

ДОДАТОК А

Перелік джерел посилання науковими напрямами керівника та науковців кафедри
Програмної інженерії

6. Alkilani M., Kobziev V. Enhancing E-government Services by Using Cloud Computing //CEUR Workshop Proceedings. – 2019. – P.66-69.

9. Лановий О., Кульмінський А. Використання даних як сервісу за допомогою хмарних технологій. БИОНИКА ИНТЕЛЛЕКТА. 2017. № 2. С. 177–182.

Додаток Б
Слайди презентації

Кваліфікаційна робота

«Дослідження методів та технологій розгортання та управління інфраструктурою хмарних застосунків»

Виконав:
Студент групи ІПЗм-22-1
Литовченко Владислав

Науковий керівник:
к.т.н., доц. Бабій А.С

ХНУРЕ 2024

Рисунок Б.1 – Слайд 1 (тема дослідження)

Актуальність роботи:

- Зростання популярності хмарних обчислень
- Зростання популярності DevOps практик та автоматизації
- Зростання кількості оброблюваних даних та необхідність в гнучкості інфраструктури
- Економія фінансів

Рисунок Б.2 – Слайд 2 (актуальність роботи)

Мета роботи:

- Аналіз ринку хмарних обчислень та виявлення викликів в створенні хмарної інфраструктури
- Дослідити та порівняти існуючі методи та принципи створення інфраструктури
- Дослідити та порівняти популярних інструментів-представників найкращого методу
- Більш детально розглянути принцип та деталі роботи найперспективнішого інструменту

Рисунок Б.3 – Слайд 3 (мета роботи)

Екскурс в історію появи «хмари»

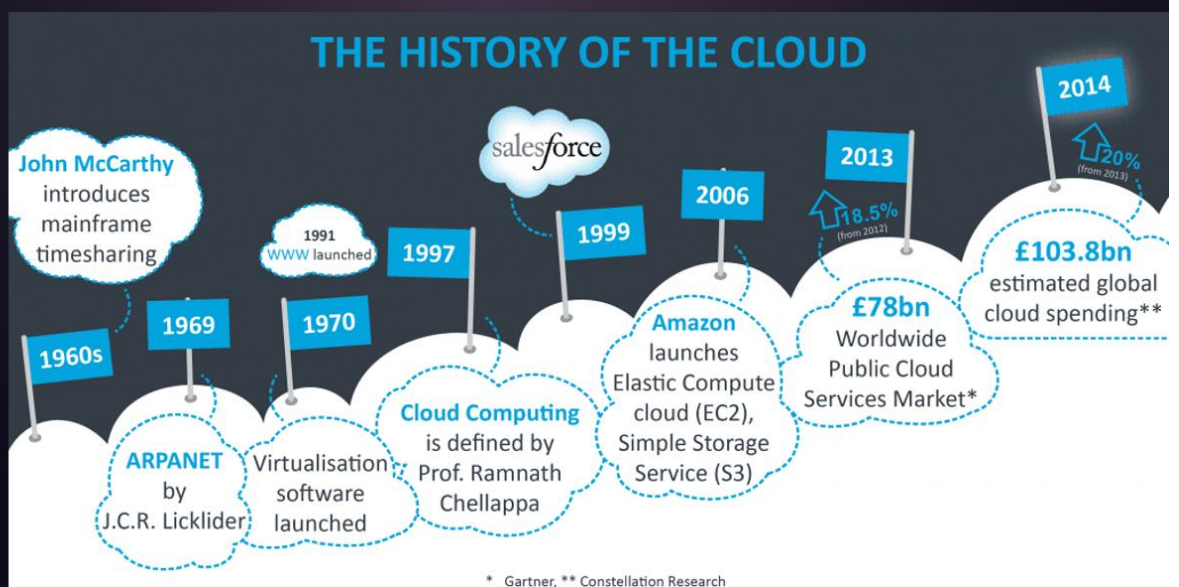



Рисунок Б.4 – Слайд 4 (історія «хмари»)

Принцип надання хмарних послуг

IaaS vs. PaaS vs. SaaS Examples

IaaS	PaaS	SaaS
Amazon Web Services	Google App Engine	HubSpot
Google Cloud	Red Hat OpenShift	JIRA
Microsoft Azure	Heroku	Dropbox
IBM Cloud	Apprenda	DocuSign



The diagram illustrates the principle of cloud service delivery, categorized into IaaS, PaaS, and SaaS. It features three columns of examples, each with a person sitting at a desk in front of a screen. The HubSpot logo is visible in the bottom right corner.

Рисунок Б.5 – Слайд 5 (принцип надання хмарних послуг)

Склад хмарної інфраструктури



The diagram shows the components of cloud infrastructure, represented by four overlapping circles within a larger light blue circle. The components are: Hardware (green circle with a circuit icon), Network (purple circle with a network icon), Storage (blue circle with a database icon), and Virtualization (orange circle with a monitor icon).

Рисунок Б.6 – Слайд 6 (склад хмарної інфраструктури)

Виклики та проблеми

- Зменшення витрат
- Можливі помилки в конфігурації
- Потреба в швидкому та гнучкому масштабуванні інфраструктури за потребою
- Неможливість міграції та відсутність резервних копій
- Час адаптації до інтерфейсу нових хмарних постачальників



Рисунок Б.7 – Слайд 7 (виклики та проблеми)

Infrastructure as a Code

INFRASTRUCTURE as CODE

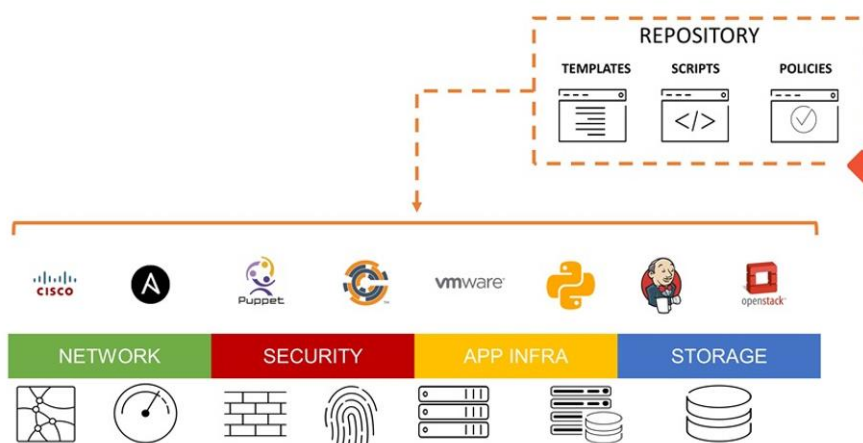


Рисунок Б.8 – Слайд 8 (Infrastructure as a Code)

Declarative vs Imperative

Imperative

```
let arr = [1, 2, 3, 4, 5],
    arr2 = [];

for (var i=0; i<arr.length; i++) {
  arr2[i] = arr[i]*2;
}

console.log(arr2);
```

Declarative

```
let arr = [1, 2, 3, 4, 5];

arr2 = arr.map(function(v, i) {
  return v * 2;
});

console.log(arr2);
```

	Declarative	Imperative
PROS	<ul style="list-style-type: none"> ✔ Less Code ✔ Faster Dev Cycles ✔ Adaptive to changes ✔ Less maintenance 	<ul style="list-style-type: none"> ✔ Flexible ✔ High levels of control
CONS	<ul style="list-style-type: none"> ✔ Domain specific ✔ Difficult to manually optimize ✔ Require annotations to override automated behaviors 	<ul style="list-style-type: none"> ✔ State management ✔ State assumptions in code ✔ Manual optimizations ✔ Integrity checks ✔ Failure management

