Харківський національний університет радіоелектроніки
(повне найменування вищого навчального закладу)

Факультет *інфокомунікацій*
(повна назва)

Кафедра *інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Другий (магістерський)
(рівень вищої освіти)

на тему *Система автоматичної перевірки навичок роботи з AWS. Підтема 1. Робота з EC2*

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

Допускається до захисту

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

Харків - 2022 року

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

Томашевській Анастасії Віталіївні
(прізвище, ім'я, по батькові)

1. Тема роботи *Система автоматичної перевірки навичок роботи з AWS. Підтема 1. Робота з EC2*

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 і 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 концепції завдань для перевірки навичок роботи зі сховищами даних

4. Практична реалізація завдання

Висновки

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

Слайди у форматі Power Point (назва та мета роботи, актуальність,слайди, система перерівки навичок, типи сховищ, концеції, тощо)

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 атестаційної роботи	Строк виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	*21.10.22*	
2	*Підбір літератури за темою роботи.*	*23.10-29.10.22*	
3	*Виконання розділу 1*	*29.10-03.11.22*	
4	*Виконання розділу 2*	*03.11-07.11.22*	
5	*Виконання розділу 3*	*07.11-11.11.22*	
6	*Виконання розділу 4*	*11.11-15.11.22*	
7	*Оформлення презентаційного матеріалу,*	*18.12-19.12.22*	
	підготовка до захисту у ЕК		

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

Студент _____ *Томашевська А.В.*
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Пояснювальна записка: 81 с., 47 рис., 1 табл., додатки А, Б, В, Г, 9 джерел.

Об'єкт роботи – система перевірки навичок роботи студентів “Cloud Mentor” на базі платформи AWS.

Мета роботи – опис системи перевірки навичок роботи студентів та розробка новий завдань.

Розглянуто розвиток хмарних технологій на сучасному ринку та тенденції їх використання. Розроблені завдання для перевірки навичок студентів різної складності та з використанням різних сервісів платформи Amazon Web Services. Розглянуто приклади завдань на основі роботи з сервісом для створення та підтримки віртуальних машин EC2.

AWS, JENKINS, DOCKER, TERRAFORM, LINUX, WINDOWS, CLI, PYTHON.

ABSTRACT

Explanatory note: 81 p., 47 figures, 1 table, appendices A, B, D, 9 sources.

Object of work - a system for testing students' skills "Cloud Mentor" based on the AWS platform.

Purpose of work - description of the system of checking students' skills and development of new tasks.

The development of cloud technologies in the modern market and trends in their use are considered. Tasks for testing students' skills of varying complexity and using various services of the Amazon Web Services platform have been developed. Examples of tasks based on working with the service to create and maintain EC2 virtual machines are considered.

AWS, JENKINS, DOCKER, TERRAFORM, LINUX, WINDOWS, CLI, PYTHON.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

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

ЦП (Центральний процесор) –призначений для інтерпретації команд;

IT (Information Technology) – інформаційні технології;

Backup – резервне копіювання даних сайту;

Amazon EC2 – це вебсервіс, який надає обчислювальні потужності в хмарі;

ПЗ – програмне забезпечення;

SaaS (Software as a Service) – додатки з доступом через Інтернет;

SRE – (Site Reliability Engineer) – практика надійної експлуатації сервісів**;**

IaaC – (Infrastructure as Code) — керування обчислювальними ресурсами;

Amazon CloudWatch – це сервіс моніторингу та управління;

Amazon S3 – об'єктне сховище;

Kubernetes – програмне забезпечення для контейнеризованих програм;

Docker Compose – інструмент для мультиконтейнерних програмам;

Amazon AMI – образ Linux для використання Amazon EC2;

Amazon IAM – дозволяє керувати доступом до ресурсів AWS;

Crontab – демон що використовується для виконання завдань у певний час;

CloudFormation – дозволяє створити інфраструктуру в Amazon;

API – опис способів взаємодії однієї комп'ютерної програми коїться з іншими;

Amazon CloudTrail – проводить аудит операційних процесів і ризиків;

Busybox – основной интерфейс во встраиваемых операционных системах;

Docker Hub – это публичный репозиторий для загрузки своих образов;

AWS Systems Manager – панель керування для ресурсів AWS;

ВСТУП

В наш час використання хмарних технологій є невід’ємною частиною кожного сучасного проекту. Завдяки використанню хмарних сервісів можна налагодити роботу багатьох процесів, від інфраструктурних технологій, інструментів для обчислення, сховищ та баз даних, до інновацій, таких як машинне навчання та штучний інтелект, великої кількості даних та аналітики, а також Інтернет речей.

Під хмарними обчисленнями розуміється доставка IT-ресурсів на вимогу через Інтернет з оплатою за фактом використання. Купувати, розміщувати й обслуговувати фізичні ЦОД і сервери не потрібно. Замість цього ви отримуєте доступ до технологічних сервісів: обчислювальних сервісів, сховищ і баз даних, якими можна користуватися в міру необхідності завдяки постачальнику хмарних послуг, такому як Amazon Web Services (AWS)[1].

Використання хмарних обчислень потрібне організаціям незалежно від типу, розміру та галузі використання. Хмарні технології використовуються для різноманітних напрямів, включно з резервним копіюванням даних (backup), аварійним відновленням даних, розробкою та тестуванням програмного забезпечення, аналізом великої кількості даних, систем електронної пошти, віртуальних машин, а також інтернет сайтів та застосунків, створених для споживачів. Так, організації працюючі з охороною здоров'я використовують хмарні технології, аби зробити розробку планів лікування та обстеження легшою і орієнтованою на окремого пацієнта. Компанії у галузі фінансових послуг застосовують хмарні технології як для систем спостереження та кібер-безпеки в будь-який проміжок часу. Розробникам та тестувальникам ігор хмарні обчислення необхідні, аби мати змогу надавати безпечний та протестований продукт користувачам в будь-якій точці планети.

Amazon Elastic Compute Cloud (Amazon EC2) забезпечує масштабовану обчислювальну потужність у Amazon Web Services (AWS) Cloud. Використання Amazon EC2 позбавляє вас необхідності інвестувати в апаратне забезпечення, тож ви можете швидше розробляти та розгортати програми. Ви можете використовувати Amazon EC2, щоб запускати стільки чи менше віртуальних серверів, скільки вам потрібно, налаштовувати безпеку та мережу, а також керувати сховищем. Amazon EC2 дає змогу збільшувати або зменшувати масштаб, щоб відповідати змінам у вимогах або стрибкам популярності, зменшуючи потребу у прогнозуванні трафіку[2].

