_for_availability
  ]
  connection {
    host        = aws_instance.root_ec2.public_ip
    type        = "ssh"
    user        = "ec2-user"
    private_key = tls_private_key.exec_key.private_key_pem
```

```
  }

  provisioner "remote-exec" {
    inline = [
      "bash /tmp/rm_files.sh"
    ]
  }
}
```

В результаті ми повинні отримати EC2 віртуальну машину з неробочим веб-сервером, який студент повинен привести до початкового стану шляхом відновлення root диска машини. Це реалізовано за допомогою повторного розгортання ресурсів, що вирішується через стан змінної *count = var.deploy_task_resources ? 0 : 1*, а так як Terraform розгортає тільки ресурси, які були змінені, або додані, то тільки запущено скрипт, що видалить html сторінку. На рис. 3.1 зображено веб-сайт, який ми отримуємо при успішному розгортанні інфраструктури:

Рисунок 3.1 – неробочий веб-сайт

Як можна побачити, сервіс не працює і це потрібно виправити. Студент це може зробити шляхом відновлення root диска, що можна зробити двома способами: через AWS консоль, або через AWS CLI.

Через консоль це можна зробити через меню *EC2 > Actions > Monitor and troubleshoot > Replace root volume*. Там можна обрати снепшот, який був створений при розгортанні інфраструктури. Після процесу відновлення root диска ми отримаємо робочу версію веб-сервера і будемо мати доступ до веб-сторінки.

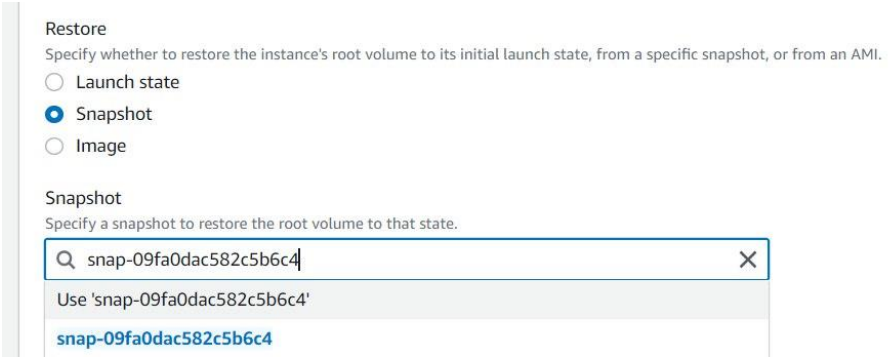


Рисунок 3.2 – відновлення root диска через AWS консоль

Статус відновлення можна подивитися в деталях віртуальної машини в секції Storage > Recent root volume replacement tasks. Після цього html сторінка буде перестворена.

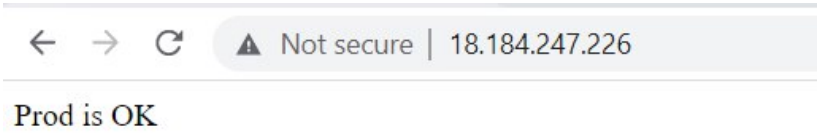


Рисунок 3.3 – Робоча веб-сторінка

Другий варіант використовує команду AWS CLI:
aws ec2 create-replace-root-volume-task --instance-id <instance-id> --snapshot-id <snapshot_id>

Розглянемо основні ресурси, які використовуються при перевірці завдання: скрипт перевірки і модулі валідації.

Скрипт перевірки отримає логи з CloudTrail і зберігає їх в логах в S3 бакеті завдання.

```
#!/bin/bash

source "../../../shared_libs/shell_functions.sh" &> /dev/null

eval "$(jq -r '@sh "WEBSITE_URL=\(.website_url)
```

```
TEST_NAME_1=\(.test_name_1) TEST_NAME_2=\(.test_name_2)
LOGS_FOLDER=\(.logs_folder) ROOT_ID=\(.root_id) OLD_ROOT=\(.old_root)
DEFAULT_REGION=\(.default_region)"')"

export AWS_DEFAULT_REGION=${DEFAULT_REGION}

Status=$(aws ec2 describe-replace-root-volume-tasks \
--filters "Name=instance-id,Values=${ROOT_ID}" \
--query "ReplaceRootVolumeTasks[0].TaskState" --output text)

