DOI: https://doi.org/10.36910/6775-2524-0560-2024-54-12
УДК 004.738+004.43
Іваненко Олександр Андрійович, бакалавр,
https://orcid.org/0009-0000-6310-020X
Марченко Олександр Іванович, к.т.н., доцент,
https://orcid.org/0000-0002-4537-3420
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікоського», м. Київ, Україна

СПОСІБ УНІФІКОВАНОГО ОПИСУ ХМАРНОЇ ІНФРАСТРУКТУРИ РІЗНИХ ПРОВАЙДЕРІВ

Іваненко О.А., Марченко О.І. Спосіб уніфікованого опису хмарної інфраструктури різних провайдерів. У даній статі запропоновано новий спосіб, який дозволяє описувати хмарну інфраструктуру різних провайдерів в простішому вигляді. Цей спосіб базується на уніфікації деталей, які описуються по різному у різних провайдерів хмарних сервісів, до більшого рівня абстракції, який є спільним для всіх провайдерів. Отриманий уніфікований опис на наступному кроці застосування способу компілюється в код у форматі Infrastructure as Code (IaC) потрібного хмарного провайдера. В результаті, запропонований спосіб, з однієї сторони, універсалізує опис хмарної інфраструктури, а, з іншої сторони, зберігає гнучкість налаштування більш комплексних систем, які потребують деталізації, характерної для кожного провайдера.
Ключові слова: хмарна інфраструктура, хмарний провайдер, Infrastructure as Code (IaC), хмарні ресурси, автоматизація опису.

Ivanenko O., Marchenko O. A unified technique of describing the cloud infrastructure of different providers. This article proposes a new technique to describe the cloud infrastructure of different providers in a simpler way. This technique is based on the unification of details that are described differently by different cloud service providers to a higher level of abstraction that is common to all providers. The resulting unified description is compiled into code in the Infrastructure as Code (IaC) format of the required cloud provider at the next step of the technique application. As a result, the proposed technique, on the one hand, universalizes the description of cloud infrastructure, and, on the other hand, retains the flexibility to configure more complex systems that require details specific to each provider.
Keywords: cloud infrastructure, cloud provider, Infrastructure as Code (IaC), cloud resources, description automation.

Постановка проблеми. Хмарні ресурси стали важливою частиною сучасної технологічної екосистеми, що дає розробникам і компаніям безмежні можливості для швидкого та гнучкого створення інфраструктури. Хмарні провайдери, такі як Amazon Web Services, Microsoft Azure, Google Cloud Platform та інші, від початку свого виникнення зробили доступними потужні обчислювальні ресурси, сервіси зберігання даних, мережеві послуги та багато інших сервісів, які дозволяють покращити технологічні елементи обробки даних. З іншого боку, з ростом кількості ресурсів, які можуть бути використані в хмарних середовищах, виникла потреба в більш керованих і ефективних засобах управління цими ресурсами. Використання засобів опису Infrastructure as Code (IaC) допомагає вирішити цю проблему. Способи опису IaC набувають все більшої популярності разом із розвитком хмарних технологій [1], оскільки у визначенні кодової інфраструктури використовуються концепції програмування, які дозволяють зберігати опис інфраструктури у вигляді коду. Це уможливлює використання контролю версій для опису, спрощує управління кодом та робить інфраструктуру кросплатформеною. Але різні провайдери хмарних ресурсів, як правило, мають свої власні способи та засоби типу IaC, що призводить до суттєвого збільшення необхідних часових та людських ресурсів при розробці комплексних хмарних рішень, в яких використовуються послуги різних провайдерів одночасно. Тому розробка універсальних способів опису хмарної інфраструктури, які дозволяли б розроблювати такі комплексні рішення з меншими витратами є актуальною.

Термінологія.

IaC — Infrastructure as Code — підхід до опису та автоматизації управління інфраструктурними ресурсами [2].

EC2 — Elastic Compute Cloud — ресурс, що надає користувачам віртуальні машини [3].

Security Group - віртуальний брандмауер, який контролює вхідний та вихідний трафік [4].

CIDR — Classless Inter-Domain Routing — використовується для представлення діапазонів IP-адрес в компактній формі [5].

SSE — Server-Side Encryption — шифрування даних у місці призначення програмою або службою, яка їх отримує [6].

