Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Коробко Романа Дмитровича

на здобуття ступеня вищої освіти Бакалавра

Система керування інформаційною безпекою для забезпечення безперервності бізнес-процесів підприємства

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітня програма Кібербезпека

Шифр КРБКБ.220110.22.01.07 ПЗ

Виконав студент 4 курсу група КБ-22-1 Роман КОРОБКО
Керівник доктор техн. наук, професор Михайло КАСЯНЧУК
Нормоконтролер д-р філософії Наталія ПЕТЛЯК

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

12.06 2026 р.

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

09 січня 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Коробко Роману Дмитровичу

1 Тема роботи Система керування інформаційною безпекою для забезпечення безперервності бізнес-процесів підприємства

sКерівник роботи канд. техн. наук, доцент, Касянчук Михайло Миколайович

Затверджено наказом ректора університету від 8 січня 2026 № 7

2 Строк подання студентом кваліфікаційної роботи на кафедру 27 травня 2026р.

3 Вихідні дані до роботи Спроєктувати та реалізувати розподілену відмовостійку платформу для забезпечення безперервності бізнес-процесів підприємства. Проаналізувати поточну AWS-інфраструктуру підприємства. Реалізувати мережевий рівень рівень оркестрації рівень даних, систему моніторингу (Prometheus, Grafana, Loki) та виявлення інцидентів безпеки.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз предметної області та постановка задачі персональної платформи як сервісу (PaaS). Побудова системи керування інформаційною безпекою для забезпечення безперервності бізнес-процесів. Оцінка ефективності системи керування інформаційною безпекою.

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

«Архітектура поточної AWS-інфраструктури підприємства (AS-IS)», «Цільова архітектура розподіленої платформи Iron Mesh PaaS (TO-BE)», «Архітектура системи моніторингу та виявлення інцидентів безпеки».

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7 Дата видачі завдання 12 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	Січень-Лютий	
Аналіз сучасних тенденцій IT-інфраструктур та хмарних сервісів	Лютий	
Дослідження поточного стану AWS-інфраструктури підприємства	Лютий	
Постановка задачі та формування вимог до системи	Березень	
Проєктування архітектури платформи Iron Mesh PaaS	Березень	
Реалізація мережевого, оркестраційного та даних рівнів	Квітень	
Розгортання моніторингу, SIEM та системи резервного копіювання	Квітень	
Тестування відмовостійкості та оцінка ефективності системи	Травень	
Оформлення пояснювальної записки згідно вимог	Травень	
Оформлення графічної частини	Травень	
Захист КР	Червень	

Студент _____ Роман КОРОБКО

Керівник кваліфікаційної роботи _____ Михайло КАСЯНЧУК

АНОТАЦІЯ

Тема кваліфікаційної роботи: Система керування інформаційною безпекою для забезпечення безперервності бізнес-процесів підприємства.

Автор роботи: Коробко Роман Дмитрович

Керівник роботи: Касянчук Михайло Миколайович

Загальний обсяг роботи: 77 сторінок, 18 рисунків, 21 таблиця, 47 посилань.

Графічна частина: 3 плакати.

Ключові слова: інформаційна безпека, безперервність бізнес-процесів, мульти-провайдерна архітектура, відмовостійкість, HashiCorp Nomad, WireGuard, MariaDB Galera, Wazuh SIEM, Prometheus, моніторинг.

Кваліфікаційна робота бакалавра присвячена проєктуванню та реалізації системи керування інформаційною безпекою для забезпечення безперервності бізнес-процесів підприємства. У роботі проаналізовано системні ризики централізованої хмарної інфраструктури на прикладі AWS EC2, виявлено сім критичних недоліків та обґрунтовано перехід до розподіленої мульти-провайдерної архітектури. Спроєктовано та реалізовано платформу Iron Mesh PaaS на базі кластера з 12 вузлів, розподілених між трьома географічно незалежними майданчиками та об'єднаних через шифровану mesh-мережу WireGuard.

У результаті роботи реалізовано комплексну систему керування інформаційною безпекою, яка поєднує оркестрацію контейнерів (HashiCorp Nomad), управління секретами (Vault), синхронну реплікацію бази даних (MariaDB Galera), централізований моніторинг (Prometheus, Grafana, Loki) та SIEM (Wazuh). Тестування підтвердило: uptime ≥ 99.95% (проти 99.0% попередньої системи), RPO = 0 для критичних даних, автоматичний failover за < 5 хвилин, зниження сукупної вартості володіння на 25–35%.

25.05.2026

ANNOTATION

Theme of qualification work: Information security management system for ensuring business continuity of enterprise processes.

Author of the work: Korobko Roman Dmytrovych.

Mentor: Kasyanchuk Mykhailo Mykolayovych.

Total volume of work: 77 pages, 18 figures, 21 tables, 47 references.

Graphic part: 3 posters.

Keywords: information security, business continuity, multi-cloud architecture, fault tolerance, HashiCorp Nomad, WireGuard, MariaDB Galera, Wazuh SIEM, Prometheus, monitoring.

The bachelor's thesis is devoted to the design and implementation of an information security management system that ensures business continuity of enterprise processes. The work analyses systemic risks of centralised cloud infrastructure based on the AWS EC2 case study, identifies seven critical deficiencies, and justifies the transition to a distributed multi-provider architecture. The Iron Mesh PaaS platform is designed and implemented on a 12-node cluster distributed across three geographically independent sites and connected via an encrypted WireGuard mesh network.

As a result of the work, a comprehensive information security management system has been implemented that combines container orchestration (HashiCorp Nomad), secret management (Vault), synchronous database replication (MariaDB Galera), centralised monitoring (Prometheus, Grafana, Loki), and SIEM (Wazuh). Testing confirmed: uptime $\geq 99.95\%$ (vs. 99.0% of the previous system), RPO = 0 for critical data, automatic failover within < 5 minutes, and a 25-35% reduction in total cost of ownership.

25.05.2026

ЗМІСТ

ВСТУП

Актуальність теми. Забезпечення безперервності бізнес-процесів є одним із пріоритетних завдань сучасних підприємств, діяльність яких критично залежить від IT-інфраструктури. Згідно зі звітом Uptime Institute (2024), 55% організацій зазнали незапланованих простоїв протягом останніх трьох років, а середня вартість однієї години простою для компаній малого та середнього бізнесу складає $137–427. Водночас зростає частота та складність кіберзагроз: за даними IBM X-Force Threat Intelligence Index (2024), середній час перебування зловмисника у скомпрометованій системі до виявлення складає 204 дні для організацій без централізованого моніторингу безпеки (SIEM), і лише 73 дні - з ним.

Масова міграція підприємств у публічні хмарні платформи (AWS, Azure, GCP) вирішила проблему фізичного обслуговування серверів, але створила нові ризики: концентрація критичної інфраструктури в одного провайдера (vendor lock-in), непрозоре ціноутворення з непрогнозованими витратами на мережевий трафік, та залежність від доступності єдиного хмарного регіону. Масштабні збої AWS us-east-1 (грудень 2021), Azure Active Directory (вересень 2023) та Cloudflare (червень 2022) підтвердили, що навіть провідні провайдери не гарантують абсолютної доступності.

Альтернативним підходом є побудова self-hosted розподілених систем на базі орендованих віртуальних серверів від кількох незалежних провайдерів, об'єднаних у єдиний кластер через шифровану mesh-мережу. Такий підхід поєднує переваги хмарних технологій (автоматизація, оркестрація контейнерів, еластичність) з незалежністю від окремого постачальника послуг. Розвиток інструментів з відкритим кодом - HashiCorp Nomad/Consul/Vault, WireGuard, Prometheus, Wazuh - зробив побудову подібних систем доступною для малого та середнього бізнесу.

Мета роботи - підвищення рівня інформаційної безпеки та забезпечення безперервності бізнес-процесів підприємства шляхом проєктування та реалізації

розподіленої відмовостійкої платформи з централізованим управлінням секретами, моніторингом та виявленням інцидентів безпеки.

Об'єкт дослідження - процес забезпечення інформаційної безпеки та безперервності бізнес-процесів підприємства.

Предмет дослідження - система керування інформаційною безпекою на базі розподіленої мульти-провайдерної платформи з оркестрацією контейнерів та централізованим моніторингом.

Методи дослідження: системний аналіз (аналіз поточної інфраструктури та виявлення недоліків), порівняльний аналіз (оцінка технологій оркестрації, VPN-протоколів, баз даних), моделювання, експеримент (тестування failover, продуктивності, безпеки на production-кластері), техніко-економічний аналіз.

Практичне значення отриманих результатів полягає у побудові працюючої production-ready платформи Iron Mesh PaaS, що забезпечує: uptime ≥ 99.95% RPO = 0 для критичних даних (проти 24 годин), автоматичний failover за < 5 хвилин, проактивне виявлення загроз безпеки через Wazuh SIEM (раніше відсутнє), та зниження сукупної вартості володіння на 25–35%. Платформа може бути адаптована для інших підприємств малого та середнього бізнесу, що потребують відмовостійкої інфраструктури без залежності від єдиного хмарного провайдера.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку джерел посилань та додатків. У першому розділі проведено аналіз предметної області та поточної інфраструктури підприємства, сформульовано постановку задачі. У другому розділі описано побудову розподіленої платформи: мережевий рівень, оркестрація, рівень даних, Edge/Ingress, моніторинг, SIEM, резервне копіювання та CI/CD. У третьому розділі виконано оцінку ефективності: порівняння з попередньою системою, результати тестування, фінансовий аналіз та рекомендації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ПЕРСОНАЛЬНОЇ ПЛАТФОРМИ ЯК СЕРВІСУ (PAAS)

1.1 Аналіз сучасних тенденцій побудови IT-інфраструктур та хмарних сервісів

Еволюція IT-інфраструктур за останні десять років пройшла шлях від фізичного володіння обладнанням до повної абстракції ресурсів у вигляді програмного коду (Infrastructure as Code, IaC). Цей перехід супроводжувався появою нових парадигм управління обчислювальними ресурсами - від монолітних фізичних серверів через гіпервізорну віртуалізацію до сучасних контейнерних платформ і розподілених mesh-кластерів. Для глибокого розуміння предметної області та обґрунтування архітектурних рішень даної роботи необхідно розглянути три ключові технологічні парадигми, що послідовно змінювали одна одну, а також проаналізувати критичні недоліки кожної з них [1, 2].

1.1.1 Традиційна модель On-Premises та її технічні обмеження

Протягом 2000-х і початку 2010-х років стандартом корпоративної IT-інфраструктури була модель On-Premises: організація купувала або орендувала фізичні сервери (Bare Metal), розміщуючи їх у власних серверних кімнатах або в орендованих стійках у комерційних ЦОД (модель Colocation). Управління такою інфраструктурою вимагало утримання власного штату системних адміністраторів, витрат на ліцензування програмного забезпечення та регулярного оновлення апаратного забезпечення за власний рахунок [3].

З технічної точки зору, головною проблемою цієї моделі є критично низька утилізація ресурсів. Дослідження McKinsey Global Institute (2022) та звіти Uptime Institute показують, що середньостатистичний корпоративний сервер використовує лише 5-15% своєї обчислювальної потужності протягом робочого часу. Причина полягає в тому, що сервери розгортаються з урахуванням пікового навантаження (наприклад, кінець кварталу або рекламні кампанії), яке може тривати лише кілька годин на місяць. Решта часу обладнання простоює, споживаючи при цьому від 60 до 80% своєї максимальної потужності за рахунком

за електроенергію - навіть у режимі idle сучасний двопроцесорний сервер споживає 150-300 Вт [1].

Другою критичною проблемою є архітектурна наявність єдиної точки відмови (Single Point of Failure, SPOF). Типова on-premises інсталяція малого та середнього бізнесу обмежена одним підключенням до Інтернет-провайдера, однією системою безперебійного живлення (UPS), одним комутатором ядра мережі. Вихід з ладу будь-якого з цих компонентів призводить до повної недоступності сервісів. При цьому відновлення після апаратного збою (Mean Time To Repair, MTTR) для on-premises середовища становить у середньому 4-8 годин, оскільки включає фізичну заміну компонентів, доставку запасних частин і ручне відновлення програмного забезпечення [4].

Третє обмеження - негнучкість масштабування (так звана проблема Capacity Planning). Якщо бізнес-процес потребує раптового збільшення обчислювальних ресурсів, цикл від замовлення сервера до його введення в експлуатацію займає від 4 до 12 тижнів з урахуванням закупівлі, доставки, монтажу, налаштування та тестування. За цей час бізнес-можливість може бути втрачена.[1]

1.1.2 Ера публічних хмарних гіперскейлерів: можливості та системні ризики

Поява Amazon Web Services (2006), Microsoft Azure (2010) та Google Cloud Platform (2011) радикально змінила парадигму споживання IT-ресурсів. Модель Pay-as-you-go дозволила бізнесу орендувати обчислювальні потужності за хвилини замість тижнів, а еластичне масштабування (Auto Scaling) вирішило проблему пікового навантаження без надмірного provisioning. Станом на 2024 рік, за даними аналітичної компанії Synergy Research Group, AWS, Azure та GCP разом контролюють понад 65% світового ринку хмарних послуг із загальним річним доходом, що перевищує 200 млрд доларів США.[5]

Проте домінування гіперскейлерів породило нові системні ризики, які стали критично важливими для забезпечення безперервності бізнесу. Перший і найбільш значущий - це Vendor Lock-in (технологічна залежність від вендора). Сучасні хмарні провайдери свідомо будують екосистеми власних пропрієтарних сервісів: AWS пропонує Lambda для serverless-обчислень, DynamoDB як NoSQL-

базу, SQS як чергу повідомлень, S3 як об'єктне сховище. Технічна інтеграція цих сервісів між собою є безшовною і значно прискорює розробку, але водночас формує глибоку залежність: архітектура, побудована навколо AWS Lambda та DynamoDB, потребує повного переосмислення та переписування при міграції на Azure або GCP. За оцінкою Gartner (2023), вартість міграції між хмарними провайдерами для підприємства середнього розміру може складати від 500 тисяч до 2 мільйонів доларів США [6].

Другий суттєвий ризик - непрозоре та нелінійне ціноутворення. На відміну від фіксованих витрат on-premises, вартість хмарних послуг складається з десятків компонентів: оплата за обчислювальні ресурси (vCPU-hours), оперативну пам'ять, мережеву передачу (Egress traffic), операції введення/виведення (I/O operations), сховище, IP-адреси тощо. Egress traffic - тобто трафік, що виходить із хмари назовні - тарифікується окремо і може стати несподіваним джерелом значних витрат. Наприклад, AWS стягує від 0.09 до 0.05 USD за гігабайт вихідного трафіку залежно від регіону та обсягу. Для відеостримінгового або SaaS-сервісу з великим обсягом вихідних даних ці витрати можуть у рази перевищувати вартість самих обчислень.

