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

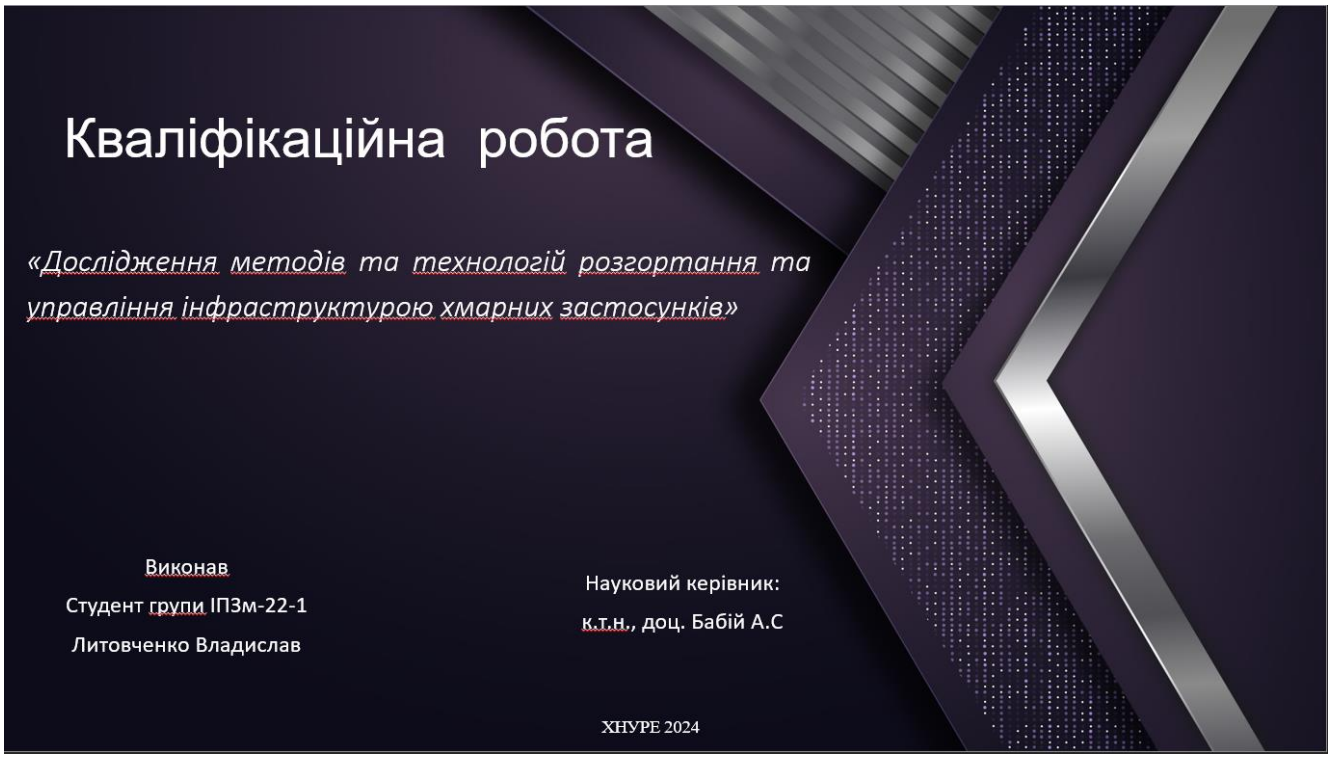


Рисунок Б.1 – Слайд 1 (тема дослідження)

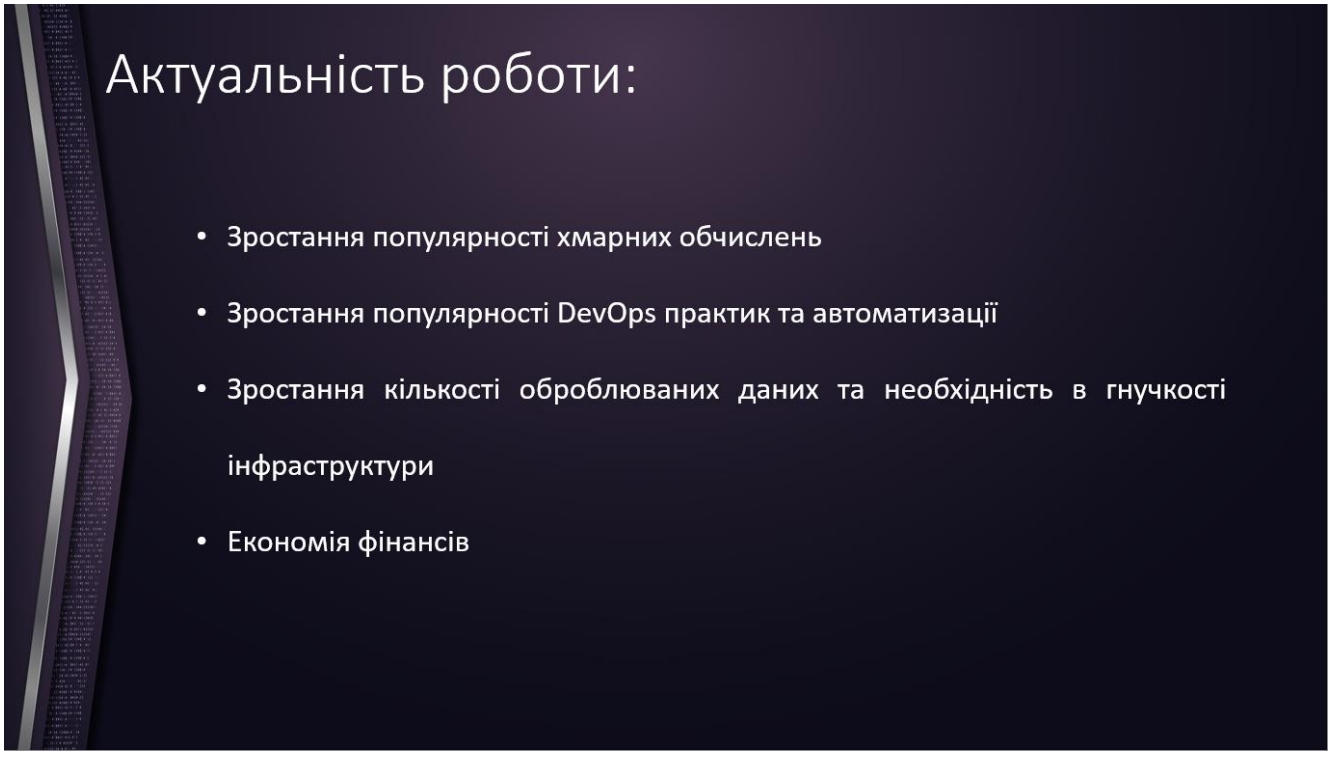


Рисунок Б.2 – Слайд 2 (актуальність роботи)

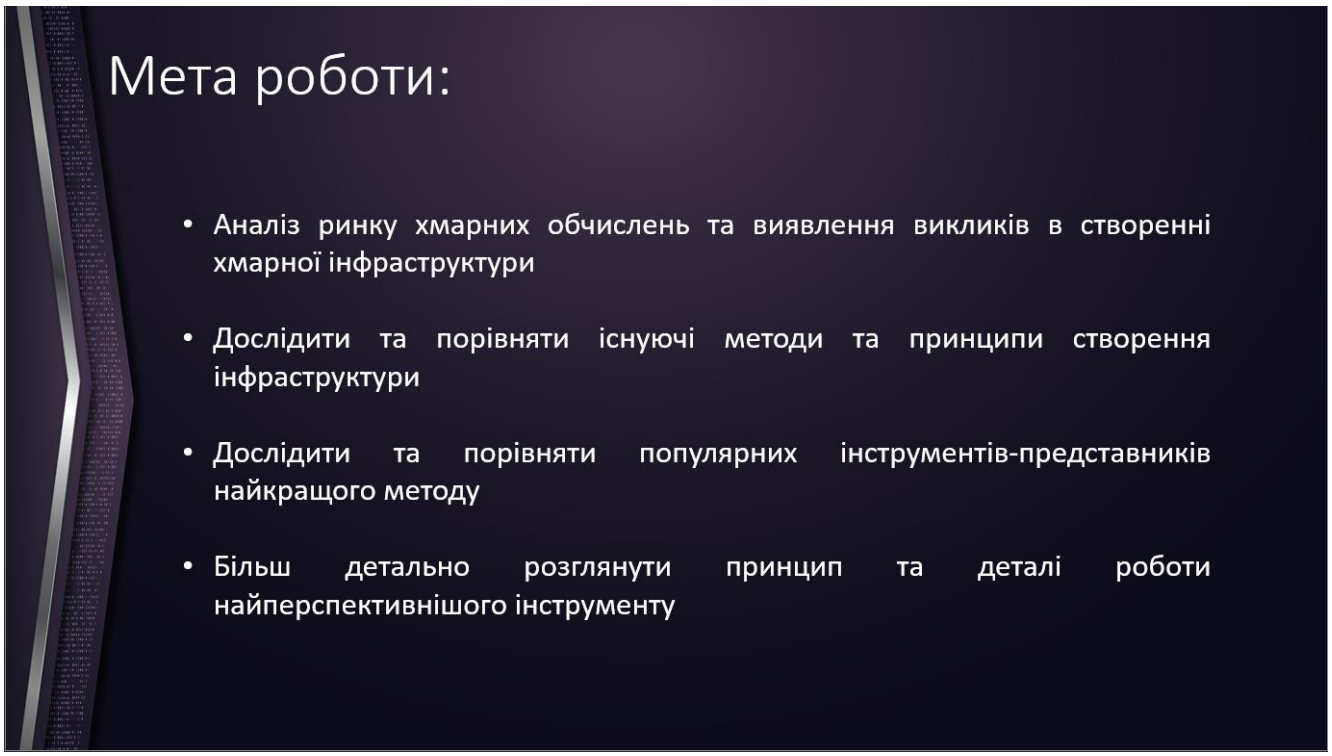


Рисунок Б.3 – Слайд 3 (мета роботи)

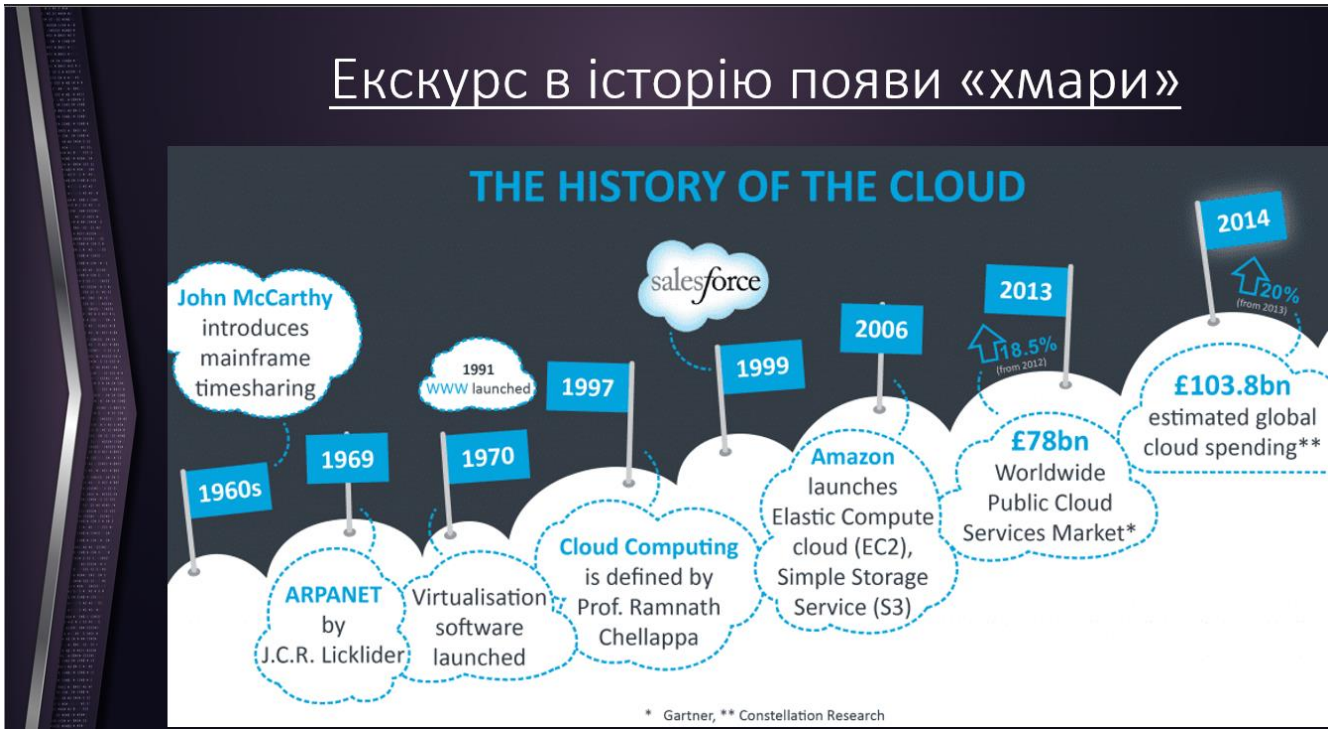


Рисунок Б.4 – Слайд 4 (історія «хмари»)

