DOI 10.15276/imms.v15.no4.636
УДК 004.75:004.77

Informatics and Mathematical Methods in Simulation
Vol.15 (2025), No. 4, pp. 636-646

ПОРІВНЯЛЬНИЙ АНАЛІЗ БЕЗПЕКОВИХ ПАТЕРНІВ ХМАРНИХ ПЛАТФОРМ У КОНТЕКСТІ НАУКОВОЇ ВІДТВОРЮВАНОСТІ

О. Г. Трофименко[1], П. О. Чикунов[1], Д. Ю. Астахов[1], Ю. В. Молоканов, Т. А. Фаріонова[2]

[1]Національний університет «Одеська юридична академія»
23, Фонтанська дорога, Одеса, 65009, Україна
[2]Національний університет кораблебудування імені адм. Макарова
9, Героїв України пр., Миколаїв, 54007, Україна
Emails: trofymenko@onua.edu.ua, pavel@onua.edu.ua, astakhovnil@gmail.com, molokanov9@gmail.com, tetyana.farionova@nuos.edu.ua

Дослідження присвячене системному аналізу реалізації ключових безпекових патернів у хмарних платформах Amazon Web Services, Microsoft Azure та Google Cloud Platform з урахуванням їх придатності для наукових досліджень у галузі комп'ютерних наук. Актуальність дослідження зумовлена потребою в забезпеченні безпеки, ізоляції, автоматизації та відтворюваності інфраструктури в умовах мультихмарного середовища. Мета дослідження полягає у систематизації та порівняльному аналізі ключових безпекових патернів у хмарних платформах AWS, Azure та GCP з урахуванням їх ефективності, обмежень та придатності для забезпечення відтворюваності, ізоляції та етичної відповідності в наукових дослідницьких середовищах. Методологія дослідження базується на системному підході, який включає порівняння конфігурацій хмарних платформ на основі практичних кейсів, застосування інструментів статичного аналізу (Checkov, tfsec) для оцінки безпеки IaC-коду, моделювання поведінки системи у сценаріях відмови та атак для емпіричної перевірки стійкості інфраструктури. У статті розглянуто три практичні кейси: 1) для AWS – реалізація Role Vending Machine для автоматизованого призначення тимчасових ролей з обмеженими правами, інтеграція з GitHub Actions, Checkov та IAM Access Analyzer; 2) для Azure – багатозонна архітектура з PIM, GitOps, ARM-шаблонами та мережевою сегментацією для моделювання відмовостійкості; 3) для GCP – використання Shared VPC, ізольованих проєктів, TPU та Vertex AI Experiments для задач NLP і глибокого навчання. Порівняльний аналіз показав: принцип найменших привілеїв значно знижує ризики витоку прав; відсутність логування та аудиту створює "мертві" права, які можуть бути точками входу для атак; ізоляція середовищ (через окремі акаунти, підписки, проєкти) критично важлива для запобігання перехресному впливу; управління змінами через CI/CD або GitOps забезпечує повторюваність і контроль; Threat modeling дозволяє виявити уразливості, пов'язані з розміщенням ресурсів; регуляторні вимоги (GDPR, шифрування, геозони) мають бути враховані на етапі архітектурного планування. Проведене дослідження дозволяє зробити висновок, що хмарна інфраструктура, налаштована з урахуванням принципів безпеки, автоматизації та етичної відповідності, може стати повноцінним науковим інструментом, а не лише технічним середовищем. Водночас нехтування цими принципами створює ризики компрометації даних, порушення регуляторних норм та втрати достовірності результатів. Наукова новизна роботи полягає у порівняльному аналізі безпекових моделей трьох хмарних платформ у контексті забезпечення достовірності та етичної відповідності наукових експериментів. Практична значущість результатів полягає у можливості їх використання для побудови безпечної, масштабованої та відтворюваної хмарної інфраструктури в наукових проєктах, у професійній діяльності DevOps-команд, які працюють з мультихмарними середовищами, як основа для розробки етичних та регуляторно відповідних архітектур у сфері комп'ютерних наук.

Ключові слова: хмарні обчислення, мультихмарне середовище, інфраструктура як код, DevOps, DevSecOps, безпекові патерни, CI/CD, GitOps, хмарна архітектура.

Вступ. Хмарні обчислювальні платформи стали фундаментальним компонентом сучасної наукової інфраструктури, особливо в галузі комп'ютерних наук. Вони

забезпечують масштабоване середовище для реалізації алгоритмічних моделей, високопродуктивних симуляцій, розгортання нейронних мереж, зберігання великих обсягів даних та автоматизації експериментальних циклів. В умовах обмежених локальних ресурсів і потреби в гнучкому масштабуванні платформи Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform (GCP) пропонують інструменти, які дозволяють не лише ефективно керувати обчисленнями, а й забезпечити наукову точність, повторюваність та відтворюваність результатів.

У контексті стрімкого розвитку хмарних технологій та DevOps-практик, хмарна інфраструктура стала критично важливою для реалізації високопродуктивних обчислень, зберігання даних і автоматизації наукових експериментів.