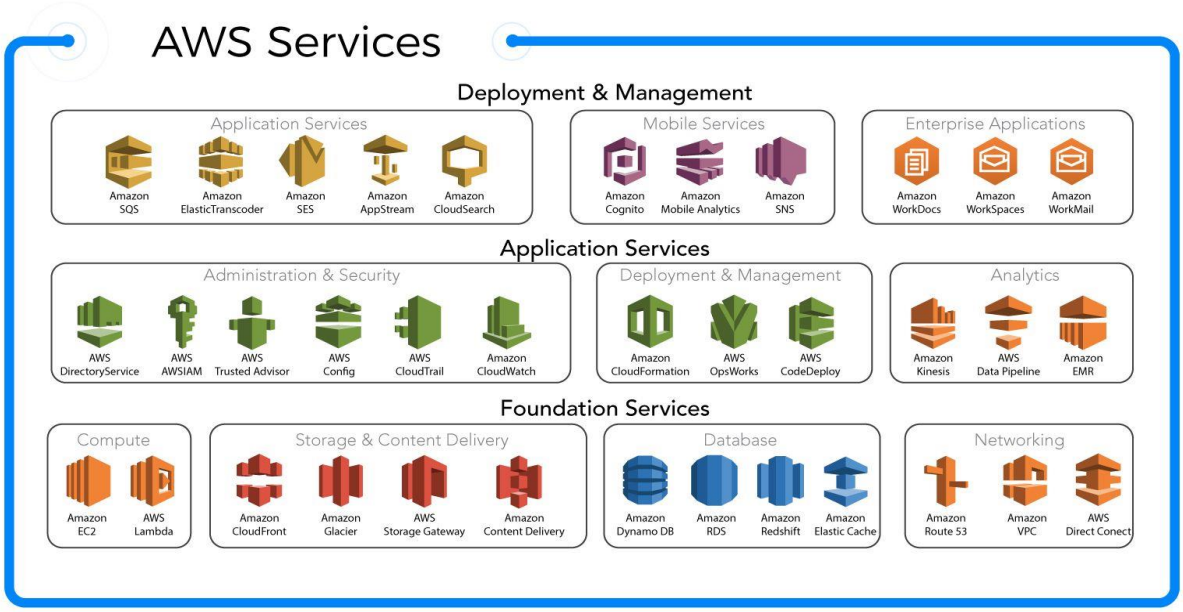

Рисунок 1 — Сервіси Amazon Web Services

Велика кількість сервісів AWS дозволяє розробникам виконувати свою роботу швидше, легше та безпечніше. Замовники можуть в режимі реального часу спостерігати за продуктами та бути впевненими що робота йде налагоджено та їх застосунки та дані є захищеними від зовнішніх чинників.

1 СУЧАСНІ ХМАРНІ ТЕХНОЛОГІЇ

1.1 Перехід існуючого бізнесу в хмарні технології

Різні мотиви можуть стимулювати бізнес-перетворення, що підтримуються використанням хмари. Декілька мотивів, швидше за все, застосовуються одночасно. Мета списку в наступній таблиці – допомогти сформувати уявлення про те, які мотиви є актуальними. Ви зможете оцінити потенційні наслідки всіх релевантних причин. Ваша команда з впровадження хмарних технологій має зустрітися із зацікавленими особами, керівниками та керівниками компаній та обговорити, які мотиви можуть допомогти у впровадженні хмарних технологій вашої компанії [8].

В таблиці 1.1 вказано причини міграції вони є найпоширенішими причинами використання хмари, але є дуже значними. Ці результати мають вирішальне значення. Цей важливий перший крок до впровадження хмари називається «міграцією у хмару».

Таблиця 1.1 — Міграція бізнесу в хмарні технології

Критичні бізнес-події	**Міграція**	**Інновації**
Припинення використання центрів обробки даних	Економія коштів	Підготовка до впровадження нових технічних можливостей
Злиття, поглинання або відчуження	Зменшення складності постачальників та технічної складності	Створення нових технічних можливостей
Скорочення капітальних витрат	Оптимізація внутрішніх операцій	Масштабування відповідно до ринкових вимог

Продовження таблиці 1.1 — Міграція бізнесу в хмарні технології

Закінчення підтримки критично важливих технологій	Підвищення гнучкості організації	Масштабування відповідно до географічних потреб
Реагування на зміни стану відповідності нормативним вимогам	Підготовка до впровадження нових технічних можливостей	Підвищення залучення клієнтів та покращення взаємодії з клієнтами
Нові вимоги до незалежності даних	Масштабування відповідно до ринкових вимог	Перетворення продуктів чи послуг
Зменшення кількості збоїв та підвищення стабільності IT-інфраструктури	Масштабування відповідно до географічних потреб	Порушення рівноваги на ринку через нові продукти чи послуги

Дані є важливою складовою проекту, а сучасні додатки — ланцюжок поставок, яким ці дані передаються у різні інтерфейси. У сучасному ринку додатків важко знайти новітній продукт або послугу, які не були б засновані на даних, аналітиці та досвіді роботи користувачів.

1.2 Хмарні технології в сучасній освіті

Технології в нашому світі розвиваються кожен день та технічний прогрес не стоїть на місті. Використання хмарних технологій доволі сильно проштовхнуло навчання про віртуалізацію вперед.

В останні декілька років світ охопило багато стихійних та техногенних лих. Освіта та бізнес в кожному куточку планети відчули це на собі та почали пристосовуватися до нових реалій існування.

За таких умов бізнес здебільшого перейшов до децентралізації і віддалені форми роботи. Для цього відбувається стрімка переорієнтація на хмарні технології і навіть змінення підходів до розробки прикладного програмного забезпечення (ПЗ) на користь моделі SaaS (Software as a Service – ПЗ як послуга) з поступовим відходом від традиційних схем дистрибуції. Нині хмарні сервіси успішно застосовують у телемедицині і дистанційній діагностиці, в інтелектуальних промислових застосунках, пов’язаних з експлуатацією виробничого устаткування, у логістиці, в управлінні транспортом, у роздрібній торгівлі, в інфраструктурі моніторингу навколишнього середовища і вимірювальних приладів тощо [3].

Однією із важливих переваг використання хмарних технологій в освіті є те, що студенти можуть слухати лекції, проходити практичні та лабораторні роботи, складати заліки та екзамени на відстані один від одного та від викладача. Використовуючи певні програми та утиліти студенти можуть спілкувати між собою за допомогою мікрофону та відеокамери. Завдяки цьому можна продовжувати навчальний процес без перешкод та отримувати всі необхідні навички та знання за розкладом програми.

Найпоширеніші хмарні платформи [4-6], які використовують в освіті:

Google Classroom пов’язує хмарні онлайн-програми Google, надає доступ до платформи з комп’ютерів, планшетів та смартфонів, дозволяє зручно планувати терміни виконання завдання студентами.

Blackboard серед інших свої послуг надає ПЗ для хмарного навчання. Так, Blackboard Classroom забезпечує організацію та проведення відеоконференцій у віртуальних класах, керування завданнями, аналітику їх виконання тощо.

Knowledge Matters дозволяє моделювати хмарні бізнес-симуляції для імітації ситуацій, з якими студенти стикатимуться у діловому середовищі, тим самим даючи студентам можливість практикувати розв’язання реальних проблем. Симуляції кейсів орієнтовані на конкретні галузеві сценарії.

Coursera – відома освітня хмарна платформа з різноманітними онлайн-курсами від відомих університетів та викладачів.

Office 365 Education від компанії Microsoft орієнтований на студентів та викладачів і призваний спростити навчання в Інтернеті.

Classflow – це хмарне ПЗ, яке допомагає викладачам створювати інтерактивні форми занять та показувати їх студентам.

Top Hat – освітній застосунок, який дозволяє студентам і викладачам взаємодіяти з матеріалами курсу та між собою проводити опитування студентів, організовувати дискусії або надсилати матеріали для читання.

D2L Brightspace – платформа онлайн-навчання для задоволення потреб педагогів і студентів, незалежно від того, де і хто вони. Її зручна інформаційна панель дозволяє викладачам відстежувати успіхи студентів.

1.3 Безперервна інтеграція, доставка та розгортання CI/CD

CI/CD — це метод частої доставки додатків клієнтам шляхом впровадження автоматизації на етапах розробки додатків. Основними концепціями CI/CD є безперервна інтеграція, безперервна доставка та безперервне розгортання. CI/CD — це рішення проблем, які інтеграція нового коду може спричинити для команд розробників (відомих також як «інтеграційне пекло»).

Зокрема, CI/CD запроваджує постійну автоматизацію та постійний моніторинг протягом життєвого циклу продуктів, від етапів інтеграції та тестування до доставки та розгортання. У сукупності ці підключені практики часто називають «конвеєром CI/CD» і підтримуються командами, які працюють разом у гнучкий спосіб за допомогою підходу DevOps або розробки надійності сайту (SRE) [7].

Безперервна інтеграція — за допомогою цієї практики ранньої та частої інтеграції всіх змін коду зроблених розробниками в головну гілку програми відстежування версій коду, автоматичного тестування змінних, як тільки їх

фіксують та об’єднують, і автоматичного запуску збірки проекту. Завдяки безперервній інтеграції помилки та проблеми з безпекою проекту можна виявити та виправити легко та швидко.