Рисунок Б.9 – Слайд 9 (declarative vs imperative)

CLI scripts

```
#!/bin/bash
REGION="us-west-2"

INSTANCE_ID=$(aws ec2 run-instances --image-id "ami-0abcdef1234567890" --count 1 --instance-type "t2.micro"
--key-name $KEY_NAME --subnet-id $SUBNET_ID --region $REGION --query 'Instances[0].InstanceId' --output text)
echo "EC2 Instance launched with ID: $INSTANCE_ID"
aws ec2 wait instance-running --instance-ids $INSTANCE_ID --region $REGION
echo "EC2 Instance is running"
PUBLIC_IP=$(aws ec2 describe-instances --instance-ids $INSTANCE_ID --region $REGION --query
'Reservations[0].Instances[0].PublicIpAddress' --output text)
echo "EC2 Instance Public IP: $PUBLIC_IP"
echo "Infrastructure setup complete."
```

Переваги:

- Простота
- Швидкість

Недоліки:

- Важко читається
- Не масштабується
- Потенціальні критичні помилки
- Не підтримує управління станом

Рисунок Б.10 – Слайд 10 (CLI scripts)

Використання SDK

```

import boto3
session = boto3.Session(region_name='us-west-2')
ec2 = session.resource('ec2')
instances = ec2.create_instances(
    ImageId='ami-0abcdef1234567890',
    InstanceType='t2.micro',
    KeyName='your-key-pair',
    NetworkInterfaces=[
        {
            'SubnetId': subnet.id,
            'DeviceIndex': 0,
            'AssociatePublicIpAddress': True,
            'Groups': [security_group.id]
        }
    ]
)
instance = instances[0]
instance.wait_until_running()
print(f"EC2 Instance launched")

```

Переваги:

- Легке тестування
- Підтримка IDE
- Краща читаємість
- Мовна уніфікованість

Недоліки:

- Залежність від певного SDK
- Швидкість
- Не підтримує управління станом
- Потенційні помилки

Рисунок Б.11 – Слайд 11 (Використання SDK)

Використання CDK

```

from aws_cdk import core
import aws_cdk.aws_ec2 as ec2

class MyCDKStack(core.Stack):
    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        instance = ec2.Instance(self, "MyInstance",
            instance_type=ec2.InstanceType.of(ec2.InstanceClass.T2, ec2.InstanceSize.MICRO),
            machine_image=ec2.MachineImage.generic_linux({"us-west-2": "ami-0abcdef1234567890" }),
            key_name="your-key-pair",
            vpc=vpc,
            vpc_subnets={"subnet_type": ec2.SubnetType.PUBLIC}
        )
        core.CfnOutput(self, "InstanceID", instance.instance_id)
app = core.App()
MyCDKStack(app, "MyCDKStack", env={'region': 'us-west-2'})
app.synth()

```

Переваги:

- Різноманітність мов
- Краща читаємість за рахунок декларативності
- Підтримка стану
- Перегляд потенційних змін
- Зменшення шансу помилки
- Можливість «відкачування» змін
- Модульність

Недоліки:

- Залежність від провайдера
- Швидкість
- Затримки в додаванні нових функцій
- Зручний перегляд розгорнутих ресурсів

Рисунок Б.12 – Слайд 12 (використання CDK)

Використання vendor-neutral інструментів

```

provider "aws" {
  region = "us-west-2"
}
resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafa1f0"
  instance_type = "t2.micro"
  tags = {
    Name = "ExampleInstance"
  }
}

```

Переваги:

- Підтримка багатьох провайдерів
- Краща читаємість за рахунок декларативності
- Можливість використання не в хмарному контексті
- Open-source, велика спільнота
- Легка інтеграція з CI/CD
- Безкоштовність

Недоліки:

- Безпека
- Швидкість
- Затримка в додаванні нових функцій/сервісів хмарних провайдерів

Рисунок Б.13 – Слайд 13 (використання vendor-neutral інструментів)

Порівняння методів згідно оцінок користувачів (малий бізнес)

Критерій/Продукт	AWS CLI	AWS CDK	Terraform	Web-console
Задовольняє потреби	<u>9.3</u>	8.6	8.9	8.9
Легкість використання	<u>9.0</u>	8.5	8.5	8.3
Легкість створення	N/A	7.5	8.3	<u>8.8</u>
Легкість адміністрування	N/A	7.7	<u>8.3</u>	7.9
Якість підтримки	8.3	<u>8.5</u>	8.2	7.3
Чи був продукт корисний для бізнесу?	N/A	8.0	N/A	<u>8.3</u>
Продуктовий напрям(% позитивності)	<u>10.0</u>	<u>10.0</u>	9.2	8.4

Рисунок Б.14 – Слайд 14 (порівняння методів для малого бізнесу)

Порівняння методів згідно оцінок користувачів (великий бізнес)

Критерій/Продукт	AWS CLI	AWS CDK	Terraform	Web-console
Задовольняє потреби	8.6	9.0	9.2	8.5
Легкість використання	8.6	8.1	8.7	8.5
Легкість створення	N/A	8.1	9.2	N/A
Легкість адміністрування	N/A	8.1	9.5	N/A
Якість підтримки	8.3	8.3	8.5	8.3
Чи був продукт корисний для бізнесу?	N/A	8.5	9.5	N/A
Продуктовий напрям(% позитивності)	8.3	9.2	10.0	6.1

Рисунок Б.15 – Слайд 15 (порівняння методів для великого бізнесу)



Рисунок Б.16 – Слайд 16 (Terraform)