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

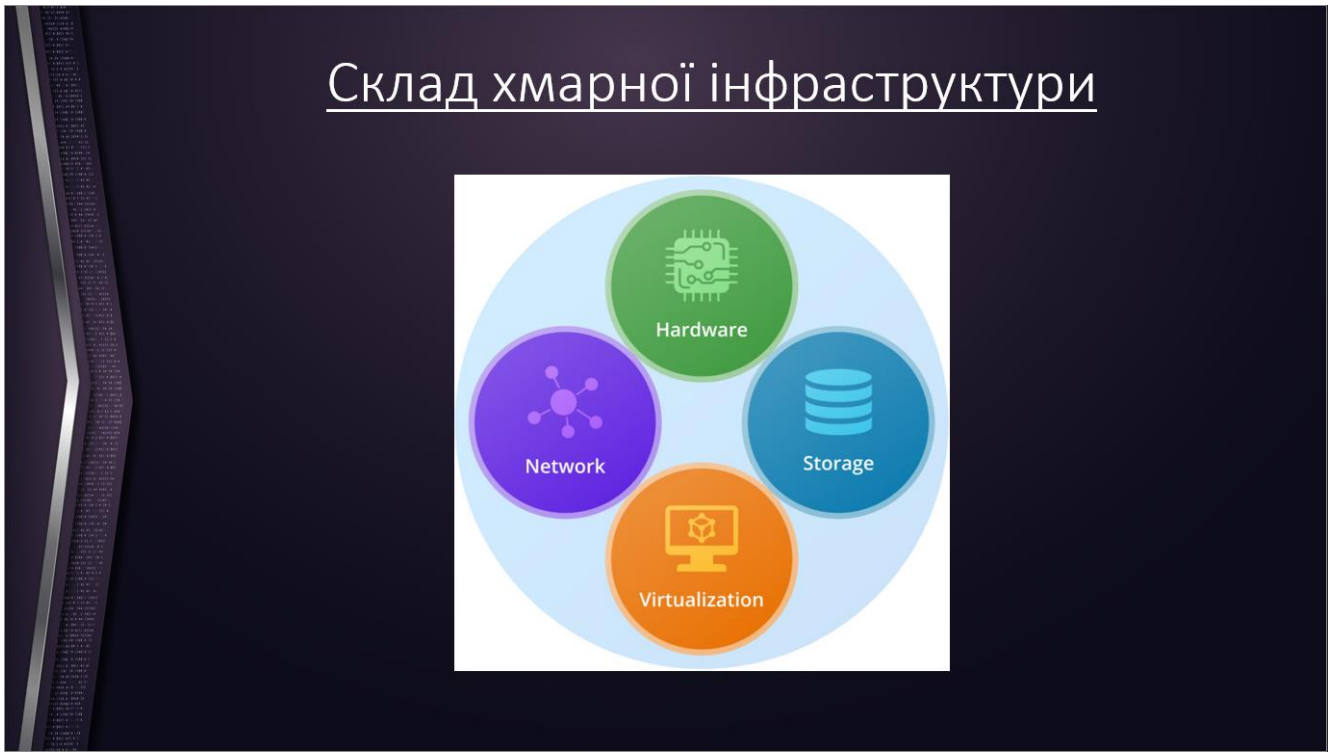


Рисунок Б.6 – Слайд 6 (склад хмарної інфраструктури)

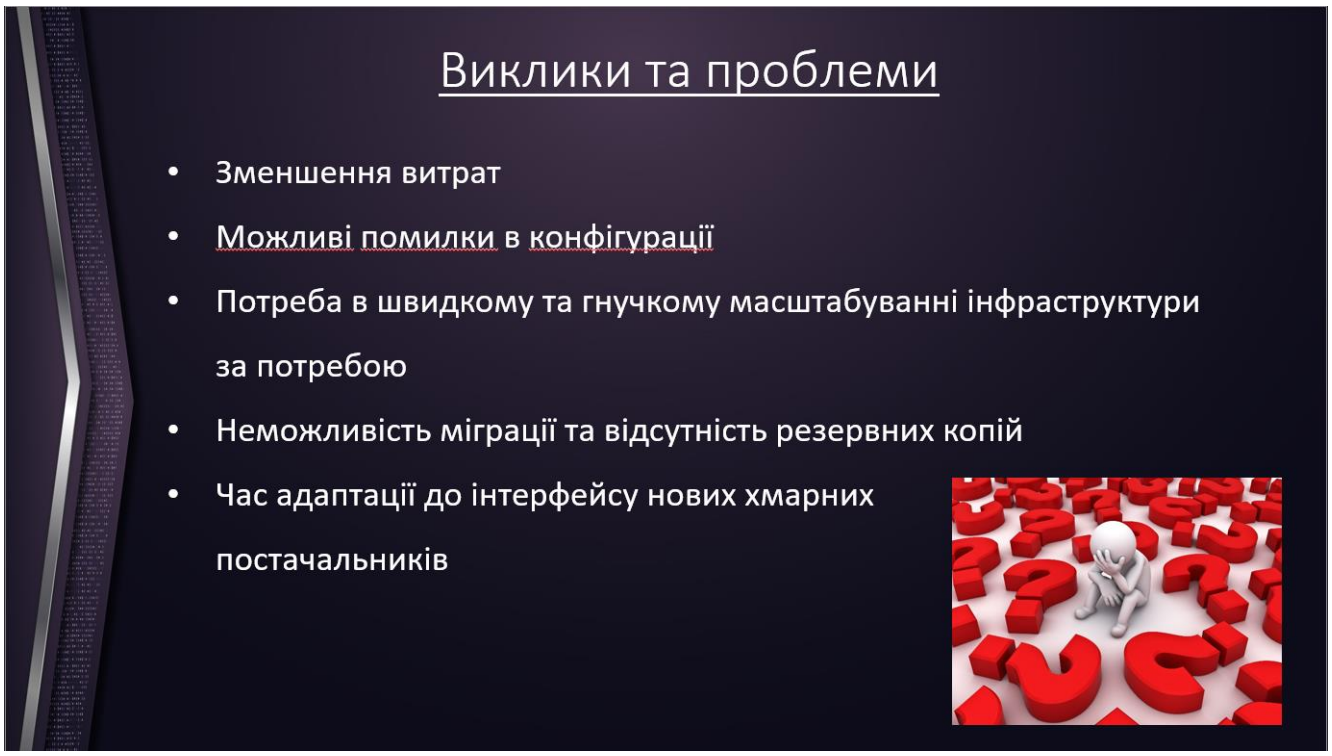


Рисунок Б.7 – Слайд 7 (виклики та проблеми)

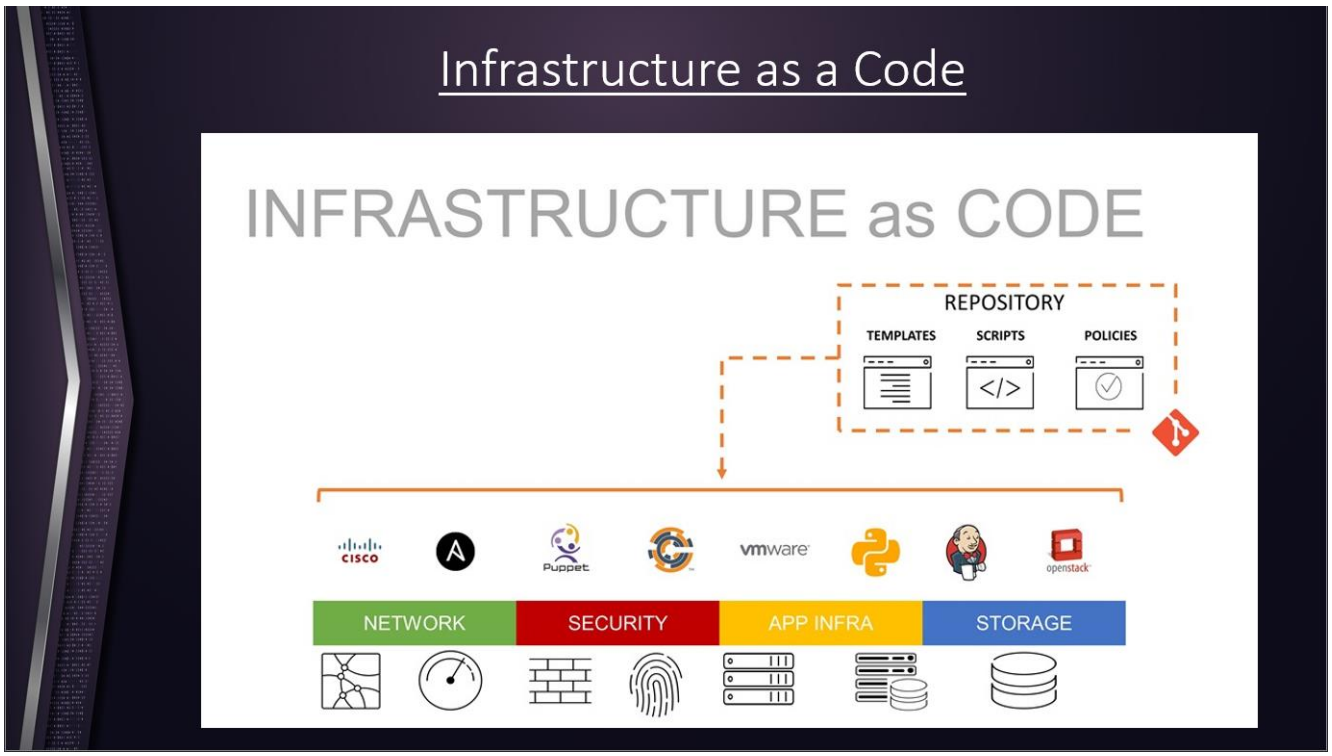


Рисунок Б.8 – Слайд 8 (Infrastructure as a Code)