Аналіз досліджень та публікацій показує активне зростання інтересу до застосування мультихмарних архітектур, реалізації ключових безпекових патернів у хмарних платформах. У статті [1] виконано огляд концепції «Інфраструктура як код» (Infrastructure as Code, IaC) для мультихмарних налаштувань, зосереджуючись на модульній архітектурі, стандартизації інструментів, управлінні, інтеграції безпеки та автоматизації через конвеєри CI/CD (Continuous Integration / Continuous Delivery). Дослідження [2] підкреслює трансформаційний потенціал використання кількох хмарних архітектур у сучасних корпоративних середовищах, наголошуючи на їхній ролі в досягненні бізнес-гнучкості, оптимізації витрат та операційної ефективності. У статтях [3, 4] обговорюються найкращі практики впровадження автоматизації безпеки за допомогою практик статичного та динамічного тестування безпеки застосунків (SAST/DAST), автоматизовані перевірки відповідності, захист під час виконання за допомогою сканера безпеки інфраструктури як коду, рішення для безпеки контейнерів та системи виявлення аномалій на основі поведінки. Дослідження [5] заглиблюється в перетин IaC, CI/CD та оркестрації Kubernetes у мультихмарних DevOps-конвеєрах. Робота [6] аналізує еволюцію мультихмарної стратегії і роль Terraform у хмарній оркестрації з невеликим порівнянням з іншими інструментами хмарної оркестрації. У статті [7] розглядається концепція інфраструктури як коду з позиції її реалізації за допомогою Terraform, висвітлюються переваги та проблеми IaC у сучасних хмарних середовищах. У роботі [8] класифіковано широко визнані методи безпеки Terraform, поширені в галузі для популярних хмарних провайдерів, таких як AWS, Azure та Google Cloud.

Проведений аналіз засвідчує наявність ґрунтовної дослідницької бази щодо хмарних платформ. Водночас, попри широке застосування хмарних сервісів у наявних публікаціях недостатньо висвітлено питання інфраструктурної безпеки, ізоляції середовищ, управління ролями та відтворюваності конфігурацій у контексті наукової методології. Зокрема, відсутній системний аналіз того, як принцип найменших привілеїв, аудит доступу та моделювання загроз впливають на достовірність і безпечність експериментальних результатів.

Метою даної роботи є систематизація та порівняльний аналіз реалізації ключових безпекових патернів у хмарних платформах Amazon Web Services, Microsoft Azure та Google Cloud Platform з урахуванням їх ефективності, обмежень та придатності для забезпечення відтворюваності, ізоляції та етичної відповідності в наукових дослідницьких середовищах.

Для досягнення поставленої мети визначено такі завдання:

– проаналізувати сучасні наукові публікації та дослідження для виявлення ключових напрямів, тенденцій та проблем у застосуванні патернів хмарної інфраструктури для наукових досліджень у галузі комп’ютерних наук, з акцентом на безпеку, ізоляцію, автоматизацію та відповідність етичним нормам;

– дослідити переваги інфраструктури як коду для DevOps;

– виконати порівняльний аналіз трьох провідних хмарних платформ AWS, Azure, GCP щодо їх придатності для дослідницьких задач;

– продемонструвати, як правильно налаштовані хмарні середовища можуть забезпечити відтворюваність, масштабованість та захист даних, необхідні для наукової достовірності.

Переваги інфраструктури як коду для DevOps. Розвиток хмарних технологій є одним із ключових напрямів сучасної інформатизації, що визначає ефективність цифрової трансформації бізнесу, науки та державного управління. Останнє десятиріччя позначене переходом від простих моделей оренди обчислювальних ресурсів до комплексних платформ, орієнтованих на масштабованість, автоматизацію та інтеграцію із сучасними методологіями розробки. Якщо на початкових етапах розвитку хмарні обчислення концентрувалися на забезпеченні гнучкого доступу до інфраструктури у форматі IaaS (Infrastructure as a Service), то наразі спостерігається зростання популярності платформних рішень (PaaS, Platform as a Service) та сервісів, орієнтованих на мікросервісну архітектуру, контейнеризацію й оркестрацію за допомогою системи Kubernetes.

Важливим трендом є поширення мультихмарних та гібридних стратегій, які дозволяють поєднувати ресурси різних провайдерів з метою оптимізації витрат, підвищення відмовостійкості та уникнення ризику «vendor lock-in». Це вимагає розробки нових інструментів управління, здатних забезпечити уніфікований підхід до оркестрації інфраструктури, незалежно від її розподілу між AWS, Microsoft Azure, Google Cloud чи приватними дата-центрами [6]. Одночасно з цим посилюється акцент на безпеці, дотриманні нормативних вимог та автоматизації процесів відповідно до DevOps-практик [7]. Як результат формується потреба у системному підході до управління інфраструктурою, що дозволяє скоротити час розгортання середовищ, підвищити їх передбачуваність та якість обслуговування.

У цьому контексті концепція «Інфраструктура як код» стала однією з базових технологічних передумов переходу до інженерії сучасних хмарних середовищ. IaC описує інфраструктуру у вигляді декларативних чи імперативних конфігураційних файлів, що дозволяє автоматизувати процеси створення, масштабування, модифікації та видалення ресурсів [8]. Тим самим IaC дозволяє командам автоматизувати та керувати інфраструктурою за допомогою програмного коду, забезпечуючи узгодженість, масштабованість та швидше розгортання. Такий підхід забезпечує прозорість і відтворюваність усіх дій, оскільки інфраструктура визначається так само, як і програмний код, а її стан зберігається у системах контролю версій.