Третій і найнебезпечніший ризик - концентрація критичної інфраструктури в руках одного провайдера призводить до того, що будь-який великий збій зачіпає тисячі непов'язаних між собою бізнесів одночасно. Цей системний ризик реалізувався неодноразово протягом 2021-2023 років. У грудні 2021 року масштабний збій в регіоні AWS US-EAST-1 (Північна Вірджинія) тривав понад 7 годин і призвів до недоступності Netflix, Disney+, Slack, Ring, Coinbase та тисяч інших сервісів. Причиною стала помилка в системі управління мережевими пристроями, яка поширилась каскадно на залежні сервіси. У березні 2021 року пожежа у ЦОД OVHcloud у Страсбурзі (Франція) знищила один дата-центр повністю та пошкодила другий, що призвело до повної та безповоротної втрати даних для клієнтів, які не мали резервних копій за межами площадки. За підрахунками, постраждало понад 3,6 мільйона сайтів. Ці інциденти наочно демонструють, що навіть найбільші та найнадійніші провайдери не є

застрахованими від катастрофічних збоїв.[7]

Таблиця 1.1 - Найбільш значущі збої хмарних провайдерів (2021-2023)

	Провайдер / Регіон	Тривалість	Наслідки для бізнесу	Першопричина
Груд. 2021	AWS US-EAST-1 (Вірджинія)	~7 год	Netflix, Slack, Coinbase, Disney+ - повна недоступність	Каскадний збій мережевого рівня (AWS Kinesis → IAM → EC2)
Берез. 2021	OVHcloud SBG2 (Страсбург)	~72 год та більше	3,6 млн сайтів офлайн, безповоротна втрата даних	Пожежа в ЦОД, відсутність автоматичного пожежогасіння
Черв. 2022	Cloudflare (глобально)	~57 хв	19 регіонів недоступні, ~50% глобального трафіку	Помилка конфігурації BGP при плановому оновленні
Січ. 2023	Microsoft Azure (глобально)	~5 год	Teams, Outlook, Azure Portal - недоступні	Збій WAN-маршрутизатора при оновленні мережевої конфігурації
Лип. 2024	CrowdStrike / Azure (глобально)	~19 год	8,5 млн Windows-пристроїв, авіалінії, банки, лікарні	Пошкоджений файл оновлення драйвера ядра (BSOD loop)

Кожен із наведених інцидентів є прямим наслідком централізованої архітектури: надмірна концентрація залежностей в єдиному вузлі (провайдері, регіоні, програмному компоненті) неминуче перетворює локальний збій на глобальну катастрофу для всієї екосистеми залежних систем.

1.1.3 Контейнеризація та оркестрація: технічне підґрунтя сучасних розподілених систем

Паралельно з розвитком хмарних провайдерів відбувалась революція у способі пакування та запуску програмного забезпечення. Технологія

контейнеризації, яку популяризував Docker (2013), вирішила одну з найболючіших проблем розробки - невідтворюваність середовища виконання. Контейнер - це ізольований процес операційної системи Linux, що використовує механізми ядра cgroups (контроль ресурсів) та namespaces (ізоляція файлової системи, мережі, PID) для створення власного ізольованого середовища без накладних витрат на повну емуляцію апаратного забезпечення, як це роблять класичні гіпервізори (VMware, KVM) [8, 9].

На відміну від віртуальної машини, яка включає повне ядро ОС і займає від 1 до 10 гігабайт дискового простору, Docker-образ складається лише з шарів файлової системи, необхідних для конкретного додатку, і може займати від кількох мегабайт до сотень мегабайт. Час запуску контейнера - від мілісекунд до секунд, тоді як запуск ВМ займає хвилини. Це забезпечує значно вищу щільність розміщення (density) і дозволяє запускати десятки ізольованих сервісів на одному фізичному або віртуальному вузлі, що безпосередньо підвищує утилізацію ресурсів.[10]

Проте масштабне розгортання контейнерів породжує нову задачу: управління сотнями і тисячами контейнерів, що динамічно створюються, зупиняються та переміщуються між вузлами кластера. Для вирішення цієї задачі виникли системи оркестрації контейнерів. Kubernetes (K8s), розроблений Google на основі внутрішнього проекту Borg і переданий у відкритий код у 2014 році, став де-факто стандартом оркестрації в корпоративному середовищі. Kubernetes вирішує задачі планування (scheduling) - розміщення контейнерів на вузлах з урахуванням доступних ресурсів і обмежень; самовідновлення (self-healing) - автоматичного перезапуску контейнерів при збоях; горизонтального масштабування (HPA) - автоматичного збільшення кількості реплік сервісу при зростанні навантаження [11].

Однак Kubernetes розроблявся для великих інфраструктур і несе значний операційний overhead. Мінімальна продуктивна конфігурація K8s вимагає розгортання etcd-кластера (3 вузли), kube-apiserver, kube-scheduler, kube-controller-manager, kubelet на кожному робочому вузлі, а також мережевого плагіна CNI

(Calico, Flannel, Cilium). Сукупне споживання ресурсів control plane K8s для малого кластера (3-5 вузлів) складає 2-4 vCPU та 4-8 GiB RAM тільки на системні компоненти. Для малого та середнього бізнесу, де загальний бюджет на інфраструктуру складає €50-200 на місяць, це неприйнятно [12].

1.1.4 HashiCorp Stack як альтернатива: Nomad, Consul, Vault

Альтернативою Kubernetes для сценаріїв малого та середнього масштабу є HashiCorp Stack - набір інструментів з відкритим вихідним кодом (до зміни ліцензії у 2023 році), що вирішують ті самі задачі з суттєво меншим операційним навантаженням [13].

HashiCorp Nomad - це оркестратор задач (workload orchestrator), що підтримує не лише Docker-контейнери, а й ізольовані системні процеси (raw_exec, exec), Java-застосунки та навіть віртуальні машини через QEMU-драйвер. Nomad реалізує алгоритм bin packing для розміщення задач - він намагається максимально ущільнити задачі на мінімальній кількості вузлів, залишаючи решту вузлів вільними для аварійного розміщення при збоях. Це безпосередньо підвищує утилізацію ресурсів до 70-85% порівняно з 10-20% при виділенні окремого сервера на кожного клієнта. Критично важливою є вбудована підтримка Raft Consensus Algorithm для забезпечення узгодженості стану кластера при розбитті мережі (network partition) - мінімальна конфігурація для кворуму при відмові одного вузла складає 3 сервери [14].

HashiCorp Consul вирішує задачу Service Discovery - автоматичної реєстрації сервісів у кластері та надання актуальних адрес для міжсервісної взаємодії без жорсткого прив'язування до IP-адрес. У поєднанні з Nomad, кожен запущений сервіс автоматично реєструється у Consul з health check, і Nomad використовує Consul для маршрутизації трафіку між репліками. Consul Connect реалізує концепцію Service Mesh з mTLS-шифруванням між сервісами - кожен сервіс отримує z

HashiCorp Vault централізує управління секретами (паролями, ключами API, сертифікатами TLS, токенами). На відміну від зберігання секретів у змінних оточення або конфігураційних файлах (що є поширеною практикою і критичним

порушенням безпеки), Vault динамічно генерує та видає короткотривалі credentials за запитом сервісу, автоматично відкликаючи їх після закінчення терміну дії. Vault підтримує численні бекенди зберігання, але при роботі в кластері використовує власний Raft-бекенд для синхронізації між вузлами без зовнішніх залежностей [16].

Таблиця 1.2 - Порівняльний аналіз систем оркестрації для self-hosted PaaS

Критерій	Kubernetes (K8s)	Docker Swarm	HashiCorp Nomad
Мін. ресурси control plane	2-4 vCPU / 4-8 GB RAM	~0.5 vCPU / 512 MB RAM	~0.5 vCPU / 256 MB RAM
Підтримка не-Docker workloads	Лише через CRI-інтерфейс	Ні	Так (exec, Java, QEMU, raw_exec)
Вбудований Service Discovery	CoreDNS (базовий)	Вбудований DNS	Consul (повноцінний SD + health)
Управління секретами	Kubernetes Secrets (base64)	Docker Secrets	HashiCorp Vault (dynamic secrets)
Складність початкового налаштування	Висока (10-20+ годин)	Низька (1-2 години)	Середня (3-6 годин)
Алгоритм планування	Bin packing + affinity rules	Spread (рівномірний розподіл)	Bin packing + spread (налаштовно)
Multi-cloud підтримка	Через Federation API	Обмежена (Swarm mode)	Нативна (будь-яка мережева зв'язність)
Утилізація ресурсів (практична)	50-65%	40-55%	70-85% (bin packing ефект)

1.1.5 Концепція Multi-Cloud та Self-Hosted PaaS як відповідь на системні ризики

Аналіз наведених вище тенденцій дозволяє сформулювати концепцію, на якій базується архітектура, що розробляється у даній роботі. Self-Hosted PaaS (Platform as a Service) - це платформа, побудована поверх орендованих «голих» VPS від кількох незалежних хостинг-провайдерів одночасно, яка надає кінцевим

користувачам (клієнтам) абстрагований інтерфейс для розгортання та управління їх додатками без необхідності розуміти деталі базової інфраструктури.

Ключова відмінність від класичного мульти-хмарного підходу (де один і той самий додаток дублюється на AWS та Azure) полягає у формуванні єдиного логічного кластера з фізично незалежних вузлів, з'єднаних через зашифровану mesh-мережу (WireGuard). З точки зору оркестратора (Nomad), всі вузли є рівнозначними - незалежно від того, чи знаходяться вони у Hetzner Cloud у Нюрнберзі, DigitalOcean у Франкфурті або Vultr у Варшаві. Така архітектура забезпечує географічне та провайдерне розосередження при збереженні операційної простоти єдиного кластера.

WireGuard - сучасний протокол VPN, реалізований безпосередньо в ядрі Linux, що забезпечує криптографічно стійке з'єднання між вузлами кластера. На відміну від IPsec або OpenVPN, WireGuard використовує фіксований набір сучасних криптографічних примітивів (Curve25519 для обміну ключами, ChaCha20-Poly1305 для шифрування, BLAKE2s для хешування), що унеможливлює вибір слабких алгоритмів унаслідок помилки конфігурації. Продуктивність WireGuard в ядрі Linux у 3-4 рази перевищує OpenVPN при значно меншому споживанні ресурсів процесора [17, 18].

Архітектура, що розробляється, реалізує принцип Vendor Independence через вимогу мінімум трьох географічно та організаційно незалежних провайдерів. Це означає, що відмова будь-якого одного провайдера (або цілого регіону) призводить лише до часткової втрати обчислювальних ресурсів, але не до зупинки сервісу: кворум для Raft Consensus (2 з 3 серверних вузлів) зберігається, задачі автоматично перепланується на вузли, що залишилися, а зовнішній трафік перенаправляється через edge-вузли інших площадок через механізм DNS-failover або Cloudflare Load Balancing [19, 20].

Таблиця 1.3 - Порівняльний аналіз стратегій побудови IT-інфраструктури

Параметр	On-Premises	Один хмарний провайдер (AWS/Azure)	Multi-Cloud (2 провайдери)	Self-Hosted PaaS (Iron Mesh, 3+ провайдери)
Утилізація CPU/RAM	5-15%	30-50%	40-60%	70-85% (bin packing)
Стійкість до збою провайдера	Відсутня (SPOF)	Відсутня (весь бізнес на 1 провайдері)	Часткова (2 площадки)	Висока (кворум 2/3, auto-failover)
Vendor Lock-in	Низький (власне залізо)	Екстремальний (пропрієтарні сервіси)	Середній	Відсутній (open source стек)
Вартість (умовно, 10 клієнтів)	Висока CAPEX	Висока OPEX (€400-800/міс)	Дуже висока OPEX	Оптимальна (€80-160/міс)
RTO при збої вузла	4-8 годин	5-30 хвилин (авто)	5-15 хвилин	< 1 хвилини (авто-репланування)
Прозорість витрат	Висока (фіксовані)	Низька (змінні + egress)	Низька	Висока (фіксовані тарифи VPS)
Незалежність від інтернет-каналу	Висока (приватна мережа)	Низька (весь трафік через хмару)	Низька	Середня (mesh через інтернет)
Автоматизація розгортання	Відсутня або ручна	Висока (managed сервіси)	Висока	Висока (Nomad + Ansible + CI/CD)

Аналіз порівняльної таблиці демонструє, що Self-Hosted PaaS на базі розподіленого multi-cloud кластера є оптимальним вибором за сукупністю параметрів для IT-компаній, що надають хостинг-послуги малому та середньому бізнесу: вона поєднує автоматизацію та еластичність хмарних рішень із прозорим ціноутворенням on-premises моделі та відсутністю залежності від конкретного вендора[21].

1.2 Аналіз поточного стану інформаційної інфраструктури підприємства

1.2.1 Загальна топологія хмарної інфраструктури на базі AWS (AS-IS)

Підприємство повністю перенесло свою виробничу інфраструктуру на платформу Amazon Web Services (AWS), використовуючи виключно один хмарний регіон -eu-west-1 (Ірландія). Всі обчислювальні ресурси розміщені в одній зоні доступності (Availability Zone), що є типовою практикою для компаній, які поступово мігрували з on-premises без попереднього архітектурного планування мульти-зональної відмовостійкості. Загальний парк складає від 5 до 10 EC2-інстансів різних типів залежно від поточного сезонного навантаження, проте базова конфігурація включає 7 постійних інстансів, детально описаних нижче.

Кожен EC2-інстанс є виділеним вузлом для конкретного сервісу або групи сервісів, що розгорнуті безпосередньо на операційній системі Amazon Linux 2 без рівня контейнеризації. Сервіси запускаються як systemd-юніти або через PM2 (для Node.js-компонентів), конфігурація зберігається у файлах на диску інстансу, а зв'язок між компонентами здійснюється за внутрішніми IP-адресами VPC (Virtual Private Cloud) без service discovery -адреси прописані статично у конфігураційних файлах кожного сервісу.

Наведена схема демонструє критичну архітектурну особливість поточної інфраструктури: всі 7 EC2-інстансів розміщені в межах однієї зони доступності (eu-west-1a). Зона доступності AWS є фізично незалежним ЦОД у межах регіону, і її збій є малоймовірним, але задокументованим сценарієм (збій AZ в us-east-1 у липні 2022 р. тривав кілька годин). Проте більш критичним є те, що всі інстанси знаходяться в єдиному регіоні: збій регіону AWS eu-west-1 або будь-яка регіонально специфічна проблема (DDoS, законодавчі обмеження, аварія мережевої інфраструктури) призводить до повної недоступності всіх сервісів підприємства без жодного механізму автоматичного відновлення.