Об’єднуючи всі зміни та запускаючи автоматичне тестування та перевірку на помилки, мінімізується ймовірність так званого «конфлікту» коду, якщо кілька розробників працюють над одним документом чи на одній гілці. Друга перевага полягає в тому, що не потрібно довго чекати результатів етапу і дозволяє виправляти помилки одразу після їх виявлення.

Безперервна доставка існує для розробки програмного забезпечення, що здійснюється разом із безперервною інтеграцією для автоматизації проекту та всіх його складових.

Після написання та тестування коду під час процесу безперервної доставки починається на останніх етапах, щоб гарантувати, що він може бути розгорнутий правильно та без помилок у будь-якому середовищі та у будь-який час. Безперервна доставка працює в багатьох частинах процесу, від надання інфраструктури до розгортання продукту в середовищі тестування та продакшену.

2 СИСТЕМА АВТОМАТИЧНОЇ ПЕРЕВІРКИ НАВИЧОК РОБОТИ З AWS

За потреби навчання студентів що спеціалізуються на роботі з хмарними технологіями, інструментами зборки, безперервної автоматизації, керування конфігураціями, користуванням системами версій, було створено систему автоматичної перевірки навичок роботи що ґрунтується на такому хмарному сервісі як Amazon Web Services.

Головною метою платформи є вивчення та навчання практичним навичкам роботи з безперервною інтеграцією та безперервною доставкою за допомогою Jenkins, робота з системою управління репозиторіями програмного коду GitLab, інструментом для створення та управління віртуальних контейнерів Docker, та вивчення IaaC з утилітою командного рядка Terraform.

Завдання подані студентам є багато направленими, різними по складності та сферою використання. Серед завдань студенти навчаються використанню таких сервісів як сервіс для контролювання доступу до ресурсів IAM, сервіс для створення та керування віртуальними машинами EC2, сервіс сховище даних S3 сервіс для аналізу CloudWatch та інші.

2.1 Концепція платформи

Концепцією платформи є створення штучного проекту та сервісу для удосконалення знань та навичок студентів шляхом створення завдань різного ступеня складності та направленостей.

2.2 Встановлення системи для розробників

1. Встановіть Rancher Desktop та виберіть dockerd як середовище виконання контейнера, kubernetes можна вимкнути.

2. Встановіть Docker Compose.

3. Підготуйте студентський обліковий запис AWS в регіоні *eu-central-1*.

4. Створіть стек CloudFormation "CloudMentorLocalConnectResources", який керує такими ресурсами:

1. Користувач IAM *cloud-mentor-connect-user* з дозволами на будь-яку роль.
2. Роль IAM *cloud-mentor-management-role* з правами адміністратора. Він налаштовує довірені дозволи для користувача IAM.
3. Роль IAM *cloud-mentor-connector-role* з правами адміністратора. Він налаштовує довірені дозволи для ролі IAM.
4. Відро S3 *cloud-mentor-debug-tf-state-<student AWS AccountID>*.

5. Склонуйте репозиторій Cloud Mentor.

6. Налаштуйте облікові дані AWS. Отримайте необхідні дані з стеку CF (перейдіть до CloudFormation service, та виберіть стек під назвою *CloudMentorLocalConnectResources*, та перейдіть до *Outputs*).

Створіть AWS Credentials file (linux *~/.aws/credentials*, windows powershell *$HOME\.aws\config*).

```
[cloud-mentor-connect-user]
aws_access_key_id = <aws_access_key_id>
aws_secret_access_key = <aws_secret_access_key>
```

Рисунок 2.1 – Конфігураційний файл для даних користувача AWS

Створіть AWS config (linux *~/.aws/config*, windows powershell *$HOME\.aws\config*).

```
[profile cloud_mentor_debug]
role_arn = arn:aws:iam::<your AWS account ID>:role/cloud-mentor-management-role
source_profile = cloud-mentor-connect-user
```

Рисунок 2.2 – Конфігурація для даних користувача AWS

7. Зберіть контейнер та запустіть його.

```
# Windows cmd envs
setx AWS_REGION eu-central-1
setx AWS_DEFAULT_PROFILE cloud_mentor_debug
setx CLOUD_MENTOR_TF_BUCKET <The bucket created in CF stack CloudMentorLocalConnectResources>
setx DOCKER_GROUP_ID 101
setx HOME %USERPROFILE%

# Linux envs
export AWS_REGION=eu-central-1
export AWS_DEFAULT_PROFILE=cloud_mentor_debug
export CLOUD_MENTOR_TF_BUCKET=<The bucket created in CF stack CloudMentorLocalConnectResources>
export DOCKER_GROUP_ID=101


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

Рисунок 2.3 – Скрипт для збірки контейнера

8. DOCKER_GROUP_ID потрібно отримати з вашого контейнера jenkins, експортувати його, а потім перезапустити контейнери *docker-compose -f docker-compose-local.yaml up*.

```
➜  cloud-mentor git:(setup_dev_env_docker_compose) X docker ps
CONTAINER ID   IMAGE                            COMMAND                  CREATED          STATUS                    PORTS
8e0ad039916a   cloud-mentor_jenkins             "/sbin/tini -- /usr/…"   30 minutes ago   Up 30 minutes             0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:5
7f079aa1a0d6   cloud-mentor_cm-api              "python -m uvicorn s…"   33 minutes ago   Up 29 minutes (healthy)   0.0.0.0:81->80/tcp, :::81->80/tcp
da1fea2a50ec   instructure/dynamo-local-admin   "/usr/bin/supervisor…"   6 weeks ago      Up 30 minutes (healthy)   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp, 8001-8002

➜  cloud-mentor git:(setup_dev_env_docker_compose) X docker exec -it cloud-mentor-jenkins-1  bash

jenkins@8e0ad039916a:/$ stat -c '%g' /var/run/docker.sock
101

jenkins@8e0ad039916a:/$ exit

➜  cloud-mentor git:(setup_dev_env_docker_compose) X export DOCKER_GROUP_ID=101
```

Рисунок 2.4 – Команди для становлення DOCKER_GROUP_ID

9. Налаштуйте Jenkins.

Ваш локальний ключ ssh *'~/.ssh/id_rsa'* буде додано до віртуальної машини автоматично. У цьому випадку вам не потрібно налаштовувати облікові дані вручну. Якщо ви використовуєте спеціальний ключ, вам слід налаштувати облікові дані вручну, перейти до облікових даних і створити

SSH *Username with private key* з ідентифікатором «*git*», вказати ім'я користувача *git* та ключ для доступу до сховища *cloud-mentor* у *git.com*. Налаштуйте «Вбудований вузол»: установіть мітку мітки *master* та «Кількість виконавців» на 5.

10. Створіть користувача Cloud-Mentor за замовчуванням.

Увійдіть до контейнера *jenkins docker exec -it <jenkins_container_name> bash* і виконайте сценарій, який створить вашого студента за замовчуванням і напише його ідентифікатор:

```
#!/bin/bash
aws_account_id=$(aws sts get-caller-identity --query 'Account')
group_id=$(curl -s -X 'POST' \
        'http://cm-api/groups' \
        -H 'accept: application/json' \
        -H 'Content-Type: application/json' \
        -d '{
            "name": "Default",
            "location": "UA",
            "description": "Default"
        }' | jq -r ".id")

student_id=$(curl -s -X 'POST' \
    'http://cm-api/students' \
    -H 'accept: application/json' \
    -H 'Content-Type: application/json' \
    -d "{
    \"name\": \"Default\",
    \"surname\": \"Default\",
    \"email\": \"user@example.com\",
    \"aws_account_id\": ${aws_account_id},
    \"group_id\": \"${group_id}\"
    }" | jq -r ".id")

echo "Default user created, id '${student_id}'."
```

Рисунок 2.5 – Скрипт для отримання student_id ідентифікатора

11. Створіть робочу область *create_user_workspace*.

Запустити конвеєр «*create_user_workspace*» із *reload_parameters* опцією. Перший запуск трубопроводу буде невдалим. Після першого запуску

правильні конфігурації конвеєра будуть отримані з git, і після цього конвеєр буде готовий до використання. Згенерувати ідентифікатор студента на попередньому кроці в *student_cm_id* параметр. Ви завжди можете отримати ідентифікатор студента або створити його через CM API.

Рисунок 2.6 – папка користувача в інтерфейсі Jenkins

12. Видаліть docker контейнери.