На рис. 1 зображено типовий робочий процес IaC, який охоплює основні компоненти та взаємозв'язки між ними.

Рис. 1. Інфраструктура як код

Процес починається з дій розробника або інженера DevOps, який створює код інфраструктури, використовуючи спеціалізовані мови опису, як-от: Terraform, Ansible або CloudFormation. Цей код містить декларативні інструкції щодо створення, налаштування та управління обчислювальними ресурсами, включаючи сервери, мережі, бази даних та інші компоненти. Після написання, код передається до системи контролю

версій, найчастіше Git, що дозволяє зберігати історію змін, здійснювати пул-реквести та забезпечувати колективну роботу над інфраструктурними сценаріями.

Затверджений код надходить до автоматизаційного сервера або API-рушія, який виконує його інтерпретацію та застосування. Цей компонент відповідає за інтеграцію з хмарними платформами або локальними дата-центрами, забезпечує розгортання інфраструктури відповідно до заданих параметрів. У результаті інфраструктура може бути реалізована як у хмарному середовищі, так і на локальних серверах, що забезпечує гнучкість та масштабованість IT-рішень.

Важливою особливістю IaC є її здатність забезпечувати ідентичність середовищ, зменшувати ризики людських помилок та сприяти впровадженню практик DevOps, таких як CI/CD. Тим самим інфографіка демонструє не лише технічну послідовність дій, а й концептуальну модель переходу від ручного управління інфраструктурою до її повної автоматизації через код.

Використання IaC у рамках DevOps дає змогу досягати низки важливих переваг. По-перше, автоматизація розгортання суттєво зменшує кількість людських помилок, характерних для ручного налаштування інфраструктури [9]. По-друге, забезпечується повторюваність середовищ, що критично важливо для тестування, розробки та продуктивної експлуатації [10]. По-третє, IaC інтегрується з CI/CD-процесами, дозволяючи створювати інфраструктуру «на вимогу» у відповідь на зміни в програмному забезпеченні. По-четверте, завдяки централізованому управлінню станом та конфігураціями підвищується рівень безпеки й керованості мультихмарних середовищ [11].

Крім того, IaC дозволяє організаціям оптимізувати витрати на хмарні ресурси, програмно керуючи ресурсами, забезпечуючи автоматичне вилучення невикористаних ресурсів. Це запобігає непотрібним витратам на хмарні ресурси та їх втратам. Сучасні інструменти IaC підтримують мультихмарні середовища, дозволяючи організаціям безперешкодно розгортати робочі навантаження в AWS, Azure та Google Cloud. Це дозволяє уникнути прив'язки до постачальника та підвищує гнучкість [9].

Отже, IaC є важливою практикою в сучасному DevOps, яка пропонує швидкість, масштабованість, автоматизацію та узгодженість в управлінні інфраструктурою. Використовуючи інструменти IaC, як-от: Terraform, Ansible та AWS CloudFormation, DevOps-фахівці можуть оптимізувати операції, знизити витрати та покращити доставку програмного забезпечення.

Порівняння ключових безпекових підходів та практик AWS, Azure та GCP. У кожній із найпотужніших хмарних платформ – AWS, Azure і GCP – реалізовано унікальні механізми, які дозволяють адаптувати інфраструктуру до специфіки дослідницьких задач.

Terraform як інструмент IaC дозволяє декларативно описувати, створювати та керувати інфраструктурою в хмарних середовищах. Його головна перевага – уніфікований підхід до роботи з різними хмарами, зокрема AWS, Azure та GCP.

Terraform використовує спеціальні провайдери для кожної хмарної платформи: aws – для Amazon Web Services, azurerm – для Microsoft Azure, google – для Google Cloud Platform. Користувач описує бажаний стан інфраструктури у .tf файлах, а Terraform самостійно визначає, які ресурси створити, змінити або видалити. При цьому один і той самий інструмент і синтаксис дозволяє працювати з різними хмарами (табл. 1), що спрощує гібридні або мультихмарні архітектури.

У табл. 2 наведено узагальнені ключові безпекові підходи та практики у розглянутих трьох провідних хмарних платформ у розрізі категорій, що мають вирішальне значення для побудови безпечної та відтворюваної інфраструктури за допомогою Terraform у мультихмарному середовищі. Порівняння дозволяє виявити як спільні стратегічні підходи, так і відмінності в реалізації конкретних механізмів, що мають критичне значення в наукових та інженерних обчисленнях.

Таблиця 1.

Особливості інтеграції з кожною платформою

Платформа	Провайдер	Приклади ресурсів	Особливості
AWS	aws	aws_instance, aws_s3_bucket, aws_lambda_function	Найбільш зрілий провайдер, підтримує майже всі сервіси AWS
Azure	azurerm	azurerm_virtual_machine, azurerm_storage_account, azurerm_function_app	Потребує автентифікацію через Azure CLI або Service Principal
GCP	google	google_compute_instance, google_storage_bucket, google_cloudfunctions_function	Підтримка через Google Cloud SDK або ключі доступу

Таблиця 2.