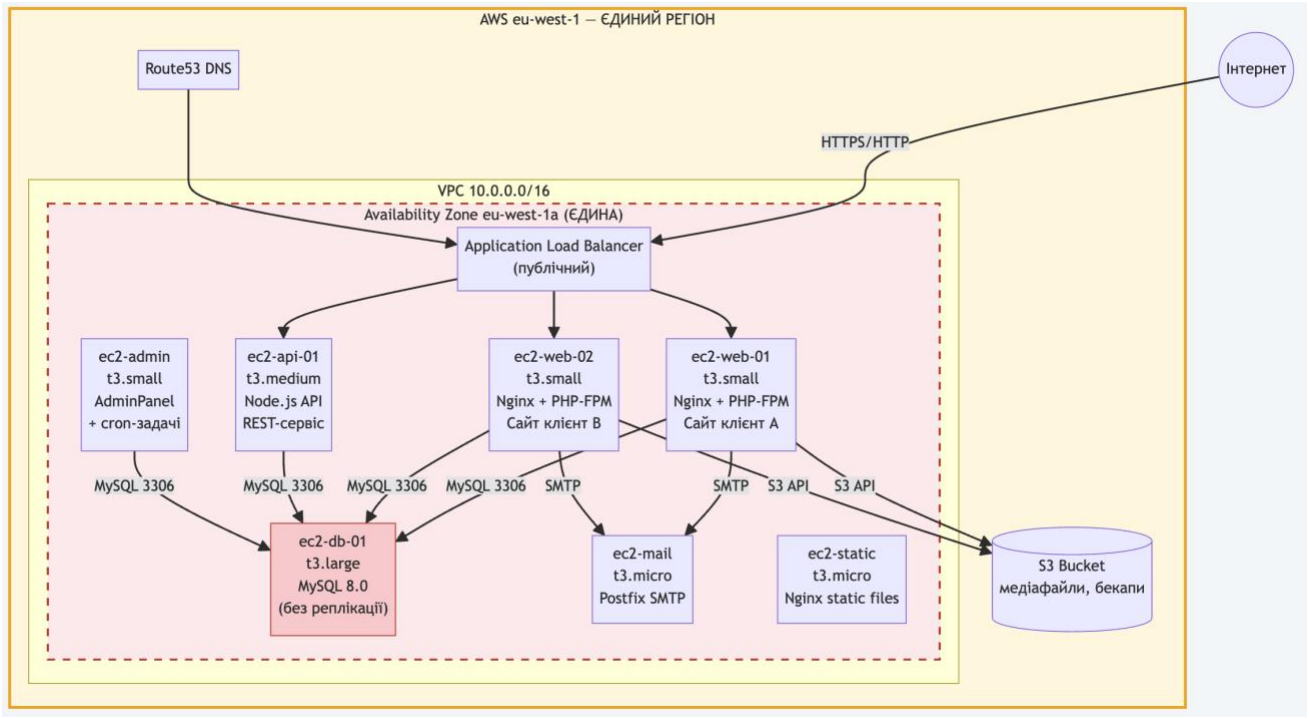


Рисунок 1.1 - Поточна топологія хмарної інфраструктури підприємства (AWS eu-west-1)

1.2.2 Характеристика EC2-інстансів та встановленого програмного забезпечення

Програмний стек підприємства розгорнутий безпосередньо на операційній системі Amazon Linux 2 без використання контейнеризації. Відсутність Docker або будь-якого іншого рівня абстракції виконання означає, що залежності кожного сервісу встановлені глобально на рівні ОС через пакетний менеджер yum/dnf, що породжує конфлікти версій при спробі розмістити кілька сервісів на одному інстансі та ускладнює відтворення середовища при відновленні після збою.

Станом на початок 2025 року для PHP 7.4 задокументовано понад 50 невиправлених CVE (Common Vulnerabilities and Exposures), серед яких є критичні вразливості класу RCE (Remote Code Execution), що дозволяють зловмиснику виконати довільний код на сервері.

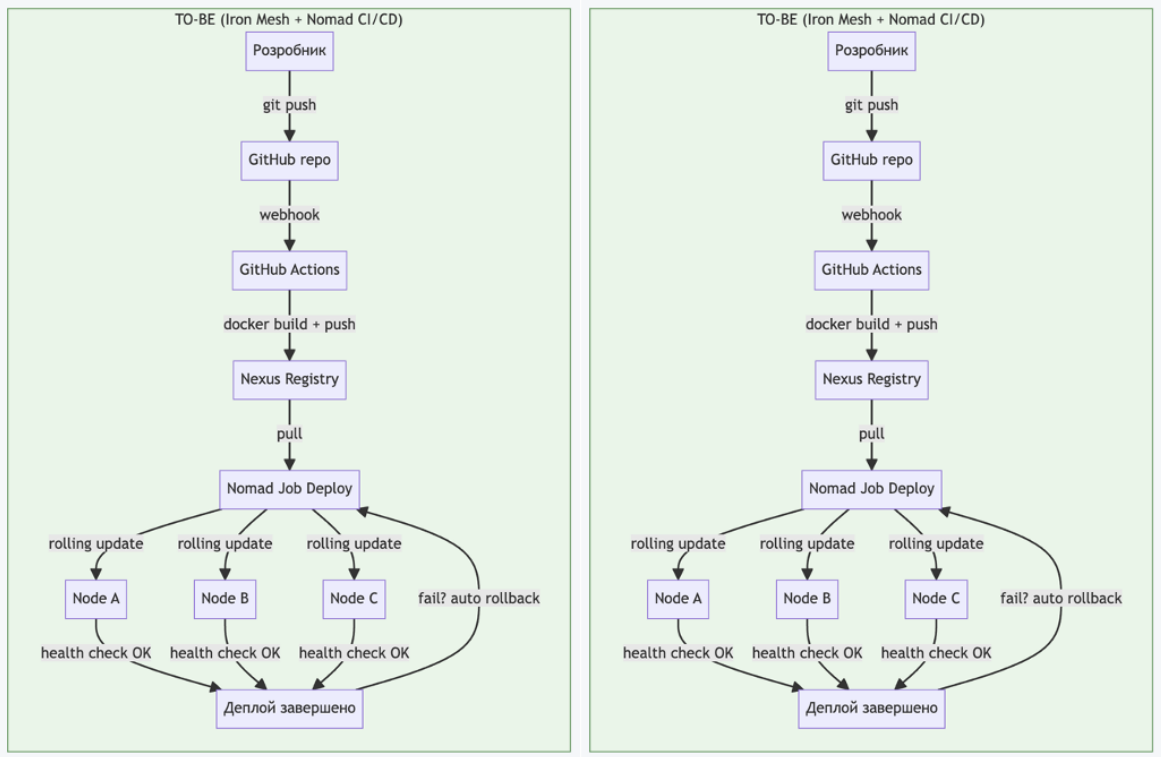


Рисунок 1.2 - Програмний стек EC2-інстансів підприємства

Особливої уваги заслуговує ec2-web-01 та ec2-web-02, на яких встановлено PHP-FPM версії 7.4. Ця версія PHP досягла статусу End-of-Life 28 листопада 2022 року і з того часу не отримує жодних оновлень безпеки від PHP Group. Продовження використання PHP 7.4 у продуктивному середовищі є прямим порушенням базових вимог інформаційної безпеки та вимог стандарту PCI DSS (якщо на сервері обробляються платіжні дані).

1.2.3 Процеси розгортання та операційного управління

Відсутність CI/CD pipeline є одним із ключових операційних недоліків поточної інфраструктури. Розгортання нових версій сервісів або конфігураційних змін здійснюється вручну через SSH-підключення до кожного EC2-інстансу окремо.

На підставі проведеного аналізу сформульовано перелік системних недоліків поточної AWS-інфраструктури, кожен з яких безпосередньо впливає на здатність підприємства забезпечувати безперервність надання послуг.

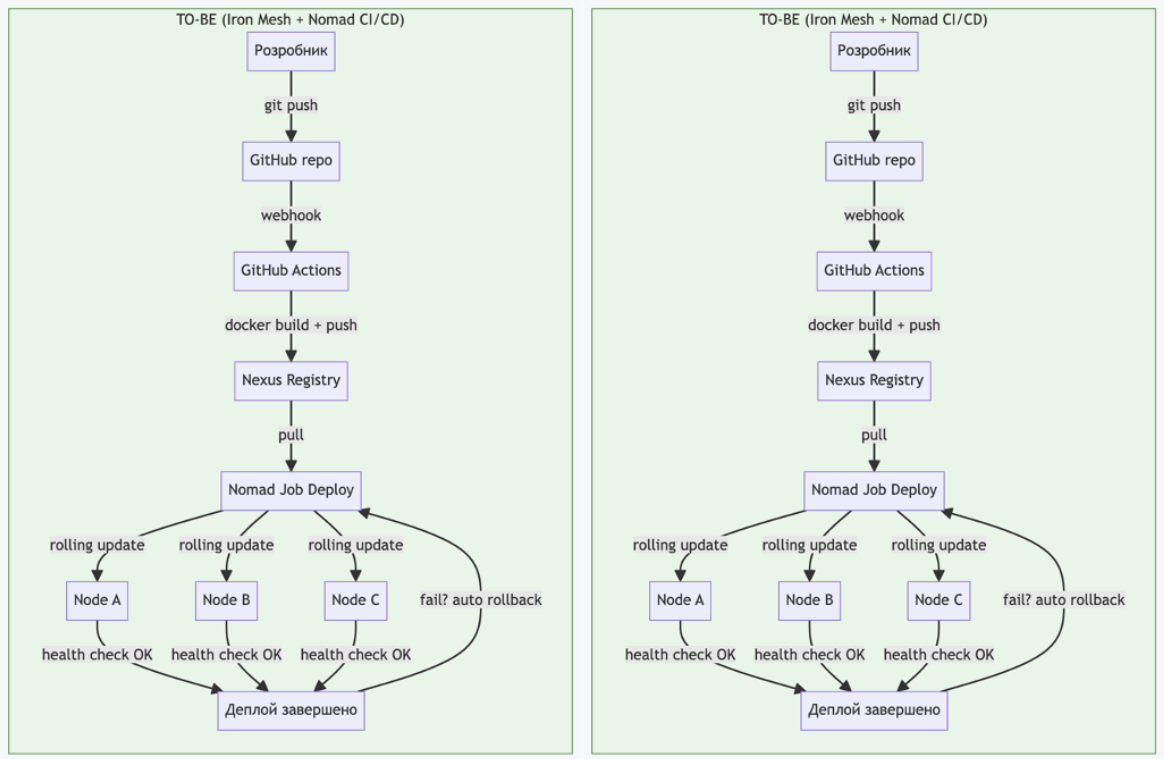


Рисунок 1.3 - Порівняння поточного (AS-IS) та цільового (TO-BE) процесу розгортання

Типова послідовність деплою включає: підключення до інстансу через AWS Session Manager або SSH-ключ, зупинку сервісу через systemctl stop або pm2 stop, оновлення коду через git pull з приватного репозиторію, ручне оновлення конфігураційних файлів (якщо потрібно), перезапуск сервісу та ручну перевірку логів через journalctl або tail -f. За відсутності автоматизованого тестування та механізму rollback, помилкове оновлення може залишатись непоміченим до першого повідомлення від клієнта.

1.2.4 Виявлені критичні недоліки та їх вплив на безперервність бізнес-процесів

Vendor lock-in на рівні регіону вказує на те що 100% інфраструктури розміщено в одному регіоні AWS eu-west-1. Будь-який регіональний збій AWS (задокументовані прецеденти: US-EAST-1 у грудні 2021 та липні 2024 р.) призводить до повної зупинки всіх бізнес-процесів. Час відновлення повністю залежить від AWS і може складати від 1 до 7+ годин без жодних можливостей впливу з боку підприємства.

Single Point of Failure бази даних ec2-db-01 з MySQL без реплікації -це єдина точка відмови для всього стеку. Відмова EBS-тому, вичерпання I/O burst balance або помилка адміністратора зупиняють всі сервіси одночасно. RPO = до 24 год (між mysqldump-бекапами), RTO = 1-4 год (відновлення дампу).

Статична адресація між сервісами а саме IP-адреси EC2-інстансів прописані статично у конфігах. Заміна будь-якого інстансу (наприклад, після збою) вимагає ручного оновлення конфігурацій на всіх залежних вузлах і перезапуску сервісів - операція, яка займає 30-90 хвилин і вимагає участі адміністратора.

Відсутність централізованого моніторингу та алертингу у вигляді того що AWS CloudWatch налаштований лише для базових метрик EC2 (CPU, мережа), але відсутні алерти на рівні застосунку: помилки HTTP 5xx, затримка відповіді API, переповнення диску. Проблеми виявляються клієнтами, а не командою підтримки.

Відсутність шифрування секретів. Паролі до MySQL, ключі AWS IAM та API-токени зберігаються у .env-файлах на дисках EC2-інстансів у незашифрованому вигляді. Компрометація будь-якого інстансу дає зловмиснику доступ до всіх облікових даних.

Ручний деплой без rollback та CI/CD. Середній час деплою оновлення на всі інстанси - 45-90 хвилин ручної роботи. Відсутність автоматичного rollback означає, що помилкове оновлення може тривати до 2-3 годин до повного відновлення попередньої версії.

Сукупний аналіз виявлених недоліків підтверджує, що поточна AWS-інфраструктура підприємства не відповідає базовим вимогам до інформаційної безпеки та безперервності бізнес-процесів. Усунення виявлених проблем неможливе точковими заходами в межах поточної архітектури і потребує проєктування принципово нової розподіленої платформи, що є предметом постановки задачі у підрозділі 1.3.

Рисунок 1.4 - Матриця ризиків поточної AWS-інфраструктури підприємства

Матриця ризиків наочно демонструє, що три ризики потрапляють у квадрант критичних (висока імовірність та критичний вплив): вразливість PHP 7.4, відсутність централізованого моніторингу та статична IP-адресація. Ще два ризики - збій регіону AWS та витік секретів через .env - знаходяться у квадранті стратегічних: їх імовірність нижча, але вплив є катастрофічним для безперервності бізнесу. Відмова ec2-db-01 є окремим сценарієм з найвищим значенням впливу (0.98) -фактично повна зупинка всіх сервісів.

1.3 Постановка задачі

На підставі аналізу предметної області (підрозділ 1.1) та критичних недоліків поточної AWS-інфраструктури (підрозділ 1.2) сформульовано задачу: спроектувати та реалізувати розподілену відмовостійку PaaS-платформу на базі кластера з мінімумс чотирьох географічно та організаційно незалежних вузлів, що забезпечує автоматичне відновлення після відмови будь-якого одного вузла або хмарного провайдера без переривання обслуговування.

Таблиця 1.6 - Нефункціональні вимоги (SLA) до платформи

Uptime	~97-98%	≥ 99.9% (≤ 8.76 год/рік)
RTO (час відновлення)	1-4 год (ручний)	< 60 сек (авто failover)
RPO (втрата даних)	До 24 год (mysqldump)	< 5 хв (Galera sync)
Час деплою	45-90 хв (SSH вручну)	< 5 хв (CI/CD pipeline)
Утилізація CPU	8-15%	≥ 60% (bin packing)
Шифрування трафіку	0% (відкритий VPC)	100% (WireGuard + mTLS)
Залежність від провайдера	100% AWS eu-west-1	0% (3 незалежні провайдери)

1.3.2 Цільова архітектура та критерії успіху

Цільова архітектура -кластер з трьох майданчиків (Hetzner / DigitalOcean / Vultr), об'єднаних через WireGuard mesh і керованих HashiCorp Nomad з Raft Consensus (кворум 2/3). Зовнішній трафік розподіляється через Cloudflare з автоматичним DNS failover між edge-вузлами.