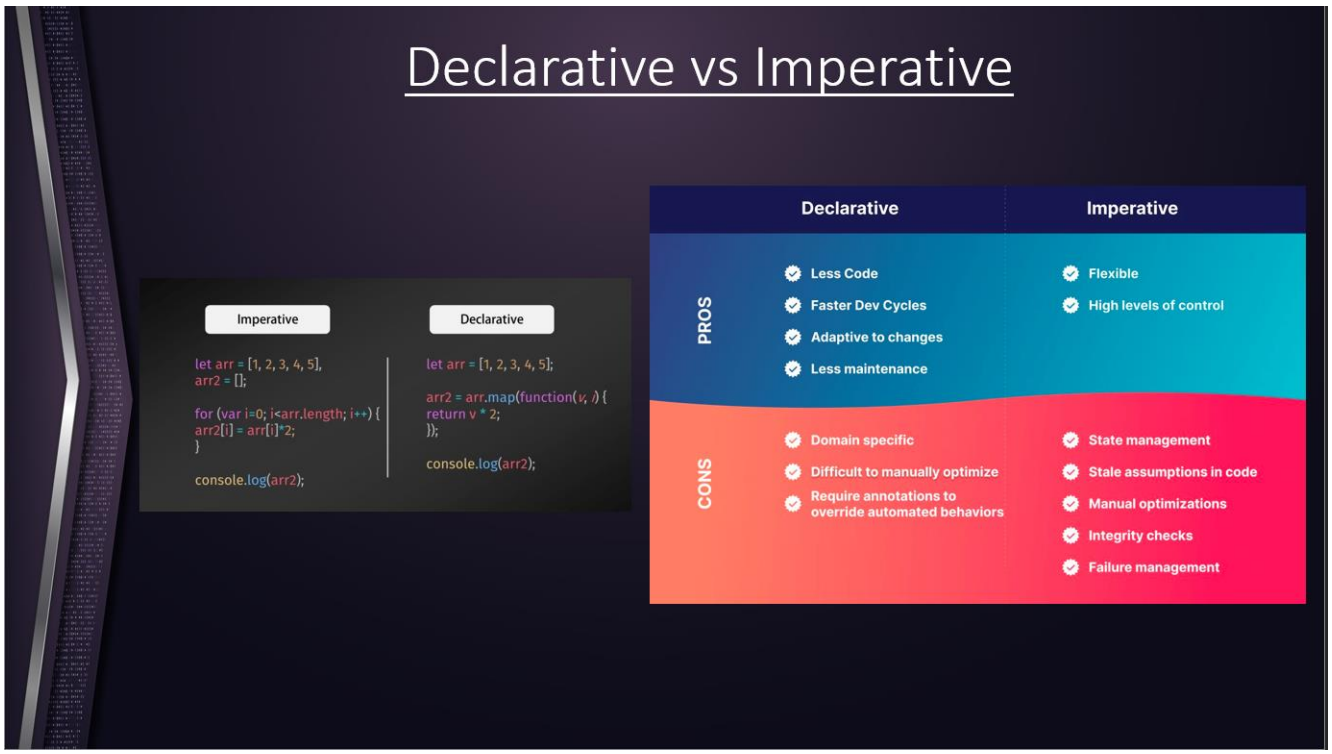


Рисунок Б.9 – Слайд 9 (declarative vs imperative)

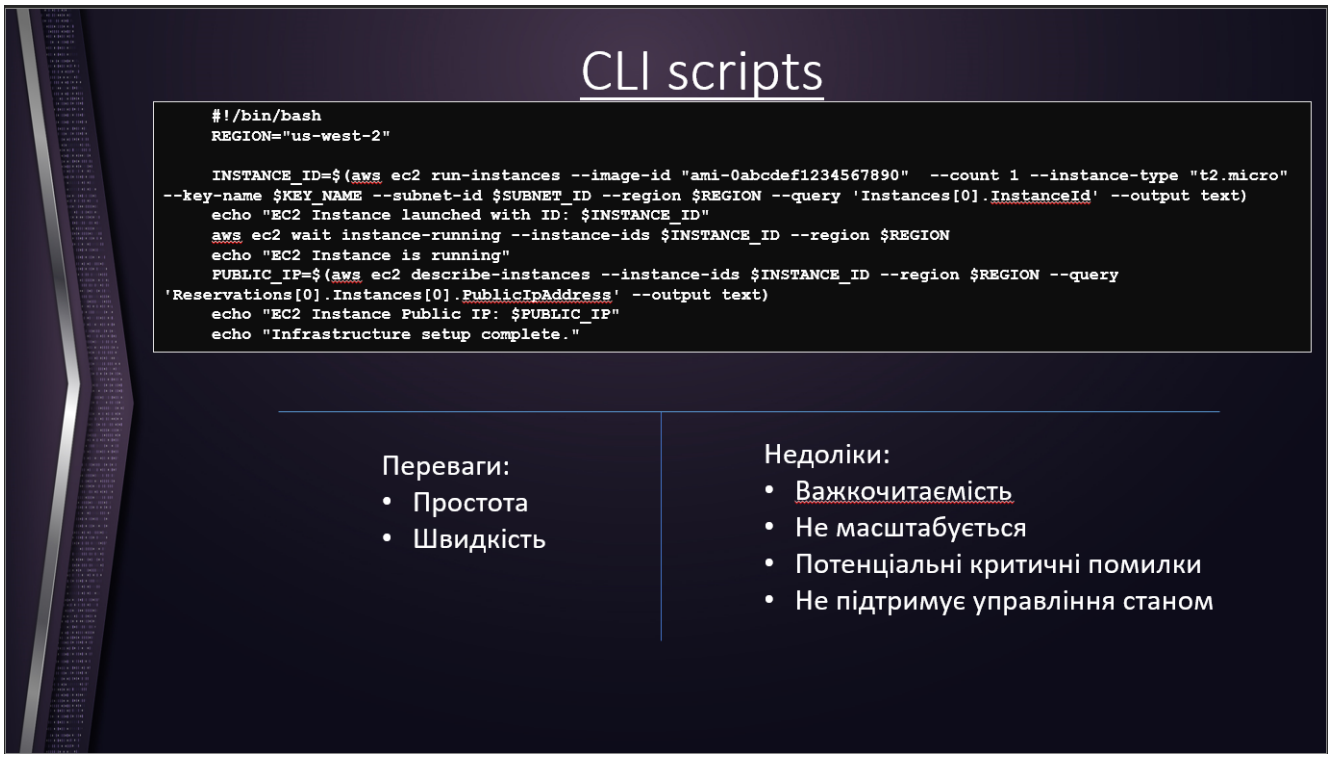


Рисунок Б.10 – Слайд 10 (CLI scripts)

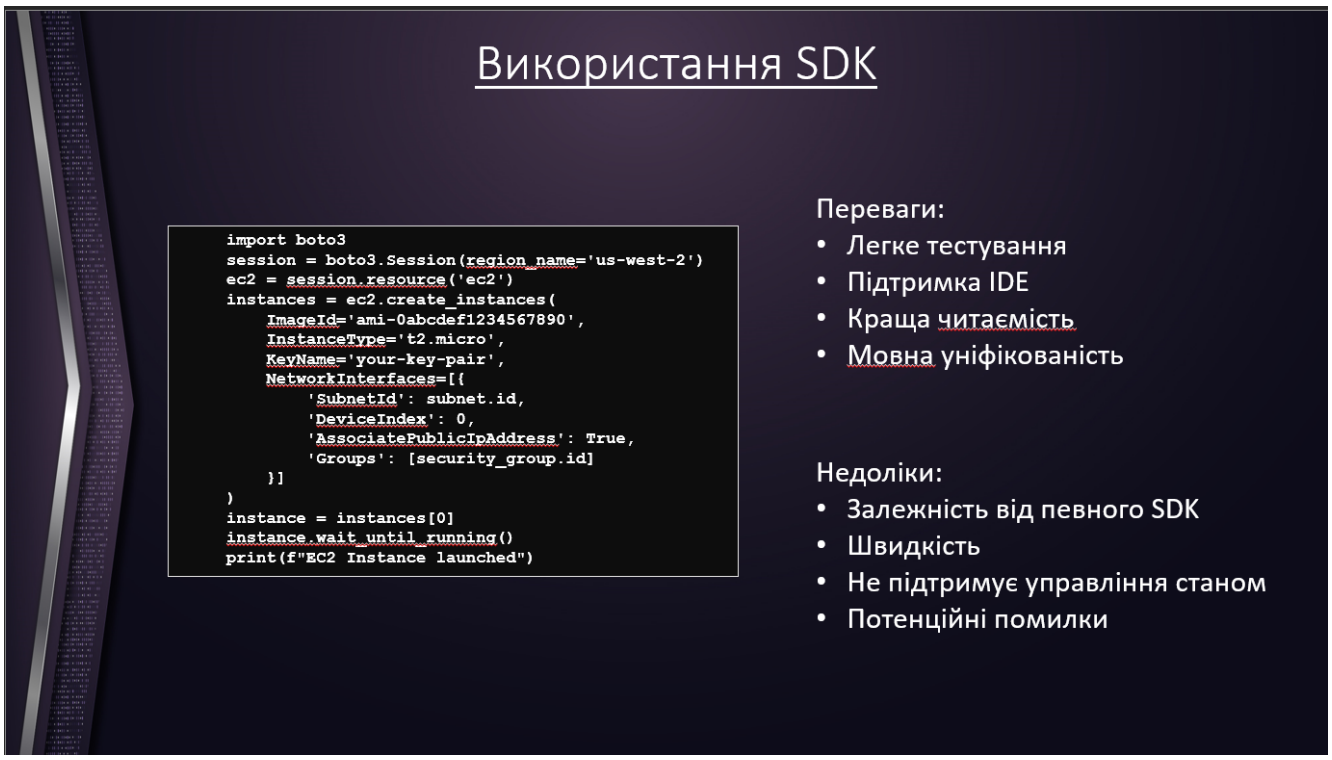


Рисунок Б.11 – Слайд 11 (Використання SDK)

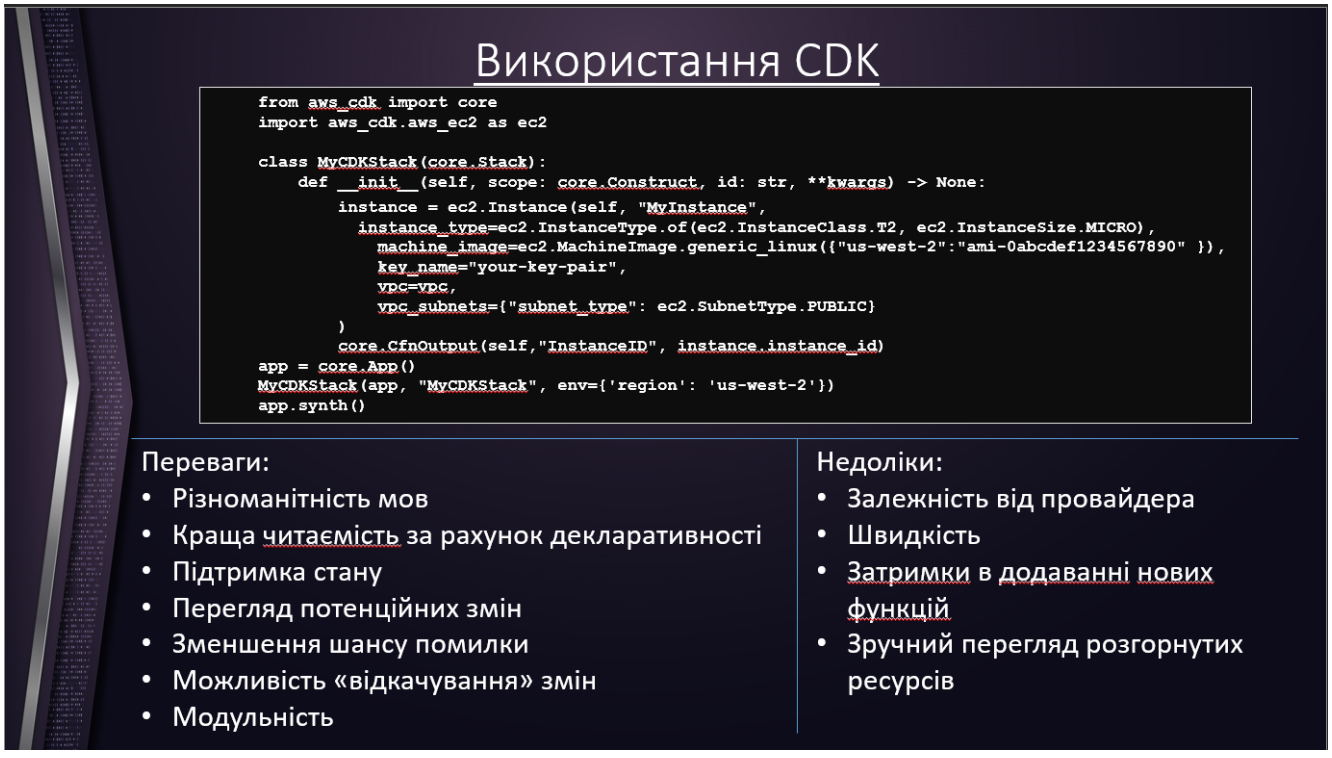


Рисунок Б.12 – Слайд 12 (використання CDK)