CNCF — Cloud Native Computing Foundation [7].

Load Balancer — мережевий сервіс, який розподіляє вхідний трафік між різними серверами чи екземплярами [8].

Аналіз останніх досліджень і публікацій. Компанія HashiCorp створила інструменти, які отримали широке визнання для опису та керування IaC на різних хмарних та віртуальних платформах, таких як AWS, Google Cloud та Azure. У роботі [9] представлено багато кодових прикладів, що ілюструють використання простої, декларативної мови програмування Terraform для розгортання та управління інфраструктурою з мінімальним набором команд. У роботі [10] розглядаються платформи та інструменти, які використовуються для створення та налаштування компонентів інфраструктури, а також надаються шаблони використання цих інструментів. У роботі [11] розглядається високорівневий спосіб опису ресурсів хмарної інфраструктури, але цей спосіб обмежується лише Terraform шаблонами.

Незважаючи на проведені дослідження, поки що відсутній єдиний інструмент, який був би уніфікованим для опису хмарної інфраструктури на різних платформах та залишався б високорівневим та простим у базовому налаштуванні. Такий інструмент дозволив би створювати описи ресурсів на більш високому рівні абстракції та спрощував б процес для менш досвідчених користувачів.

Спосіб опису хмарної інфраструктури CloudFormation. Зростання популярності хмарних сервісів вимагало нових методів автоматизації та управління ресурсами. У цьому контексті виникли засоби інфраструктурного програмування (IaC), що спрощують процеси розгортання та конфігурації хмарної інфраструктури.

Одним із ключових інструментів IaC є AWS CloudFormation [12], який надає можливість визначення інфраструктури у вигляді коду. За допомогою CloudFormation, розробники можуть описати всі компоненти своєї хмарної інфраструктури, включаючи віртуальні машини, бази даних, мережі та інші ресурси, у файлі JSON або YAML. Це дозволяє забезпечити повну та репродуктивну інфраструктуру, а також легко впроваджувати зміни та відновлювати систему у випадку неполадок. Недоліком цього способу є те, що він інтегрується лише з хмарним провайдером AWS. Якщо є потреба у використанні іншого хмарного провайдера, потрібно використовувати інші інструменти, що ускладнює розробку та робить проєкт менш гнучким.

Способи опису хмарної інфраструктури Terraform та Crossplane. Важливим інструментом опису хмарної інфраструктури є Terraform [13], розроблений компанією HashiCorp. Terraform є мультихмарним інструментом IaC, що дозволяє визначати інфраструктуру в коді та працювати з різними хмарними провайдерами, такими як AWS, Azure, Google Cloud та інші. Завдяки своїй гнучкості, Terraform дозволяє розробникам створювати конфігурації, які можна легко адаптувати для різних середовищ та вимог.

Додатковим прикладом еволюції у сфері автоматизації хмарної інфраструктури може слугувати Crossplane [14], розроблений у межах CNCF. Цей інструмент пропонує єдиний інтерфейс для керування ресурсами різних провайдерів, створюючи абстракцію від конкретних деталей кожного. Crossplane дозволяє користувачам спростити процес створення та керування хмарними ресурсами без необхідності глибокого вивчення конкретного інструментарію кожного провайдера. Не дивлячись на інноваційність та переваги, Crossplane не вирішує всіх проблем автоматизації хмарної інфраструктури. Деякі користувачі вказують на складність конфігурації та недостатню документацію, особливо для менш популярних хмарних провайдерів чи специфічних сценаріїв використання.

Загальні недоліки існуючих способів. Незважаючи на значний внесок інструментів IaC у спрощення та автоматизацію управління інфраструктурою, існують загальні недоліки. Висока деталізація ресурсів може призвести до складних конфігурацій, ускладнюючи розуміння та підтримку коду. Крім того, прив'язка до конкретного хмарного провайдера або інструменту може обмежити переносимість між різними середовищами. На сучасному етапі розвитку кодової інфраструктури все ще не існує повністю уніфікованого та високорівневого інструменту, який відповідав би всім потребам та вимогам користувачів. Існуючі способи та засоби, хоч і надають

значний рівень автоматизації, все ще виявляються специфічними для кожного хмарного провайдера та вимагають глибоких знань щодо їхньої роботи.