Порівняльна таблиця безпекових патернів AWS, Azure та GCP

Категорія	AWS	Asure	GCP
Найменші привілеї	IAM Role Vending Machine (RVM)	Privileged Identity Management (PIM)	Custom IAM Policies + Time-bound Access
Логування та аудит	IAM Access Analyzer, CloudTrail	Activity Logs, PIM Audit	VPC Flow Logs, Cloud Audit Logs
Ізоляція середовищ	Окремі VPC, окремі акаунти	Окремі підписки, групи ресурсів	Shared VPC + окремі проєкти
Управління змінами	CI/CD з GitHub Actions + Checkov	GitOps + ARM Templates	Terraform + Cloud Build
Моделювання загроз	Placement-aware EC2, network ACLs	WAF, NSG, segmentation	Co-location analysis, TPU isolation
Регуляторна відповідність	KMS, region control, data lifecycle	Encryption at rest/in transit, GDPR zones	Data residency, DLP API
Економічна ефективність	Auto-stop EC2, spot instances	Budget alerts, cost analysis	Preemptible VMs, region-aware planning

У категорії найменших привілеїв AWS демонструє високий рівень автоматизації, завдяки реалізації патерна Role Vending Machine (RVM), який забезпечує контрольований процес призначення тимчасових ролей з обмеженими правами доступу в хмарній інфраструктурі. Основною перевагою цього підходу є зменшення ризику надмірного доступу або помилок прав, а також спрощення аудиту через стандартні інструменти (IAM Access Analyzer, Checkov). IAM (Identity and Access Management) є системою керування ідентифікацією та доступом у хмарній платформі. Важливо розуміти і зважати на виклики, пов’язані зі складністю підтримки шаблонів і рев’ю-процесів при великій кількості проєктів. Azure натомість пропонує Privileged Identity Management (PIM) як сервіс із вбудованою тимчасовою ескалацією прав доступу, що дозволяє обмежити постійне використання адміністративних ролей. Тим самим Azure через PIM надає гнучке управління привілейованими правами, дозволяє активувати їх лише на час, необхідний для адміністративних дій. Це корисно, коли дослідницька команда має змінні ролі, і потрібно обмежити загрозу постійного адміністративного доступу. Але недоліком є те, що PIM залежить від правильного налаштування політик та умов активації; якщо активація занадто проста або не перевіряється, роль може бути зловживана. GCP реалізує подібну концепцію через кастомізовані IAM-політики з можливістю тимчасового надання доступу, часто інтегровані з TTL-механізмами, наприклад через Cloud Scheduler або сторонні інтеграції. Це дозволяє зберігати автономію обчислювальних середовищ, сприяє повторюваності експериментів, бо всі команди працюють в узгоджених мережевих умовах. Проте виклик полягає в тому, що не всі сервіси GCP повністю підтримують Shared VPC або мають обмеження щодо сервісних агентів, підмереж або прав, що можуть бути делегованими. Також, при

масштабуванні мережевих правил і firewall конфігурацій витрати на адміністрування зростають, і ризик помилкових правил або надмірних дозволів може збільшуватися.

У сфері логування та аудиту всі три платформи надають інструменти для детального моніторингу змін і дій користувачів, але з відмінностями в інтеграції та гнучкості. AWS поєднує IAM Access Analyzer з CloudTrail для детекції надмірних прав та історії доступу. Azure забезпечує аудит через Activity Logs і PIM Audit, з фокусом на привілейовані дії. GCP пропонує Cloud Audit Logs разом із VPC Flow Logs, що робить акцент як на дії користувачів, так і на мережевих з'єднаннях.

Ізоляція середовищ реалізується через архітектурні механізми розмежування: AWS використовує окремі облікові записи та VPC, Azure – підписки й групи ресурсів, тоді як GCP робить ставку на поєднання окремих проєктів із Shared VPC для централізованого управління мережею. Кожен із підходів має свої переваги в масштабованості та адмініструванні – зокрема, підхід GCP дозволяє зменшити дублювання мережевої конфігурації без втрати ізоляції.

Щодо управління змінами, то всі три платформи підтримують інфраструктурний підхід з використанням Terraform, однак мають різні точки інтеграції з CI/CD. AWS застосовує GitHub Actions у поєднанні з інструментами статичного аналізу (наприклад, Checkov), що дозволяє виявляти потенційні проблеми в конфігураціях ще до їхнього застосування. Azure традиційно використовує ARM-шаблони та GitOps-підходи з прив'язкою до Azure DevOps. GCP орієнтується на поєднання Terraform з Cloud Build, що забезпечує нативну інтеграцію з іншими сервісами GCP у пайплайні.

У контексті моделювання загроз кожна платформа реалізує власні механізми попередження ризиків: AWS враховує політику розміщення ресурсів у хостах (placement), мережеві ACL і фізичну ізоляцію в EC2; Azure фокусується на використанні WAF, мережевих груп безпеки (NSG) і сегментації, а GCP надає розширені можливості аналізу розміщення процесів на спільних вузлах (особливо для TPU) і підтримку зонованої ізоляції. Це набуває особливої важливості у дослідницьких сценаріях, де є ризик атак побічними каналами в мульти-тенантних середовищах.

У частині регуляторної відповідності всі три платформи підтримують вимоги до захисту даних як у спокої, так і в русі. AWS забезпечує контроль регіонів зберігання та гнучке управління життєвим циклом даних у поєднанні з KMS. Azure надає механізми шифрування та підтримку збереження даних у зонах, що відповідають вимогам GDPR. GCP дозволяє реалізувати політику data residency та надає інструменти на кшталт DLP API для захисту конфіденційної інформації.