```
docker-compose -f docker-compose-local.yaml down
```

Рисунок 2.7 – Команда для видалення контейнера

2.3 Загальний підхід до розгортання інфраструктури і перевірки завдань

В цьому розділі буде йти мова про те, як саме запустити перевірку завдання виконаного студентом, валідувати його, відправити результат та видалити.

Щоб написати перевірку CLI для завдання, вам потрібно створити файл із назвою *task_name_validation.py* у папці *cli_tasks_validations* . Шаблон сценарію, який можна використовувати для написання власної перевірки CLI, наведено в кінці цієї документації.

Визначте команди, за допомогою яких можна виконати завдання, і зверніть увагу, що деякі завдання можна розв'язувати за допомогою взаємозамінних команд. Наприклад, IAM-політику можна приєднати до ролі двома різними способами: виконання *aws iam attach-role-policy* або *aws iam put-role-policy.*

Ми використовуємо лише події керування для перевірки CLI, оскільки за реєстрацію подій даних стягується додаткова плата. Таким чином, ми не можемо відстежити деякі команди, які виконує учень, наприклад *aws s3 cp command.*

Наприклад, ваші дії можуть бути такими:

1. Запустіть завдання, для якого ви хочете написати перевірку.
2. Виконайте всі необхідні команди в CLI, щоб виконати завдання.
3. Переконайтеся, що завдання виконано успішно.
4. Шукайте події, що відповідають раніше виконаним командам, і їх вміст, який можна використовувати для перевірки.

Для кожної команди, за допомогою якої можна вирішити завдання, ви повинні написати шаблон регулярного виразу, який може містити ім'я ресурсу, ARN, частину політики тощо.

Щоб отримати імена ресурсів у сценарії перевірки CLI, додайте відповідний вивід *terraform y task_folder /infra/outputs.tf* .

```
output "cli_usage_check" {
    value = {
      assume_role_name   = aws_iam_role.assume_role.name
      readonly_role_name = aws_iam_role.readonly_role.name
    }
  }
```

Рисунок 2.8 – Скрипт для отримання імен ресурсів

Потім цей *output* зчитується в конвеєрі Jenkins, об'єднується у файл *json* разом з іншими параметрами, такими як *task_name*, *taskStartTimestamp*,

taskEndTimestamp, aws_region, aws_role_arn і передається до сценарію *python*. Таким чином, наступні рядки є обов'язковими та завжди присутніми.

```
role_arn = pipeline_info['aws_role_arn']
aws_region = pipeline_info['aws_region']
global_region = "us-east-1"
pipeline_start_time = pipeline_info['taskStartTimestamp']
pipeline_end_time = pipeline_info['taskEndTimestamp']
task_name = pipeline_info['task_name']
```

Рисунок 2.9 – Скрипт параметрів pipeline

Після цих рядків коду додайте змінні з результату terraform і дайте їм описові імена.

```
cli_usage_check_output_value_1 = pipeline_info['tf_output']['output_value_1']
cli_usage_check_output_value_2 = pipeline_info['tf_output']['output_value_2']
```

Рисунок 2.10 – Скрипт для перевірки check_output_value

Потім *sleep* команда використовується для очікування, поки всі події CloudTrail будуть зареєстровані.

Далі ми викликаємо функцію, яка перевіряє, чи було виконано завдання за допомогою CLI. Цю функцію потрібно назвати task_name*_validation()***.**

У цій функції ви маєте вказати шаблони регулярних виразів, які використовуватимуться для фільтрації подій CloudTrail. Не забудьте правильно екранувати символи. Ось кілька прикладів:

```
attach_readonly_pattern = r'(\"roleName\":\"' + readonly_role_name + r'\").*(\"policyArn\":\"arn:aws:iam::aws:policy/ReadOnlyAccess\")'
attach_profile_pattern = r'(\"InstanceId\":\")' + instance_id + r'\".*(\"Name\":\"' + iam_role_name + r'\")'
```

Рисунок 2.11 – Шаблони регулярних виразів

У цих прикладах використовуються змінні, які були передані сценарію як вихід із terraform.

Наступним кроком є створення змінної з назвою *commands_with_parameters*. Ця змінна є словником, який містить власні унікальні назви команд як ключі та параметри команд у словнику як значення.

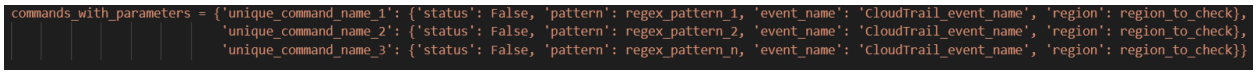

Рисунок 2.12 – Створення змінної *commands_with_parameters*

Давайте ближче розглянемо цю змінну. Ключі словника мають бути унікальними описовими іменами, які використовуються для ідентифікації кожної команди, за допомогою якої можна виконати завдання. Значення словника мають бути словником із 4 обов'язковими ключами (*status* , *pattern* , *event_name* , *region*).

Ключ статусу – завжди має бути встановлено значення *False* , це вказує, чи була команда знайдена в подіях CloudTrail чи ні.

Ключ шаблону - повинен бути необробленим рядком із шаблоном, який буде використовуватися надалі для пошуку події, що відповідає пошуковій команді CLI.

Event_name key – це ім'я API, пов'язане з подією в CloudTrail.

Ключ регіону - має бути регіон, де шукати події з указаною назвою події. Може бути одним із двох можливих значень: *aws_region* , *global_region*.

Для більшості сервісів події фіксуються в регіоні, де відбулася дія. Для глобальних служб, таких як AWS Identity and Access Management (IAM), AWS STS і Amazon CloudFront, події записуються в регіоні Схід США (Північна Вірджинія) (us-east-1).

Якщо завдання має взаємозамінні команди, створіть змінну *interchangeable_commands*. Це буде використано для перевірки виконання принаймні однієї команди зі списку можливих команд.

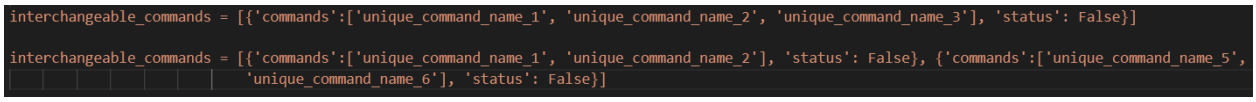

Рисунок 2.13 – Створення змінної *interchangeable_commands*

Змінна *interchangeable_commands* — це список словників, де кожен словник має два ключі **:** *commands* і *status* .

Ключ команд - це список взаємозамінних команд

Ключ статусу – завжди має бути встановлено значення *False* , це вказує, чи була одна з команд у списку знайдена в подіях CloudTrail чи ні.

Наступним кроком є створення змінної результату**.** Ця змінна має чотири обов’язкові аргументи:

1. *role_arn;*
2. *pipeline_start_time;*
3. Поточний час;
4. Команди з параметрами.

І один не обов'язковий - *interchangeable_commands*.

Функція *find_events_in_logs* повертає число, отримане діленням кількості команд, виконаних студентом через CLI, *на* загальну кількість перевірених команд.

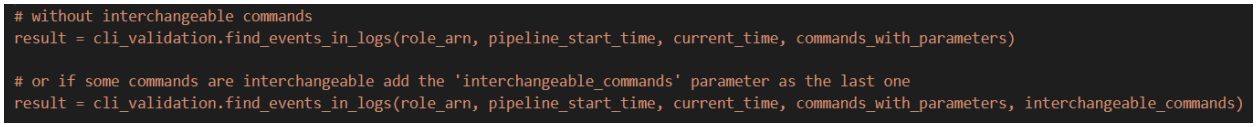

Рисунок 2.14 – Output функції *find_events_in_logs*

Шаблон сценарію перевірки CLI додано в додаток А.

3 РОЗРОБКА КОНЦЕПЦІЇ ЗАВДАНЬ ДЛЯ ПЕРЕВІРКИ НАВИЧОК РОБОТИ ЗІ СХОВИЩАМИ ДАНИХ

Сховища даних в AWS є таких типів як об'єктне, файлове та блочне сховище.

Основними сервісами сховищ даних в Amazon Web Services є Amazon Simple Storage Service (S3), Amazon Elastic File System (EFS), Amazon FSx, та Amazon Elastic Block Store (EBS).

Детальніша інформація про кожне сховище буде описана нижче:

Amazon Simple Storage Service (S3) – сховище об'єктів, яке пропонує найкращу в галузі масштабованість, доступність і безпеку та дає змогу зберігати й витягувати будь-який обсяг даних із будь-якого місця;

Amazon Elastic File System (EFS) – проста без серверна гнучка файлова система, яка не потребує постійного контролю, призначена для обміну даними без управління сховищем;

Amazon FSx – повністю кероване, економічне файлове сховище, яке пропонує можливості та продуктивність популярних комерційних файлових систем і систем з відкритим вихідним кодом;

Amazon Elastic Block Store (EBS) – простий у використанні, високопродуктивний сервіс, що забезпечує пропускну спроможність та інтенсивність транзакцій робочих навантажень за будь-якого масштабу.

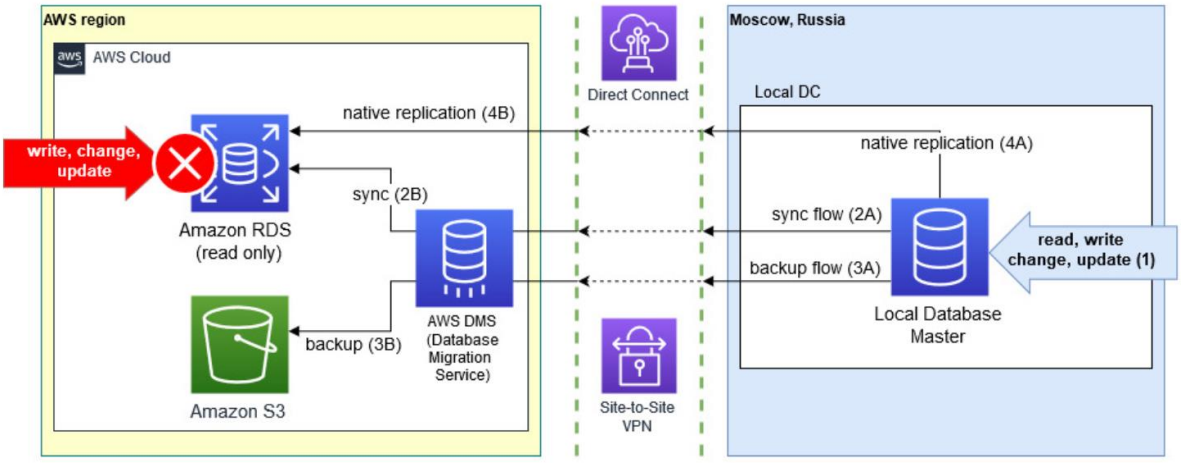

Рисунок 3.1 – Гібридний сценарій зберігання даних в AWS

Використання хмарних технологій можливе завдяки налаштуванням віртуалізації, більше про віртуалізацію та віртуальні машини можна подивитись в тезисах написаних мною, що знаходяться в додатку В.

3.1 Розробка завдання по роботі з EC2. Створення кластера EC2 за допомогою інтерфейсу командного рядка Amazon CLI

В результаті виконання цього завдання студент має налаштувати кластер, зареєструвати задачу, виконати сценарії в Amazon ECS за допомогою командного рядка Amazon CLI. За замовчуванням використовується стандартний кластер *default* при створенні екземпляру контейнера. Задача створити власний кластер з унікальною назвою.

Крок 1: Встановлення Amazon CLI.

Завантажте файл для встановлення утиліти.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Рисунок 3.2 – Команди для встановлення Amazon CLI

Використайте *curl* команду *--option -o*, щоб вказати ім'я файлу, в який має бути записаний завантажений пакет.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

Рисунок 3.3 – Використання команди curl

Розархівуйте інсталятор за допомогою команди *unzip* та запустіть розархівований файл.

```
unzip awscliv2.zip
sudo ./aws/install -i /usr/local/aws-cli -b /usr/local/bin
```

Рисунок 3.4 – Розархівування інсталятору

Крок 2: Створення кластеру.

Використайте наступну команду для створення кластеру з унікальною назвою.

```
aws ecs create-cluster --cluster-name MyCluster
```

Рисунок 3.5 – Команда для створення кластеру в Amazon ECS

В результаті використання команди наведеної на рисунку 3.4 ви отримаєте результат наведений на рисунку 3.5.

```
{
    "cluster": {
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/MyCluster"
    }
}
```

Рисунок 3.6 – *Output* команди для створення кластеру

Крок 3: Створення екземпляру контейнера Amazon ECS.

Ви можете запустити екземпляр різними методами, включаючи консоль Amazon EC2, Amazon CLI та SDK. В цій задачі вам треба використовувати консоль Amazon EC2.

Відкрийте консоль Amazon EC2 за адресою https://console.amazonaws.cn/ec2/.

На панелі навігації у верхній частині екрана відображається поточний регіон. Виберіть регіон, у якому потрібно запустити екземпляр.

На інформаційній панелі консолі Amazon EC2 виберіть «Запустити екземпляр».

Крок 4: Створення списку екземплярів контейнера.

Аби отримати список всіх екземплярів контейнера треба виконати наступну команду.

```
aws ecs list-container-instances --cluster default
```

Рисунок 3.7 – Команда для виводу списку контейнерів

В результаті команди з рисунку 3.3.6 отримуємо такий *output*:

```
{
    "containerInstanceArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:container-instance/container_instance_ID"
    ]
}
```

Рисунок 3.8 – *Output* команди для виводу списку екземплярів контейнера

Крок 5: Охарактеризуйте екземпляр контейнера.

Отримавши ARN або ідентифікатор екземпляра контейнера, ви можете використовувати команду *describe-container-instances*, щоб отримати цінну інформацію про екземпляр, таку як залишкові та зареєстровані ресурси CPU і пам’яті.

```
aws ecs describe-container-instances --cluster default --container-instances container_instance_ID
```

Рисунок 3.9 – Команда для виводу даних про екземпляр

В результаті отримуємо обширну інформацію про екземпляр частина якої наведена нижче.

```
{
    "failures": [],
    "containerInstances": [
        {
            "status": "ACTIVE",
            "registeredResources": [
                {
                    "integerValue": 1024,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
```


Рисунок 3.10 – Порти, пам'ять, ЦП екземпляру

Крок 6: Зареєструйте Task Definition.

Визначення завдань – це списки контейнерів, згрупованих разом. У наступному прикладі наведено просте визначення завдання, яке використовує busybox зображення з Docker Hub і просто перебуває в режимі сну на 360 секунд.

Рисунок 3.11 – Приклад json файлу task definition

Щоб використовувати файл JSON для визначення контейнера:

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/sleep360.json
```

Рисунок 3.12 – Json файл для визначення контейнера

Register *-task-definition* повертає опис визначення завдання після завершення реєстрації.