Відсутність єдиного стандарту опису ресурсів для різних провайдерів та невизначеність у визначенні високорівневих концепцій ускладнюють розробку універсального інструменту [15]. Зробити кодову інфраструктуру універсальною та зрозумілою для розробників буде викликом, який вимагає подальших досліджень та стандартизації в галузі автоматизації хмарної інфраструктури.

Постановка завдання. Запропонувати спосіб опису хмарної інфраструктури, який мав би високий рівень абстракції та уніфікації опису і забезпечував би універсальність для використання на різних платформах та у середовищах різних хмарних провайдерів, а також дозволяв би компіляцію цього опису у деталізований код для відповідних провайдерів, спрощуючи процес розгортання і керування хмарною інфраструктурою та забезпечуючи можливість такого керування користувачами незалежно від їхнього досвіду у галузі хмарних технологій.

Порівняння деталей опису базової інфраструктури існуючими способами. Для з'ясування можливостей універсалізації опису хмарної інфраструктури та пропозиції нового уніфікованого способу опису розглянемо деталі опису базової інфраструктури способами Crossplane, CloudFormation та Terraform у порівняльній формі. Для такого порівняльного аналізу та виокремлення однакових елементів різних інструментів опису хмарної інфраструктури у якості прикладу було обрано створення віртуальної машини EC2 з відповідною Security Group та механізму розподілу навантаження Load Balancer. Кожен з зазначених способів потребує оголошення провайдера хмарних послуг, де буде відбуватися наступний опис ресурсів (рис. 1).

```
Crossplane
apiVersion: aws.crossplane.io/v1alpha3


Cloudformation
AWSTemplateFormatVersion: '2010-09-09'

Terraform
Textprovider "aws" {
  region = "us-east-1"
}
```

Рис. 1 – Оголошення провайдера хмарних послуг

Оголошення нової групи безпеки потребує явного вказання типу ресурсу. В нашому випадку синтаксис опису типу ресурсу – групи *AWS Security Group* – є різним у різних провайдерів. Також потрібно вказати назву ресурсу – *SecurityGroupName* (рис. 2).

```
Crossplane
kind: SecurityGroup
metadata:
  name: SecurityGroupName

Cloudformation
Resources:
  SecurityGroupName:
    Type: AWS::EC2::SecurityGroup


Terraform
resource "aws_security_group" "SecurityGroupName"
  name        = "SecurityGroupName"
  description = "Security Group"
```

Рис. 2 – Оголошення типу та імені для групи безпеки

При оголошенні групи безпеки обов'язковим є блок базових правил доступу до групи. Ці налаштування є базовими для всіх способів опису. Правило з назвою *protocol* вказує на те, що воно стосується транспортного протоколу TCP. Правила з назвами *from_port* та *to_port* встановлюють проміжок діапазону для вхідного порту. В нашому випадку цими правилами обмежується трафік,

який проходить через порт 80. Всі ці налаштування є базовими та використовуються у такому форматі в багатьох проєктах. Основним динамічним правилом є *cidrIP*, що вказує діапазон IP-адрес, які мають доступ до вхідного трафіку групи безпеки. Security Group створюється саме для того, щоб дати дозвіл відповідним адресам до потрібного ресурсу, тому це правило буде часто змінюватись. У наведеному прикладі використовується 0.0.0.0/0 – дозвіл будь-яким IP-адресам в мережі інтернет (рис 3).

```
Crossplane
spec:
  forProvider:
    ingress:
      - fromPort: 80
        toPort: 80
        protocol: tcp
        cidrBlocks:
          - 0.0.0.0/0
```

```
Cloudformation
Properties:
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
```

```
Terraform
ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
```

Рис. 3 – Оголошення правил доступу до групи безпеки

Розглянемо тепер створення віртуальної машини – екземпляру EC2. Для створення цього ресурсу також потрібно вказати тип ресурсу *AWS Instance* та ім'я *EC2InstanceName* при використанні кожного із способів опису (рис. 4).

```
Crossplane
kind: Instance
metadata:
  name: EC2InstanceName

Cloudformation
EC2InstanceName:
  Type: AWS::EC2::Instance

Terraform
resource "aws_instance" "EC2InstanceName"
```