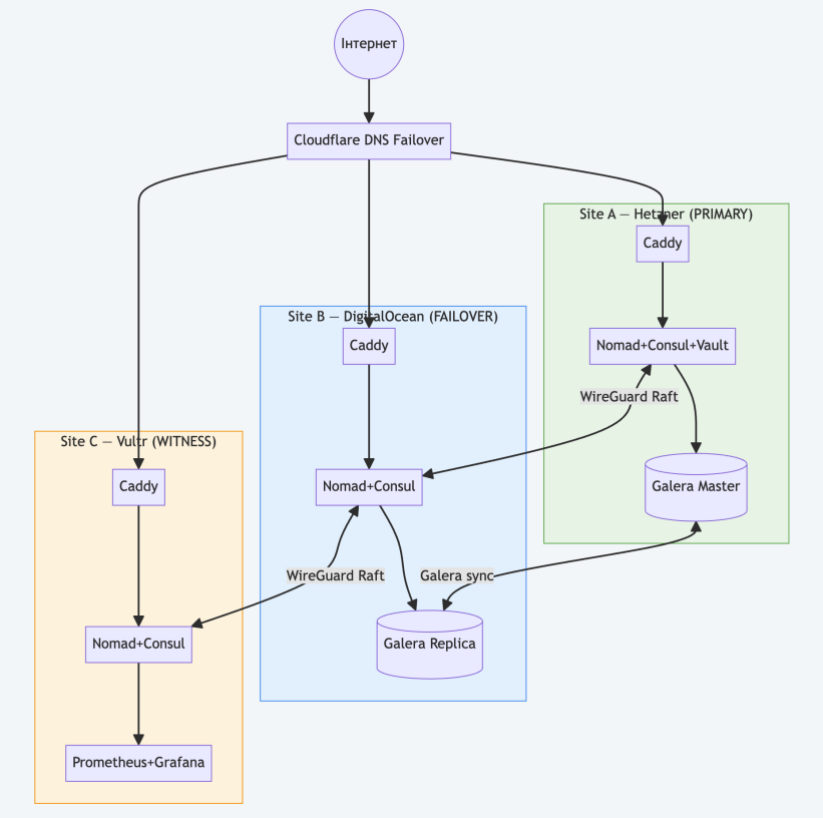


Рисунок 1.5 - Цільова архітектура Iron Mesh PaaS (TO-BE)

Задача вважається вирішеною при одночасному виконанні п'яти критеріїв: (1) збереження доступності сервісів при вимкненні Site A протягом ≤ 60 сек; (2) відсутність втрати даних при failover (RPO = 0); (3) успішний CI/CD деплой з rollback менш ніж за 5 хвилин; (4) відсутність plain-text секретів на файловій системі; (5) середня утилізація CPU кластера ≥ 60%.

1.4 Висновки

У першому розділі проведено аналіз еволюції IT-інфраструктур та підтверджено системні ризики концентрації сервісів у єдиного хмарного провайдера на реальних інцидентах AWS і Azure 2021-2024 рр. із задокументованими збитками понад 10 млрд дол. США. Аналіз поточної AWS EC2-інфраструктури підприємства виявив 7 критичних недоліків: відсутність failover, SPOF бази даних, EOL-компоненти (PHP 7.4), plain-text секрети та відсутність CI/CD. Сформульовано задачу проектування розподіленої PaaS-платформи Iron Mesh з 6 функціональними вимогами, SLA-метриками та верифікованими критеріями успіху (uptime ≥ 99.9%, RTO < 60 сек, RPO < 5 хв).

2 ПОБУДОВА СИСТЕМИ КЕРУВАННЯ ІНФОРМАЦІЙНОЮ БЕЗПЕКОЮ ДЛЯ ЗАБЕЗПЕЧЕННЯ БЕЗПЕРЕРВНОСТІ БІЗНЕС-ПРОЦЕСІВ

2.1 Специфікація та архітектура системи

На підставі виявлених у розділі 1 критичних недоліків AWS-інфраструктури підприємства та сформульованих функціональних і нефункціональних вимог (підрозділ 1.3), у даному розділі описано процес проєктування та реалізації розподіленої відмовостійкої платформи Iron Mesh PaaS. Система побудована на принципах мульти-провайдерної архітектури, шифрованого mesh-з'єднання та оркестрації контейнерів із централізованим управлінням секретами, моніторингом та виявленням інцидентів безпеки.

Перед описом реалізації необхідно зафіксувати основні проєктні обмеження. По-перше, бюджетне: загальна вартість інфраструктури не повинна перевищувати еквівалент поточних витрат на AWS (~$400-500/міс). По-друге, операційне: адміністрування кластера має здійснюватися однією особою без цілодобового чергування. По-третє, масштабування: архітектура має дозволяти горизонтальне розширення від 3 до 100+ вузлів без зміни базових компонентів.

2.1.1 Архітектурні принципи побудови системи

Проєктування системи базується на семи фундаментальних принципах, що утворюють взаємопов'язану систему: реалізація одного підсилює дію іншого. Наприклад, Infrastructure as Code (№4) є передумовою для Cattle, not Pets (№2), а Zero Trust Network (№3) - фундаментом Defense in Depth (№7). Систематизацію принципів подано у таблиці 2.1.

Принцип Cattle, not Pets концептуально змінює підхід до управління серверами. Дані зберігаються у реплікованих сховищах (MariaDB Galera, MinIO), конфігурація - у Git. У попередній AWS-інфраструктурі кожен EC2-інстанс був «улюбленцем» (pet) - з унікальним набором пакетів та конфігурацій, встановлених вручну протягом місяців.

Таблиця 2.1 - Архітектурні принципи побудови системи та їх зв'язок із виявленими проблемами

№	Принцип	Опис	Проблема, що вирішується
1	No Single Point of Failure	Кожен критичний компонент має мінімум 3 екземпляри, об'єднані Raft-консенсусом. Відмова будь-якого одного вузла або майданчика не зупиняє систему	SPOF бази даних (1.2.4, п.2), vendor lock-in (1.2.4, п.1)
2	Cattle, not Pets	Сервери - замінні одиниці. Будь-який вузол перебудовується з нуля Ansible за 15-30 хвилин без втрати даних	Ручний деплой (1.2.4, п.7), відсутність стандартизації
3	Zero Trust Network	Весь міжвузловий трафік зашифрований WireGuard (ChaCha20-Poly1305). Публічно доступні лише 80/443 на edge. Кожен запит автентифікується токеном	Відсутність шифрування секретів (1.2.4, п.6), статична адресація (1.2.4, п.4)
4	Infrastructure as Code	Конфігурація описана у файлах під контролем версій: Ansible playbooks, Nomad HCL, Consul ACL policies	Ручний деплой через SSH (1.2.3)
5	Push-to-Deploy	git push ініціює збирання Docker-образу, публікацію у реєстр та rolling update через Nomad без SSH	Ручний деплой: 45-90 хв. (1.2.3)
6	Data Gravity	Бази реплікуються синхронно (MariaDB Galera, RPO=0). Файли - через S3 API (MinIO) з daily mirror та Restic backup	SPOF бази (1.2.4, п.2), RPO = 24 год.
7	Defense in Depth	Ешелонований захист: Cloudflare → UFW → WireGuard → ACL → Vault → Container. Компрометація одного рівня не відкриває наступний	EOL-компоненти (1.2.4, п.3), відсутність моніторингу (1.2.4, п.5)

Втрата такого інстансу вимагала тривалого відновлення. У новій архітектурі сервер - це «худоба» (cattle): стандартизований вузол, який можна знищити та відтворити за 15-30 хвилин одним Ansible playbook.

2.1.2 Загальна архітектура платформи

Цільова архітектура являє собою кластер із 12 серверних вузлів, розподілених між трьома географічно та організаційно незалежними майданчиками (Sites). Кожен майданчик розміщений у окремого хостинг-провайдера для виключення залежності від єдиного постачальника. Навіть повне припинення роботи одного провайдера залишає два інших працездатними (рисунок 2.1).

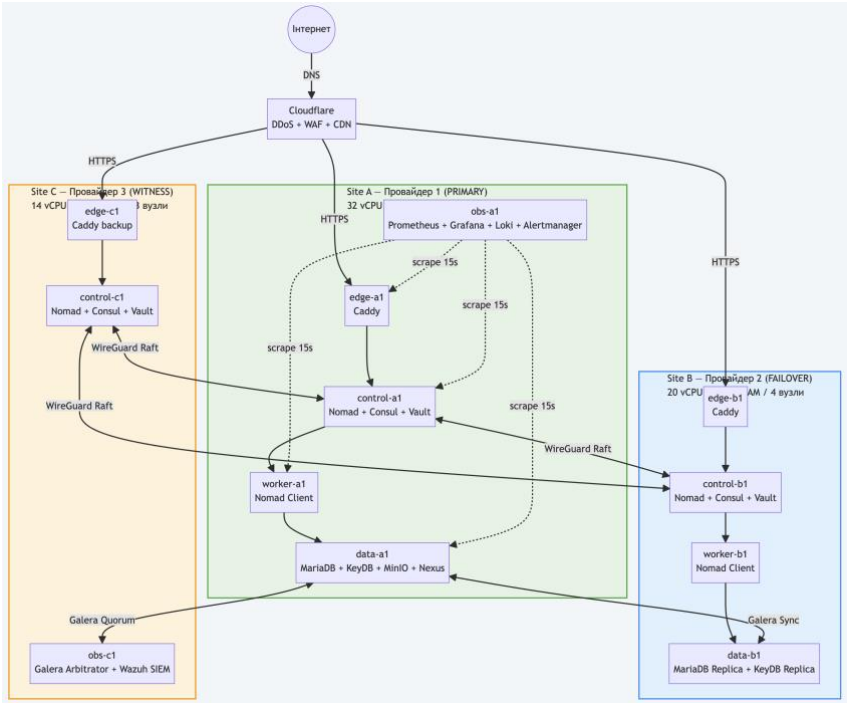


Рисунок 2.1 - Загальна архітектура платформи Iron Mesh PaaS

Архітектура побудована за моделлю Primary-Failover-Witness. Site A (Primary) - основний майданчик із найбільшою потужністю (32 vCPU, 64 ГБ RAM, 5 вузлів): edge-проксі (edge-a1), панель керування з Nomad Server, Consul Server, Vault (control-a1), обчислювальний вузол для Docker-контейнерів (worker-a1), вузол даних з MariaDB Galera, KeyDB Master, MinIO, Nexus (data-a1) та вузол моніторингу (obs-a1).

Таблиця 2.2 - Розподіл вузлів кластера за майданчиками та ролями

Вузол	Майданчик	Роль	vCPU	RAM	Disk	WG IP
edge-a1	Site A	Caddy reverse proxy, SSL termination	4	8 GB	80 GB	10.10.1.1
control-a1	Site A	Nomad Server, Consul Server, Vault	8	8 GB	100 GB	10.10.1.2
worker-a1	Site A	Nomad Client (Docker workloads)	8	16 GB	200 GB	10.10.1.3
data-a1	Site A	MariaDB Galera, KeyDB Master, MinIO, Nexus	6	16 GB	300 GB	10.10.1.4
obs-a1	Site A	Prometheus, Grafana, Loki, Alertmanager	6	16 GB	200 GB	10.10.1.5
edge-b1	Site B	Caddy reverse proxy	4	8 GB	80 GB	10.10.2.1
control-b1	Site B	Nomad Server, Consul Server, Vault	4	8 GB	100 GB	10.10.2.2
worker-b1	Site B	Nomad Client (Docker workloads)	6	12 GB	150 GB	10.10.2.3
data-b1	Site B	MariaDB Galera Replica, KeyDB Replica	6	12 GB	200 GB	10.10.2.4
edge-c1	Site C	Caddy reverse proxy (резервний)	4	8 GB	80 GB	10.10.3.1
control-c1	Site C	Nomad Server, Consul Server, Vault	4	8 GB	100 GB	10.10.3.2
obs-c1	Site C	Galera Arbitrator, Wazuh SIEM	6	16 GB	300 GB	10.10.3.3
Разом			66	136 GB	1890 GB	

Site B (Failover) дублює критичні компоненти з меншою потужністю edge-проксі, панель керування, обчислювальний вузол та вузол реплікації бази даних. Site C (Witness) виконує роль арбітра для кворуму з мінімальними ресурсам: резервний edge-проксі, панель керування та спеціалізований вузол безпеки (obs-c1) з Galera Arbitrator та Wazuh SIEM. Розміщення SIEM на окремому

майданчику є свідомим рішенням: у разі компрометації Site A журнали безпеки на Site C залишаються недоторканими.

Трисайтова архітектура обрана через вимоги алгоритму Raft: кворум Q = ⌊N/2⌋ + 1, де N - кількість серверів. Для N=3 кворум = 2, що дозволяє продовжити роботу при падінні одного майданчика. Конфігурація з 5 серверів (кворум 3, витримує падіння 2) була б надмірною для поточного масштабу та збільшила б вартість на ~40% [21]. Повний розподіл вузлів подано у таблиці 2.2.

Сукупна потужність: 66 vCPU, 136 ГБ RAM, 1890 ГБ SSD. Порівняно з попередньою AWS-інфраструктурою (7 EC2, ~14 vCPU, 28 ГБ RAM, один регіон одного провайдера) - 4,7-кратне збільшення ресурсів при зниженні вартості за рахунок VPS замість хмарних ресурсів.

2.1.3 Технологічний стек

Вибір кожного компонента визначається критеріями: відкритий вихідний код (уникнення vendor lock-in на рівні ПЗ), активна спільнота, мінімальне споживання ресурсів (важливо для VPS), сумісність з іншими компонентами та вбудовані механізми високої доступності.

Щодо ліцензування HashiCorp: у серпні 2023 року ліцензію Nomad, Consul та Vault змінено з MPL 2.0 на BSL 1.1 (Business Source License). BSL 1.1 дозволяє вільне використання для будь-яких цілей, окрім надання цих продуктів як керованого хмарного сервісу конкурентам HashiCorp Cloud Platform. Для self-hosted інфраструктури підприємства зміна ліцензії не створює обмежень.

2.1.4 Процедура автоматизованого розгортання (Ansible)

Розгортання кластера автоматизовано через набір Ansible playbooks - декларативних сценаріїв, що описують бажаний стан кожного вузла. Ansible працює за push-моделлю: адміністратор запускає playbook зі своєї робочої станції, і Ansible підключається до вузлів через SSH без необхідності встановлення агента на керованих серверах.

Процес розбито на 7 послідовних фаз: (1) WireGuard mesh + UFW + fail2ban (~5 хв), (2) Docker CE + базові пакети (~10 хв), (3) HashiCorp Stack: Consul + Vault + Nomad з ACL bootstrap (~15 хв), (4) рівень даних: MariaDB Galera + KeyDB +

MinIO + Nexus (~20 хв), (5) Edge: Caddy на 3 вузлах + consul-template (~5 хв), (6) моніторинг: Prometheus + exporters + Grafana + Loki + Promtail + Alertmanager + Wazuh SIEM (~25 хв), (7) резервне копіювання: Restic repos + cron + Pushgateway (~5 хв). Повне розгортання з 12 вузлів «з нуля» - приблизно 85 хвилин. Для порівняння: попередня AWS-інфраструктура з 7 EC2 налаштовувалась вручну кілька робочих днів без документованої процедури відтворення.

2.2 Мережевий рівень: WireGuard mesh, міжмережевий екран та DNS

2.2.1 Аналіз та обґрунтування вибору VPN-протоколу

Мережевий рівень є фундаментом платформи: від його надійності залежить функціонування всіх вищих рівнів. Основне завдання - створити єдиний захищений адресний простір для 12 вузлів у різних датацентрах таким чином щоб для сервісів кластера вони виглядали як вузли однієї локальної мережі. Зробивши це ми зможемо якісно організувати нашу мережу. Порівняння VPN-протоколів подано у таблиці 2.3.

WireGuard обрано за сукупністю: найвища продуктивність (~900 Мбіт/с), мінімальна латентність (< 1 мс), вбудованість у ядро Linux, компактна кодова база що пройшла формальну верифікацію, та найпростіша конфігурація. Критично важливим для кластера є stateless roaming: при зміні IP вузла WireGuard автоматично оновлює маршрут при першому пакеті від peer .