Рисунок 3.13 – Output команди task definition

Крок 7: Список Task Definition

За допомогою команди *list-task-definitions* ви можете будь-коли перерахувати визначення завдань для свого облікового запису . Результат цієї команди показує значення familyта revision, які можна використовувати разом під час виклику *run-task* або *start-task*.

```
aws ecs list-task-definitions
```

Рисунок 3.14 – Команда для перерахунку визначення завдань

В результаті виконання задачі ви отримаєте такий *output*:


```
{
    "taskDefinitionArns": [
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:2",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep360:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:3",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:4",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:5",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:6"
    ]
}
```


Рисунок 3.15 – *Output* команди list- task-definitions

Крок 8: Запустіть завдання.

Після реєстрації завдання та запуску екземпляра, зареєстрованого у кластері, ви можете запустити зареєстроване завдання у кластері.

```
aws ecs run-task --cluster default --task-definition sleep360:1 --count 1
```

Рисунок 3.16 – Команда для запуску завдання кластера

Після виконання команди з рисунку 3.16 ви отримаєте такий *output:*

```
{
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "PENDING",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/sleep360:1",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "PENDING",
                    "name": "sleep"
                }
            ]
        }
    ]
}
```


Рисунок 3.17 – Результат виконання команди *run-task*

Крок 9: Складіть список завдань.

Перелічіть завдання для вашого кластера. Ви повинні побачити завдання, яке ви запустили в попередньому розділі. Ви можете взяти ідентифікатор завдання або повний ARN, який повертає ця команда, і використовувати його для опису завдання пізніше.

```
aws ecs list-tasks --cluster default
```

Рисунок 3.18 – Команда для виводу списку завдань


```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID"
    ]
}
```


Рисунок 3.19 – *Output* команда для виводу списку завдань

Крок 10: Опишіть поточне завдання.

Опишіть завдання, використовуючи ідентифікатор завдання, отриманий раніше, щоб отримати більше інформації про завдання.

```
aws ecs describe-tasks --cluster default --task task_ID
```

Рисунок 3.20 – Команда для опису завдання


```
{
    "failures": [],
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "RUNNING",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/sleep360:1",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "RUNNING",
                    "name": "sleep",
                    "networkBindings": []
                }
            ]
        }
    ]
}
```


Рисунок 3.21 – Output команди *describe-tasks*

В результаті виконання цих кроків ми отримуємо кластер із завданням EC2.

3.2 Розробка завдання по роботі з EC2. Створення скрипту для резервного копіювання EC2-instance в Amazon AMI

Метою завдання є створення резервного копіювання екземплярів в Amazon EC2 сервісі. Маємо декілька EC2-серверів в AWS, розташованих у

різних регіонах. Потрібно автоматизувати їхнє резервне копіювання так, щоб відновлення було легким та швидким.

Для виконання завдання потрібно:

1. Встановити пакет *ec2-api-tools.*
2. Оновити скрипт вказавши шлях до logs файлу та *ec2-automate-backup2ami.sh*
3. Створити користувача якого ви будете використовувати для backup та додати до нього відповідну політику в Amazon IAM.

Для створення backup потрібен скрипт *ec2-backup-wrapper.sh* для запуску по cron.

```
#!/bin/bash

#-- stage/prod/etc
MODE=$1

BACKUP_SCRIPT=/usr/local/bin/ec2-automate-backup2ami.sh
LOG=ec2-automate-backup2ami.$MODE.log
LOG_ERR=ec2-automate-backup2ami.$MODE.err
EMAILS=$2

#-- set AWS_ELB_HOME, EC2_HOME and PATH
source /etc/profile.d/ec2-api.sh

source .$MODE

$BACKUP_SCRIPT -s tag -t "Backup=true" -k 14d -p -h -u -n -y "Copy=true" -o "us-west-1 eu-west-1" > $LOG 2>$LOG_ERR
RES=$?

if [ ! "x$RES" = "x0" ]; then
#    cat $LOG | grep -q UnauthorizedOperation &&
    cat $LOG | mail -s "EC2 AMI auto-backup $MODE: failed" $EMAILS
fi

if [ -s $LOG_ERR ]; then
    cat $LOG_ERR | mail -s "EC2 AMI auto-backup $MODE: failed" $EMAILS
```

Рисунок 3.22 – Код скрипту для запуску backup по cron

Також знадобиться основний скрипт зборка backup, він знаходиться в додатку Б.

Крок 1: Створення користувача в Amazon IAM.

Рисунок 3.23 – Створення користувача з політиками EC2

Крок 2: Створіть файл з параметрами для створеного користувача. Додайте параметри файлу конфігурації та облікових даних за допомогою наступних команд.

```
cat .stage
export AWS_ACCESS_KEY=access_key
export AWS_SECRET_KEY=secret_key
export AWS_ACCESS_KEY_ID=access_key
export AWS_SECRET_ACCESS_KEY=secret_key
```

Рисунок 3.24 – Команди AWS credentials

Крок 3: Встановлення crontab

Вкажіть розташування домашнього файлу EC2_HOME.

За допомогою інтерфейсу AWS console позначте кожен екземпляр тегом backup з відзнакою true.

Напишіть скрипт для запуску crontab, приклад описаний нижче.

```
crontab -l
PATH=$PATH:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin
EC2_HOME=/usr/local
SHELL=/bin/bash

00 2 * * * ./ec2-backup-wrapper.sh stage "alerts0@mydomain.cc alerts01@mydomain.cc"
```

Рисунок 3.25 – Опис команди *crontab*

Після запуску та роботи скрипту його результат буде записаний у log файл ec2-automate-backup2ami.stage.log. Якщо результатом скрипту буде помилка, то на вказані email адреси буде надіслане повідомлення. Також у Amazon AMI з'явиться новий образ з назвою *ec2ab_server.domain.cc_YYYY-MM-DD* (дата створення) та такими тегами:

Name: назва екземпляру Amazon EC2;

InitianingHost: FQDN backup сервера;

PurgeAfterFE: дата видалення образу (unix time);

PurgeAfter: дата видання образу (YYYY-MM-DD);

PurgeAllow: по замовчуванню true, для автоматичного видалення образу;

Instance: ID екземпляра EC2;

Created: дата створення образу (YYYY-MM-DD);

Крок 4: Налаштування автоматичного копіювання образів в різні регіони.

Для автоматичного копіювання образів створених в процесі backup під час запуску скрипту потрібно додати ключ *-y*, та додати в налаштування

екземпляру додати тег. Також треба вказати ключ *-o* та написати список регіонів, один з них буде обраний випадковим чином під час запуску скрипта. Слід зауважити що всі створені backup будуть додані до одного регіону.

```
/usr/local/bin/ec2-automate-backup2ami.sh -s tag -t "Backup=true" -k 14d -p -h -u -n -y "CopyRegion=true" -o "us-west-1 eu-west-1"
```

Рисунок 3.26 – Приклад команди для запуску скрипта з тегами

На рисунку 3.26 ми можемо побачити що скрипт зробить backup буде створений на всіх екземплярах з тегом *backup* (з поміткою *true*), видалить всі старі backup (створені більше 13 днів тому), та скопіює всі копії екземплярів в один із регіонів та добавить тег до першого AMI з назвою *CopyRegion*, а до всіх нових тег *SourceRegion.*

Виконавши це завдання можна налаштувати автоматичне створення backup екземплярів EC2 сервісу.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАВДАННЯ

4.1. Постановка задачі розробки завдання «Cloudwatch agent»

Мета цього завдання полягає в тому, щоб навчитися створювати власну метрику Cloudwatch. Перед студентами буде стояти задача у такій формі:

Ви маєте:

–Екземпляр EC2 *cloudmentor-ec2_cloudwatch_metrics-instance-public*.

–Сигнал тривоги хмарного моніторингу (Cloudwatch Alarm) *cloudmentor-ec2_cloudwatch_metrics-alarm.*

–Параметр SSM *cloudmentor-ec2_cloudwatch_metrics-parameter*.

–*cloudmentor-ec2_cloudwatch_metrics-iam_role* – ця роль необхідна для коректної роботи агенту cloudwatch.