Рис. 4 – Оголошення типу та імені для екземпляру EC2

Для створення екземпляру EC2 потрібно вказати такі характеристики як *imageId* та *instanceType*, де *imageId* – це ідентифікатор образу Amazon Machine Image (AMI), що визначає образ операційної системи, який буде використаний для створення екземпляра EC2. В даному прикладі *ami-34gh6h1o4782ga42n* є ідентифікатором конкретного AMI, який було обрано для екземпляра. *InstanceType* – це тип екземпляра EC2. В даному прикладі обрано найпростіший тип *t2.micro* (рис. 5).

```
Crossplane
forProvider:
  imageID: ami-34gh6h1o4782ga42n
  instanceType: t2.micro

Cloudformation
Properties:
   ImageId: ami-34gh6h1o4782ga42n
   InstanceType: t2.micro

Terraform
ami = "ami-34gh6h1o4782ga42n"
instance_type = "t2.micro"
```

Рис. 5 – Оголошення imageId та instaceType для екземпляру EC2

Наступним необхідним елементом для створення віртуальної машини є посилання на групу безпеки, яка буде контролювати вхідний трафік до створеного екземпляра. Кожен екземпляр пов'язується з однією чи кількома групами безпеки, які визначають правила фільтрації трафіку для цього екземпляра. У даному прикладі визначене посилання на створену вище групу безпеки *SecurityGroupName* (рис. 6).

```
Crossplane
securityGroupIDs:
   - SecurityGroupName


Cloudformation
SecurityGroupIds:
   - !Ref SecurityGroupName


Terraform
security_group = [aws_security_group.MySecurityGroup.id]
```

Рис. 6 – Оголошення посилання на групу безпеки

Ще одним ресурсом, який потрібно описати для даного прикладу хмарної інфраструктури, є механізм розподілу навантаження. Механізм розподілу навантаження, або LoadBalancer, використовується для рівномірного розподілу трафіку або робочих навантажень між різними серверами чи ресурсами в мережі. Як і в попередніх випадках, для створення цього ресурсу необхідно вказувати тип та ім'я (рис. 7).

```
Crossplane
kind: LoadBalancer
metadata:
  name: LoadBalancerName

Cloudformation
LoadBalancerName:
  Type: AWS::ElasticLoadBalancing::LoadBalancer


Terraform
resource "aws_elb" "LoadBalancerName"
```

Рис. 7 – Оголошення типу та імені для механізму розподілу навантаження

Для коректної роботи механізму розподілу навантаження потрібно описати доступні зони, в яких буде розгорнуто ресурс. У цьому прикладі для доступу були обрані зони *us-east-1a* та *us-east-1b* (рис. 8).

```
Crossplane
availabilityZones:
    - us-east-1a
    - us-east-1b

Cloudformation
AvailabilityZones:
     - us-east-1a
     - us-east-1b

Terraform
availability_zones = ["us-east-1a", "us-east-1b"]
```

Рис. 8 – Оголошення зон доступності для механізму розподілу навантаження

Блок *Listeners* визначає порти, на яких LoadBalancer слухатиме трафік, та протокол, по якому він буде перенаправляти його до екземплярів. В цьому прикладі, LoadBalancer слухатиме трафік на порту 80 і перенаправлятиме його по протоколу HTTP також на порт 80 екземплярів (рис. 9).

```
Crossplane
listeners:
    - loadBalancerPort: 80
      instancePort: 80
      protocol: HTTP

Cloudformation
Listeners:
    - LoadBalancerPort: 80
      InstancePort: 80
      Protocol: HTTP

Terraform
listener {
    instance_port     = 80
    instance_protocol = "HTTP"
    lb_port           = 80
    lb_protocol       = "HTTP"
  }
```

Рис. 9 – Оголошення портів та мережевого протоколу для механізму розподілу навантаження

Останнім полем опису механізму розподілу навантаження є блок *Instances*. Він визначає екземпляри EC2, які будуть обслуговувати трафік, отриманий від механізму розподілу навантаження. В даному прикладі використовується посилання на значення, що зазначене в ресурсі *EC2InstanceName*. Таким значенням може бути ім'я або ідентифікатор іншого ресурсу, наприклад, EC2 екземпляра (рис. 10).

```
Crossplane
instances:
    - EC2InstanceName

Cloudformation
Instances:
    - !Ref EC2InstanceName

Terraform
instances = [aws_instance.MyEC2Instance.id]
```

Рис. 10 – Оголошення посилання на екземпляр EC2

У всіх зазначених описах спостерігається висока деталізація та високі вимоги до кваліфікації розробників у контексті опису конфігурації інфраструктури. Але важливо відзначити, що частина налаштувань інфраструктурного коду є базовою, і ці налаштування можуть бути уніфіковані та приховані від розробника.