aws ec2 delete-volume --volume-id ${OLD_ROOT}

command_log "${TEST_NAME_1}" "curl -sb -H 'Accept: text/html'
${WEBSITE_URL}" "${LOGS_FOLDER}"
command_log "${TEST_NAME_2}" "echo ${Status}" "${LOGS_FOLDER}"

jq -n --arg test_name_1 "${TEST_NAME_1}"  \
    --arg test_name_2 "${TEST_NAME_2}"  \
    --arg logs_folder "${LOGS_FOLDER}" '{"test_name_1":$test_name_1,
"test_name_2":$test_name_2, "logs_folder": $logs_folder}'
```

Модулі валідації завдають патерни, за якими слід шукати логи в CloudTrail. З коду можна побачити, що перевірка виконується в два етапи: перевірка доступності веб-сайту через curl та перевірка створення заміни root диска.

```
module "step_validation_1" {
 source = "../../../shared_tf_modules/step_validation"

 pattern_std_out     = ".*Prod is OK.*"
 pattern_status_code = "^0$"
 pattern_std_err     = ""
```

```
  logs_folder         = data.external.this.result.logs_folder

  test_name   = data.external.this.result.test_name_1
  description = "The test gets production website main page with curl and check its content"
}

module "step_validation_2" {
  source = "../../../shared_tf_modules/step_validation"

  pattern_std_out     = ".*succeeded*"
  pattern_status_code = "^0$"
  pattern_std_err     = ""
  logs_folder         = data.external.this.result.logs_folder

  test_name   = data.external.this.result.test_name_2
  description = "The test checks if there was a restore root volume task"
}
```

Тільки після проходження двох кроків валідації пайплайн видаляє ресурси завдання. Результат перевірки можна бачити на рис. 3.4:

```
11:54:36  result = "success"
11:54:36  step_validation_1 = {
11:54:36    "description" = "The test gets production website main page with curl and check its content"
11:54:36    "status_code" = {
11:54:36      "actual_output" = <<-EOT
11:54:36      0
11:54:36
11:54:36      EOT
11:54:36      "validation_pattern" = "^0$"
11:54:36    }
11:54:36    "std_err" = {
11:54:36      "actual_output" = ""
11:54:36      "validation_pattern" = ""
11:54:36    }
11:54:36    "std_out" = {
11:54:36      "actual_output" = <<-EOT
11:54:36      Prod is OK
11:54:36
11:54:36      EOT
11:54:36      "validation_pattern" = ".*Prod is OK.*"
11:54:36    }
11:54:36    "step_passed" = true
11:54:36    "step_result" = "pass"
11:54:36  }
11:54:36  step_validation_2 = {
11:54:36    "description" = "The test checks if there was a restore root volume task"
11:54:36    "status_code" = {
11:54:36      "actual_output" = <<-EOT
11:54:36      0
11:54:36
11:54:36      EOT
11:54:36      "validation_pattern" = "^0$"
11:54:36    }
11:54:36    "std_err" = {
11:54:36      "actual_output" = ""
11:54:36      "validation_pattern" = ""
11:54:36    }
11:54:36    "std_out" = {
11:54:36      "actual_output" = <<-EOT
11:54:36      succeeded
11:54:36
11:54:36      EOT
11:54:36      "validation_pattern" = ".*succeeded*"
11:54:36    }
11:54:36    "step_passed" = true
11:54:36    "step_result" = "pass"
11:54:36  }
```

Рисунок 3.4 – Результати валідації

3.3 Розробка завдання по роботі з EFS

Це завдання описує розподілену на декілька віртуальних машин файлову систему з важливими бізнес даними. Через збій одна машина не може отримати дані зі спільної файлової системи, студенту потрібно відновити доступ.

Для цього завдання основними ресурсами при створенні інфраструктури є: дві віртуальні системи EC2, екземпляр файлової системи EFS, скрипт готовності віртуальних машин, скрипт, який би виконував симуляцію збою (фактично робив би відключення файлової системи EFS від віртуальної машини).

На етапі верифікації використовуються модулі валідації для завдання патерна перевірки і скрипт, що отримає логи з CloudTrail і зберігає їх в логах в S3 бакеті завдання. Перевірка враховує два пункти: чи є у другої віртуальної машини доступ до розподіленої файлової системи EFS і чи наявних в логах CloudTrail запит на приєднання файлової системи до цієї машини.

3.4 Розробка завдання по роботі з S3

В цьому завданні студент бакет з увімкненим версіюванням. Через помилку, один із важливих файлів був видалений. Студент повинен відновити потрібний файл, використовуючи попередню версію файлу.

Основні файли при розгортанні інфраструктури – бакет, скрипт, який імітує версіювання. При перевірці завдання використовуются модулі валідації для завдання патерна перевірки і скрипт, що отримає логи з CloudTrail і зберігає їх в логах в S3 бакеті завдання. Перевірка передбачає дві умови – наявність бакета з актуальними даними і наявність запиту на відновлення бакета.

ВИСНОВКИ

У ході виконання даної кваліфікаційної роботи розглядалися варіанти роботи з різними типами сховища даних, що використовуються в хмарних обчисленнях. Провайдером хмарних послуг був обраний Amazon Web Services через доступність і великий об’єм сервісів роботи зі сховищами даних, а також великим обсягом технічної документації. Для демонстрації роботи зі сховищами даних була представлена система автоматичної перевірки навичок роботи із сервісами AWS і розроблено декілька практичних завдань, які навчають студентів певним аспектам роботи з EBS, EFS та S3.

Більшу частину роботи складає докладний опис того, як працює система автоматичної перевірки навичок роботи з AWS, приділяється увага тому, чому важливо мати ці навички і як хмарні технології впливають на сучасний ринок IT-послуг на прикладі статистики крупних компаній і імплементації хмарних обчислень на практиці. Також зроблено порівняння сервісів зберігання даних і визначено умови, при яких кожен сервіс (EBS, EFS та S3) будуть ефективно застосовані за своїм призначенням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Livingstone A. Disaster Recovery of Workloads on AWS: Recovery in the Cloud [Електронний ресурс] / Alex Livingstone. – 2021. – Режим доступу до ресурсу: https://docs.aws.amazon.com/pdfs/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-workloads-on-aws.pdf

2. Nizami K. Backup and recovery approaches on AWS [Електронний ресурс] / Khurram Nizami. – 2022. – Режим доступу до ресурсу: https://docs.aws.amazon.com/pdfs/prescriptive-guidance/latest/backup-recovery/backup-recovery.pdf#welcome

3. Flexera. 2021 State of Cloud Report [Електронний ресурс] / Flexera. – 2021. – Режим доступу до ресурсу: https://resources.flexera.com/web/pdf/report-cm-state- of- the- cloud- 2021.pdf?elqTrackId=28d62429a6ec40d0bb8e92159e68d63a&elqaid=6545&elqat=2

4. Cloud computing - statistics on the use by enterprises [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Cloud_computing_-_statistics_on_the_use_by_enterprises#Enterprises.E2.80.99_dependence_on_cloud_computing

5. Cloud Storage Market Size [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.fortunebusinessinsights.com/cloud-storage-market-102773

6. Cloud computing for business [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Cloud_computing_-_statistics_on_the_use_by_enterprises

7. DCI Saves 27% on Cloud Costs, Gains Support for Long-Term Growth Using AWS [Електронний ресурс]. – 2022. – Режим доступу до ресурсу:

https://aws.amazon.com/solutions/case- studies/dci- case- study/?did=cr_card&trk=cr_card

8. Skoda Case Study [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://aws.amazon.com/solutions/case- studies/skoda- case- %20study/?did=cr_card&trk=cr_card

9. Caltech Uses Amazon EFS to Automate Academic Computing File Management [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://aws.amazon.com/solutions/case- studies/caltech- case- study/?did=cr_card&trk=cr_card

10. Jones S. State of Continuous Delivery Report Q2022 [Електронний ресурс] / S. Jones, K. Korakitis. – 2022. – Режим доступу до ресурсу: https://cd.foundation/wp-content/uploads/sites/78/2022/06/The-State-of-CD-Q1-2022.pdf

11. Лещенко М. Р. Система автоматичної перевірки відновлення сховища даних на AWS / М. Р. Лещенко, А. І. Костромицький. // Тези доповідей десятої міжнародної науково-технічної конференції. – 2022. – С. 108.