За три ходи слід прикріпити політику для агенту до *cloudmentor-ec2_cloudwatch_metrics-iam_role*. Далі необхідно оновити параметр SSM, додавши конфігурацію для агенту. Також необхідно запустити агент cloudwatch і створити з ним метрику пам'яті. Якщо все налаштовано правильно, тривога змінить статус на *OK*.

Бонусний розділ.

Ви можете отримати додаткові бали, якщо вирішите завдання, використовуючи команди AWS CLI для виконання необхідних дій.

4.2. Послідовність дій при виконанні завдання

Після налаштування особистого облікового запису та користувача в Jenkins студент повинен запустити pipeline task_pipeline та обрати завдання яке потрібно буде виконати.

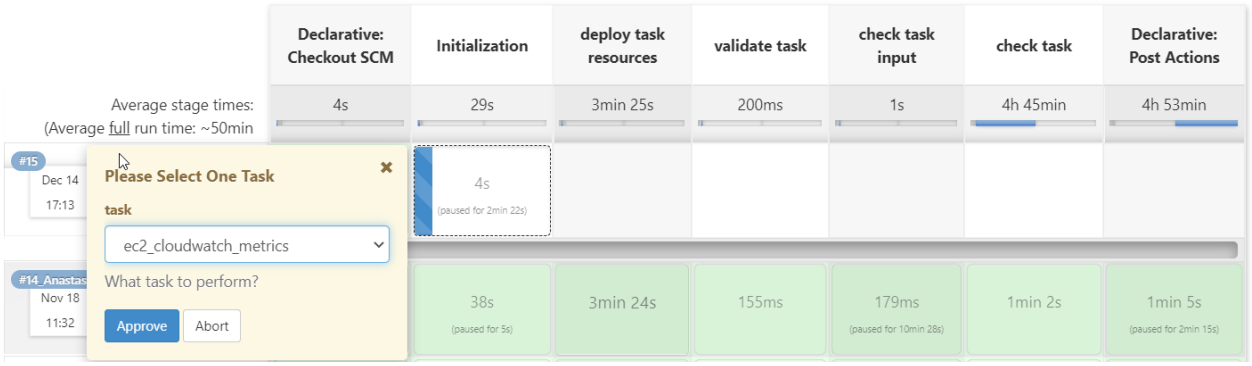

Рисунок 4.1 – Вибір завдання після запуску pipeline

Після вибору завдання студент може подивитися його деталі в секції інтерфейсу Jenkins Task Definition.

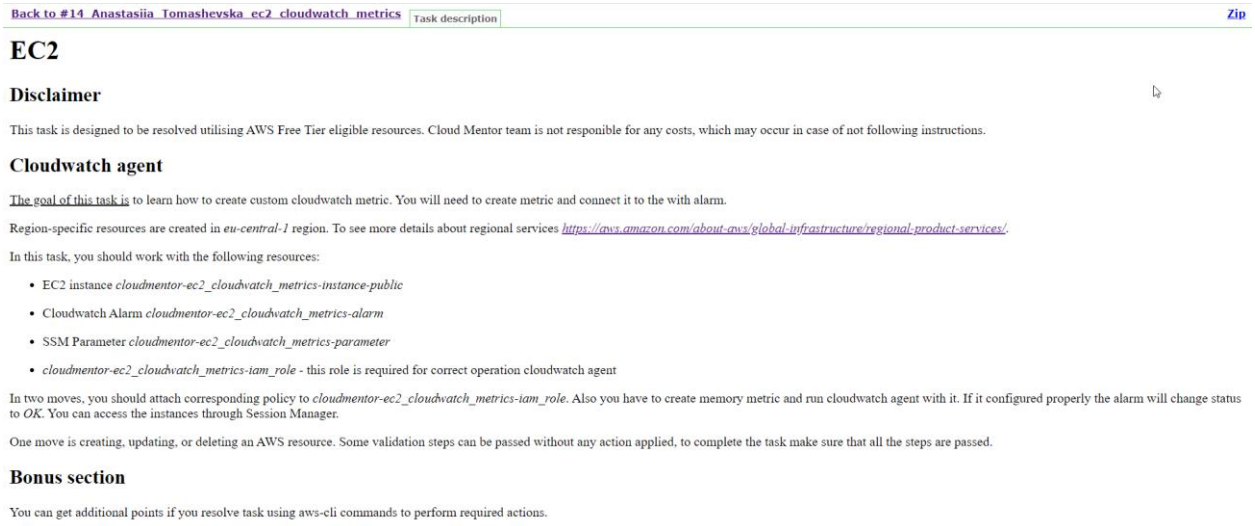

Рисунок 4.2 – Опис завдання cloudwatch_metrics

Крок 1: налаштування користувача в AWS AIM.

За умовами завдання студент має роль *cloudmentor-ec2_cloudwatch_metrics-iam_role* до якої треба приєднати відповідну політику. Для цього треба зайти в консоль AWS, перейти до сервісу AIM, зайти в відповідну роль та додати політику *CloudWatchAgentAdminPolicy*.

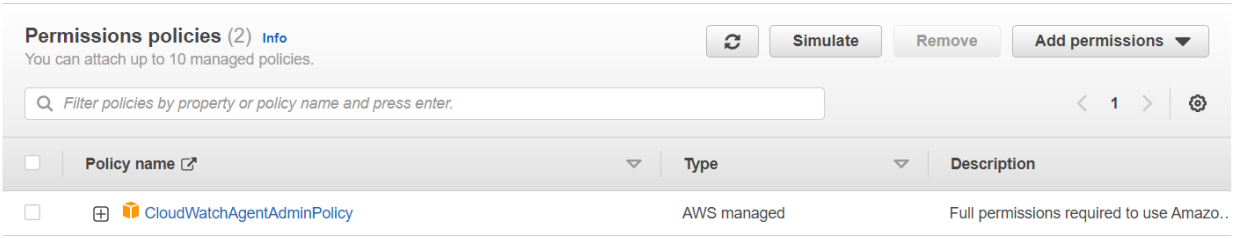

Рисунок 4.2 – Роль з прикріпленою політикою *CloudWatchAgentAdminPolicy*

Крок 2: Оновлення параметру SSM.

Студенту потрібно зайти в сервіс AWS Systems Manager та оновити параметру SSM додавши до нього скрипт описаний на рисунку 4.3.

```
{
  "agent": {
      "metrics_collection_interval": 10,
      "run_as_user": "root"
  },
  "metrics": {
      "namespace": "CWAgent",
      "metrics_collected": {
          "mem": {
          "measurement": [
              "mem_used_percent"
          ]
          },
          "disk": {
          "measurement": [
              "used_percent"
          ],
          "resources": [
              "*"
          ]
          }
      },
      "append_dimensions": {
          "ImageId": "${aws:ImageId}",
          "InstanceId": "${aws:InstanceId}",
          "InstanceType": "${aws:InstanceType}"
      }
  }
}
```

Рисунок 4.3 – Скрипт конфігурація cw_agent_config.json

Крок 3: Запуск агенту.

За допомогою наступних команд студент може інсталювати пакет cloudwatch agent та запустити його.

```
#!/bin/bash

sudo yum install -y amazon-cloudwatch-agent
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c ssm:${ssm_cloudwatch_config}
```

Рисунок 4.4 – Запуск cloudwatch agent

Крок 4: Завершення завдання.

Після виконання всіх попередніх кроків студент повинен повернутися до інтерфейсу Jenkins та виповнити перевірку завдання.

Якщо завдання виконано правильно, то pipeline завершиться успішно.