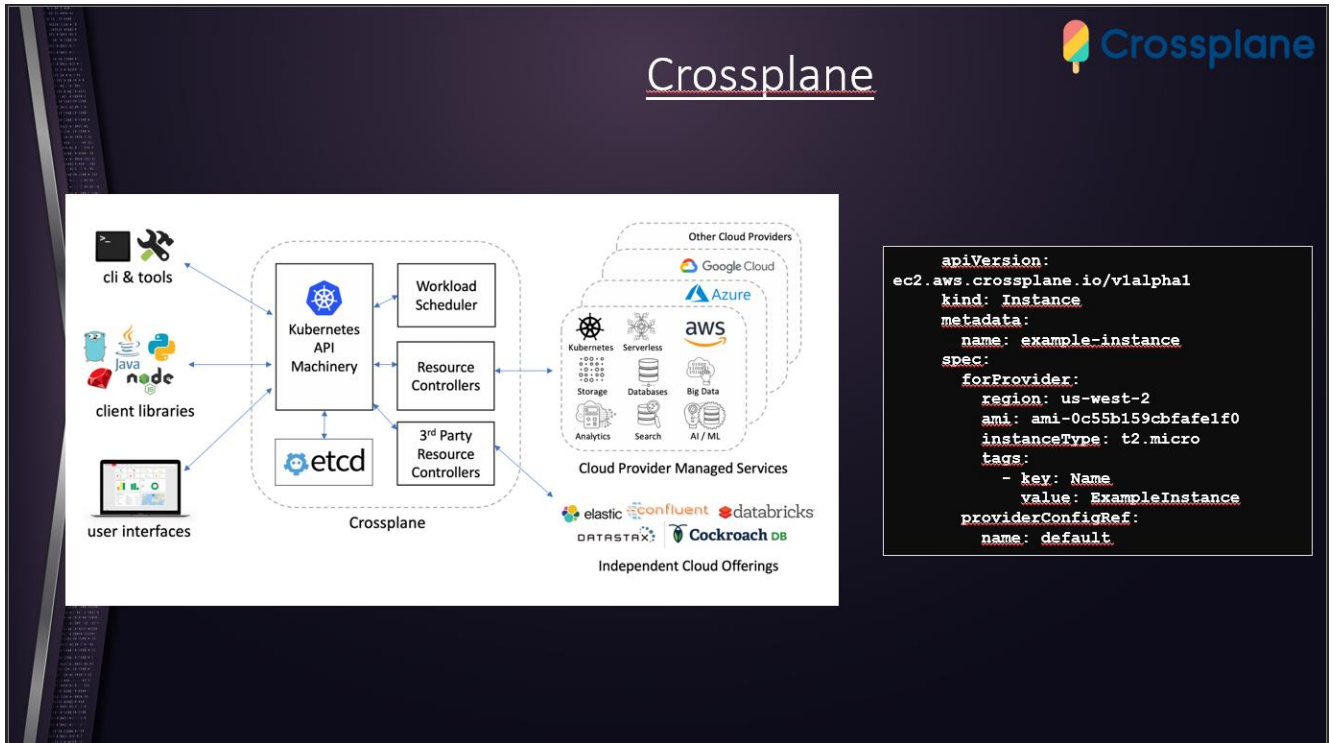


Рисунок Б.17– Слайд 17 (Crossplane)

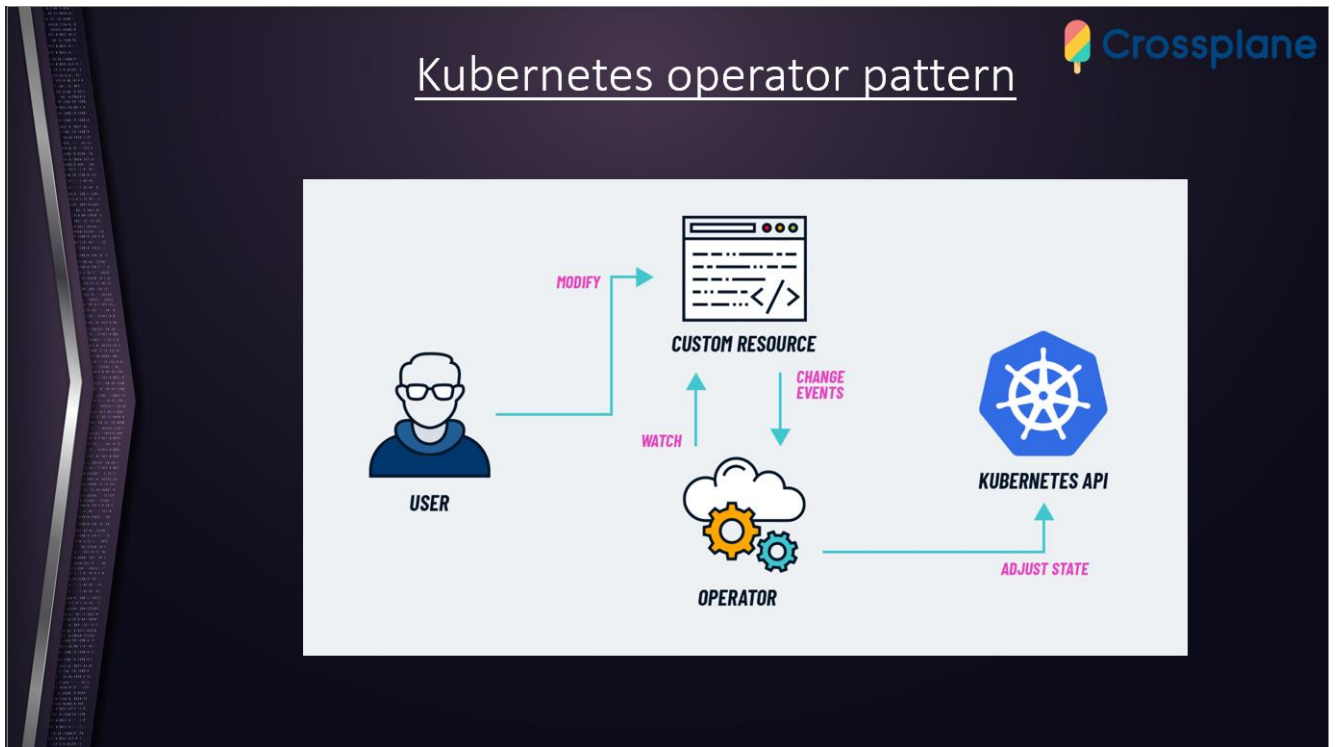


Рисунок Б.18 – Слайд 18 (Kubernetes operator pattern)

Crossplane vs Terraform

Критерій	Terraform	Crossplane
Велика спільнота	+	+/-
Планування змін перед розгортанням	+	-
Легке перенесення стану в інші місця	+	-
Можливість паралельної роботи	-	+
Повторні спроби розгортання при помилці	-	+
Автоматичне виявлення та відкат "drift changes"	-	+
Паралелізм	Асинхронність	Паралелізм за типом ресурсу
Зручна інтеграція з CI/CD	+/-	+
Вбудований RBAC	-	+
Вбудовані можливості масштабування	-	+

Рисунок Б.19 – Слайд 19 (Crossplane vs Terraform)

Швидкість

Terraform

Кількість\Операція	Створення	Видалення
1	15.87 с	14.03 с
30	38.12 с	27.27 с
100	109.75 с	75.97 с
250	224.27 с	176.63 с
500	434.72 с	402.25 с

Crossplane

Кількість\Операція	Створення	Видалення
1	11.74 с	24.40 с
30	21.07 с	27.98 с
100	35.98 с	43.04 с
250	61.74 с	77.38 с
500	116.40 с	150.55 с

Рисунок Б.20 – Слайд 20 (порівняння швидкості)