Запропонований спосіб опису хмарної інфраструктури. Авторами пропонується новий спосіб опису хмарної інфраструктури, який базується на уніфікації деталей, що описуються по різному у різних провайдерів хмарних сервісів, до більш високого рівня абстракції, та який має JSON-подібний синтаксис.

Для досягнення більшого рівня абстракції спрощуються або приховуються загальні блоки налаштування яких, зазвичай, повторюються у більшості проектів. Прикладами спрощених блоків можуть бути блоки *ingress* та *egress*, які вказують правила доступу до ресурсу, або блок *server_side_encryption_configuration*, який вказує SSE правила для хмарного сховища даних. При використанні запропонованого способу опису хмарної інфраструктури блоки *ingress* та *egress* спрощуються до визначення доступних IP-адрес (*cidrBlocks*), а всі інші елементи приховуються. Аналогічно блок SSE правил спрощується до вказання лише потрібного алгоритму шифрування.

Наприклад, на рис. 11 показаний повний опис шаблону мовою Terraform, який буде згенерований компілятором зі спрощеного опису, зробленого запропонованим способом опису (рис. 12).

```
ingress {
  from_port   = 80
  to_port     = 80
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

egress {
  from_port   = 0
  to_port     = 0
  protocol    = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}

 server_side_encryption_configuration {
   rule {
     apply_server_side_encryption_by_default {
       sse_algorithm = "AES256"
      }
    }
 }
```

Рис. 11 – Приклад опису блоків шаблону, які будуть згенеровані зі спрощеного опису запропонованим способом

```
ingress / egress
cidrBlocks: "0.0.0.0/0"

SSE config
sse.algorithm = "AES256"
```

Рис. 12 – Приклад спрощеного опису блоків шаблону запропонованим способом

Прикладом блоку опису, який може бути прихований, є блок *listener*, в якому вказуються основні порти та протокол доступу до ресурсу (рис. 13). Цей блок опису також буде присутнім у вихідному згенерованому шаблоні для можливості додавання додаткових налаштувань, але з

початкового опису за запропонованим способом він виключається, оскільки ці налаштування є базовими та розповсюдженими серед багатьох проектів.

```
  listener {
    instance_port     = 80
    instance_protocol = "HTTP"
    lb_port           = 80
    lb_protocol       = "HTTP"
  }
```

Рис. 13 – Приклад блоку опису шаблона, який може бути прихований в початковому описі за запропонованим способом

Передбачається, що опис хмарної інфраструктури за запропонованим способом буде компілюватися у опис цієї інфраструктури мовами трьох розглянутих вище способів опису CloudFormation, Crossplane та Terraform для забезпечення гнучкості розробки та ефективної підтримки комплексної хмарної інфраструктури в різних проектах. Тобто, в залежності від необхідності використовувати сервіси певного хмарного провайдера, компілятором буде генеруватися код опису інфраструктури способом саме цього провайдера, а початковий опис запропонованим способом буде залишатися сталим. Назва провайдера хмарних послуг, для якого компілятор буде генерувати код опису хмарної інфраструктури, зазначається на самому початку у вигляді, показаному на рис. 14.

```
provider: "aws"
```

Рис. 14 – Оголошення хмарного провайдера в описі запропонованим способом

Розглянемо опис ресурсів для того самого прикладу створення віртуальної машини EC2. Оголошення групи безпеки забезпечується вказанням тільки типу та імені ресурсу. Важливий елемент з визначення доступних IP-адрес також залишається, а всі інші деталі приховуються від користувача (рис. 15). Але всі ці деталі будуть присутні в згенерованому компілятором коді опису конкретного провайдера.

```
securityGroup SecurityGroupName {
    cidrBlocks: "0.0.0.0/0"
}
```