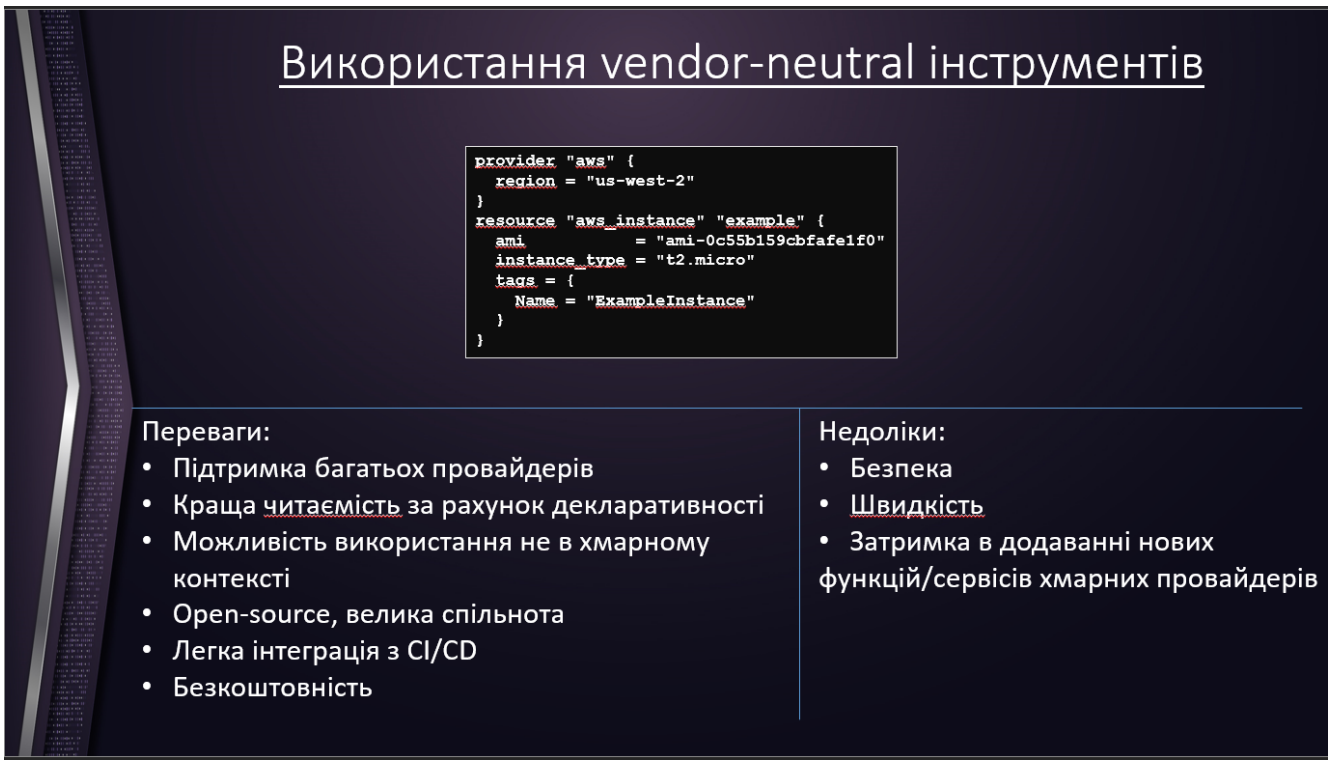


Рисунок Б.13 – Слайд 13 (використання vendor-neutral інструментів)

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

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

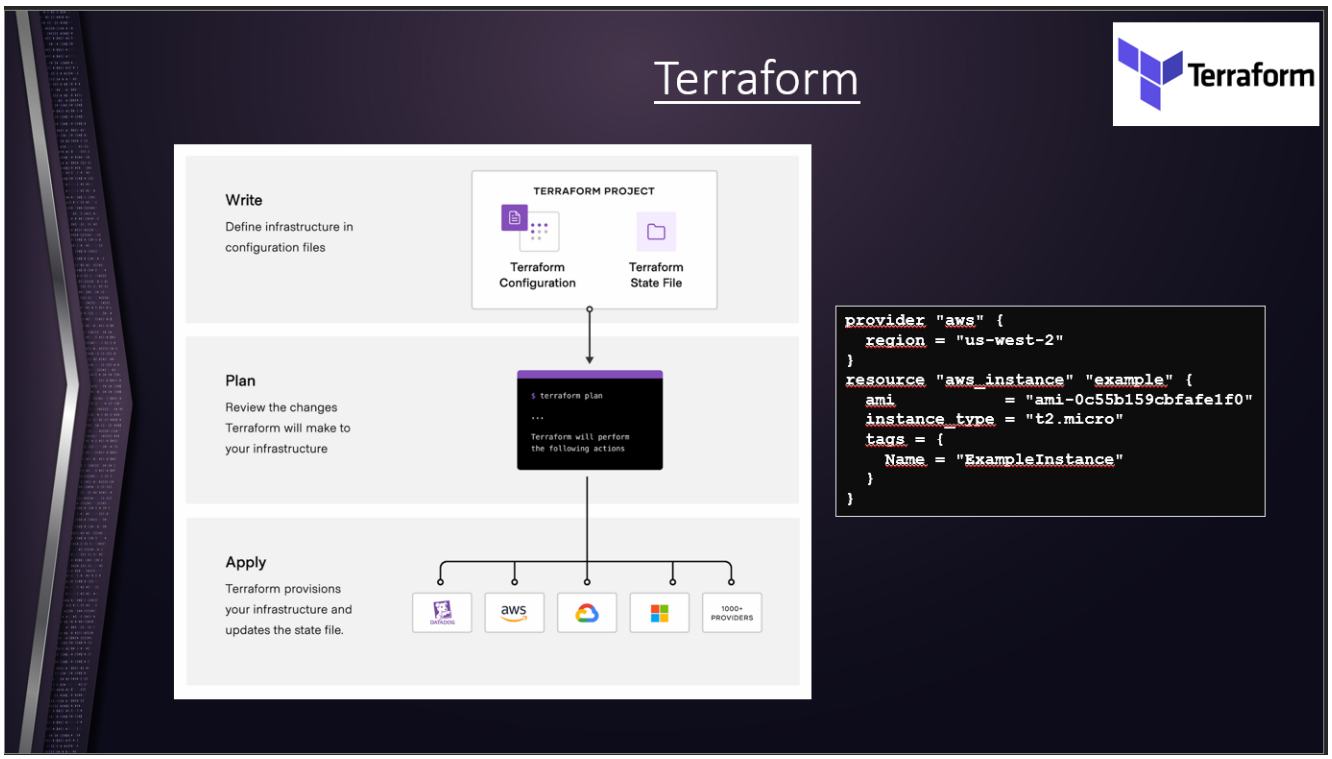


Рисунок Б.16 – Слайд 16 (Terraform)

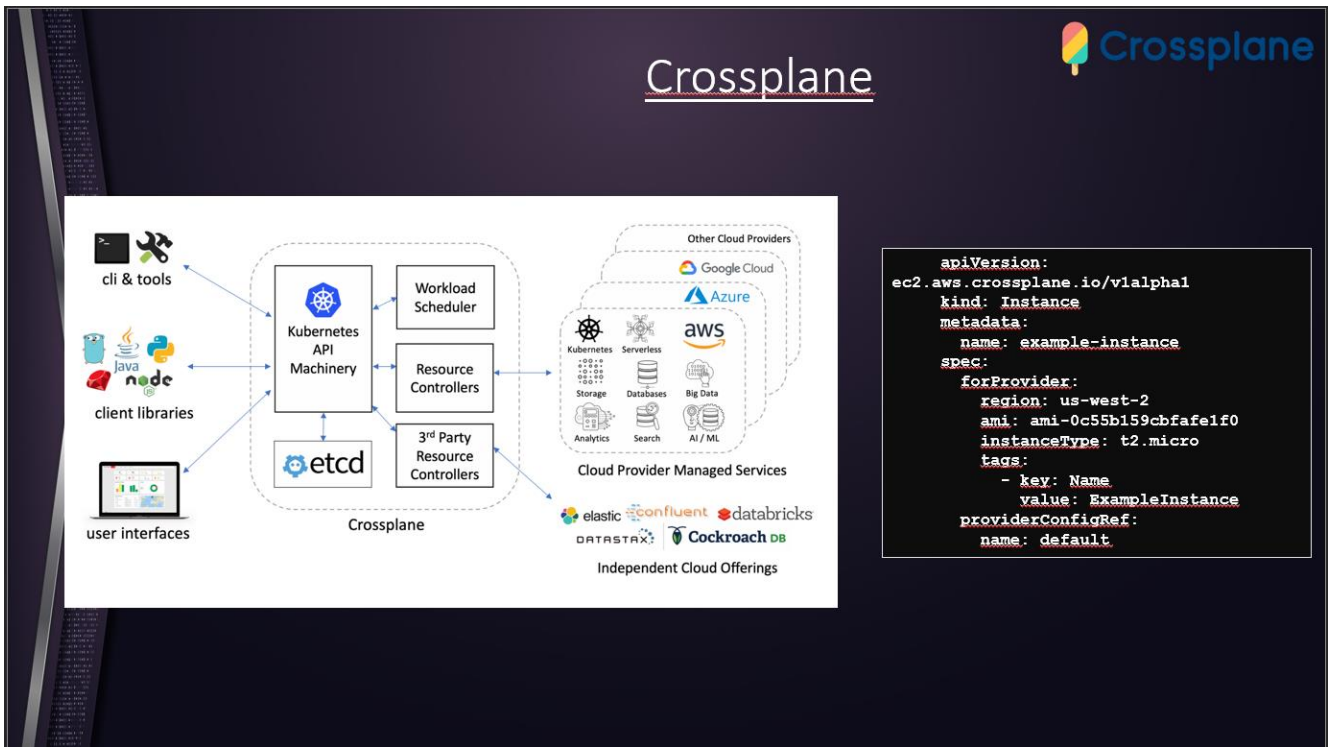


Рисунок Б.17– Слайд 17 (Crossplane)