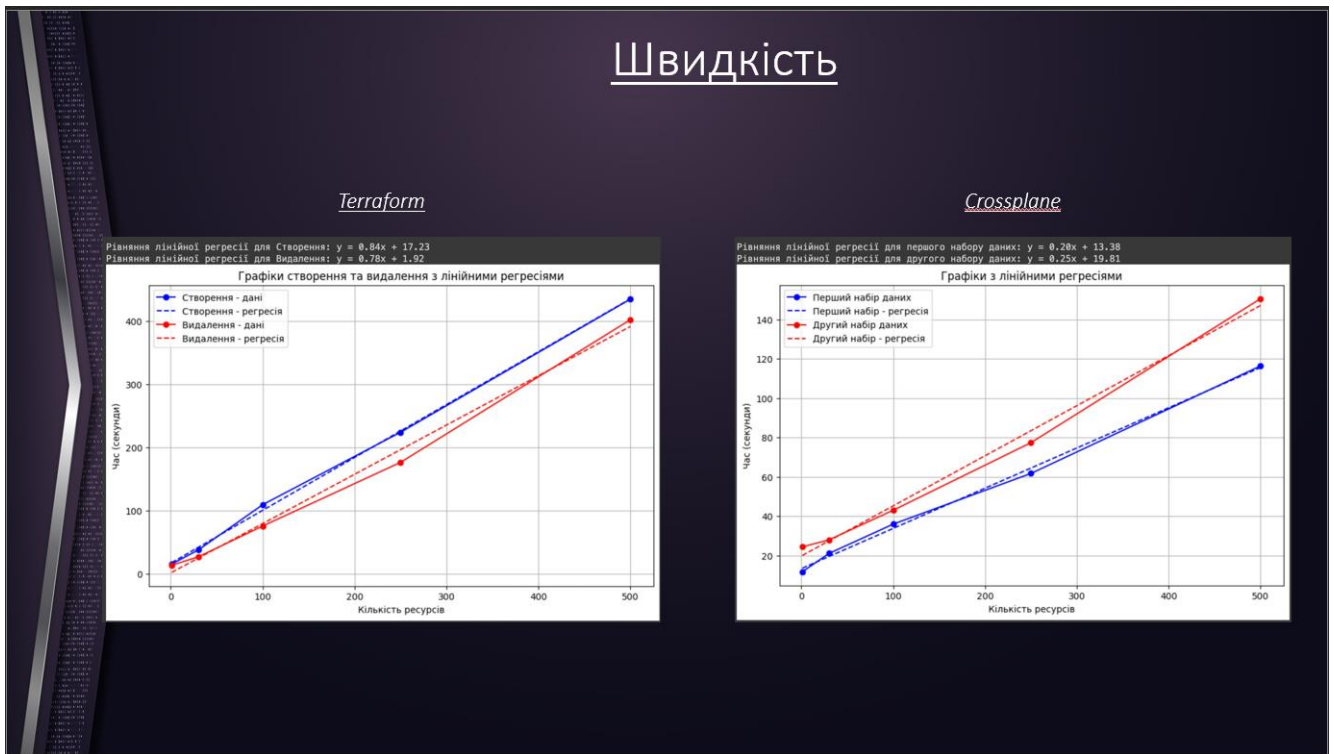


Рисунок Б.21 – Слайд 21 (порівняння швидкості)

Імплементация провайдера Crossplane

Кроки:

1. Створення Custom Resource Definition GoLang структури
2. Створення Custom Resource Definition для Provider Config
3. Створення параметрів та статусу відносно реального ресурсу
4. Створення оператора
 - a) Отримання облікових даних з Provider Config CRD
 - b) Реалізація методів Observe, Create, Update, Delete
5. Генерація yamI для Custom Resource Definition з 1 та 2 кроку
6. Застосування yamI маніфестів на Kubernetes кластерів
7. Створення docker image з операторами та створення поду на кластері

Рисунок Б.22 – Слайд 22 (для імплементации провайдера Crossplane)

Імплементація провайдера Crossplane

Структури для конфігурації Network CRD

```
type NetworkParameters struct {
    Name string json:"name" // Required
    Subnet []*Subnet json:"subnets" // Optional
    // +crossplane:generate:reference:type=topology/v1.Datacenter
    // +crossplane:generate:reference:refFieldName=DatacenterIDRef
    // +optional
    DatacenterID string json:"datacenter"
    DatacenterIDRef *xpv1.Reference json:"datacenterIdRef,omitEmpty"
    DatacenterIDSelector *xpv1.Selector json:"datacenterIdSelector,omitEmpty"
}

type Subnet struct {
    IP string json:"ip" // Required
    Bit int json:"bit" // Required
    Gateway *string json:"gateway,omitEmpty" // Optional
    ...
}

type NetworkObservation struct {
    Subnets []*SubnetStatus json:"subnets,omitEmpty"
    Id int json:"id"
}

type SubnetStatus struct {
    Datacenter string json:"datacenter"
    VlanID int json:"vlanId"
    SubnetID int json:"subnetId"
    StartRange string json:"startRange"
    EndRange string json:"endRange"
    SubnetIP string json:"subnetIp"
    ...
}
```

Рисунок Б.23 – Слайд 23 (CRD для імплементації Network)

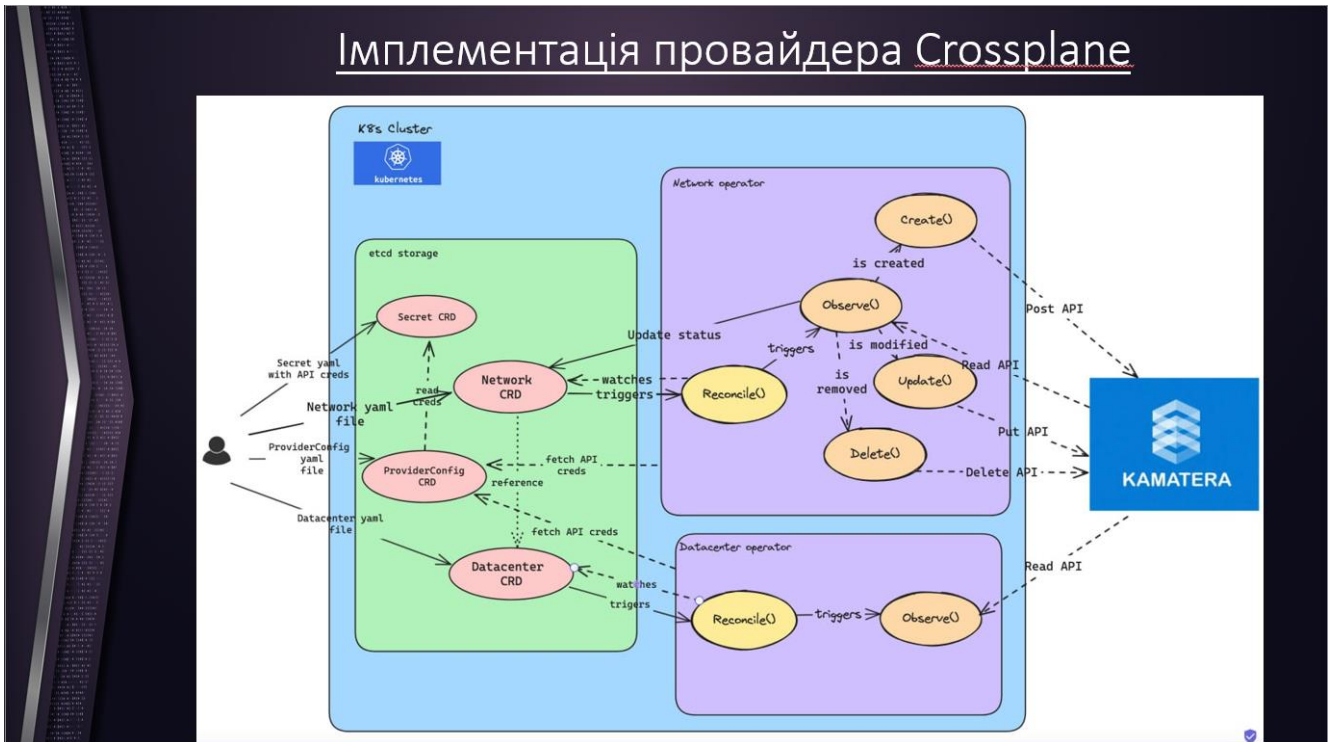


Рисунок Б.24 – Слайд 24 (діаграма роботи реалізованого провайдера)