Економічна ефективність управління інфраструктурою також варіюється залежно від платформи. AWS пропонує автозупинку EC2 і використання spot-інстансів для зниження витрат. Azure реалізує моніторинг бюджету та аналітику витрат як частину Azure Cost Management. GCP, своєю чергою, оптимізує витрати за рахунок preemptible VM і планування розміщення ресурсів у найвигідніших регіонах.

Тим самим аналіз показує, що кожна з хмарних платформ має власні сильні сторони та підходи до вирішення спільних викликів безпеки, що формують ґрунт для застосування Terraform у мультихмарному науковому середовищі. Обґрунтований вибір залежить від вимог до ізоляції, регуляторної відповідності, масштабу експериментів і операційної стійкості.

Порівняння продуктивності та безпеки при реалізації основних безпекових патернів у хмарних середовищах AWS, Azure та GCP. Для проведення порівняльного аналізу було вибрано експериментальний підхід із побудовою типових сценаріїв розгортання інфраструктури у кожному з трьох хмарних середовищ – AWS, Azure та GCP – із використанням однакової логіки розгортання за допомогою Terraform. Усі експерименти виконувались із метою верифікації функціональності, вимірювання операційної продуктивності, а також оцінки рівня безпеки, реалізованого через відповідні нативні сервіси.

Застосування уніфікованих сценаріїв і автоматизованих засобів розгортання забезпечує наукову відтворюваність експериментів, оскільки створює можливість їх повторення в аналогічних умовах іншими дослідниками.

Кожна реалізація мала такі базові компоненти: мережеву інфраструктуру з ізоляцією середовищ; обчислювальні ресурси (2 віртуальні машини) для запуску навантаження; політики доступу з реалізацією найменших привілеїв; логування та моніторинг змін; пайплайни змін (CI/CD); активацію тимчасових привілеїв адміністратора. Розгортання інфраструктури проводилось через модульну архітектуру Terraform, з використанням однакових патернів розгортання (адаптованих під кожного провайдера) із відкритих репозиторіїв GitHub та офіційних модулів з Terraform Registry. Наприклад, модуль terraform-aws-vpc для AWS, Azure Network Module для Azure та terraform-google-network для GCP.

Для емпіричної оцінки продуктивності було застосовано таку стратегію:

– час повного розгортання інфраструктури (команда terraform apply) вимірювався для кожної платформи з однаковою конфігурацією (2 віртуальні машини, окрема мережа, IAM/role/policy); в середньому: AWS ~95 с, Azure ~123 с, GCP ~102 с;

– продуктивність середовища виконання вимірювалась при запусканні однакових бенчмарків CPU/GPU (Sysbench, MLPerf inference) на екземплярах із однаковими характеристиками (4 vCPU, 16 ГБ RAM, Linux Ubuntu 22.04 LTS). У CPU-бенчмарках AWS c5.large та GCP n2-standard-4 показали близьку продуктивність (~26k events/sec), тоді як Azure D4s_v3 відставала (~23k events/sec). У GPU-тестах AWS p3 і GCP A100 продемонстрували схожу продуктивність у TensorFlow-бенчмарках, проте GCP показала меншу латентність за рахунок TPU;

– для латентності політик доступу вимірювалась затримка між запитом тимчасових прав (через RVM, PIM, або IAM TTL) та фактичним набуттям прав доступу: AWS (RVM + OIDC) ~15-20 с, Azure (PIM) ~45-60 с, GCP (IAM TTL via Cloud Scheduler) ~25-30 с (рис. 2).

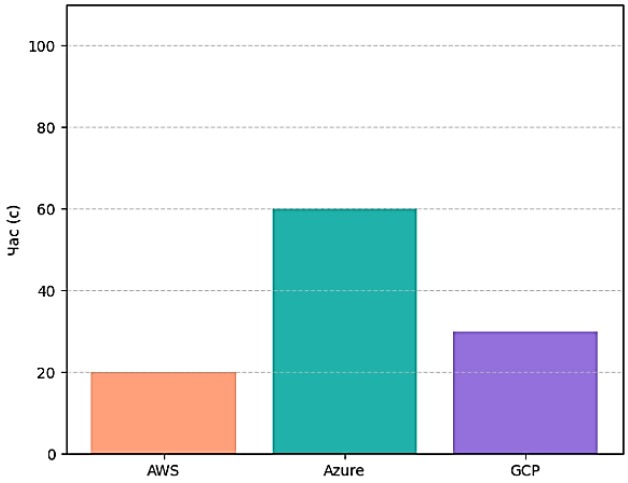

Рис. 2. Затримка надання тимчасових привілеїв доступу в хмарних провайдерах

Наведена на рис. 2 діаграма демонструє середню латентність (затримку) між запитом тимчасових прав доступу (через механізми RVM, PIM або TTL-based IAM) та моментом фактичного надання прав. Найшвидший час надання доступу продемонструвала AWS (приблизно 20 с) завдяки механізму Role Vending Machine з інтеграцією OIDC. Найбільшу затримку зафіксовано у Microsoft Azure (понад 60 с) через архітектурні особливості PIM, що може негативно впливати на гнучкість доступу в DevSecOps-сценаріях.