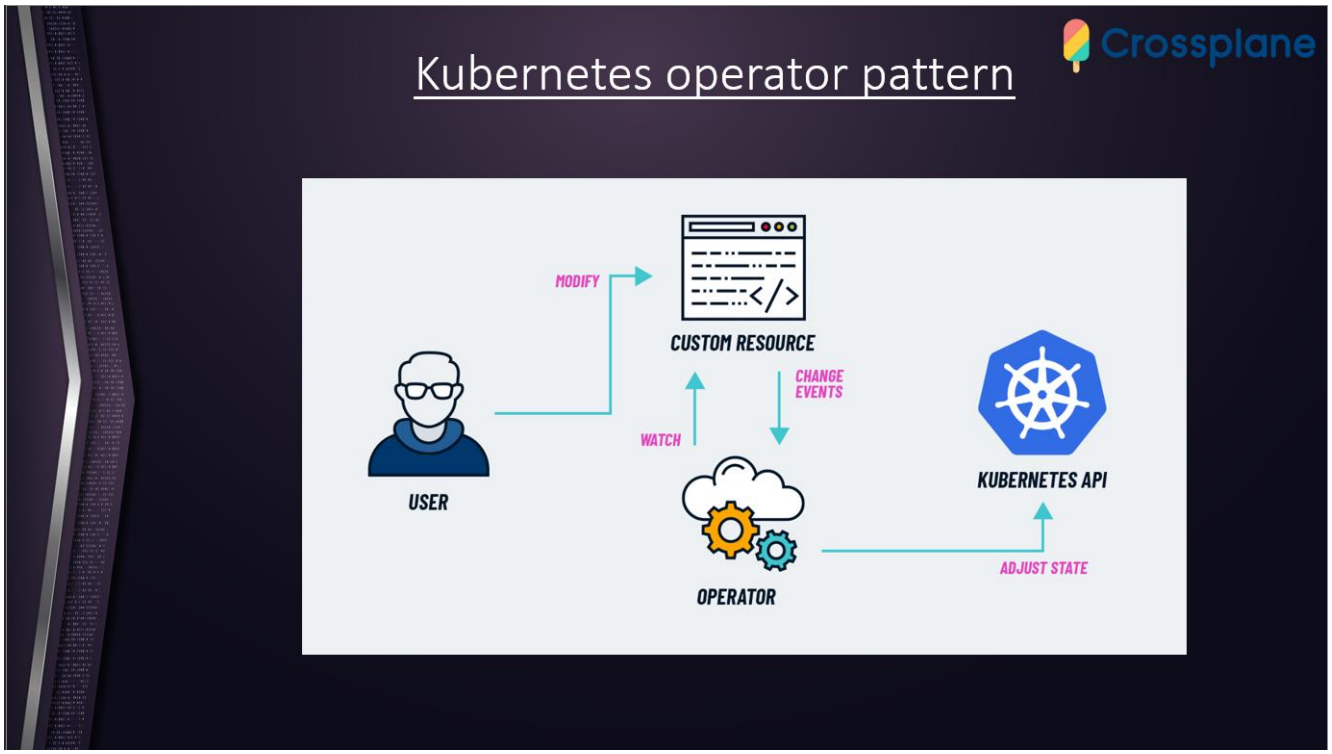


Рисунок Б.18 – Слайд 18 (Kubernetes operator pattern)

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)

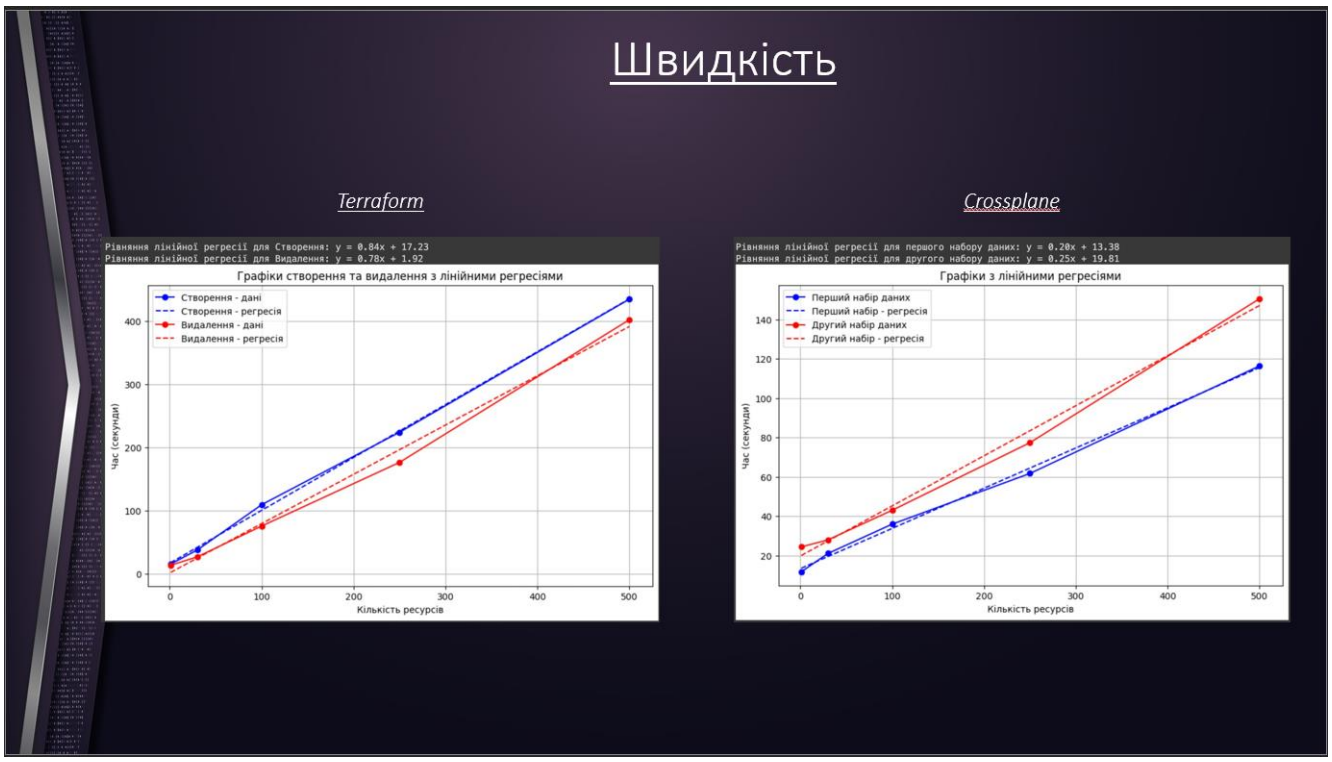


Рисунок Б.21 – Слайд 21 (порівняння швидкості)

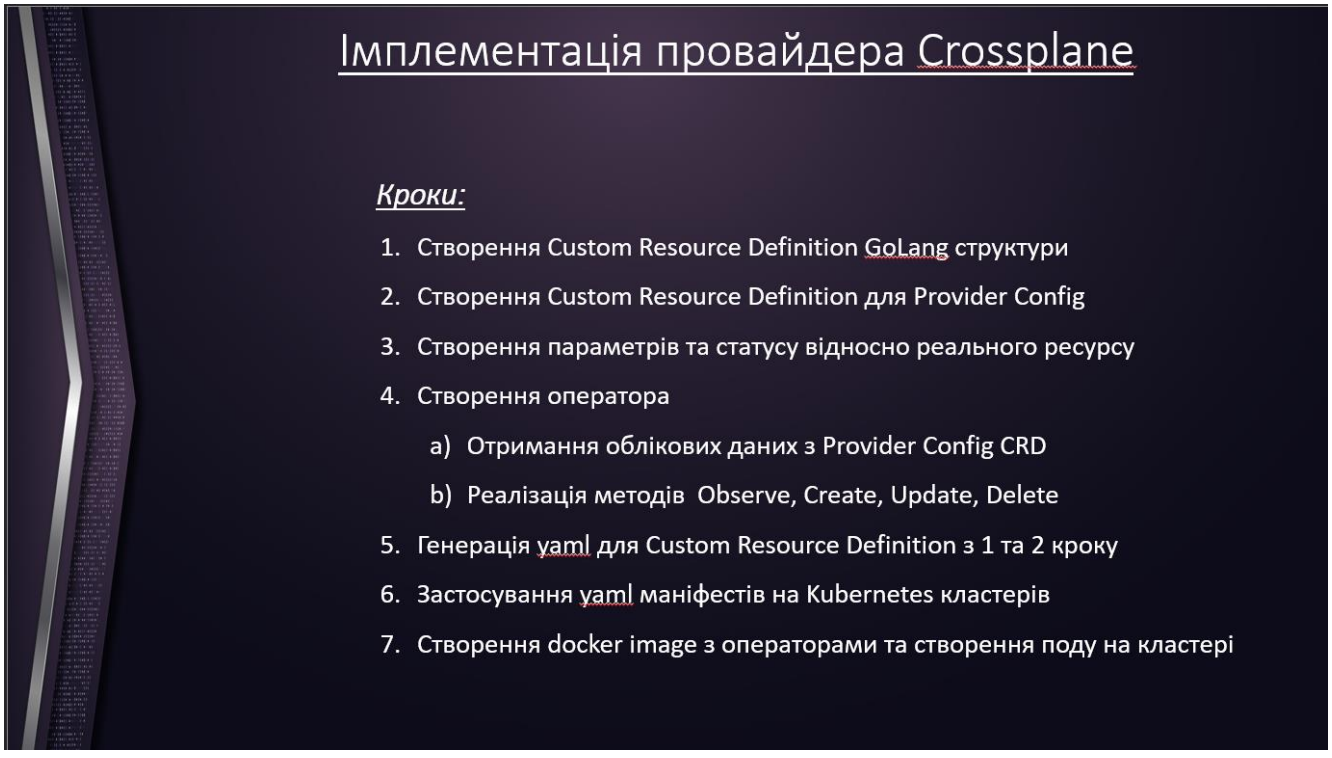


Рисунок Б.22 – Слайд 22 (для імплементації провайдера Crossplane)

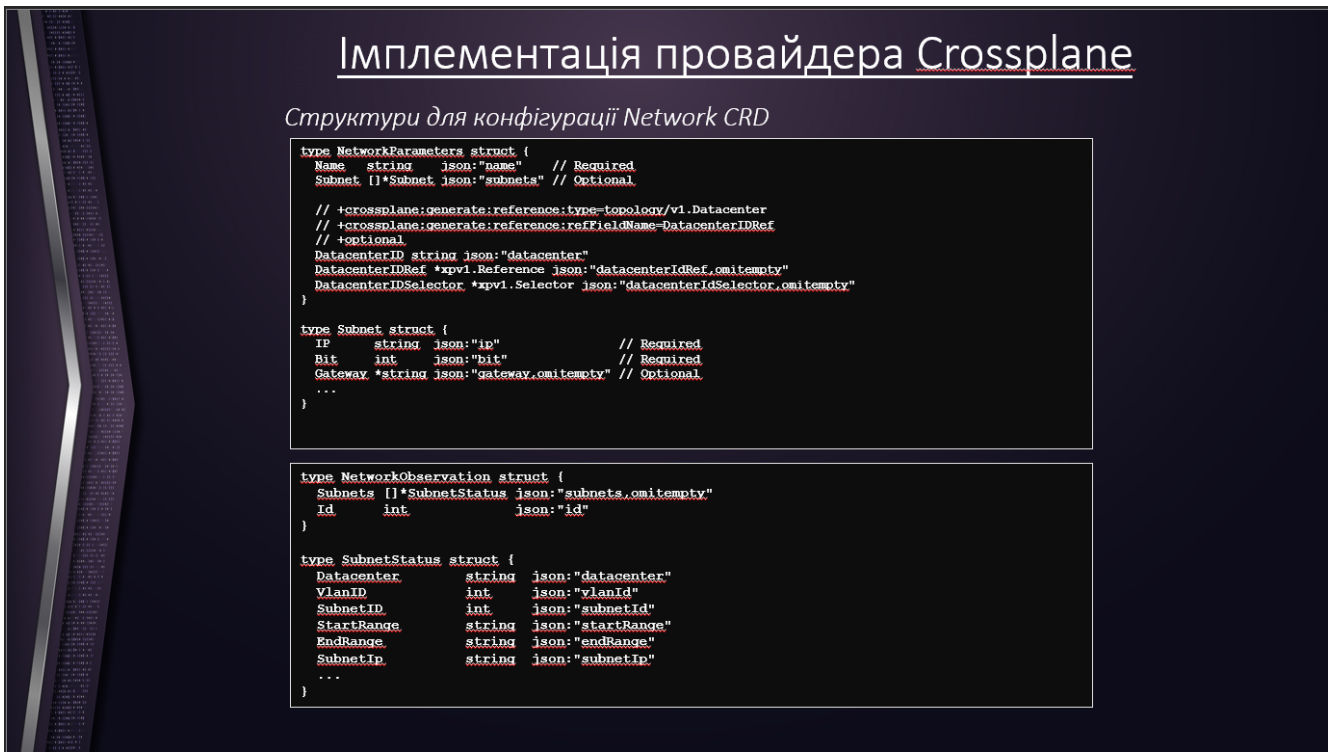


Рисунок Б.23 – Слайд 23 (CRD для імплементації Network)