Рис. 15 – Оголошення групи безпеки в описі запропонованим способом

Оголошення екземпляру майже не зазнало змін, але запропонований JSON-подібний формат робить його більш простим і зручним. Як і раніше, вказано тип ресурсу, ім'я ресурсу, ідентифікатор віртуального образу та тип екземпляра. За допомогою символа «&» вказується посилання на будь-який інший ресурс. Наприклад, на рис.16 вказане посилання на групу безпеки.

```
instance EC2InstanceName {
    id: "ami-34gh6h1o4782ga42n"
    type: "t2.micro"

  securityGroups: &SecurityGroupName
}
```

Рис. 16 – Оголошення екземпляру EC2 в описі запропонованим способом

В оголошенні механізму розподілу навантаження було приховано базовий блок *listener*. Доступні зони вказуються за допомогою переліку декількох елементів, що записуються через кому. Посилання на екземпляр реалізовано за допомогою символа «&» (рис. 17).

```
loadBalancer LoadBalancerName {
    zones: "us-east-1a", "us-east-1b"
    instances: &EC2InstanceName
}
```

Рис. 17 – Оголошення екземпляру EC2 в описі запропонованим способом

У результаті можна побачити, що при використанні запропонованого способу деталі опису деяких ресурсів приховуються, а опис інших елементів налаштування, що були розглянуті у прикладах, подається в більш читабельному та зручнішому форматі.

Висновки. Авторами запропоновано уніфікований спосіб опису хмарної інфраструктури, який робить процес опису більш абстрактним та простим. Для прикладу було показано опис створення групи безпеки, віртуальної машини та механізму розподілу навантаження способами AWS CloudFormation, Terraform та Crossplane. Було зазначено недоліки існуючих рішень та виділено схожі елементи, які можна уніфікувати. Особливість запропонованого способу полягає у тому, що для його використання не вимагається глибоких знань мережевих технологій, а процес конфігурації хмарної інфраструктури стає більш доступним для широкого кола користувачів.

Запропонований спосіб опису хмарної інфраструктури дозволяє створювати конфігурації для роботи з різними хмарними провайдерами, які використовують різні способи опису хмарної інфраструктури, в результаті чого конфігурування та управління інфраструктурою стають більш універсальними та простими. Важливо зазначити, що типи ресурсів, в залежності від провайдера, можуть відрізнятись, але запропонований спосіб надає можливість описувати інфраструктуру з генерацією детального опису різних типів ресурсів будь-яким з існуючих способів опису, що будуть реалізовані в компіляторі, на вхід якого подається уніфікований опис інфраструктури запропонованим способом. Результатом компіляції буде повноцінний скрипт, записаний відповідно до способу опису потрібного хмарного провайдера. Такий спосіб опису дозволяє командам розробників, що працюють над різними проєктами, використовувати стандартизовані практики незалежно від обраного провайдера та інструментів, що використовуються в конкретному проєкті.

Важливим аспектом запропонованого способу є те, що він реалізований мовою, яка визначається формальною граматикою, що дозволяє легко розширювати можливості цього способу і реалізовувати компілятори з цієї мови у мови опису будь-яких провайдерів хмарних сервісів. Ця властивість запропонованого способу опису робить його перспективним для майбутнього використання, оскільки він може легко адаптуватися до змін в екосистемах хмарних сервісів, відображаючи нові тенденції у розробці інфраструктурного коду, які будуть з'являтися.

Список бібліографічного опису

1. Google Trends "Infrastructure as Code" . 2024. URL: https://trends.google.com/trends/explore?date=2013-01-01%202023-01-01&q=Infrastructure%20as%20Code&hl=uk
2. Infrastructure as Code . 2024. URL: https://docs.aws.amazon.com/whitepapers/latest/introduction-devops-aws/infrastructure-as-code.html
3. Amazon EC2. 2024. URL: https://docs.aws.amazon.com/whitepapers/latest/ec2-networking-for-telecom/amazon-ec2.html
4. Security groups and network ACLs. 2024. URL: https://docs.aws.amazon.com/whitepapers/latest/aws-best-practices-ddos-resiliency/security-groups-and-network-acls-bp5.html
5. VPC CIDR blocks . – 2024. URL: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-cidr-blocks.html
6. Protecting data with server-side encryption. 2024. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html
7. Alysa Fallon Cloud Native Computing Foundation . 2024. URL: https://www.techtarget.com/searchitoperations/definition/Cloud-Native-Computing-Foundation-CNCF