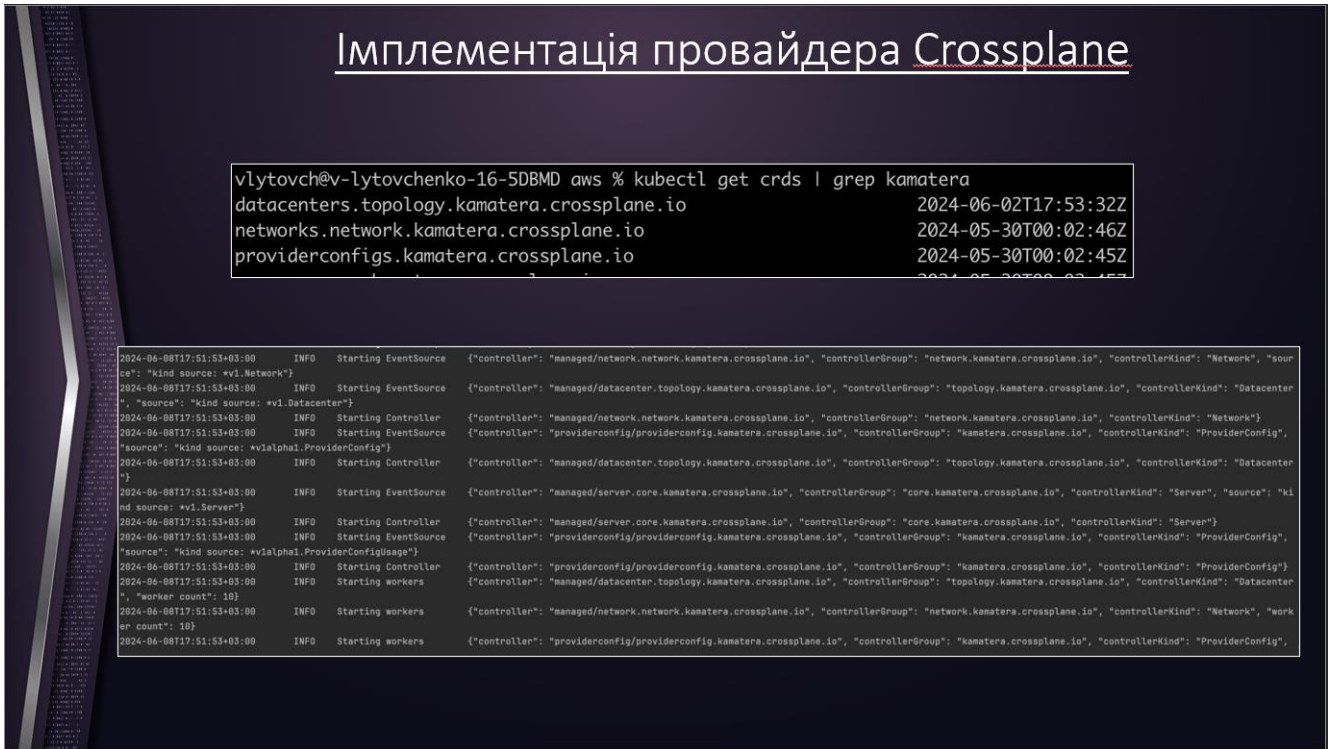


Рисунок Б.25 – Слайд 25 (робота провайдера на кластері)

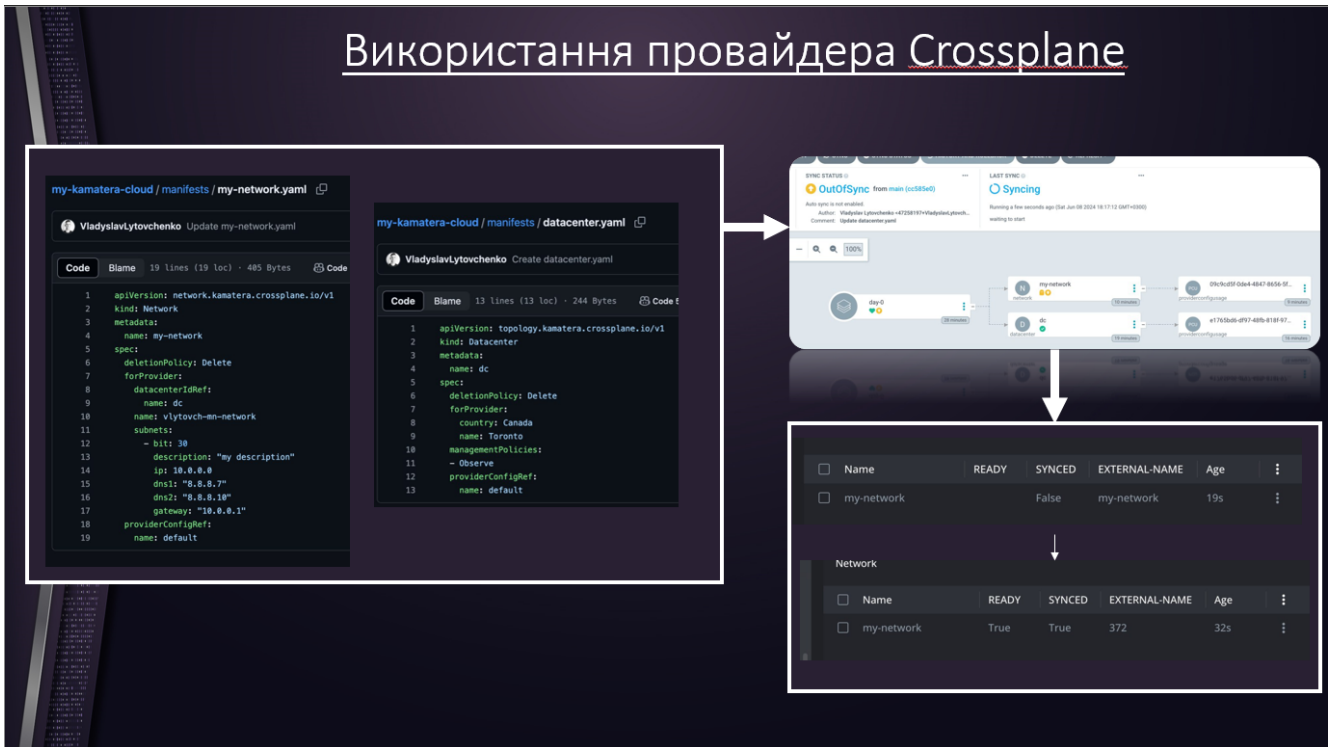


Рисунок Б.26 – Слайд 26 (схема використання провайдера)

Використання провайдера Crossplane

The screenshot displays the 'My Cloud' NETWORK MANAGEMENT interface. The 'Zone' is set to 'CA-TR - Toronto, Canada'. A search filter is applied for 'lan-4934155-vlytovch-mn-network'. The interface shows a table with network details:

Name	ID	Actions
lan-4934155-vlytovch-mn-...	372	Actions Close

Below the table, it indicates 'Showing 1 - 1 of 1 networks.' and '10 Per page'. A green '+ Create New Network' button is visible.

The detailed view for 'lan-4934155-vlytovch-mn-network' shows the following configuration:

IP Address	Subnet Mask	Range	Gateway	Description	Actions
10.0.0.0/30	255.255.255.252	10.0.0.1 - 10.0.0.2 1 used of 2 addresses	10.0.0.1 DNS: 8.8.8.7, 8.8.8.10	my description	Open

An 'Add IP Address Scope' button is located at the bottom right of the detailed view.