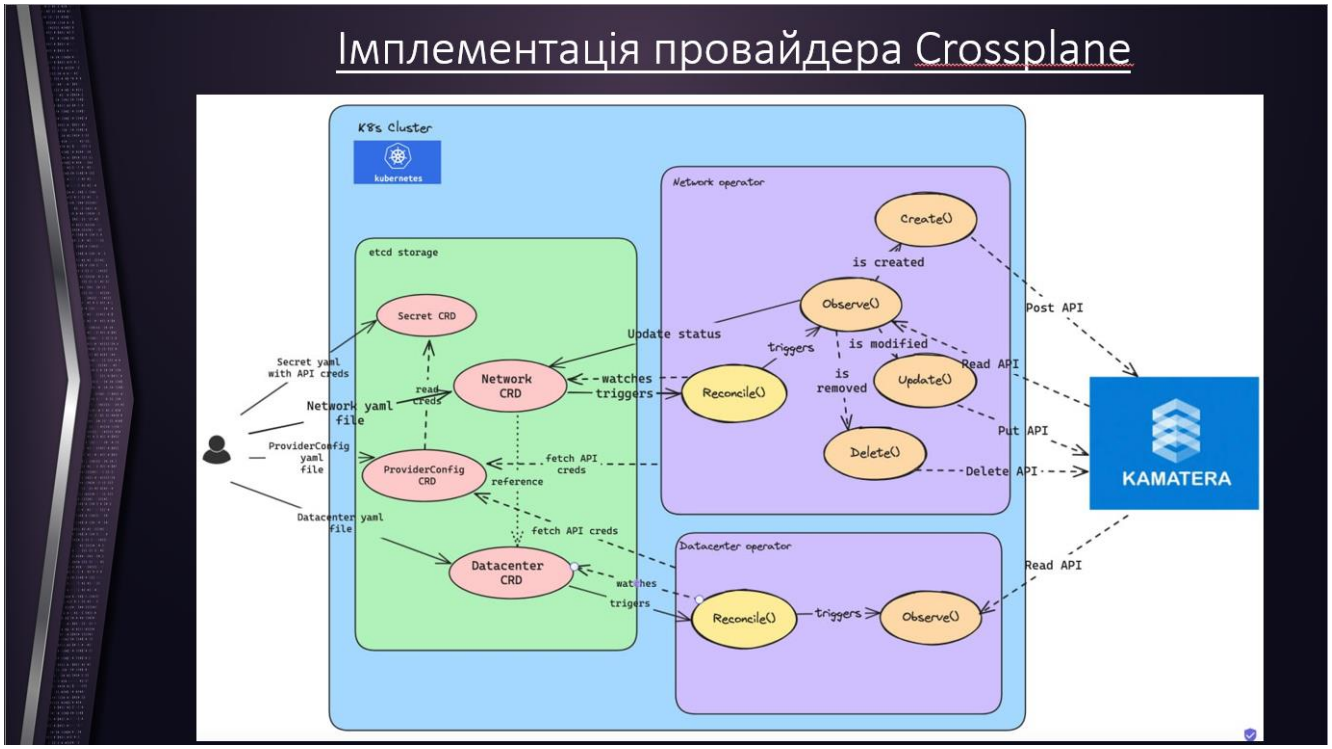


Рисунок Б.24 – Слайд 24 (діаграма роботи реалізованого провайдера)

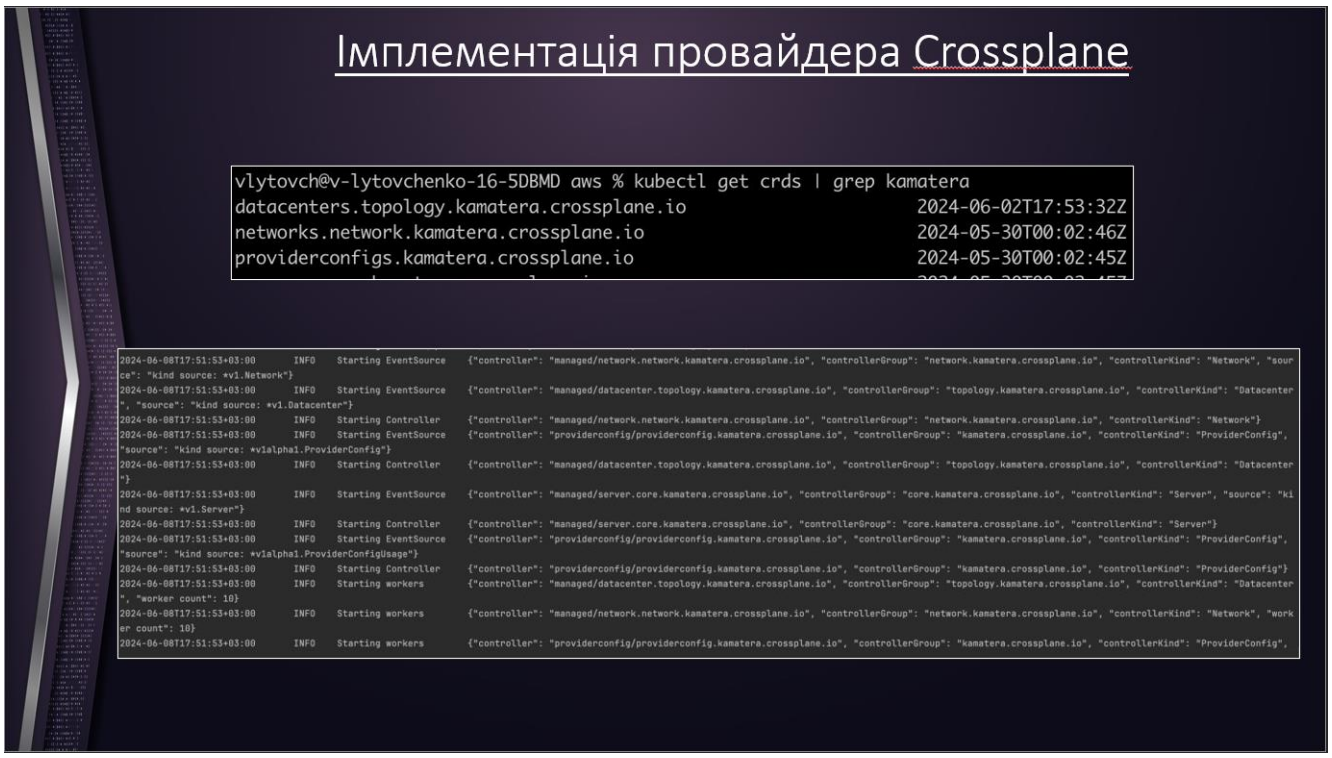


Рисунок Б.25 – Слайд 25 (робота провайдера на кластері)

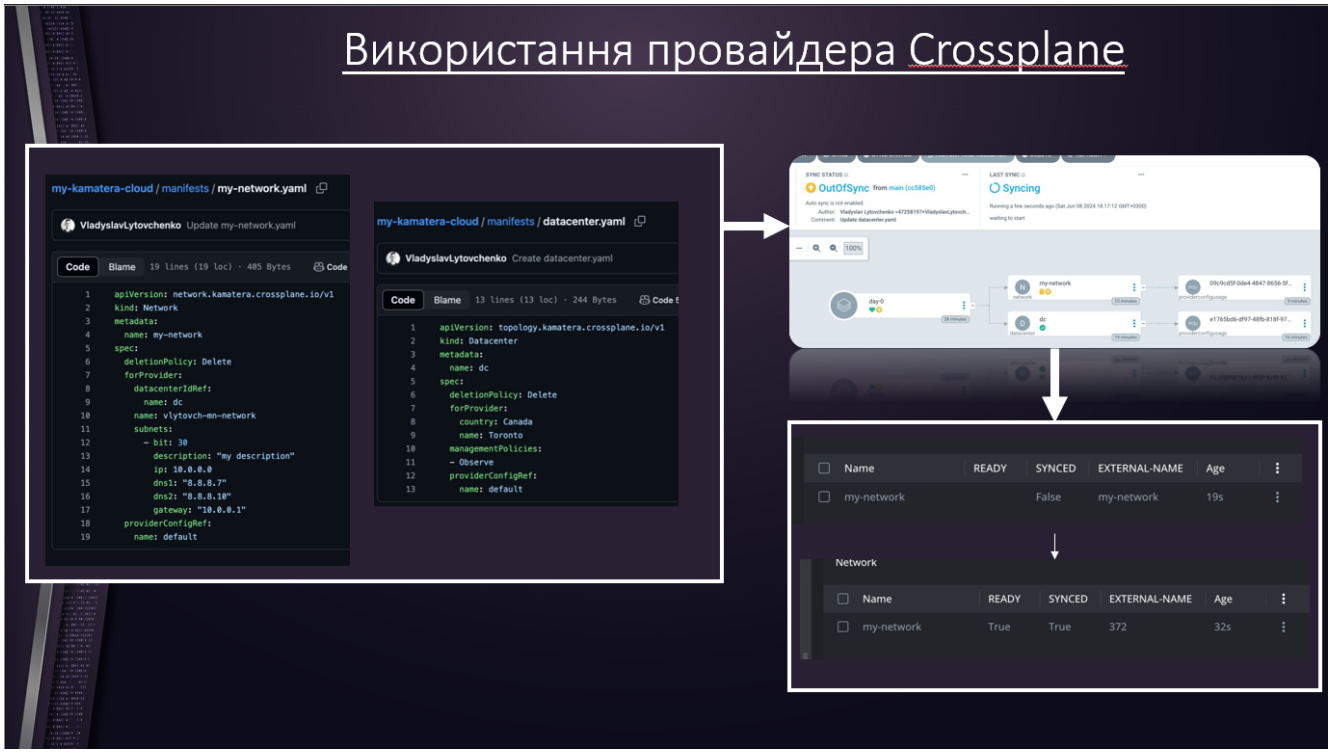


Рисунок Б.26 – Слайд 26 (схема використання провайдера)

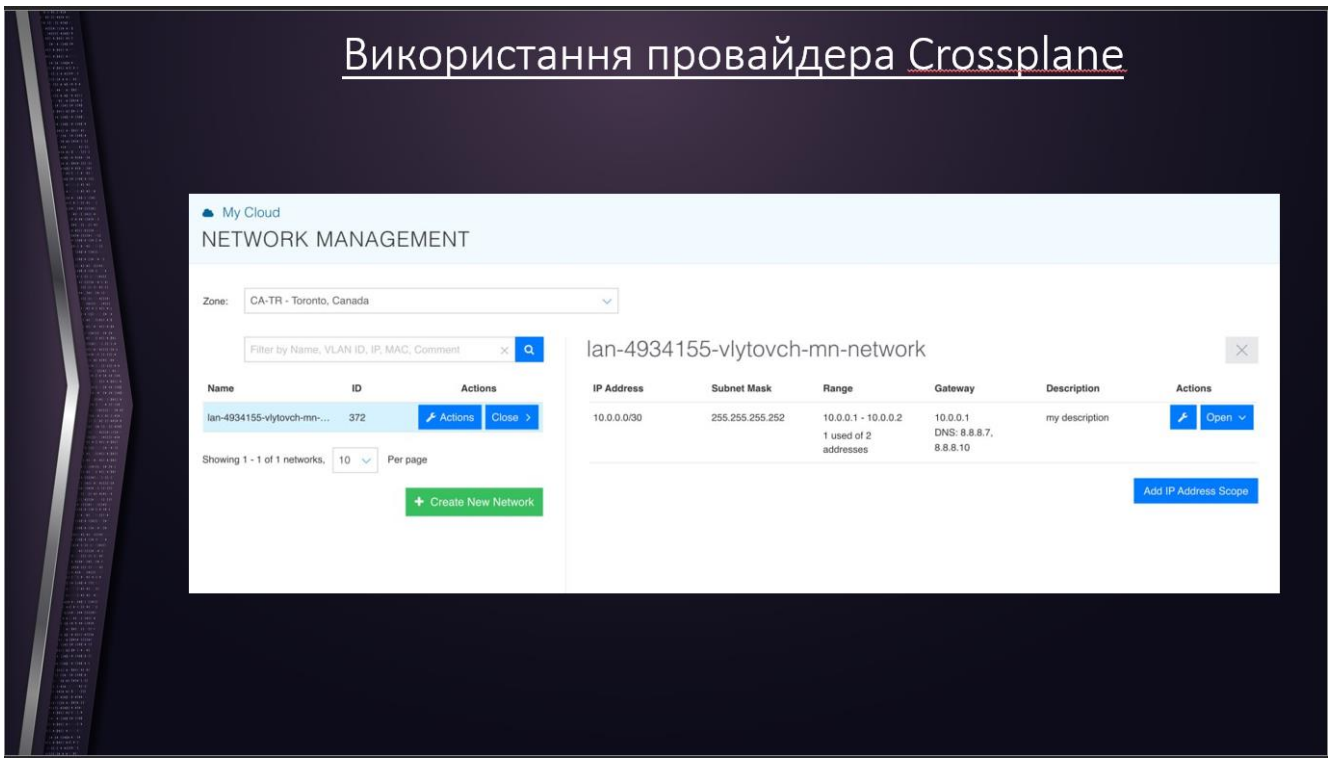


Рисунок Б.27 – Слайд 27 (створений ресурс на хмарі)