Криптографічний стек WireGuard використовує фіксований набір примітивів без можливості конфігурування (на відміну від OpenVPN/IPsec, де адміністратор може обрати слабкий алгоритм): Curve25519 для обміну ключами, ChaCha20 для шифрування (оптимізований для CPU без апаратного AES, типових для VPS), Poly1305 для MAC, BLAKE2s для хешування. Рівень безпеки еквівалентний 128-бітному симетричному ключу - достатній на 20+ років за оцінками NIST SP 800-57 [22].

Таблиця 2.3 - Порівняльний аналіз VPN-протоколів для mesh-мережі кластера

Характеристика	WireGuard	OpenVPN	IPsec (StrongSwan)
Реалізація	Ядро Linux (з v5.6)	Userspace (tun/tap)	Ядро Linux (xfrm)
Кодова база	~4 000 рядків	~100 000 рядків	~400 000 рядків
Криптографія	Curve25519, ChaCha20-Poly1305, BLAKE2s	OpenSSL (конфігурується)	IKEv2, AES-GCM (конфігурується)
Throughput (1 Гбіт лінк)	~900 Мбіт/с	~300-500 Мбіт/с	~700-800 Мбіт/с
Латентність (overhead)	< 1 мс	2-5 мс	1-2 мс
Час з'єднання	~100 мс (1-RTT)	~1000 мс (TLS)	~500 мс (IKE SA)
Roaming (зміна IP)	Автоматично (stateless)	Перепідключення	MOBIKE (IKEv2)
Конфігурація	~15 рядків/peer	~50-100 рядків	~100-200 рядків
Аудит безпеки	Формальна верифікація (2018)	Часткові аудити	Складно через обсяг коду

2.2.2 Проєктування Full Mesh топології

Обрана топологія - Full Mesh, де кожна пара вузлів має прямий тунель. Альтернатива Hub-and-Spoke (зірка) відкинута з трьох причин: SPOF (падіння hub розриває весь кластер), подвоєна латентність (трафік між spoke проходить через hub двічі), та вузьке місце на hub (весь міжвузловий трафік проходить через один вузол). Full Mesh забезпечує: відсутність SPOF, мінімальну латентність (один хоп), рівномірний розподіл навантаження.

Кількість тунелів: $T = N \times (N - 1) / 2$. Для 12 вузлів: 66 тунелів. Для масштабу 50+ вузлів (1225 тунелів) планується перехід на автоматизовані контролери Netmaker або Headscale (рисунок 2.2).

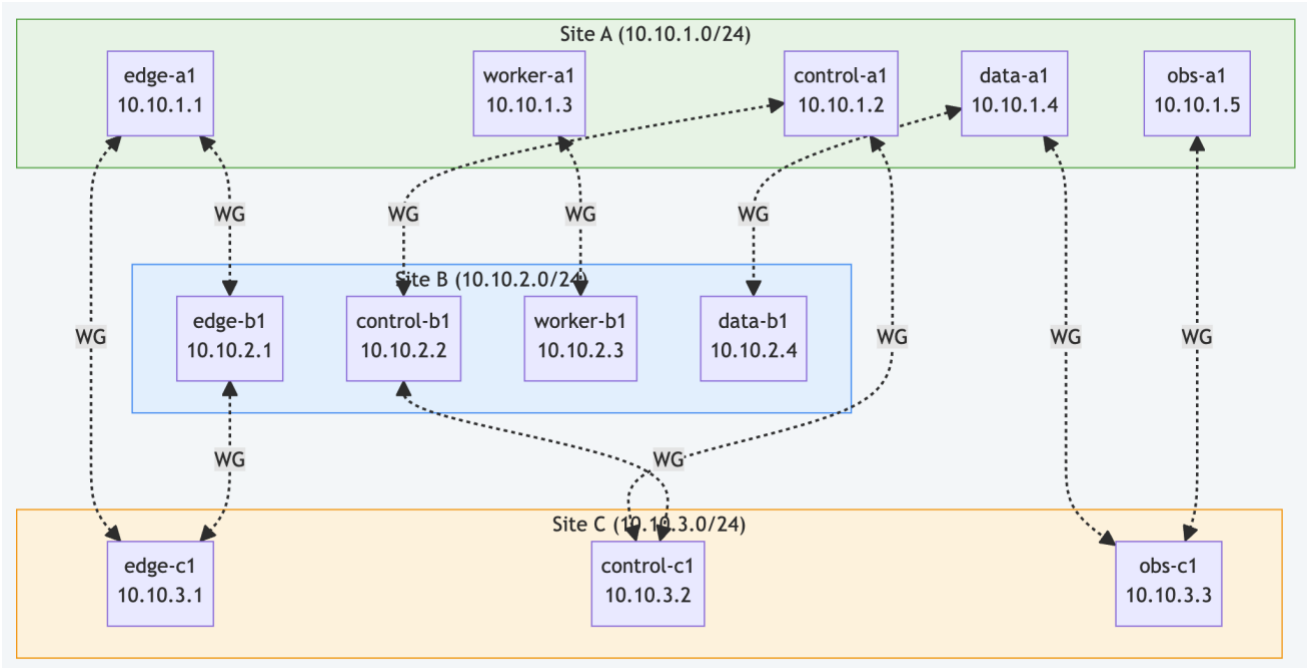


Рисунок 2.2 - Топологія WireGuard Full Mesh (12 вузлів, 66 тунелів)

Адресний простір: приватна підмережа 10.10.0.0/16, розділена за майданчиками - Site A: 10.10.1.0/24, Site B: 10.10.2.0/24, Site C: 10.10.3.0/24. Усередині кожного сегменту адреси за роллю: .1 - edge, .2 - control, .3 - worker, .4 - data, .5 - obs. Резерв 10.10.4.0/24 - 10.10.255.0/24 для майбутніх майданчиків (до 252 сайтів × 254 вузлів).

2.2.3 Налаштування міжмережевого екрану (UFW)

Для реалізації Zero Trust на мережевому рівні кожен вузол має UFW з implicit deny: весь вхідний трафік заборонений, дозволені лише явно вказані порти.

Всі сервісні порти прив'язані виключно до WireGuard-інтерфейсу (bind_addr = "10.10.x.x"). Це створює подвійний захист: UFW + bind address - навіть якщо правило UFW буде видалено, сервіси не слухають на публічному інтерфейсі. Додатково fail2ban блокує IP після 5 невдалих SSH-спроб за 10 хвилин (блокування на 1 годину).

2.2.4 DNS-резолюція через Consul та dnsmasq

На кожному вузлі налаштовано dnsmasq (127.0.0.1:53), який розподіляє

DNS-запити: зона .consul → Consul DNS (127.0.0.1:8600), решта → Cloudflare (1.1.1.1) та Google (8.8.8.8) з DNSSEC (рисунок 2.3).

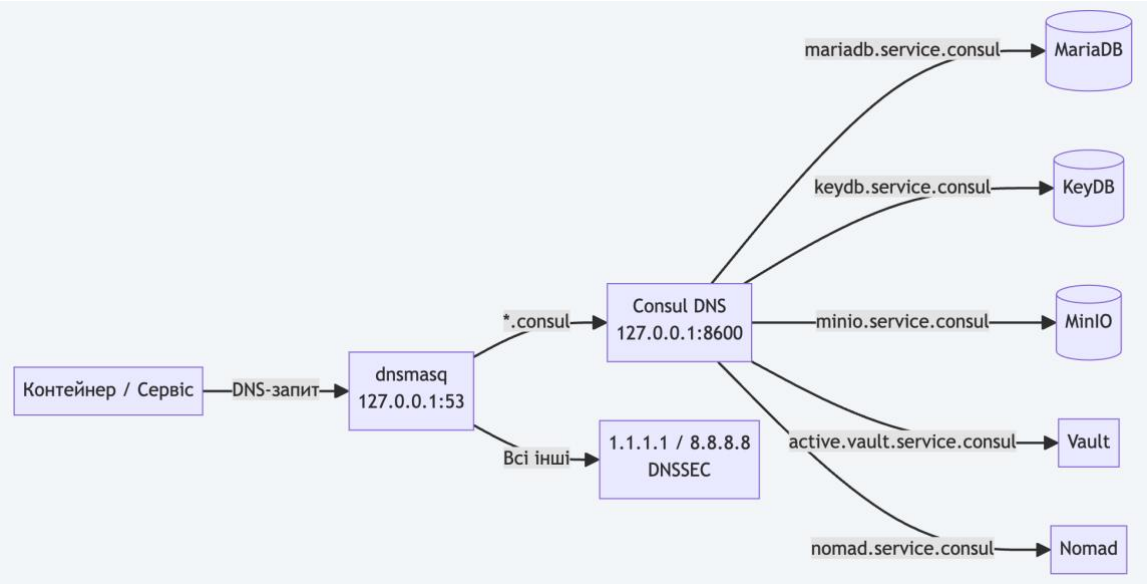


Рисунок 2.3 - DNS-резолюція через dnsmasq та Consul

Consul DNS забезпечує автоматичне перетворення імен сервісів у IP здорових вузлів. Замість статичної адреси 10.10.1.4:3306 додатки використовують mariadb.service.consul, яке Consul розрізнює в адресу поточного здорового вузла з урахуванням health check. Основні записи: mariadb.service.consul (TCP port 3306, 10c інтервал, round-robin між data-a1 та data-b1), keydb.service.consul (TCP 6379), active.vault.service.consul (HTTP /v1/sys/health, повертає лише unsealed лідера), minio.service.consul (HTTP /minio/health/live), nomad.service.consul (HTTP /v1/status/leader).

При виході вузла з ладу Consul виключає його з DNS за 10-30 секунд. При відновленні - включає назад після проходження health check. Весь процес автоматичний, що вирішує проблему статичної IP-адресації (підрозділ 1.2.4, п.4): замість 30-90 хвилин ручного оновлення конфігурацій - 10-30 секунд автоматичного DNS failover.

2.2.5 Зовнішній DNS та DDoS-захист через Cloudflare

Cloudflare (Free tier) забезпечує: проксування через глобальну мережу (300+ PoP), фільтрацію DDoS на L3/L4/L7, WAF (OWASP), CDN та DNS Health Check

failover. Доменні записи (*.arctum.tech → 3 edge IP, delta-consul/nomad/vault/grafana/wazuh/minio/nexus/registry.arctum.tech → edge-a1 → відповідний internal сервіс) захищені Cloudflare Proxy.

Health Checks: HTTP probe кожні 60 секунд, 3 невдачі = виключення з DNS (~3 хв), 1 успіх = повернення (~1 хв). Edge-вузли на інших майданчиках маршрутизують через WireGuard до контейнерів на будь-якому вузлі: edge-b1 може проксувати до worker-a1, якщо Site A частково працює.

2.3 Рівень оркестрації: HashiCorp Nomad, Consul та Vault

2.3.1 Архітектура панелі керування (Control Plane)

Панель керування побудована на трьох компонентах HashiCorp Stack: Nomad, Consul, Vault. Кожен розгорнутий у конфігурації 3 серверів (control-a1, control-b1, control-c1) з незалежним Raft-кластером. Три окремих Raft-кластери (а не один спільний) підвищують стійкість: збій Raft у Consul не впливає на Nomad або Vault (рисунок 2.4).

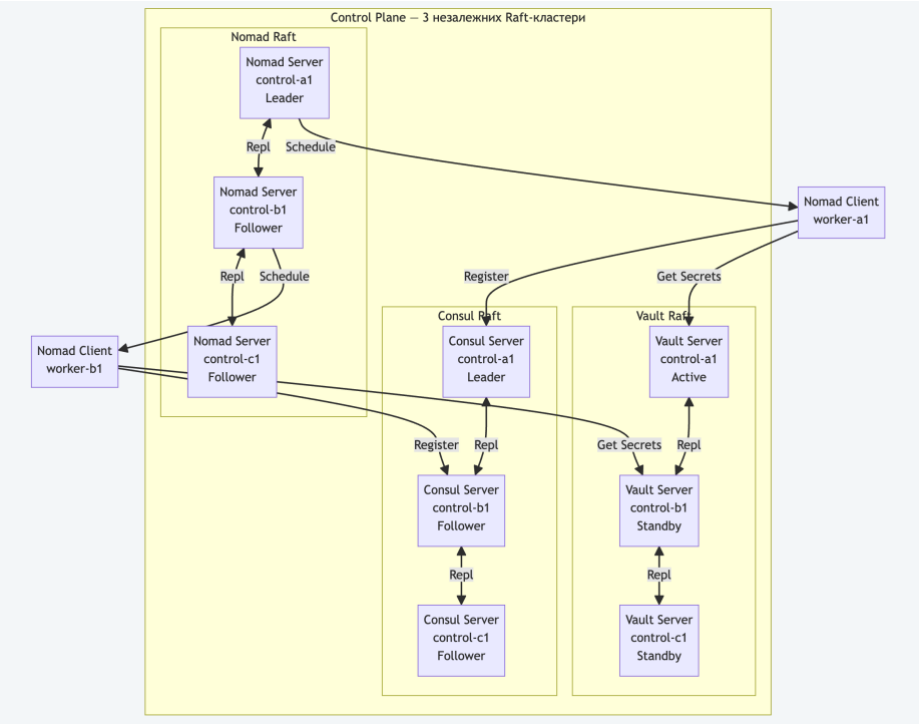


Рисунок 2.4 - Три незалежних Raft-кластери HashiCorp Stack

Raft - алгоритм розподіленого консенсусу (Ongaro & Ousterhout, Stanford, 2014), що гарантує лінеаризуємість: будь-яке читання повертає результат останнього запису, всі операції впорядковані. Працює за принципом leader election: лідер приймає записи, реплікує на followers, підтверджує після кворуму. При недоступності лідера - перевибори за ~5 секунд. Стан кластерів: Consul - 3 servers + 9 clients (ACL: deny), Vault - 3 voters (Raft storage, Shamir unsealed), Nomad - 3 servers + 2 clients (ACL).

2.3.2 HashiCorp Nomad - оркестрація контейнерів

Nomad приймає декларативний опис сервісу і розподіляє контейнери по вузлах з урахуванням ресурсів та відмовостійкості. На відміну від Kubernetes, не потребує окремих CNI/CSI/DNS - ці функції забезпечують Consul та Vault.

Ключові можливості. Rolling Updates з auto_revert: нові контейнери запускаються по одному, мають бути здоровими 30 секунд, при провалі за 5 хвилин - автоматичний відкат. Порівняно з попередніми 45-90 хв ручного деплою без rollback. Bin Packing: щільне розміщення контейнерів на вузлах з урахуванням декларованих ресурсів.

Інтеграція з Consul: при запуску контейнера Nomad реєструє його в Consul Інтеграція з Vault: блок template у job spec динамічно підставляє секрети у змінні оточення контейнера; при ротації секрету - автоматичний restart (рис 2.5).