Рисунок Б.27 – Слайд 27 (створений ресурс на хмарі)

ВИСНОВКИ



Рисунок Б.28 – Слайд 28 (висновки)

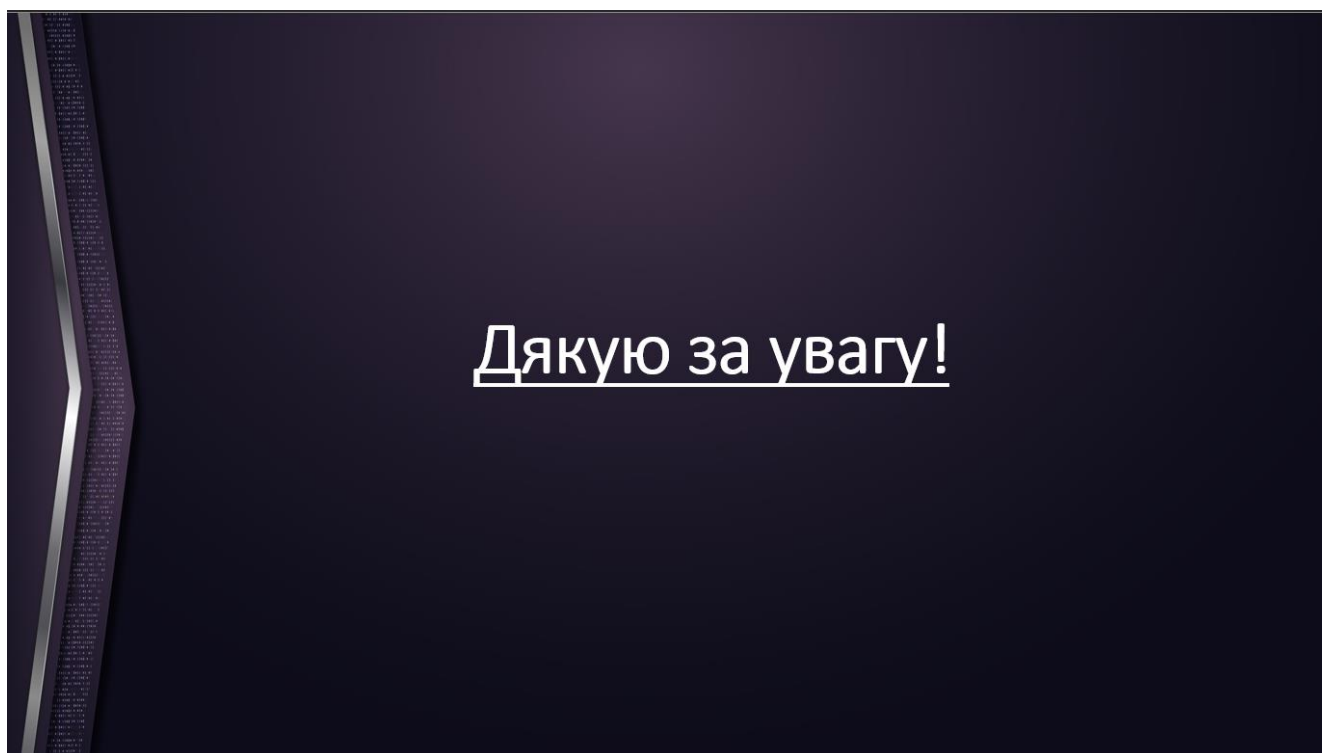


Рисунок Б.29 – Слайд 29 (подяка)

ДОДАТОК В

Апробація у вигляді тез



Рисунок В.1 – Титульна сторінка тез

Льчишин М.З., Запісоцький І.В., Матвієвський Н.А. НАПРЯМИ РОЗВИТКУ ІННОВАЦІЙНОГО ПІДПРИЄМНИЦТВА В УКРАЇНІ.....	62
SECTION: FINANCE AND BANKING	
Білоцерківський О., Кузьмичов О. УПРАВЛІННЯ ФІНАНСОВИМИ АКТИВАМИ ТОРГОВЕЛЬНОГО ПІДПРИЄМСТВА.....	66
Sharenko M. LEGAL REGULATION OF FINTECH IN UKRAINE AND THE EUROPEAN UNION.....	69
SECTION: INFORMATION TECHNOLOGY & CYBERSECURITY	
Литовченко В.Ю., Бабій А.С. МЕТОДИ ТА ІНСТРУМЕНТИ АВТОМАТИЗАЦІЇ СТВОРЕННЯ ТА УПРАВЛІННЯ ХМАРНОЮ ІНФРАСТРУКТУРОЮ.....	72
Ллюхіна К.В., Бутенко О.С. ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ У РОЗМІНУВАННІ СІЛЬСЬКОГОСПОДАРСЬКИХ ТЕРИТОРІЙ.....	74
Стеф'юк В., Штаср Л., Белей О. ІНФОРМАЦІЙНИЙ МОДУЛЬ ОПРАЦЮВАННЯ ЦИФРОВИХ ЗОБРАЖЕНЬ З ТЕПЛОВІЗОРА.....	77
Hunko I. LACK OF REQUIREMENTS TESTING AS A FACTOR IN INCREASING COSTS AND DELAYS IN THE DEVELOPMENT PROCESS.....	83
Mazurets O., Uspenska K., Vit R., Tyschenko O. INTELLIGENT SYSTEM FOR DETERMINING THE OBJECT ATTRIBUTES VALUES BY NEURAL NETWORKS MEANS BY GRAPHIC IMAGES IN DATABASES.....	86
Кириченко І.В., Олійник А.Є. ДОСЛІДЖЕННЯ МЕТОДІВ СТИСНЕННЯ ЗОБРАЖЕНЬ В БЛОКЧЕЙН СХОВИЩАХ.....	92

**SECTION: INFORMATION TECHNOLOGY AND
CYBERSECURITY****МЕТОДИ ТА ІНСТРУМЕНТИ АВТОМАТИЗАЦІЇ
СТВОРЕННЯ ТА УПРАВЛІННЯ ХМАРНОЮ
ІНФРАСТРУКТУРОЮ****Литовченко Владислав Юрійович**

здобувач вищої освіти

Факультет комп'ютерних наук

vladyslav.lytovchenko@nure.ua

Бабій Андрій Степанович

доцент

Кафедра програмної інженерії

andrii.babii@nure.ua

Національний технічний університет

«Харківський національний університет радіоелектроніки», Україна

У сучасному динамічному світі підприємства стикаються з постійною потребою в масштабуванні та гнучкості своїх ІТ-систем. Хмарні технології пропонують нові можливості для вирішення цих проблем, надаючи доступ до ресурсів за потребою та економлячи час і кошти. Однак, ручне створення та управління хмарною інфраструктурою може бути складним і трудомістким процесом, що може призвести до помилок, неефективності та затримок. Автоматизація створення та управління хмарною інфраструктурою може значно покращити роботу компаній.

Одним із популярних імперативних методів є Command Line Interface[1]. За допомогою CLI можна автоматизувати велику кількість завдань, починаючи від створення віртуальних машин та мережевих ресурсів, і закінчуючи управлінням контейнерами і розгортанням застосунків в хмарному середовищі. Використання команд під час роботи з CLI дозволяє швидко та ефективно виконувати завдання, які раніше можливо було виконати лише через графічний інтерфейс. Крім того, використання CLI у сфері хмарних технологій дозволяє інженерам працювати зі своїми конфігураціями як з кодом. Це важливо для ведення інфраструктурного коду, де весь конфігураційний код системи знаходиться у версійному контролі і може бути легко відслідкований, виправлений та відтворений.

Наступним імперативним підходом для створення інфраструктури є використання SDK інструментів[2]. SDK (набір розробника) представляє собою набір інструментів, бібліотек, API та ресурсів, які створені для спрощення розробки додатків та сервісів, орієнтованих на роботу у хмарному середовищі конкретного провайдера. Такий метод є дуже схожим на використання CLI, але

головними перевагами є те, що використовується будь яка комфортна існуюча мова розробки замість навантажених команд інтерфейсу командного рядку. Завдяки популярним середовищам для розробки (IDE) можна набагато швидше та надійніше писати алгоритм конфігурації, так як середовище буде постійно підказувати та вказувати на можливі помилки в коді.