Для порівняння рівня безпеки було використано два підходи: статичний аналіз конфігурацій і перевірка поведінки системи у сценаріях відмови та атаки. Статичний аналіз здійснювався за допомогою Checkov та tfsec, зокрема для перевірки: відсутності

надмірних дозволів в IAM політиках, обов'язковості шифрування дисків і трафіка, увімкнення логування (CloudTrail / Audit Logs / Activity Logs). Другий підхід до оцінювання рівня безпеки хмарної інфраструктури стосувався емпіричного тестування поведінки системи у сценаріях відмови та потенційних атак [12]. На відміну від статичного аналізу конфігурацій, цей метод орієнтований на динамічну перевірку реакції середовища на порушення безпеки, що дозволяє виявити неочевидні вразливості, пов’язані з логікою доступу, обробкою помилок та збереженням аудиту [13]. Зокрема, було реалізовано низку контрольованих симуляцій, які моделюють типові загрози:

– спроба доступу до ресурсу без належних прав дозволяє перевірити ефективність механізмів автентифікації та авторизації, а також виявити можливі обхідні шляхи до критичних компонентів;

– зміна конфігурації мережі або параметрів міжмережевого екрану (firewall) імітує внутрішню або зовнішню атаку, спрямовану на порушення ізоляції середовища, що дає змогу оцінити стійкість системи до несанкціонованих змін;

– видалення або модифікація журналів аудиту (наприклад, CloudTrail, Audit Logs, Activity Logs) дозволяє визначити, чи здатна система виявити спроби приховати сліди активності, що критично важливо для забезпечення цілісності та трасування подій.

Застосування цього підходу дозволяє оцінити ефективність реалізованих патернів у реальних умовах та виявити потенційні прогалини, які не фіксуються засобами статичного аналізу. Він є важливим елементом комплексного аудиту безпеки хмарних рішень, особливо в контексті багатоплатформного середовища, де взаємодія компонентів може створювати складні сценарії ризику.

Після впровадження всіх патернів: в AWS блокування доступу спрацьовувало миттєво, логування відбувалось через CloudTrail, IAM Access Analyzer фіксував спроби зміни політик; в Azure аналогічну функцію виконував PIM Audit + Activity Log; в GCP – Cloud Audit Logs + Policy Analyzer. У результаті статичного аналізу найменше порушень виявлено в AWS (середній security score – 94/100), GCP – 91/100, Azure – 88/100 (через складніші ARM шаблони, які складно перевіряти автоматично) (рис. 3).

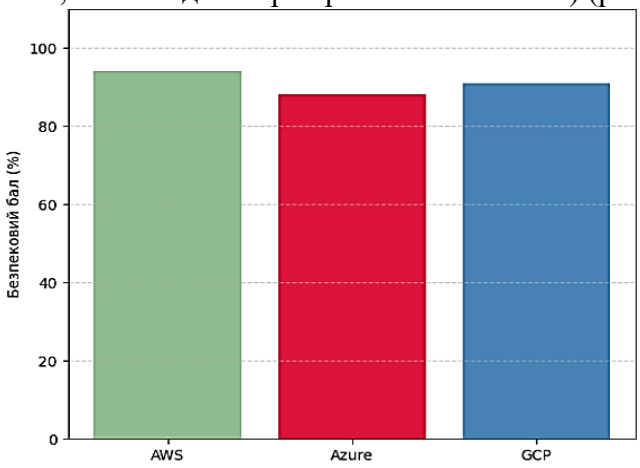

Рис. 3. Оцінка безпеки конфігурації хмарної інфраструктури

Статичний аналіз безпеки конфігурацій, розгорнутих у трьох провайдерах, із використанням інструментів Checkov та tfsec, охоплював політики доступу, наявність шифрування, активацію логування та відповідність рекомендаціям безпекових фреймворків (CIS Benchmarks). AWS продемонстрував найвищу оцінку (94%) завдяки чіткій структурі IAM та розвиненим сервісам моніторингу. Найменша оцінка у Azure (88%) зумовлена переважно складністю аналізу ARM-шаблонів та частковою відсутністю автоматизованого логування змін у політиках.

Результати та обговорення. Конкретні приклади налаштування інфраструктури за допомогою Terraform у трьох провідних хмарних платформах AWS, Azure та GCP продемонстрували різні підходи до безпеки, розмежування доступу, ізоляції середовищ

і централізованого управління мережею, зокрема через механізми RVM, PIM і Shared VPC відповідно. У випадку з AWS приклад реалізації Role Vending Machine довів можливість централізованого та автоматизованого надання прав доступу через опис ролей у вигляді коду, що в поєднанні з перевірками безпеки (IAM Access Analyzer, Checkov) забезпечує як гнучкість експериментального середовища, так і захист від неконтрольованого розширення прав. Аналіз Azure продемонстрував важливість привілейованого управління доступом у наукових системах із підвищеним рівнем відповідальності. Застосування Azure PIM у поєднанні з ARM-шаблонами забезпечило не лише ізоляцію середовищ, а й контроль над тимчасовими правами адміністративного рівня, що є ключовим для захисту життєвого циклу дослідження. У Google Cloud Platform приклад Shared VPC дав змогу реалізувати централізовану мережеву архітектуру з різними обчислювальними доменами, що дозволило виконувати порівняльні експерименти з мовними моделями в узгоджених умовах, забезпечуючи реплікованість результатів і контроль над витратами.