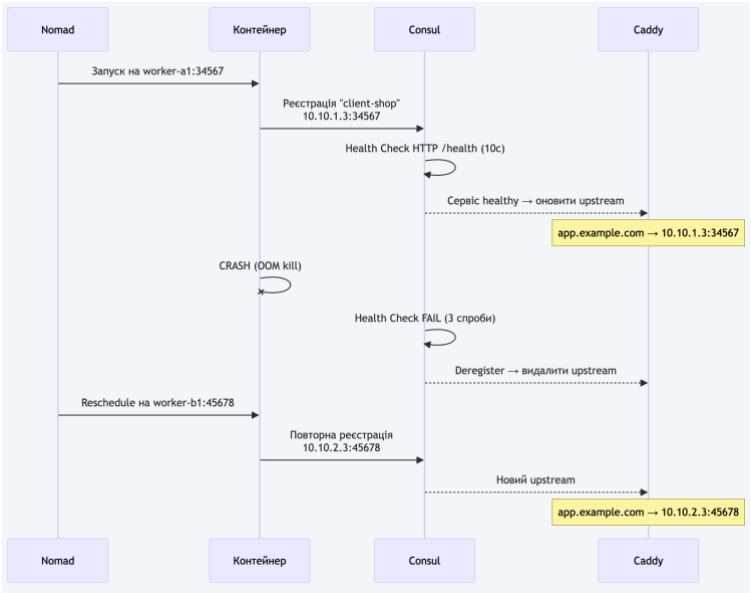


Рисунок 2.5 - Реєстрація сервісу та failover через Consul

2.3.3 HashiCorp Consul - Service Discovery та ACL

Consul забезпечує: реєстр сервісів, DNS-резолювання, Health Checking та ACL. Процес автоматичний: Nomad запускає контейнер → реєстрація у Consul → health check кожні 10с → при success включає в DNS → Caddy маршрутизує трафік. При crash: health check fail → виключення з DNS → Nomad reschedule → повторна реєстрація.

ACL налаштовано з default policy: deny. Для кожного компонента - окремий токен з мінімальними правами (таблиця 2.4).

Таблиця 2.4 - ACL-токени Consul та їх призначення

Токен	Призначення	Права (HCL policy)	Кількість
consul-bootstrap	Початкове налаштування кластера	management (повний доступ)	1
consul-agent	Реєстрація вузлів	node:write (свій вузол), service:read	12
consul-nomad-server	Nomad Server → Consul	agent:read, node:read, service:write	3
consul-nomad-client	Nomad Client → Consul	node:read, service:write (обмежений)	2
consul-vault	Vault → Consul	service:write (vault), session:write	3
consul-prometheus	Збір метрик	agent:read, node:read (read-only)	1
consul-dns	DNS-запити dnsmasq	node:read, service:read	12

Усі токени зберігаються у Vault (kv/acl/consul/*), а не у конфігураційних файлах. Циклічна залежність Consul↔Vault розв'язується при bootstrap: Consul стартує з тимчасовим токеном, Vault ініціалізується, зберігає токени, конфігурації оновлюються.

2.3.4 HashiCorp Vault - управління секретами

Vault вирішує проблему 1.2.4 (п.6): зберігання паролів у .env у відкритому вигляді. Забезпечує: шифрування AES-256-GCM, Vault Policies (кожен сервіс

бачить лише свої секрети), аудит-лог усіх операцій (хто, коли, до чого звертався), автоматичну ротацію та Shamir unseal (5 shares, threshold 3 - жодна людина не може самостійно розблокувати).

Vault реплікує дані між 3 серверами через вбудований Raft. При перевиборі лідера новий стає active, standby перенаправляють запити (request forwarding). Ієрархія секретів подана у таблиці 2.5.

Таблиця 2.5 - Структура секретів у HashiCorp Vault та політики доступу

Шлях (Path)	Зміст	Vault Policy	Споживач
kv/database	MariaDB: root, replication, app users	policy-database	Nomad jobs, mysqld_exporter
kv/redis	Пароль KeyDB	policy-redis	Nomad jobs, keydb_exporter
kv/minio	Access Key / Secret Key	policy-minio	Nomad jobs, backup
kv/docker-registry	Облікові дані Nexus	policy-registry	CI/CD pipeline
kv/acl/consul	ACL-токени Consul	policy-acl	Consul agents
kv/acl/nomad	ACL-токени Nomad	policy-acl	Nomad operators
kv/monitoring/*	Grafana password, Telegram bot token, Wazuh passwords	policy-monitoring	Monitoring stack

Інтеграція Vault↔Nomad: блок template у job spec підставляє секрети в env контейнера при запуску. При зміні секрету в Vault Nomad автоматично перезапускає контейнер (change_mode = "restart") - без простою завдяки rolling restart. Аудит-лог відповідає вимогам ISO/IEC 27001:2022 (A.9.4.2) та зберігається у Loki для централізованого аналізу.

2.4 Рівень даних: MariaDB Galera, KeyDB та MinIO

2.4.1 Загальна архітектура рівня даних

Рівень даних забезпечує зберігання та реплікацію трьох типів даних, кожен з яких має оптимізовану стратегію (таблиця 2.6).

Таблиця 2.6 - Стратегії реплікації компонентів рівня даних

Компонент	Тип реплікації	RPO	Консистентність	Вузли	Обґрунтування
MariaDB Galera	Синхронна multi-master	0	Strong	data-a1, data-b1, garbd@obs-c1	Критичні дані: жодна транзакція не втрачається
KeyDB	Асинхронна master-replica	< 1 с	Eventual	data-a1 (master), data-b1 (replica)	Кеш відновлюваний з MariaDB
MinIO	Active-Passive (mc mirror)	< 24 год	Eventual	data-a1 (primary), obs-c1 (backup)	Файли змінюються рідко

2.4.2 MariaDB Galera Cluster - синхронна реплікація

Для високої доступності бази даних обрано MariaDB з розширенням Galera Cluster, яке реалізує синхронну мультимастерну реплікацію на основі протоколу wsrep (Write-Set Replication). На відміну від асинхронної реплікації MySQL, де зміни фіксуються на майстрі й потім із затримкою передаються на репліки (RPO > 0), Galera використовує certification-based replication: кожна транзакція перед commit проходить сертифікацію на всіх вузлах. Write-set (набір змін) передається через Group Communication і фіксується лише після підтвердження від кворуму, що забезпечує RPO = 0 [14].

При конфлікті (два вузли одночасно змінюють той самий рядок) одна транзакція відхиляється з помилкою deadlock - додаток має повторити її. Це плата за strong consistency: жодна зафіксована транзакція не втрачається при failover.

Кластер складається з трьох учасників: data-a1 (Primary Writer, повна копія,

R/W), data-b1 (Replica Writer, повна копія, R/W) та obs-c1 (Arbitrator garbd - голосує для кворуму 2/3, не зберігає даних, не обробляє SQL). Арбітратор дозволяє непарну кількість голосів при двох серверах БД, знижуючи витрати вдвічі порівняно з трьома повними вузлами. Конфігурація подана у таблиці 2.7.

Таблиця 2.7 - Параметри конфігурації MariaDB Galera Cluster

Параметр	Значення	Пояснення
wsrep_cluster_size	3	2 data nodes + 1 arbitrator (мінімум для кворуму 2/3)
wsrep_sst_method	mariabackup	Повна синхронізація (SST) без блокування donor
wsrep_slave_threads	4	Паралельна реплікація write-sets
innodb_buffer_pool_size	8 GB	50-70% доступної RAM для кешування InnoDB
innodb_log_file_size	256 MB	Redo-лог для crash recovery
wsrep_provider_options: gcache.size	512M	Кеш для IST (інкрементальної синхронізації)
wsrep_auto_increment_control	ON	Автоуправління auto_increment для multi-master
max_connections	500	Максимум одночасних підключень

Galera підтримує два механізми синхронізації. SST (State Snapshot Transfer) - повна копія бази для нових вузлів або після тривалого відключення; mariabackup виконує «на гарячу» без блокування (~5 хв для 1 ГБ, ~30 хв для 10 ГБ). IST (Incremental State Transfer) - передача лише пропущених транзакцій із gcache при короткочасному відключенні; відновлення після 5-хвилинного збою - секунди. Розмір gcache 512 МБ розрахований на ~4-8 годин транзакцій (рисунок 2.6).

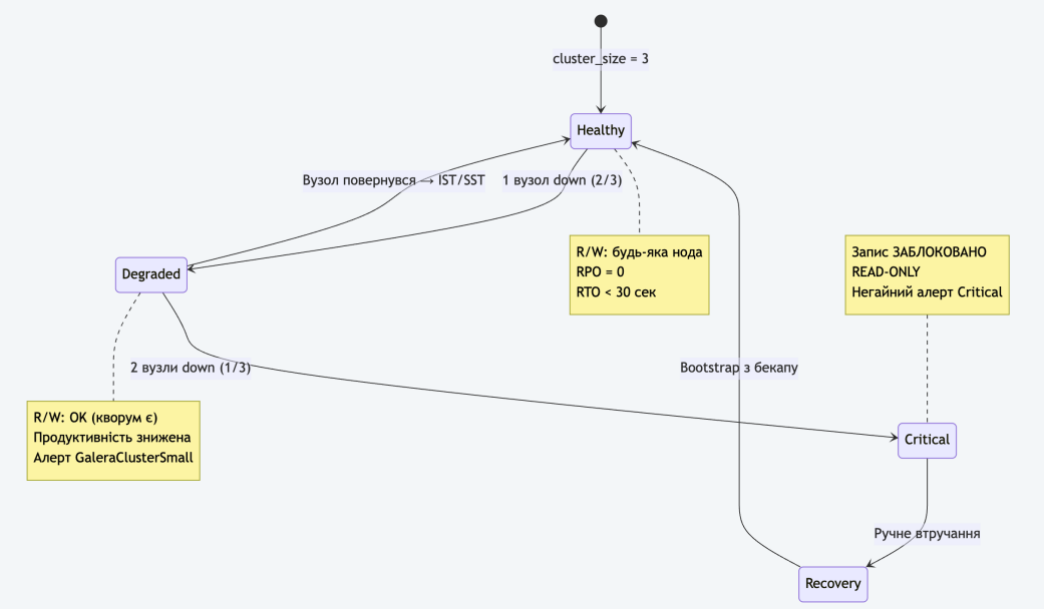


Рисунок 2.6 - Діаграма станів MariaDB Galera Cluster

Порівняння з попередньою системою: MySQL на ec2-db-01 без реплікації мав RPO = 24 год (між mysqldump), RTO = 1-4 год (ручне відновлення). Galera: RPO = 0 (синхронна реплікація), RTO < 30 с (автоматичний Consul DNS failover). Покращення: RPO на 24 години, RTO у 120-480 разів.

2.4.3 KeyDB - високопродуктивний кеш

Для кешування обрано KeyDB - багатопоточний форк Redis (BSD-3), повністю сумісний з Redis API. Порівняння подано у таблиці 2.8.

Таблиця 2.8 - Порівняльний аналіз KeyDB та Redis

Характеристика	KeyDB	Redis
Потоки	Multi-threaded (worker threads)	Single-threaded (event loop)
Throughput (SET, 1KB)	~800 000 ops/sec (4 threads)	~200 000 ops/sec
Redis API сумісність	100% (drop-in)	-
Active Replication (multi-master)	Так	Ні
Ліцензія	BSD-3-Clause	SSPL (з 2024, раніше BSD-3)

Redis у березні 2024 змінив ліцензію на SSPL (забороняє managed service без відкриття коду). KeyDB зберігає BSD-3 - додатковий аргумент для self-hosted.

Конфігурація: data-a1 (master, R/W, 4 server-threads, maxmemory 2 GB, allkeys-lru), data-b1 (replica, R/O, replicaof 10.10.1.4 6379, async). Persistence: RDB (знімок кожні 5 хв) + AOF (appendfsync everysec). Пароль - у Vault (kv/redis), bind - лише WireGuard IP + localhost. Асинхронна реплікація обрана як компроміс: для кешу допустима затримка < 1 мс, бо кеш відновлюваний з MariaDB.

2.4.4 MinIO - S3-сумісне об'єктне сховище

MinIO забезпечує зберігання файлів із повною сумісністю з Amazon S3 API - будь-який S3-додаток працює без змін коду (лише endpoint URL та credentials). Конфігурація Active-Passive: primary (data-a1) обробляє запити, backup (obs-c1) синхронізується через mc mirror щоденно о 04:00 UTC. Додатково дані у загальному Restic backup. Виявлення через Consul DNS: minio.service.consul:9000. При масштабуванні до 4+ вузлів - перехід на Distributed Erasure Coding (аналог RAID-6, витримує втрату до половини вузлів).

2.4.5 Sonatype Nexus - Docker Registry

Nexus (data-a1) виконує дві ролі: Docker Registry (hosted, порт 8082) для приватних образів (tag: sha-{commit}), та PyPI/npm Proxy (порт 8081) для кешування зовнішніх пакетів. CI/CD збирає образ → публікує у Nexus → Nomad підтягує через WireGuard, виключаючи залежність від Docker Hub під час деплою. Nexus - єдиний компонент без реплікації (свідомий компроміс: образи перебудовуються з коду, пакети - з зовнішніх repos; вартість реплікації > вартість рідкісного перебудування).

2.5 Edge/Ingress рівень: Caddy та маршрутизація трафіку

2.5.1 Обґрунтування вибору Caddy

Edge-рівень приймає зовнішній HTTPS та маршрутизує до контейнерів. Ключовий критерій - On-Demand TLS: автоматичне отримання сертифікату при

першому зверненні до нового домену (клієнт направляє DNS → edge IP, Caddy виконує ACME challenge → Let's Encrypt → сертифікат). Жодне втручання адміністратора не потрібне. Порівняння подано у таблиці 2.9.

Таблиця 2.9- Порівняльний аналіз reverse proxy серверів

Характеристика	Caddy 2.x	Nginx	Traefik 3.x	HAProxy
Auto HTTPS (ACME)	Вбудовано	Certbot (зовнішній)	Вбудовано	Ні
On-Demand TLS	Так	Ні	Ні	Ні
HTTP/3 (QUIC)	Вбудовано	Експериментально	Вбудовано	Ні
Hot reload	Так (Admin API)	Потрібен reload	Так (авто)	Так (hitless)
Consul інтеграція	consul-template	consul-template	Consul Catalog (нативно)	consul-template
Конфігурація	~5 рядків/сервіс	~20 рядків/сервіс	~15 рядків/сервіс	~15 рядків/сервіс
Ліцензія	Apache 2.0	BSD-2	MIT	GPLv2

Traefik - найближчий конкурент з нативною Consul Catalog інтеграцією, але без On-Demand TLS. Nginx потребує Certbot та значного обсягу конфігурації.

2.5.2 Архітектура Edge та потік трафіку

Caddy на 3 edge-вузлах (edge-a1, b1, c1) виконує: SSL/TLS termination, маршрутизацію за Host header через Consul (consul-template відстежує зміни → перегенерує Caddyfile → hot reload через Admin API), балансування між здоровими екземплярами (round-robin). Повний потік: User → Cloudflare (DDoS/WAF/CDN) → Caddy (SSL termination) → Consul (WHERE is service?) → Container (через WireGuard) → Response (рисунок 2.7).