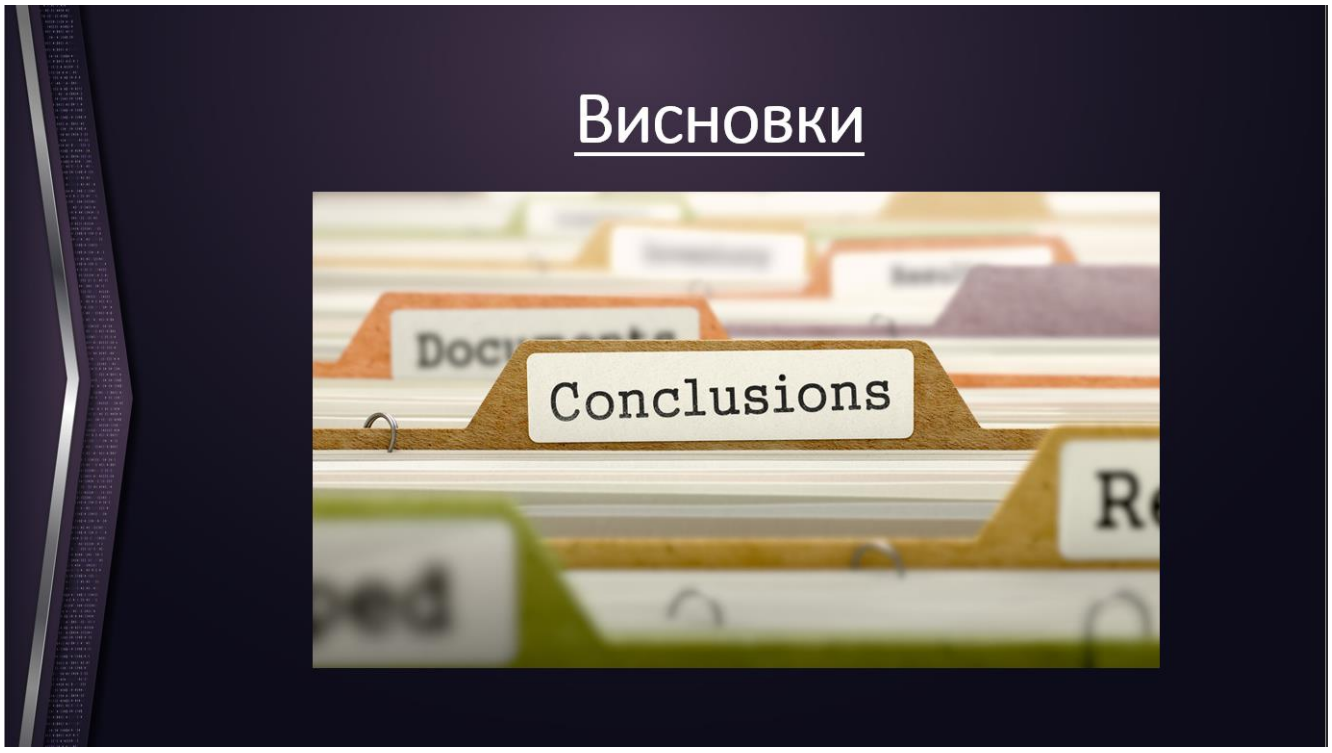


Рисунок Б.28 – Слайд 28 (висновки)

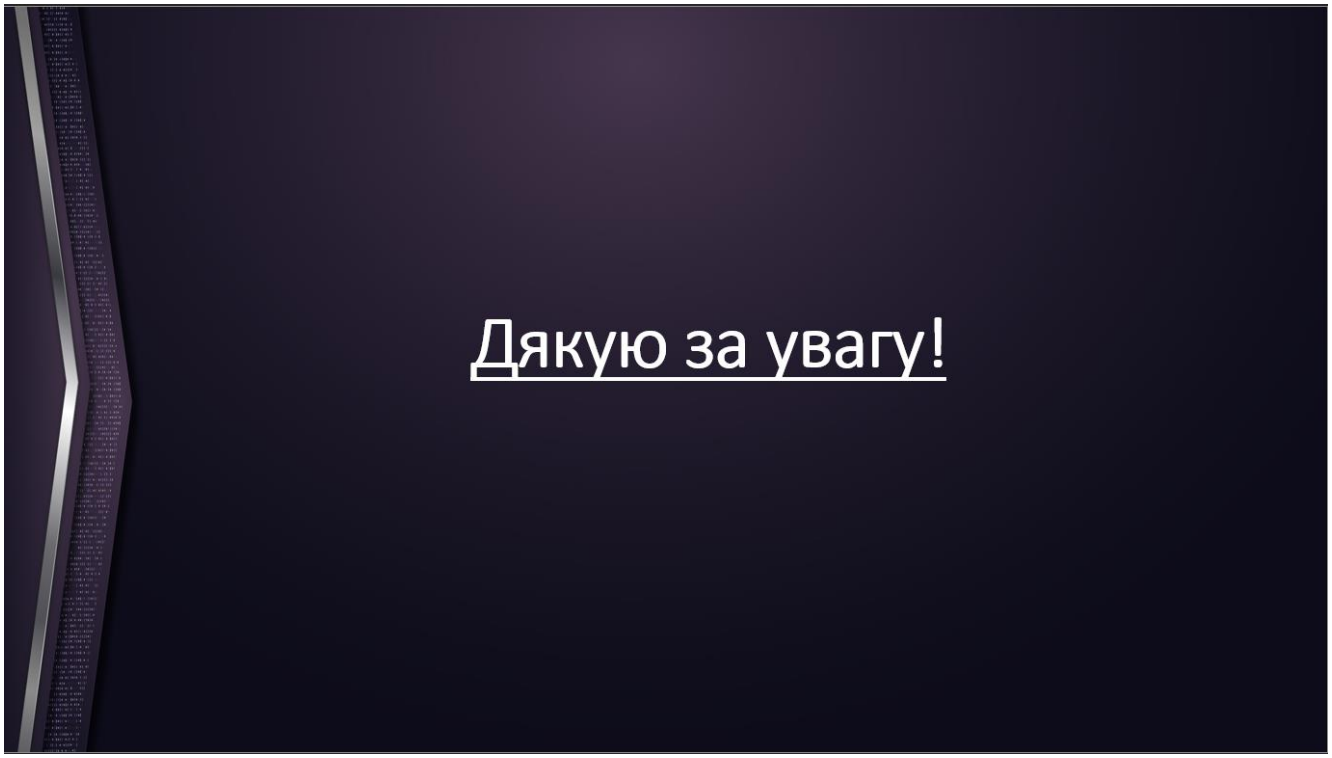


Рисунок Б.29 – Слайд 29 (подяка)

ДОДАТОК В

Апробація у вигляді тез

Рисунок В.1 – Титульна сторінка тез

Рисунок В.2 – Зміст збірника тез

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 IT-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

Рисунок В.3 – Перша сторінка тез

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]. Прикладам 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 гарантування 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.

Рисунок В.4 – Друга сторінка тез

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

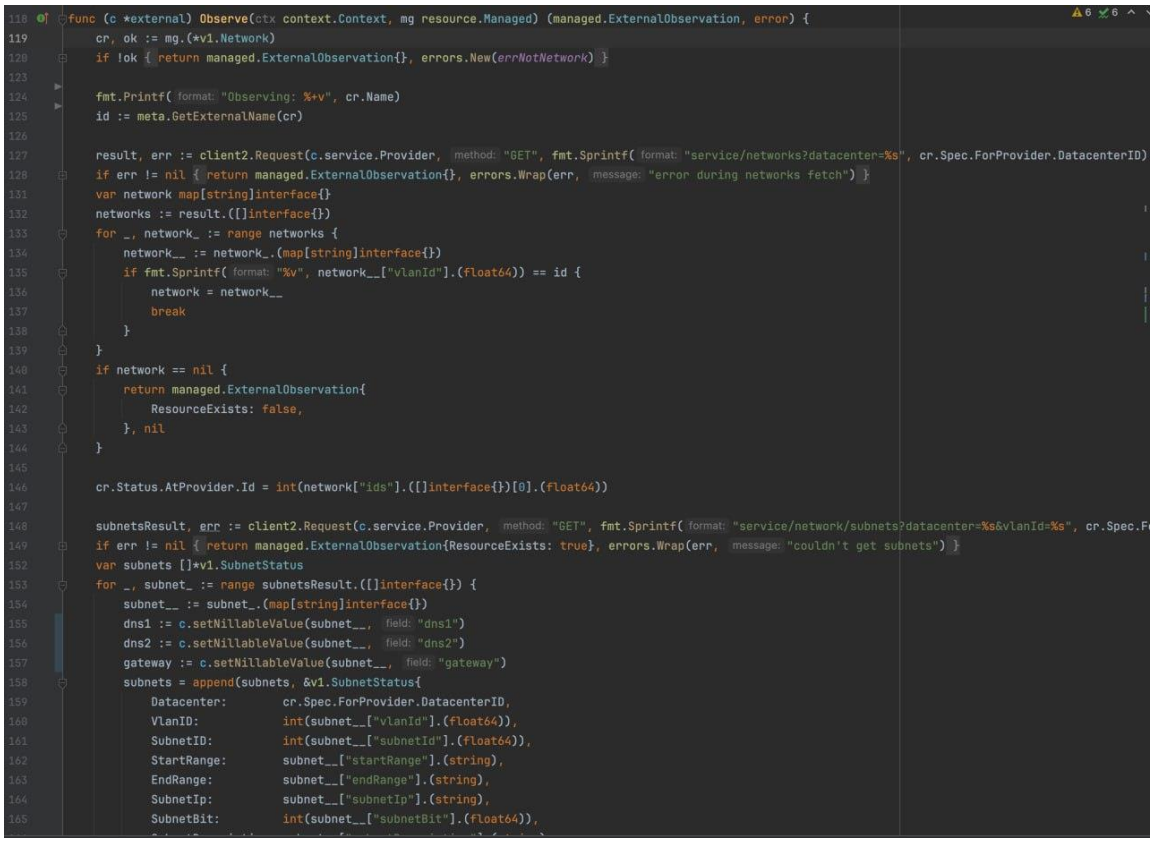
Величезні мінні поля поставили під загрозу життя близько шести мільйонів мирних жителів і зробили непридатними частинами найцінніших сільськогосподарських угідь країни, що вплинуло як на економіку країни, так і на світове постачання продовольства.