8. Elastic Load Balancer . 2024. URL: https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html
9. Yevgeniy Brikman, Terraform: Up and Running: Writing Infrastructure as Code – 2017 – С. 6-78. URL: https://www.amazon.com/Terraform-Running-Writing-Infrastructure-Code/dp/1492046906
10. Infrastructure as Code: Managing Servers in the Cloud 2016. С. 5-123. URL: https://www.amazon.com/Infrastructure-Code-Managing-Servers-Cloud/dp/1491924357
11. Хомутник Д. Ю., Марченко О.І. Високорівневий спосіб опису ресурсів хмарної інфраструктури. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2022. № 48. – с.117-123. URL: https://doi.org/10.36910/6775-2524-0560-2022-48-18
12. AWS CloudFormation URL: https://aws.amazon.com/cloudformation/
13. Terraform documentation URL: https://www.terraform.io/docs.
14. Crossplane Knowledge Base URL: https://docs.crossplane.io/knowledge-base/
15. Shein E. The most important cloud advances of the decade / Esther Shein // TechRepublic. 2019. URL: https://www.techrepublic.com/article/the-most-important-cloud-advances-of-the-decade/.

References

1. Google Trends, Infrastructure as Code, https://trends.google.com/trends/explore?date=2013-01-01%202023-01-01&q=Infrastructure%20as%20Code&hl=uk. Accessed 5 Jan. 2024
2. "Infrastructure as Code." *Amazon*, Amazon Web Services, Inc., https://docs.aws.amazon.com/whitepapers/latest/introduction-devops-aws/infrastructure-as-code.html. Accessed 5 Jan. 2024
3. "Amazon EC2." *Amazon*, Amazon Web Services, Inc., https://aws.amazon.com/ec2/. Accessed 6 Jan. 2024
4. "Security groups and network ACLs." *Amazon*, Amazon Web Services, Inc., https://docs.aws.amazon.com/whitepapers/latest/aws-best-practices-ddos-resiliency/security-groups-and-network-acls-bp5.html. Accessed 6 Jan. 2024
5. "VPC CIDR blocks." *Amazon*, Amazon Web Services, Inc., https://docs.aws.amazon.com/vpc/latest/userguide/vpc-cidr-blocks.html. Accessed 6 Jan. 2024
6. "Protecting data with server-side encryption" *Amazon*, Amazon Web Services, Inc., https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html. Accessed 6 Jan. 2024
7. Cloud Native Computing Foundation, Alyssa Fallon, https://www.techtarget.com/searchitoperations/definition/Cloud-Native-Computing-Foundation-CNCF. Accessed 7 Jan. 2024
8. "Elastic Load Balancer" *Amazon*, Amazon Web Services, Inc., https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html. Accessed 7 Jan. 2024
9. Yevgeniy Brikman, Terraform: Up and Running: Writing Infrastructure as Code, 2017, p. 6-78., https://www.amazon.com/Terraform-Running-Writing-Infrastructure-Code/dp/1492046906
10. Infrastructure as Code: Managing Servers in the Cloud, 2016, p. 5-123., https://www.amazon.com/Infrastructure-Code-Managing-Servers-Cloud/dp/1491924357
11. Marchenko O.I., Khomutnyk D.Yu., High-level technique for description of cloud infrastructure resources, Computer-integrated technologies: education, science, production. 2022, № 48. – с.117-123.
12. "AWS CloudFormation" *Amazon*, Amazon Web Services, Inc., https://aws.amazon.com/cloudformation/. Accessed 8 Jan. 2024
13. "Terraform Documentation." *Terraform*, HashiCorp, https://www.terraform.io/docs/. Accessed 9 Sep. 2022
14. Crossplane Knowledge Base, *Crossplane.* https://docs.crossplane.io/knowledge-base/. Accessed 9 Jan 2024
15. Shein, Esther. "The Most Important Cloud Advances of the Decade." *TechRepublic*, https://www.techrepublic.com/article/the-most-important-cloud-advances-of-the-decade/. Accessed 7 Sep. 2022