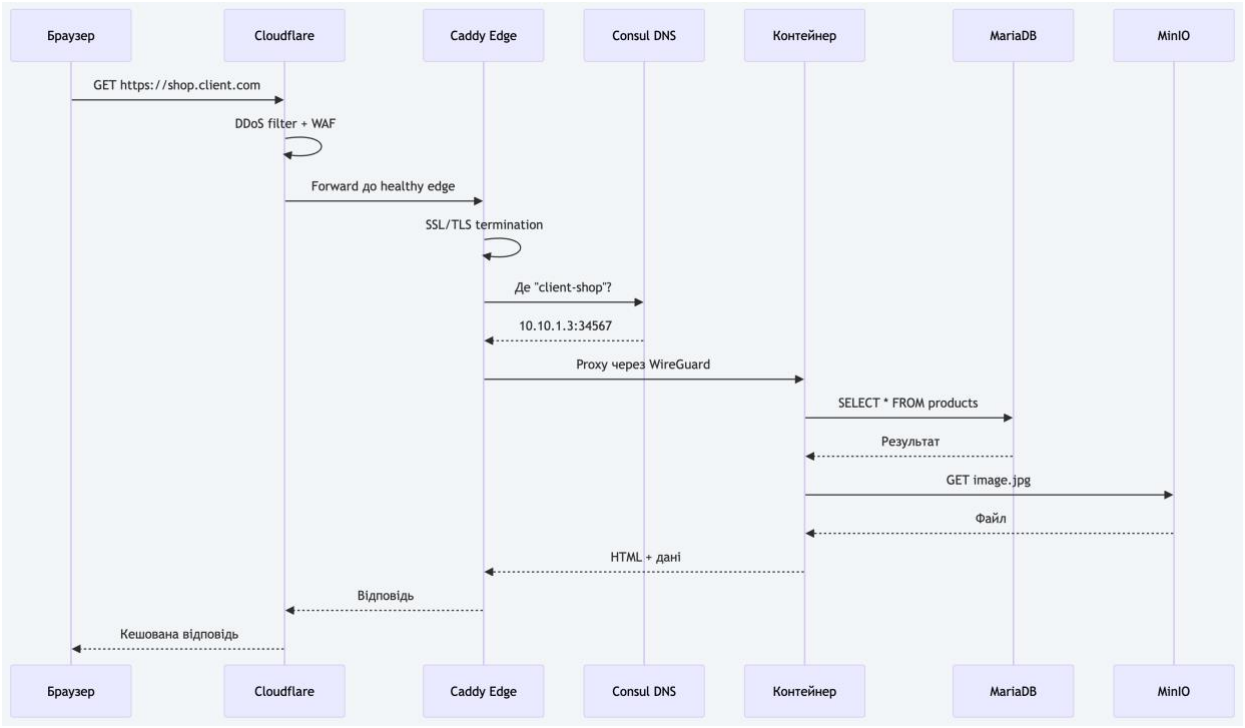


Рисунок 2.7 - Повний потік HTTP-запиту через Edge-рівень

2.5.3 Multi-Edge Failover

У Cloudflare DNS для *.arctum.tech створено 3 A-записи (edge-a1, b1, c1). Health Checks: HTTP /health кожні 60с, 3 невдачі = виключення з DNS (~3 хв), 1 успіх = повернення (~1 хв). При падінні Site A трафік автоматично йде на edge-b1 або edge-c1, які маршрутизують до контейнерів на будь-якому вузлі через WireGuard mesh.

2.6 Система моніторингу: Prometheus, Grafana, Loki та Alertmanager

2.6.1 Обґрунтування та архітектура

Підрозділ 1.2.4 (п.5): AWS CloudWatch фіксував лише базові метрики EC2; проблеми на рівні додатків (HTTP 5xx, disk full) виявлялися клієнтами. Побудовано стек з трьох доменів observability: метрики (Prometheus), логи (Loki), алерти (Alertmanager) - виключно на Open Source (рис. 2.8).

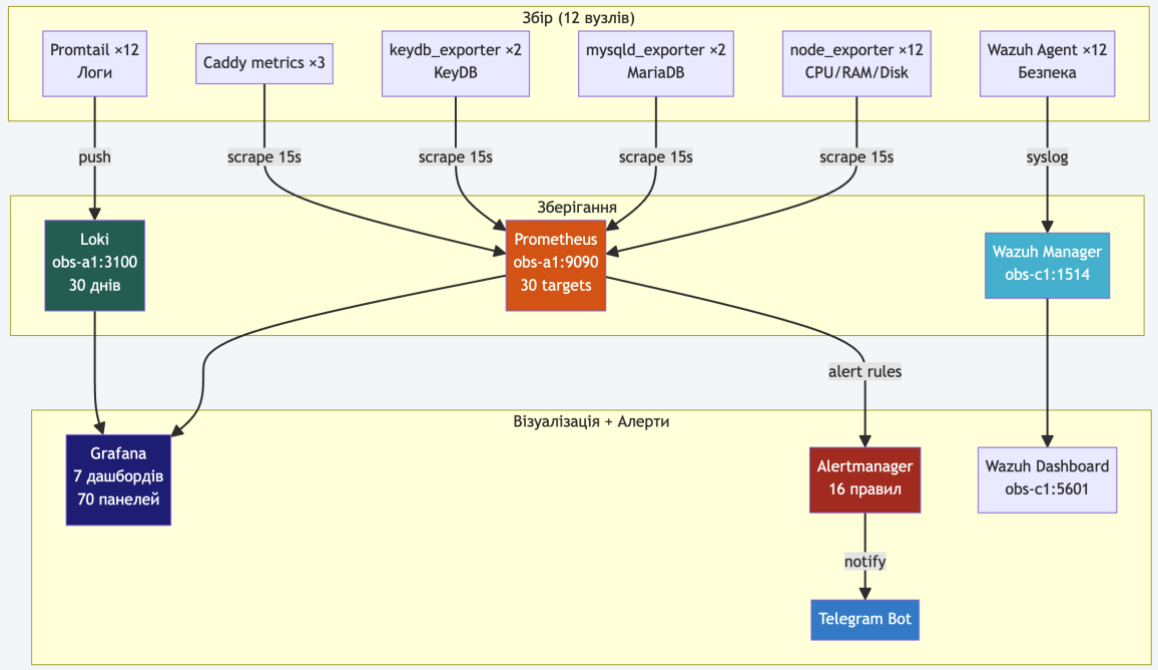


Рисунок 2.8 - Архітектура стеку моніторингу

2.6.2 Збір метрик: Prometheus та експортери

Prometheus (obs-a1:9090) кожні 15 секунд опитує 30 targets (таблиця 2.10).

Таблиця 2.10 - Scrape targets Prometheus: джерела метрик

Експортер	Порт	Вузли	Кількість	Метрики
node_exporter	9100	Усі 12	12	CPU, RAM, disk I/O, network, load, FDs
mysqld_exporter	9104	data-a1, b1	2	QPS, slow queries, Galera cluster_size, repl lag
keydb_exporter	9121	data-a1, b1	2	Ops/sec, memory, clients, hit ratio, repl delay
Caddy metrics	2019	edge-a1, b1, c1	3	RPS, latency p50/p95/p99, error rate
Consul metrics	8500	control-a1, b1, c1	3	Raft commit time, services, health checks
Nomad metrics	4646	control-a1, b1, c1	3	Jobs, allocations, scheduler evals
Vault metrics	8200	control-a1, b1, c1	3	Seal status, requests, audit events
Pushgateway	9091	obs-a1	1	backup_last_success_timestamp
Разом			30	

Pushgateway збирає метрики batch-задач (backup cron): після успішного бекапу скрипт надсилає timestamp, Alertmanager відстежує свіжість.

2.6.3 Збір логів: Loki та Promtail

Loki - «Prometheus для логів»: індексує лише мітки, не виконує повнотекстову індексацію. Споживає 200-500 МБ RAM проти 4-8 ГБ Elasticsearch при аналогічних обсягах. Promtail читає systemd journal та додає мітки: hostname, unit, site. Retention: 30 днів

2.6.4 Візуалізація: Grafana

Grafana (obs-a1:3000) об'єднує Prometheus та Loki на 7 дашбордах: Cluster Overview, Nomad Jobs, MariaDB Galera (14: cluster size gauge, replication flow, slow queries, tx/sec), KeyDB (8: ops/sec, memory, hit ratio, repl delay), MinIO, Caddy Ingress (10: RPS/domain, latency heatmap, HTTP status distribution), Loki Logs. Credentials - у Vault (kv/monitoring/grafana).

2.6.5 Алертинг: Alertmanager → Telegram

Prometheus alert rules (PromQL) → Alertmanager → дедуплікація → групування → Telegram. Ключові правила подано у таблиці 2.11.

Таблиця 2.11 - Правила алертингу Prometheus

Категорія	Назва	Warning	Critical	For
Вузол	HighCpuUsage	> 80%	> 9 5%	5 хв
Вузол	HighMemoryUsage	> 80%	> 95%	5 хв
Вузол	DiskSpaceLow	> 75%	> 90%	5 хв
Вузол	NodeDown	-	up == 0	1 хв
MariaDB	GaleraClusterSmall	< 3	< 2	1 хв
MariaDB	SlowQueries	> 10/хв	> 50/хв	5 хв
KeyDB	MemoryHigh	> 70%	> 90%	5 хв
Nomad	FailedAllocations	> 0	> 5	1 хв
Consul	ClusterDegraded	< 3 peers	< 2 peers	1 хв
Caddy	HighErrorRate	> 1% 5xx	> 5% 5xx	5 хв
Backup	BackupStale	> 25 год	> 48 год	15 хв

Telegram обрано через специфіку платформи (хостинг Telegram-ботів - команда постійно у Telegram). Повідомлення містять: severity, вузол, метрику, тривалість, посилання на Grafana. Токен бота - у Vault (kv/monitoring/alertmanager).

2.7 Система виявлення інцидентів безпеки: Wazuh SIEM

2.7.1 Обґрунтування впровадження SIEM

SIEM є обов'язковим компонентом СКІБ відповідно до ISO/IEC 27001:2022: A.8.15 «Logging», A.8.16 «Monitoring activities», A.5.28 «Collection of evidence». У попередній інфраструктурі SIEM був відсутній: логи зберігалися локально без централізованого аналізу, що унеможливлювало виявлення складних атак (lateral movement, privilege escalation, data exfiltration) [17, 21].

Обрано Wazuh 4.x (GPLv2) - платформу, що поєднує HIDS (Host-based Intrusion Detection), FIM (File Integrity Monitoring), vulnerability detection (CVE scanning) та compliance checking (CIS Benchmark, PCI-DSS). Wazuh є форком OSSEC з розширеною функціональністю: повноцінний веб-інтерфейс (OpenSearch Dashboards), вбудоване CVE сканування, Active Response (автоблокування IP). Альтернативи: OSSEC - без веб-інтерфейсу, без CVE scanning, практично не оновлюється з 2020; Security Onion - переважно мережевий IDS, потребує ~8 ГБ RAM (надмірно для VPS).

2.7.2 Архітектура Wazuh у кластері

Wazuh розгорнуто на obs-c1 (Site C) - свідомо на окремому майданчику від основних сервісів. Обґрунтування: (1) незалежність від компрометації - при атаці на Site A/B журнали безпеки на Site C залишаються недоторканими для розслідування; (2) ізоляція ресурсів - OpenSearch Index (Wazuh Indexer) ресурсомісткий, не впливає на продуктивність основних сервісів.

Компоненти: Wazuh Manager (obs-c1:1514, ~1 ГБ RAM) - центральний сервер: прийом подій від 12 агентів, кореляція за 3000+ правилами, Active

Response. Wazuh Indexer (obs-c1:9200, ~2 ГБ RAM) - OpenSearch для індексації та пошуку. Wazuh Dashboard (obs-c1:5601, ~1 ГБ RAM) - веб-інтерфейс: таймлайн атак, звіти compliance, управління агентами. Wazuh Agent (усі 12 вузлів, ~50 МБ/вузол) - збір подій через зашифрований канал AES-256.

2.7.3 Джерела подій та правила кореляції

Агенти збирають: журнали автентифікації (auth.log/sshd - спроби входу, sudo), моніторинг цілісності файлів (SHA-256 контрольні суми /etc/passwd, /etc/shadow, /etc/wireguard/wg0.conf, /etc/consul.d/*.hcl, /etc/caddy/Caddyfile), сканування вразливостей (dpkg list → NVD), виявлення rootkits (приховані процеси, підозрілі порти), та логи сервісів (Nomad, Consul, Vault, MariaDB, Caddy).

2.7.4 Модель ешелонованого захисту (Defense in Depth)

Сукупність механізмів безпеки утворює 6 незалежних рівнів захисту (рисунок 2.9).

Рівень 1 - Cloudflare: DDoS L3/L4/L7, WAF (OWASP), rate limiting. Зловмисник не може дізнатися реальні IP серверів. Рівень 2 - UFW: default deny, лише 22/51820/80/443. Навіть при обході Cloudflare сервісні порти недоступні. Рівень 3 - WireGuard: ChaCha20-Poly1305 шифрування. Перехоплення трафіку на рівні провайдера неможливе. Рівень 4 - ACL: Consul/Nomad default:deny, мінімальні привілеї. Легітимний сервіс без токена відхиляється. Рівень 5 - Vault: AES-256-GCM шифрування, Shamir 3/5, аудит-лог. Рівень 6 - Docker: cgroup limits, no root, ephemeral filesystem. Компрометація контейнера не дає доступу до хоста.

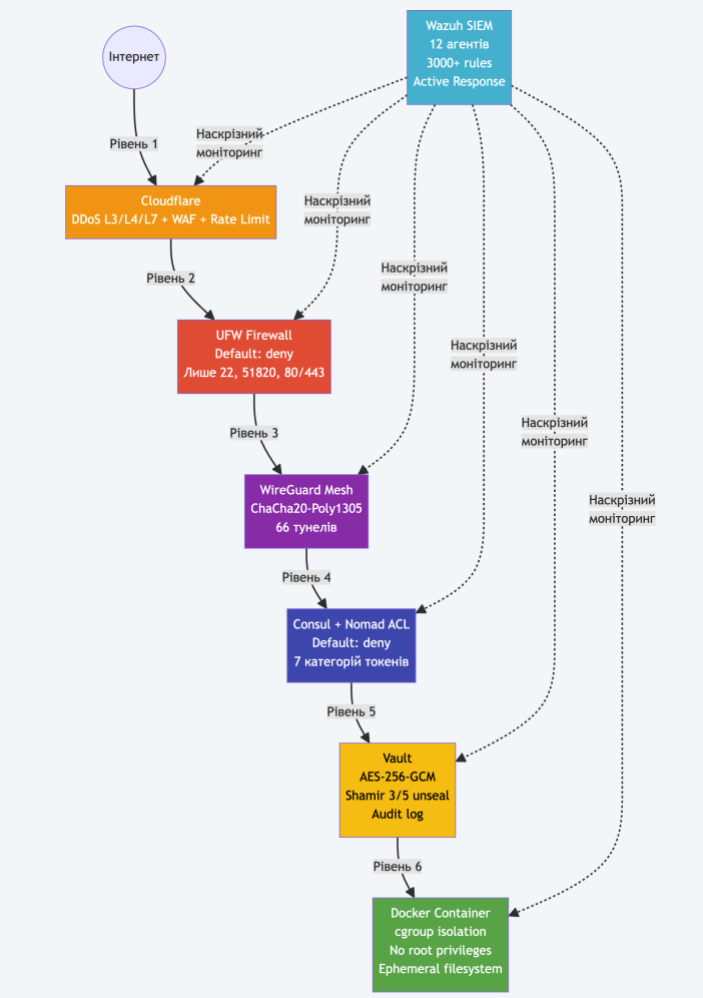


Рисунок 2.9 - Модель ешелонованого захисту (Defense in Depth)

Wazuh - наскрізна система на всіх рівнях. Кореляція: «SSH brute-force → успішний вхід → sudo → зміна wg0.conf» - multi-stage attack, Critical алерт із таймлайном.