Одним із декларативних методів можна вважати використання вбудованих "Infrastructure as Code" інструментів від хмарних провайдерів[3]. Прикладами таких інструментів є Cloud Formation для AWS, Cloud Deployment Manager для Google Cloud тощо. На відміну від минулих підходів тут ви описуєте бажаний стан вашої інфраструктури, а не послідовність дій для його створення чи конфігурації. Одним із головних переваг цього підходу є те, що такі інструменти автоматично підтримують та зберігають поточний стан системи. Тобто, якщо ми вносимо певні зміни в файл конфігурації, сервіс автоматично розуміє, які операції потрібно провести над зміненими ресурсами, щоб привести їх до вигляду, який потрібен користувачу. Крім того, в сервісі можна чітко побачити, які компоненти і ресурси на даний час відносяться до того чи іншого стеку та як саме вони взаємодіють.

Іншим схожим декларативним методом є використання зовнішніх IaC інструментів, а не наданих хмарним провайдером. Прикладом інструментів для використання такого підходу є Crossplane, Terraform, Pulumi тощо[4]. Такі сторонні інструменти дозволяють розробникам визначати конфігурацію інфраструктури як коду. Це може включати в себе віртуальні машини, бази даних, мережеві налаштування та інші ресурси.

Щоб описати цю конфігурацію, розробники можуть використовувати зручні та знайомі їм мови програмування, такі як HCL (HashiCorp Configuration Language) для Terraform, YAML для Ansible тощо. Сторонні інструменти надають можливість використовувати модульні конфігураційні файли або шаблони, щоб повторно використовувати певні частини конфігурації та створювати більш складні інфраструктурні рішення.

Після написання коду конфігурації, розробники можуть використовувати ці інструменти для автоматизації розгортання інфраструктури на різноманітних хмарних платформах, таких як AWS, Azure чи Google Cloud або інших, в залежності від існування відповідного хмарного провайдера на сайті інструмента.

Інструменти постійно підтримують стан інфраструктури, що дозволяє виявляти зміни та вносити в них корективи. Це сприяє гарантуванню відповідності стану до коду конфігурації.

Конфігураційні файли можуть легко інтегруватися в процеси неперервної інтеграції та постачання (CI/CD), що дозволяє автоматично та безперервно розгортати та оновлювати інфраструктуру при змінах в коді додатків.

Такі інструменти дуже схожі на вищезгадані вбудовані IaC інструменти, але мають ряд переваг таких як універсальність, кращий синтаксис, розвинені плагіни та модулі, велике ком'юніті тощо.

Current Trends in the Development of Scientific Research in Today's Conditions

Автоматизація створення та управління хмарною інфраструктурою є ключовим фактором успіху для підприємств, які прагнуть до гнучкості, ефективності та економії коштів. Завдяки автоматизації можна значно покращити роботу IT-систем, підвищити продуктивність та отримати конкурентну перевагу. Зважаючи на великий вибір інструментів та методів автоматизації буде доречним провести дослідження для більш поглибленого вивчення проблеми.

Список використаних джерел

1. What is a CLI? - command line interface explained - AWS. Amazon Web Services, Inc. URL: <https://aws.amazon.com/what-is/cli/> (дата звернення: 20.05.2024).
2. Software development kit - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Software_development_kit (дата звернення: 20.05.2024).
3. What is infrastructure as code? - Azure DevOps. Microsoft Learn: Build skills that open doors in your career. URL: <https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code> (дата звернення: 20.05.2024).
4. Crossplane vs terraform - iac tools comparison. Spacelift. URL: <https://spacelift.io/blog/crossplane-vs-terraform> (дата звернення: 20.05.2024).

ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ У РОЗМІНУВАННІ СІЛЬСЬКОГОСПОДАРСЬКИХ ТЕРИТОРІЙ

Ілюхіна К.В.

аспірант

Бутенко О.С.

д-р. техн. наук, професор

Кафедра 407

Національний аерокосмічний університет

ім. М.Є. Жуковського «ХАІ»

Від початку повномасштабного вторгнення Росії Україна випередила Афганістан і Сирію та стала найбільш замінованою країною на землі.

Третина країни всяєна мільйонами нерозірваних мін і касетних бомб, а також дротами, мінами-пастками й осколками снарядів.

Величезні мінні поля поставили під загрозу життя близько шести мільйонів мирних жителів і зробили непридатними частинами найцінніших сільськогосподарських угідь країни, що вплинуло як на економіку країни, так і на світове постачання продовольства.



Рисунок В.4 – Сертифікат про участь у конференції

Додаток Г

Фрагменти коду

```

118 func (c *external) Observe(ctx context.Context, mg resource.Managed) (managed.ExternalObservation, error) {
119     cr, ok := mg.(*v1.Network)
120     if !ok { return managed.ExternalObservation{}, errors.New(errNotNetwork) }
121
122     fmt.Printf("Observing: %v", cr.Name)
123     id := meta.GetExternalName(cr)
124
125     result, err := client2.Request(c.service.Provider, method: "GET", fmt.Sprintf("service/networks?datacenter=%s", cr.Spec.ForProvider.DatacenterID))
126     if err != nil { return managed.ExternalObservation{}, errors.Wrap(err, message: "error during networks fetch") }
127     var network map[string]interface{}
128     networks := result.([]interface{})
129     for _, network_ := range networks {
130         network__ := network_.(map[string]interface{})
131         if fmt.Sprintf("v%v", network__["vlanId"]).(float64) == id {
132             network = network__
133             break
134         }
135     }
136     if network == nil {
137         return managed.ExternalObservation{
138             ResourceExists: false,
139         }, nil
140     }
141
142     cr.Status.AtProvider.Id = int(network["ids"].([]interface{})[0]).(float64)
143
144     subnetsResult, err := client2.Request(c.service.Provider, method: "GET", fmt.Sprintf("service/network/subnets?datacenter=%s&vlanId=%s", cr.Spec.ForProvider.DatacenterID, cr.Status.AtProvider.VlanID))
145     if err != nil { return managed.ExternalObservation{ResourceExists: true}, errors.Wrap(err, message: "couldn't get subnets") }
146     var subnets []v1.SubnetStatus
147     for _, subnet_ := range subnetsResult.([]interface{}) {
148         subnet__ := subnet_.(map[string]interface{})
149         dns1 := c.setNilableValue(subnet__, field: "dns1")
150         dns2 := c.setNilableValue(subnet__, field: "dns2")
151         gateway := c.setNilableValue(subnet__, field: "gateway")
152         subnets = append(subnets, &v1.SubnetStatus{
153             Datacenter: cr.Spec.ForProvider.DatacenterID,
154             VlanID: int(subnet__["vlanId"]).(float64),
155             SubnetID: int(subnet__["subnetId"]).(float64),
156             StartRange: subnet__["startRange"].(string),
157             EndRange: subnet__["endRange"].(string),
158             SubnetIp: subnet__["subnetIp"].(string),
159             SubnetBit: int(subnet__["subnetBit"]).(float64),
160         })
161     }
162     cr.Status.AtProvider.Subnets = subnets
163     isChanged, err := IsNetworkDifferent(cr)
164     if err != nil {
165         return managed.ExternalObservation{ResourceExists: true, ResourceUpToDate: false}, errors.Wrap(err, message: "couldn't get ch

```

Рисунок Г.1 – Реалізація Observe методу

```

165         SubnetBit: int(subnet__["subnetBit"]).(float64),
166         SubnetDescription: subnet__["subnetDescription"].(string),
167         DNS1: dns1,
168         DNS2: dns2,
169         Gateway: gateway,
170         InUse: int(subnet__["inUse"]).(float64),
171     })
172 }
173
174 cr.Status.AtProvider.Subnets = subnets
175 isChanged, err := IsNetworkDifferent(cr)
176 if err != nil {
177     return managed.ExternalObservation{ResourceExists: true, ResourceUpToDate: false}, errors.Wrap(err, message: "couldn't get ch
178 }
179 cr.SetConditions(xpv1.Available())
180 return managed.ExternalObservation{
181     ResourceExists: true,
182     ResourceUpToDate: !isChanged,
183     ConnectionDetails: managed.ConnectionDetails{},
184 }, nil
185 }