Рисунок В.5 – Третя сторінка тез

Рисунок В.4 – Сертифікат про участь у конференції

Додаток Г

Фрагменти коду

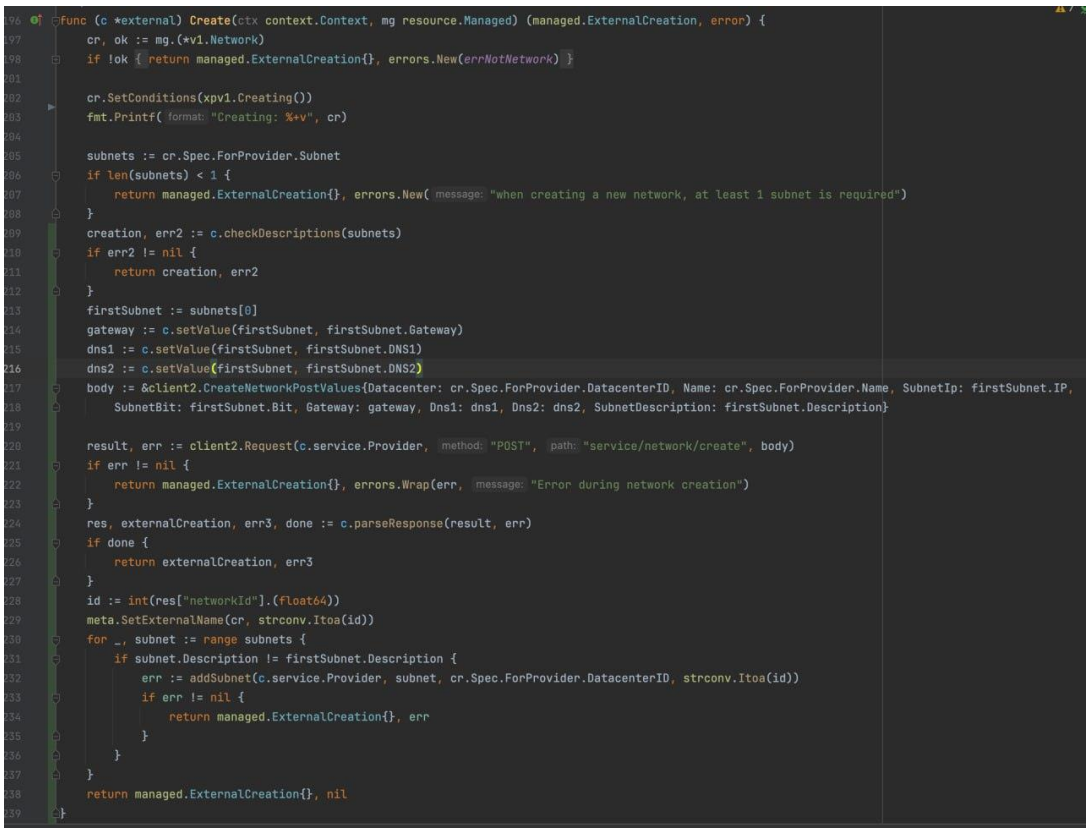


```
func (c *external) Observe(ctx context.Context, mg resource.Managed) (managed.ExternalObservation, error) {
    cr, ok := mg.(*v1.Network)
    if !ok { return managed.ExternalObservation{}, errors.New(errNotNetwork) }

    fmt.Printf( format: "Observing: %+v", cr.Name)
    id := meta.GetExternalName(cr)

    result, err := client2.Request(c.service.Provider, method: "GET", fmt.Sprintf( format: "service/networks?datacenter=%s", cr.Spec.ForProvider.DatacenterID)
    if err != nil { return managed.ExternalObservation{}, errors.Wrap(err, message: "error during networks fetch") }
    var network map[string]interface{}
    networks := result.([]interface{})
    for _, network_ := range networks {
        network__ := network_.(map[string]interface{})
        if fmt.Sprintf( format: "%v", network__["vlanId"].(float64)) == id {
            network = network__
            break
        }
    }
    if network == nil {
        return managed.ExternalObservation{
            ResourceExists: false,
        }, nil
    }

    cr.Status.AtProvider.Id = int(network["ids"].([]interface{})[0].(float64))

    subnetsResult, err := client2.Request(c.service.Provider, method: "GET", fmt.Sprintf( format: "service/network/subnets?datacenter=%s&vlanId=%s", cr.Spec.F
    if err != nil { return managed.ExternalObservation{ResourceExists: true}, errors.Wrap(err, message: "couldn't get subnets") }
    var subnets []*v1.SubnetStatus
    for _, subnet_ := range subnetsResult.([]interface{}) {
        subnet__ := subnet_.(map[string]interface{})
        dns1 := c.setNillableValue(subnet__, field: "dns1")
        dns2 := c.setNillableValue(subnet__, field: "dns2")
        gateway := c.setNillableValue(subnet__, field: "gateway")
        subnets = append(subnets, &v1.SubnetStatus{
            Datacenter:        cr.Spec.ForProvider.DatacenterID,
            VlanID:            int(subnet__["vlanId"].(float64)),
            SubnetID:          int(subnet__["subnetId"].(float64)),
            StartRange:        subnet__["startRange"].(string),
            EndRange:          subnet__["endRange"].(string),
            SubnetIp:          subnet__["subnetIp"].(string),
            SubnetBit:         int(subnet__["subnetBit"].(float64)),
```

Рисунок Г.1 – Реалізація Observe методу

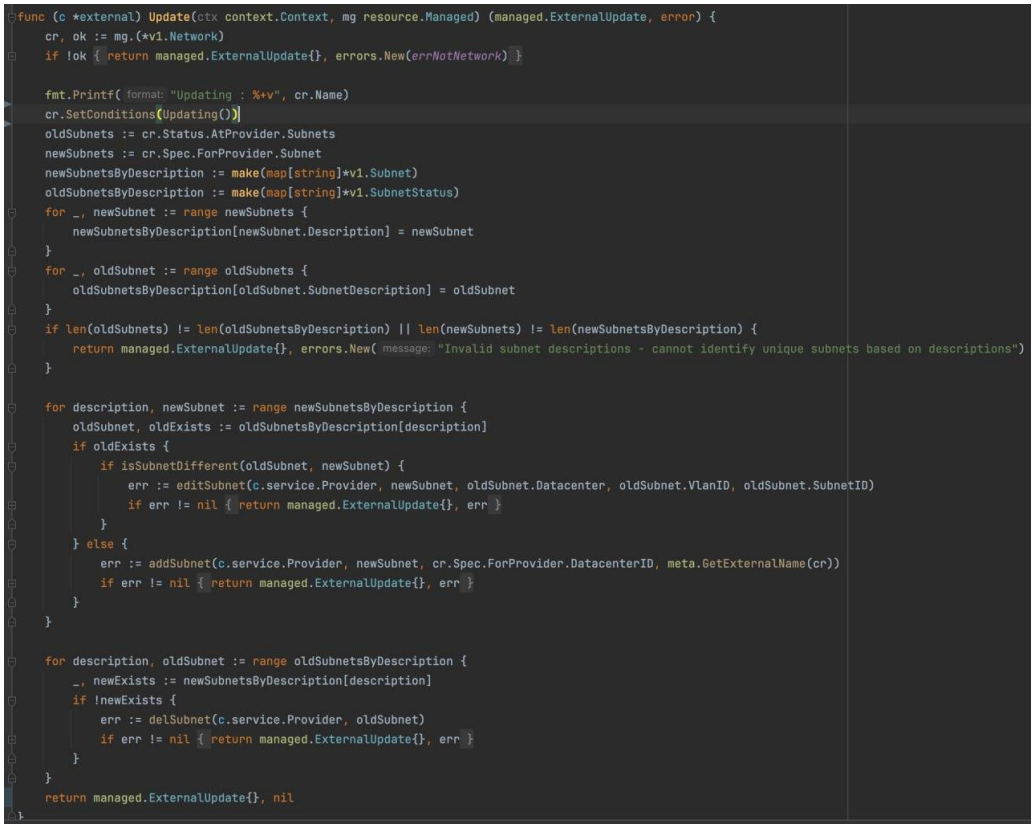
```
            SubnetBit:         int(subnet__["subnetBit"].(float64)),
            SubnetDescription: subnet__["subnetDescription"].(string),
            DNS1:              dns1,
            DNS2:              dns2,
            Gateway:           gateway,
            InUse:             int(subnet__["inUse"].(float64)),
        })
    }

    cr.Status.AtProvider.Subnets = subnets
    isChanged, err := IsNetworkDifferent(cr)
    if err != nil {
        return managed.ExternalObservation{ResourceExists: true, ResourceUpToDate: false}, errors.Wrap(err, message: "couldn't get ch
    }
    cr.SetConditions(xpv1.Available())
    return managed.ExternalObservation{
        ResourceExists:    true,
        ResourceUpToDate:  !isChanged,
        ConnectionDetails: managed.ConnectionDetails{},
    }, nil
}
```

Рисунок Г.2 – Продовження реалізації Observe методу



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

    cr.SetConditions(xpv1.Creating())
    fmt.Printf( format: "Creating: %+v", cr)

    subnets := cr.Spec.ForProvider.Subnet
    if len(subnets) < 1 {
        return managed.ExternalCreation{}, errors.New( message: "when creating a new network, at least 1 subnet is required")
    }
    creation, err2 := c.checkDescriptions(subnets)
    if err2 != nil {
        return creation, err2
    }
    firstSubnet := subnets[0]
    gateway := c.setValue(firstSubnet, firstSubnet.Gateway)
    dns1 := c.setValue(firstSubnet, firstSubnet.DNS1)
    dns2 := c.setValue(firstSubnet, firstSubnet.DNS2)
    body := &client2.CreateNetworkPostValues{Datacenter: cr.Spec.ForProvider.DatacenterID, Name: cr.Spec.ForProvider.Name, SubnetIp: firstSubnet.IP,
        SubnetBit: firstSubnet.Bit, Gateway: gateway, Dns1: dns1, Dns2: dns2, SubnetDescription: firstSubnet.Description}

    result, err := client2.Request(c.service.Provider,  method: "POST",  path: "service/network/create", body)
    if err != nil {
        return managed.ExternalCreation{}, errors.Wrap(err,  message: "Error during network creation")
    }
    res, externalCreation, err3, done := c.parseResponse(result, err)
    if done {
        return externalCreation, err3
    }
    id := int(res["networkId"].(float64))
    meta.SetExternalName(cr, strconv.Itoa(id))
    for _, subnet := range subnets {
        if subnet.Description != firstSubnet.Description {
            err := addSubnet(c.service.Provider, subnet, cr.Spec.ForProvider.DatacenterID, strconv.Itoa(id))
            if err != nil {
                return managed.ExternalCreation{}, err
            }
        }
    }
    return managed.ExternalCreation{}, nil
}
```

Рисунок Г.3 – Реалізація Create методу



```go
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( format: "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

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

    cr.SetConditions(xpv1.Deleting())

    fmt.Printf( format: "Deleting: %+v", cr.Name)

    for _, subnet := range cr.Status.AtProvider.Subnets {
        err := delSubnet(c.service.Provider, subnet)
        if err != nil { return err }
    }
    body := &client2.DeleteNetworkPostValues{
        Datacenter: cr.Spec.ForProvider.DatacenterID,
        Id:         cr.Status.AtProvider.Id,
    }
    _, err := client2.Request(c.service.Provider,  method: "POST",  path: "service/network/delete", body)
    if err != nil { return errors.Wrap(err,  message: "error during network delete") }

    return nil
}
```

Рисунок Г.5 – Реалізація Delete методу

Додаток Д

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016345940

Дата перевірки:
11.06.2024 08:50:19 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2024 08:57:39 EEST

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://kafedratnopd.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 вимкнене

Вилучення списку бібліографічних посилань вимкнене

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) | ІПЗм-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 Рисунки	
	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 Виноски	
	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