Поєднання різних патернів у мультихмарному середовищі дає змогу досягти високої безпеки, ізоляції, відтворюваності та контролю над витратами, якщо реалізовано належним чином. Водночас саме грамотне налаштування, з урахуванням принципів безпеки, автоматизації та відтворюваності, перетворює хмару на повноцінний науковий інструмент. Нехтування цими принципами може призвести до втрати даних, порушення етичних норм обробки чутливої інформації та компрометації результатів. Тому хмарна інфраструктура потребує не лише технічного опанування, а й методологічного осмислення як частини наукової етики та практики.

Розглянуті безпекові патерни, зокрема принцип найменших привілеїв, ізоляція середовищ, централізоване логування, управління ролями та моделювання загроз, мають критичне значення не лише для забезпечення безпеки хмарної інфраструктури, а й для гарантування відтворюваності наукових досліджень. По-перше, ізоляція середовищ через окремі акаунти, підписки або проєкти дозволяє уникнути небажаного перехресного впливу між експериментами, забезпечує стабільність конфігурацій та чистоту результатів. Це особливо важливо для задач, які потребують високої точності, як-от: моделювання, машинне навчання, обробка чутливих даних. По-друге, управління доступом через тимчасові ролі (RVM, PIM, TTL) забезпечує контрольоване середовище виконання, де кожна дія має чітко визначені права та часові межі. Це сприяє трасуванню експериментальних дій, зменшує ризик втручання сторонніх компонентів і дозволяє точно відтворити умови дослідження. По-третє, логування та аудит (CloudTrail, Audit Logs, Access Analyzer) створюють повну історію змін і доступу до ресурсів, що є основою для верифікації результатів та аналізу впливу інфраструктурних факторів на перебіг експерименту. По-четверте, автоматизація через IaC та CI/CD дозволяє зберігати конфігурації в системах контролю версій, забезпечуючи повторюваність середовищ та можливість точного відтворення інфраструктури в будь-який момент часу.

Тим самим безпекові патерни, інтегровані в хмарну архітектуру, не лише знижують ризики, а й створюють технічно контрольоване, стабільне, прозоре середовище, яке відповідає вимогам наукової достовірності та відтворюваності.

Висновки. Проведене дослідження підтвердило, що хмарні платформи AWS, Azure та GCP забезпечують широкий спектр механізмів для реалізації безпекових патернів, необхідних у наукових дослідницьких середовищах. Водночас їхня ефективність залежить від правильного налаштування ролей, ізоляції середовищ та інтеграції з інструментами автоматизації. Порівняльний аналіз показав, що принцип найменших привілеїв, централізоване логування та моделювання загроз є критично важливими для запобігання витоку прав, прихованих точок доступу та перехресного впливу між середовищами. Відсутність цих механізмів створює ризики компрометації даних і порушення регуляторних норм. Використання IaC у поєднанні з CI/CD або GitOps-підходами забезпечує повторюваність, контрольованість і прозорість конфігурацій, що є

необхідною умовою для достовірності наукових експериментів. Практичні кейси підтвердили, що автоматизоване управління ролями (RVM в AWS, PIM в Azure, TTL-доступ у GCP) дозволяє зменшити ризики надмірного доступу та підвищити гнучкість адміністрування в мультихмарному середовищі.

Результати дослідження можуть бути використані для побудови безпечної, масштабованої та етично відповідної хмарної інфраструктури, що функціонує як повноцінний науковий інструмент, а не лише технічне середовище. Це відкриває перспективи для подальшої стандартизації хмарних архітектур у сфері комп'ютерних наук. Перспективи подальших досліджень лежать у площині розширення аналізу на інші хмарні платформи та гібридні моделі, а також у розробці освітніх модулів з DevSecOps для академічного середовища.