```

Рисунок Г.2 – Продовження реалізації Observe методу

```

176 func (c *external) Create(ctx context.Context, mg resource.Managed) (managed.ExternalCreation, error) {
177     cr, ok := mg.(*v1.Network)
178     if !ok { return managed.ExternalCreation{}, errors.New(errNotNetwork) }
179
180     cr.SetConditions(xpv1.Creating())
181     fmt.Printf("Creating: %v", cr)
182
183     subnets := cr.Spec.ForProvider.Subnet
184     if len(subnets) < 1 {
185         return managed.ExternalCreation{}, errors.New(message: "when creating a new network, at least 1 subnet is required")
186     }
187     creation, err2 := c.checkDescriptions(subnets)
188     if err2 != nil {
189         return creation, err2
190     }
191     firstSubnet := subnets[0]
192     gateway := c.setValue(firstSubnet, firstSubnet.Gateway)
193     dns1 := c.setValue(firstSubnet, firstSubnet.DNS1)
194     dns2 := c.setValue(firstSubnet, firstSubnet.DNS2)
195     body := &client2.CreateNetworkPostValues{Datacenter: cr.Spec.ForProvider.DatacenterID, Name: cr.Spec.ForProvider.Name, SubnetIP: firstSubnet.IP,
196         SubnetBit: firstSubnet.Bit, Gateway: gateway, Dns1: dns1, Dns2: dns2, SubnetDescription: firstSubnet.Description}
197
198     result, err := client2.Request(c.service.Provider, method: "POST", path: "service/network/create", body)
199     if err != nil {
200         return managed.ExternalCreation{}, errors.Wrap(err, message: "Error during network creation")
201     }
202     res, externalCreation, err3, done := c.parseResponse(result, err)
203     if done {
204         return externalCreation, err3
205     }
206     id := int(res["networkId"].(float64))
207     meta.SetExternalName(cr, strconv.Itoa(id))
208     for _, subnet := range subnets {
209         if subnet.Description != firstSubnet.Description {
210             err := addSubnet(c.service.Provider, subnet, cr.Spec.ForProvider.DatacenterID, strconv.Itoa(id))
211             if err != nil {
212                 return managed.ExternalCreation{}, err
213             }
214         }
215     }
216     return managed.ExternalCreation{}, nil
217 }

```

Рисунок Г.3 – Реалізація Create методу

```

func (c *external) Update(ctx context.Context, mg resource.Managed) (managed.ExternalUpdate, error) {
    cr, ok := mg.(*v1.Network)
    if !ok { return managed.ExternalUpdate{}, errors.New(errNotNetwork) }

    fmt.Printf("Updating: %v", cr.Name)
    cr.SetConditions(Updating())
    oldSubnets := cr.Status.AtProvider.Subnets
    newSubnets := cr.Spec.ForProvider.Subnet
    newSubnetsByDescription := make(map[string]*v1.Subnet)
    oldSubnetsByDescription := make(map[string]*v1.SubnetStatus)
    for _, newSubnet := range newSubnets {
        newSubnetsByDescription[newSubnet.Description] = newSubnet
    }
    for _, oldSubnet := range oldSubnets {
        oldSubnetsByDescription[oldSubnet.SubnetDescription] = oldSubnet
    }
    if len(oldSubnets) != len(oldSubnetsByDescription) || len(newSubnets) != len(newSubnetsByDescription) {
        return managed.ExternalUpdate{}, errors.New(message: "Invalid subnet descriptions - cannot identify unique subnets based on descriptions")
    }

    for description, newSubnet := range newSubnetsByDescription {
        oldSubnet, oldExists := oldSubnetsByDescription[description]
        if oldExists {
            if isSubnetDifferent(oldSubnet, newSubnet) {
                err := editSubnet(c.service.Provider, newSubnet, oldSubnet.Datacenter, oldSubnet.VlanID, oldSubnet.SubnetID)
                if err != nil { return managed.ExternalUpdate{}, err }
            }
        } else {
            err := addSubnet(c.service.Provider, newSubnet, cr.Spec.ForProvider.DatacenterID, meta.GetExternalName(cr))
            if err != nil { return managed.ExternalUpdate{}, err }
        }
    }

    for description, oldSubnet := range oldSubnetsByDescription {
        _, newExists := newSubnetsByDescription[description]
        if !newExists {
            err := delSubnet(c.service.Provider, oldSubnet)
            if err != nil { return managed.ExternalUpdate{}, err }
        }
    }
    return managed.ExternalUpdate{}, nil
}

```

Рисунок Г.4 – Реалізація Update методу

```
21 func (c *external) Delete(ctx context.Context, mg resource.Managed) error {  
22     cr, ok := mg.(*v1.Network)  
23     if !ok { return errors.New(errNotNetwork) }  
24  
25     cr.SetConditions(xpv1.Deleting())  
26  
27     fmt.Printf(format: "Deleting: %+v", cr.Name)  
28  
29     for _, subnet := range cr.Status.AtProvider.Subnets {  
30         err := delSubnet(c.service.Provider, subnet)  
31         if err != nil { return err }  
32     }  
33  
34     body := &client2.DeleteNetworkPostValues{  
35         Datacenter: cr.Spec.ForProvider.DatacenterID,  
36         Id:           cr.Status.AtProvider.Id,  
37     }  
38  
39     _, err := client2.Request(c.service.Provider, method: "POST", path: "service/network/delete", body)  
40     if err != nil { return errors.Wrap(err, message: "error during network delete") }  
41  
42     return nil  
43 }
```

Рисунок Г.5 – Реалізація Delete методу

Додаток Д

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

Дата перевірки:
11.06.2024 08:50:19 EEST

Дата звіту:
11.06.2024 08:57:39 EEST

ID перевірки:
1016345940

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗм-22-1_Литовченко_В_Ю_скорочений

Кількість сторінок: 66 Кількість слів: 13745 Кількість символів: 102007 Розмір файлу: 2.25 MB ID файлу: 1016147704

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

1.52%
Схожість

Найбільша схожість: 0.39% з Інтернет-джерелом (http://kafedratn opd.inf.ua/user-files/hmarn_tehnolog_.pdf)

1.24% Джерела з Інтернету 46

Сторінка 68

0.84% Джерела з Бібліотеки 36

Сторінка 68

0% Цитат

Вилучення цитат вимкнено

Вилучення списку бібліографічних посилань вимкнено

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 4

Підозріле форматування 15 сторінок

Додаток Е

Експертний висновок результатів перевірки кваліфікаційної роботи на
відповідність оформлення вимогам ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент
(посада)

програмної інженерії
(кафедра)

ПЗМ-22-1
(група)

Литовченко В.Ю

(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунок	
	7.6 Таблиці	
	7.7 Переліки	
7.7.2	Якщо подають переліки одного рівня підпорядкованості, на які у звіті немає посилань, то перед кожним із переліків ставлять знак «тире». Якщо у звіті є посилання на переліки, підпорядкованість позначають малими літерами української абетки, далі — арабськими цифрами, далі — через знаки «тире». Після цифри або літери певної позиції переліку ставлять круглу дужку.	49, далі за текстом.
	7.8 Примітки	
	7.9 Висновки	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні позначки	
	7.15 Додатки	

Експерт

(підпис)

Вадим НЕЧВОЛОД

(прізвище, ініціали)

13.06.2024

Рисунок Е.1 – Результат перевірки записки на відповідність оформлення вимогам ДСТУ 3008:2015