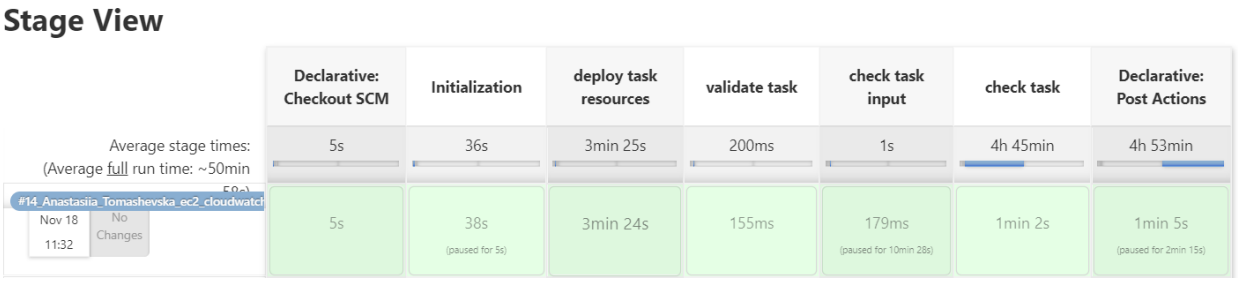

Рисунок 4.4 – Успішне виконання завдання

4.3. Спосіб валідації та перевірки завдання.

Тестування правильності виконання завдання виконується за допомогою скрипту test.sh.

```
#!/bin/bash

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

eval "$(jq -r '@sh "
  AWS_ROLE_ARN=\(.aws_role_arn)
  AWS_REGION=\(.aws_region)
  PUBLIC_INSTANCE_ID=\(.public_instance_id)
  ALARM_NAME=\(.alarm_name)
  TEST_1=\(.test_1)
  LOGS_FOLDER=\(.logs_folder)
  "')"

export PATH=$PATH:/usr/bin:/bin
export AWS_DEFAULT_REGION=${AWS_REGION}
export AWS_PAGER=""

assume_aws_role "${AWS_ROLE_ARN}" "cloud-mentor-validation"

CHECK_ALARM_STATUS=$(aws cloudwatch describe-alarms --alarm-names "${ALARM_NAME}" --state-value OK --region "${AWS_DEFAULT_REGION}")
WAIT_PERIOD=0
while true
do
    WAIT_PERIOD=$(("$WAIT_PERIOD"+10))
    if [ "$WAIT_PERIOD" -gt 180 ] || [[ "$CHECK_ALARM_STATUS" == *"$ALARM_NAME"* ]];then
        break
    else
        sleep 10
        "$CHECK_ALARM_STATUS"
    fi
done
```

Рисунок 4.5 – Скрипт тестування правильності виконання завдання (1 частина)

```
done

command_log "${TEST_1}" "aws cloudwatch describe-alarms --alarm-names ${ALARM_NAME} --state-value OK --region ${AWS_DEFAULT_REGION}" "${LOGS_FOLDER}"

jq -n --arg test_1 "${TEST_1}" \
      --arg logs_folder "${LOGS_FOLDER}" \
      '{
        "test_1":$test_1,
        "logs_folder":$logs_folder
      }'
```

Рисунок 4.6 – Скрипт тестування правильності виконання завдання (2 частина)

Скрипт terraform файлу для налаштувань cloudwatch agent описаний на рисунках 4.7.

```
resource "aws_ssm_parameter" "cw_agent" {
  description = "Cloudwatch agent config to configure custom log"
  name        = module.naming.resource_prefix.ssm_param
  type        = "String"
  value       = var.deploy_task_resources ? file("userdata/cw_agent_config.json") : 0
}

resource "aws_cloudwatch_metric_alarm" "server_mem_used_percent" {
  alarm_name          = module.naming.resource_prefix.cw_alarm
  comparison_operator = "GreaterThanOrEqualToThreshold"
  evaluation_periods  = 1
  metric_name         = "mem_used_percent"
  namespace           = "CWAgent"
  period              = 60
  statistic           = "Average"
  threshold           = 70
  datapoints_to_alarm = 1
  treat_missing_data  = "breaching"
  alarm_description   = "This metric monitors ec2 memory utilization exceeding 70%"

  dimensions = {
    InstanceId   = "${aws_instance.public.id}"
    ImageId      = "${aws_instance.public.ami}"
    InstanceType = "${aws_instance.public.instance_type}"
  }
}
```

Рисунок 4.7 – Налаштування cloudwatch_agent.tf

ВИСНОВКИ

Сенс хмарних технологій полягає в тому, що з їх допомогою вдається надавати необмежений доступ до будь-яких конфігурацій обчислювальних ресурсів та сервісів. Тобто сервери, мережі, додатки, сховища тощо. Все це можна легко і швидко використовувати а потім відмовитися чи видалити. Керування абсолютно нескладне, при цьому не потрібен безпосередній контакт із провайдером.

Приватна хмара (private cloud) являє собою інфраструктуру, якою користується одна компанія, але безпосередньо учасників споживання може бути кілька (підрозділи цієї компанії, її клієнти, підрядники).

Володіти правом власника, управляти і користуватися приватною хмарою може сама компанія або будь-яка третя сторона (або вони можуть робити це спільно). Фізично хмара може розташовуватися в юрисдикції власника або за її межами.

Хмарні технології дають змогу перенести обчислювальні процеси та зберігання даних на віддалені сервери, а користувачеві для доступу до них і роботи з ними достатньо мати вихід в інтернет зі свого особистого пристрою.

В використанні хмарних технологій є ряд переваг, таких як:

– Дають змогу отримати потрібні ресурси за запитом, без придбання обладнання. Хмарні ресурси дають можливість гнучко реагувати на потреби компанії або ситуацію на ринку.

– Дають змогу зняти частину навантаження з IT-відділу. Розгортання серверів та їх адміністрування - непрофільне завдання для більшості бізнесів.

– Дозволяють швидко запускати нові проекти або MVP для стартапів і дослідницьких відділів. Можна орендувати обчислювальні ресурси і з їхньою допомогою швидко розробляти мінімально життєздатні продукти (MVP), щоб тестувати бізнес-гіпотези. Це можуть бути мобільні додатки та ігри, програми на основі машинного навчання, SaaS-рішення, які компанія продає клієнтам.

Хмарні технології чудово підходять для проектів з високою часткою невизначеності, даючи змогу швидко запустити проект, заощадити на обладнанні і не зазнати великих збитків, якщо гіпотеза не спрацює. А якщо спрацює - легко масштабувати рішення.

– Дають змогу знизити ризики під час збоїв і аварій. У хмарі можливі різні сценарії аварійного відновлення: періодичні бекапи, резервна інфраструктура та інші. Відновлення з бекапів, що знімаються з певною періодичністю, підходить, коли утримання резервної інфраструктури обходиться дорожче, ніж простій сервісів протягом якогось часу.

Якщо ж простій неприпустимий, компанії запускають у хмарі резервну інфраструктуру, в яку дані постійно копіюються з основної. У цьому разі простій сервісів після збою якщо і буде, то мінімальний.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is cloud computing https://aws.amazon.com/ru/what-is-cloud-computing/?nc1=f_cc

2. What is Amazon EC2. User Guide. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

3. The Main Benefits & Challenges of Cloud Computing in Education. URL: https://www.buchanan.com/benefits-cloud-computing-education/

4. Gottsegen G. Cloud Computing & Education. https://builtin.com/cloudcomputing/cloud-computing-and-education.

5. The Main Benefits & Challenges of Cloud Computing in Education. https://www.buchanan.com/benefits-cloud-computing-education/.

6. Riddle J. Cloud Technologies in the Education System. https://www.computer.org/publications/tech-news/build-your-career/cloudtechnologies-in-the-education-system.

7. What is CI/CD? RedHat documentation. https://www.redhat.com/en/topics/devops/what-is-ci-cd .

8. Причини переходу в хмару. Microsoft documentation. https://learn.microsoft.com/ru-ru/azure/cloud-adoption-framework/strategy/motivations.

9. Тези доповідей десятої міжнародної науково-технічної конференції на тему «Віртуалізація та віртуальні машини» 24 – 25 листопада 2022 року Том 2: секція 4.