Список літератури

1. Dasari H. Infrastructure as Code (IaC) Best Practices for Multi-Cloud Deployments in Enterprises. *International journal of networks and security*. 2025. Vol. 05, Issue 01. P. 174-186. DOI: https://doi.org/10.55640/ijns-05-01-10.
2. Johnson O., Olamijuwon J., Cadet E., Osundare O., Samira Z. Designing multi-cloud architecture models for enterprise scalability and cost reduction. *Open Access Research Journal of Engineering and Technology*. 2024. Vol. 07(02). P. 101-113. DOI: https://doi.org/10.53022/oarjet.2024.7.2.0061
3. Thota R. Ch. Cloud-Native DevSecOps: Integrating Security Automation into CI/CD Pipelines. *International Journal of Innovative Research and Creative Technology*. 2024. Vol. 10. P. 1-19. DOI: https://doi.org/10.5281/zenodo.15036934
4. Трофименко О.Г., Дика А.І., Лобода Ю.Г. Аналіз інструментів тестування вебзастосунків. *Кібербезпека: освіта, наука, техніка.* 2023. № 4(20). С. 62-71. DOI: https://doi.org/10.28925/2663-4023.2023.20.6271.
5. Vidyasagar V. Multi-Cloud DevOps Automation for An Empirical Study on IaC, CI/CD, and Kubernetes Orchestration. *International Journal of Innovative Research in Education.* 2020. Vol. 01(01). P. 605-617. DOI: https://doi.org/10.5281/zenodo.15556322.
6. Bhat K., Prashanth K. Multicloud Orchestration using Terraform. *International Journal for Research in Applied Science and Engineering Technology.* 2022. DOI: https://doi.org/10.22214/ijraset.2022.44760.
7. Mulpuri G. Infrastructure as Code (IaC): Best Practices of Implementing IaC, Especially in Automating Infrastructure Provisioning and Management Using Terraform. *European Journal of Advances in Engineering and Technology*. 2023. Vol. 10(4). P. 56-62. URL: https://zenodo.org/records/11078219
8. Verdet A., Hamdaqa M., Da Silva L., Khomh F. Exploring Security Practices in Infrastructure as Code: An Empirical Study. *arXiv*. 2023. URL: https://arxiv.org/abs/2308.03952
9. Sonawane Y. Infrastructure as Code (IaC): Why It's a Game Changer in DevOps. URL: https://dev.to/yash_sonawane25/infrastructure-as-code-iac-why-its-a-game-changer-in-devops-1e1p
10. Michalowski M. Best Practices for Using Terraform: An In-depth Guide. *IEEE Computer Society*. 2024. URL: https://www.computer.org/publications/tech-news/trends/terraform-guide
11. Fuchs M. D. Policy as Code, Policy as Type. *arXiv*. 2025. DOI: https://doi.org/10.48550/arXiv.2506.01446
12. Трофименко О.Г., Пастернак Ю.Ю., Манаков С.Ю., Лобода Ю.Г. Автоматизація тестування вебсайтів електронної комерції. *Сучасна спеціальна техніка*. 2021. № 2(65). С. 46-59. DOI: https://doi.org/10.36486/mst2411–3816.2021.2(65).5.
13. Трофименко О.Г., Дика А.І., Лобода Ю.Г. Аналіз уразливостей та проблем безпеки вебзастосунків. *Системні технології.* 2023. № 3(146). С. 25-37. DOI: https://doi.org/10.34185/1562-9945-3-146-2023-03.

COMPARATIVE ANALYSIS OF SECURITY PATTERNS OF CLOUD PLATFORMS IN THE CONTEXT OF SCIENTIFIC REPRODUCABILITY

O. G. Trofymenko[1], P. O. Chykunov[1], D. Yu. Astakhov[1],
Yu. V. Molokanov, T. A. Farionova[2]

[1]National University "Odesa Law Academy"
23, Fontans'ka doroga st., Odesa, 65009, Ukraine
[2]Admiral Makarov National University of Shipbuilding
9, Geroiv Ukrainy Ave, Mykolaiv, 54007, Ukraine
Emails: trofymenko@onua.edu.ua, pavel@onua.edu.ua, astakhovnil@gmail.com,
molokanov9@gmail.com, tetyana.farionova@nuos.edu.ua

The study is devoted to a systematic analysis of the implementation of key security patterns in the cloud platforms Amazon Web Services, Microsoft Azure and Google Cloud Platform, considering their suitability for scientific research in the field of computer science. The relevance of the study is due to the need to ensure security, isolation, automation and reproducibility of the infrastructure in a multi-cloud environment. The purpose of the study is to systematize and comparatively analyze key security patterns in the cloud platforms AWS, Azure and GCP, considering their effectiveness, limitations and suitability for ensuring reproducibility, isolation and ethical compliance in scientific research environments. The research methodology is based on a systematic approach, which includes a comparison of cloud platform configurations based on practical cases, the use of static analysis tools (Checkov, tfsec) to assess the security of IaC code, modeling system behavior in failure scenarios and attacks to empirically verify the stability of the infrastructure. The article considers three practical cases: 1) for AWS – implementation of Role Vending Machine for automated assignment of temporary roles with limited rights, integration with GitHub Actions, Checkov and IAM Access Analyzer; 2) for Azure – multi-zone architecture with PIM, GitOps, ARM templates and network segmentation for fault tolerance modeling; 3) for GCP – use of Shared VPC, isolated projects, TPU and Vertex AI Experiments for NLP and deep learning tasks. Comparative analysis showed: the principle of least privilege significantly reduces the risks of rights leakage; the lack of logging and auditing creates "dead" rights that can be entry points for attacks; isolation of environments (through separate accounts, subscriptions, projects) is critical to prevent cross-influence; change management through CI/CD or GitOps provides repeatability and control; threat modeling allows you to identify vulnerabilities associated with resource placement; regulatory requirements (GDPR, encryption, geofencing) should be taken into account at the architectural planning stage. The research conducted allows us to conclude that a cloud infrastructure configured considering the principles of security, automation and ethical compliance can become a full-fledged scientific tool, and not just a technical environment. At the same time, neglecting these principles creates risks of data compromise, violation of regulatory norms and loss of reliability of results. The scientific novelty of the work lies in the comparative analysis of security models of three cloud platforms in the context of ensuring the reliability and ethical compliance of scientific experiments.

The practical significance of the results lies in the possibility of their use for building a secure, scalable and reproducible cloud infrastructure in scientific projects, in the professional activities of DevOps teams working with multi-cloud environments, as a basis for developing ethical and regulatory-compliant architectures in the field of computer science.

Keywords: cloud computing, multi-cloud environment, Infrastructure as Code, DevOps, DevSecOps, security patterns, CI/CD, GitOps, cloud architecture.