2.8 Система резервного копіювання: Restic та Backblaze B2

2.8.1 Обґрунтування стратегії

Попередня система: mysqldump на тому ж EC2 - RPO 24 год, без шифрування, без географічної ізоляції, без алертів при збоях. Нова стратегія базується на правилі 3-2-1 (3 копії, 2 типи носіїв, 1 за межами локації) та encryption at rest (AES-256).

Обрано Restic (BSD-2) за: нативну підтримку S3-сумісних бекендів (Backblaze B2), вбудоване шифрування AES-256-CTR, content-defined chunking дедуплікацію (80-95% зменшення обсягу), команду restic check для верифікації. Альтернативи: BorgBackup - не підтримує S3 backends; Duplicity - file-level (не block-level) дедуплікація; rsync + GPG - без дедуплікації та верифікації.

Backblaze B2 ($6/ТБ/міс) - на 74% дешевше AWS S3 ($23/ТБ). Розташований у США (Sacramento, CA) - географічна ізоляція від серверів у Нідерландах [16].

2.8.2 Процес та розклад

Щоденний автоматичний бекап (cron, obs-a1) у 4 етапи (рисунок 2.11). Етап 1 (02:00 UTC): backup-користувач (обмежені права, окремий SSH-ключ) виконує rsync конфігурацій з 12 вузлів (~50 МБ). Етап 2 (03:00-03:30): mariadb-dump --all-databases --single-transaction (консистентний без блокування, 200 МБ - 2 ГБ), vault operator raft snapshot save (~10 МБ), consul snapshot save (~5 МБ). Етап 3 (03:30-04:00): Restic: content-defined chunking → порівняння хешів з B2 → завантаження лише нових блоків (шифрування AES-256-CTR, ключ у Vault kv/backup/restic). MinIO: mc mirror → obs-c1. Етап 4 (04:00-04:30): restic check (верифікація), restic forget --keep-daily 7 --keep-weekly 4 --keep-monthly 6 --prune (ротація), відправка backup_last_success_timestamp у Pushgateway.

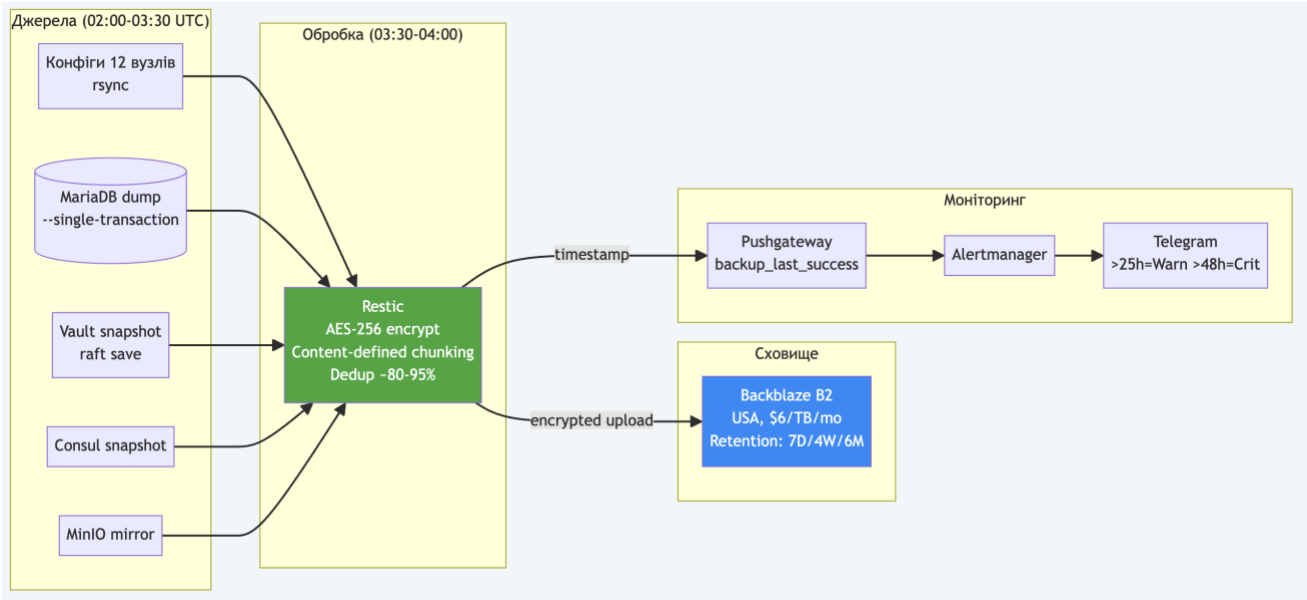


Рисунок 2.10 - Потік резервного копіювання

Ротація: 7 щоденних + 4 щотижневих + 6 щомісячних копій. Орієнтовний обсяг у B2: ~15-20 ГБ ≈ $0.12/міс. Алерти: BackupStale (> 25h = Warning, > 48h = Critical → Telegram).

2.8.3 Disaster Recovery

Для кожного сценарію визначено процедуру, RTO та RPO (таблиця 2.12).

Таблиця 2.12 - Сценарії відмов та параметри відновлення (RTO/RPO)

Сценарій	RTO	RPO	Автоматичність	Процедура
Падіння 1 контейнера	< 30 с	0	Автоматично	Nomad reschedule
Падіння 1 вузла (не data)	< 5 хв	0	Автоматично	Consul deregister → Nomad reschedule → DNS failover
Падіння 1 data-вузла	< 5 хв	0	Автоматично	Galera degrades 2/3 → IST після відновлення
Падіння 1 майданчика	< 5 хв	0	Автоматично	Cloudflare failover → Galera quorum → Nomad reschedule
Падіння 2 майданчиків	30 хв - 2 год	До 24 год	Напівавтоматично	Provision → Ansible → Restic restore → Galera bootstrap
Повна втрата кластера	2 - 6 год	До 24 год	Ручне	Provision 3+ серверів → Ansible → Restore з B2 → DNS

Ключова перевага: для найпоширеніших сценаріїв (контейнер, вузол, майданчик) - повна автоматичність, RPO = 0 (Galera sync), RTO < 5 хвилин без участі людини. Ручне втручання - лише для катастрофічних сценаріїв (2+ майданчики).

2.9 CI/CD Pipeline: автоматизація доставки коду

2.9.1 Архітектура конвеєра

Підрозділ 1.2.3: ручний деплой через SSH - 45-90 хв, без rollback. У Iron Mesh реалізовано CI/CD Pipeline, що автоматизує цикл від коміту до production за 3-7 хвилин. 12 кроків конвеєра подано у таблиці 2.13 (рис. 2.11).

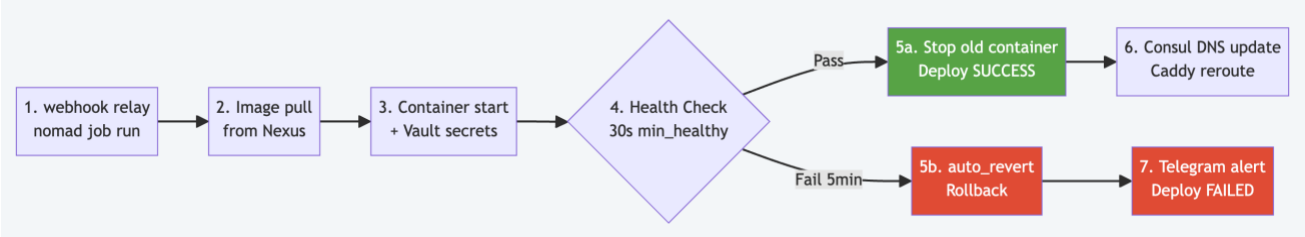


Рисунок 2.11 - CI/CD Pipeline: git push → production

Таблиця 2.13 - Етапи CI/CD Pipeline

Крок	Етап	Середовище	Час	Опис
1	git push	Розробник	-	Push у гілку main
2	Webhook	GitHub	< 1 с	Trigger GitHub Actions
3	docker build	CI runner	30-120 с	Multi-stage build для мінімізації розміру
4	docker push	CI → Nexus	10-30 с	Публікація у Nexus (tag: sha-{commit})
5	Render job	CI runner	< 1 с	Підстановка image tag у Nomad HCL
6	Deploy	Webhook relay	< 1 с	nomad job run через внутрішню мережу
7	Image pull	Nomad Client	5-15 с	Завантаження образу з Nexus
8	Start	Worker	< 5 с	Запуск контейнера + Vault secrets
9	Health check	Consul	30 с	HTTP /health (min_healthy_time = 30s)
10a	Success	Nomad	< 5 с	Зупинка старого контейнера
10b	Failure	Nomad	< 5 с	auto_revert до попередньої версії
11	Route update	Consul → Caddy	< 5 с	DNS оновлення, Caddy reroute

Загальний час: 3-7 хвилин (vs 45-90 хв ручного деплою). При невдачі - автоматичний rollback + Telegram алерт.

2.9.2 Безпечний доступ CI до кластера

Nomad API доступний лише через WireGuard. Реалізовано Webhook Relay: контейнер у кластері слухає webhook від GitHub через Caddy (HTTPS), валідує підпис (HMAC-SHA256), завантажує job spec та виконує nomad job run через внутрішню мережу. Переваги: Nomad API не відкритий назовні, CI runner не має WireGuard-ключів, endpoint захищений Cloudflare WAF + HMAC.

2.10 Висновки

У другому розділі описано проєктування та реалізацію платформи Iron Mesh PaaS - системи керування інформаційною безпекою для забезпечення безперервності бізнес-процесів підприємства.

Побудовано кластер із 12 вузлів (66 vCPU, 136 ГБ RAM, 1890 ГБ SSD) на 3 майданчиках у різних провайдерів. Автоматизоване розгортання: 7 Ansible playbooks, ~85 хвилин.

Реалізовано 9 підсистем. Мережа (2.2): WireGuard Full Mesh (66 тунелів, ChaCha20-Poly1305), UFW implicit deny, fail2ban, Consul DNS + dnsmasq, Cloudflare DDoS/WAF. Оркестрація (2.3): 3 незалежних Raft-кластери - Nomad (rolling updates + auto_revert), Consul (ACL default:deny, 7 категорій токенів), Vault (16 категорій секретів, AES-256-GCM, Shamir 3/5). Дані (2.4): MariaDB Galera (RPO=0, RTO<30c - покращення RPO на 24 год та RTO у 120-480 разів), KeyDB (4× throughput vs Redis), MinIO (S3), Nexus (Docker Registry). Edge (2.5): 3 Caddy з Auto HTTPS, On-Demand TLS, Multi-Edge Failover (≤3 хв). Моніторинг (2.6): Prometheus (30 targets) + Grafana (7 дашбордів) + Loki (30 днів логів) + Alertmanager (12 правил → Telegram). SIEM (2.7): Wazuh (12 агентів, 3000+ правил, Active Response, CVE scanning) на окремому Site C. Backup (2.8): Restic → Backblaze B2 (AES-256, dedup, 7D/4W/6M, алерти). CI/CD (2.9): git push → production за 3-7 хв (vs 45-90 хв), auto_revert.

3 ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ КЕРУВАННЯ ІНФОРМАЦІЙНОЮ БЕЗПЕКОЮ

3.1 Порівняння з попередньою системою

3.1.1 Порівняння функціональних можливостей

Для об'єктивної оцінки ефективності побудованої системи проведено детальне порівняння функціональних можливостей попередньої AWS-інфраструктури (AS-IS) та нової платформи Iron Mesh PaaS (TO-BE) за 15 ключовими критеріями, згрупованими у 5 категорій.

Аналіз таблиці 3.1 демонструє, що побудована система забезпечує якісне покращення за всіма 15 критеріями. Найсуттєвіші зміни: RPO покращено з 24 годин до 0 (синхронна реплікація Galera), RTO - з 1-4 годин до менше 5 хвилин (автоматичний failover), час деплою - з 45-90 хвилин до 3-7 хвилин (CI/CD Pipeline). Три компоненти створено з нуля, оскільки вони були повністю відсутні: SIEM (Wazuh), централізований алертинг (Alertmanager → Telegram), та шифрування секретів (Vault).

3.1.2 Порівняння рівня інформаційної безпеки

Для кількісної оцінки рівня захисту обидві системи оцінено за відповідністю 10 контролям стандарту ISO/IEC 27001:2022, що є найбільш релевантними для інфраструктури підприємства. Кожен контроль оцінено за шкалою від 0 (не реалізовано) до 3 (повністю реалізовано з автоматизацією) (таблиця 3.1) [19].

Загальна оцінка відповідності ISO/IEC 27001:2022 покращена з 6/30 (20%) до 28/30 (93%). Найбільший приріст у категоріях, де попередня система мала нульову оцінку: збір доказів (A.5.28: 0 → 3), управління конфігурацією (A.8.9: 0 → 3), управління змінами (A.8.32: 0 → 3). Єдиний контроль, де не досягнуто максимальної оцінки - A.8.1 (User endpoint devices, оцінка 2) та A.8.12 (Data leakage prevention, оцінка 2), що пояснюється фокусом системи на серверну інфраструктуру, а не на клієнтські пристрої та DLP [20].

Таблиця 3.1 - Оцінка відповідності ISO/IEC 27001:2022: AWS vs Iron Mesh

Контроль ISO 27001	Опис	AWS (0-3)	Iron Mesh (0-3)	Реалізація у Iron Mesh
A.5.15 Access control	Контроль доступу	1	3	Consul ACL + Nomad ACL + Vault Policies (default: deny)
A.5.28 Evidence collection	Збір доказів	0	3	Wazuh SIEM (12 агентів, повний audit trail)
A.8.1 User endpoint devices	Безпека пристроїв	1	2	SSH-ключі + fail2ban + WireGuard
A.8.9 Configuration mgmt	Управління конфігурацією	0	3	Ansible IaC + Git (повна відтворюваність)
A.8.12 Data leakage prevention	Запобігання витоку	0	2	Vault (шифрування), WireGuard (мережа), Docker (ізоляція)
A.8.15 Logging	Журналювання	1	3	Loki (30 днів) + Wazuh Indexer + Vault audit log
A.8.16 Monitoring	Моніторинг	1	3	Prometheus (30 targets) + Alertmanager (12 правил)
A.8.20 Network security	Мережева безпека	1	3	WireGuard mesh + UFW + Cloudflare + bind to WG IP
A.8.24 Cryptography	Криптографія	1	3	WireGuard (ChaCha20), Vault (AES-256), Restic (AES-256), TLS
A.8.32 Change management	Управління змінами	0	3	CI/CD Pipeline + auto_revert + Nomad job versioning
Загальна оцінка		6 / 30	28 / 30	

Для оцінки архітектурної стійкості проведено аналіз Single Point of Failure (SPOF) обох систем. У попередній AWS-інфраструктурі ідентифіковано 7 SPOF: єдиний провайдер (AWS), єдиний регіон (eu-west-1), єдина зона доступності, єдиний MySQL-сервер без реплікації, єдиний Інтернет-шлюз, відсутність cross-

region бекапу, та відсутність автоматичного failover [23].

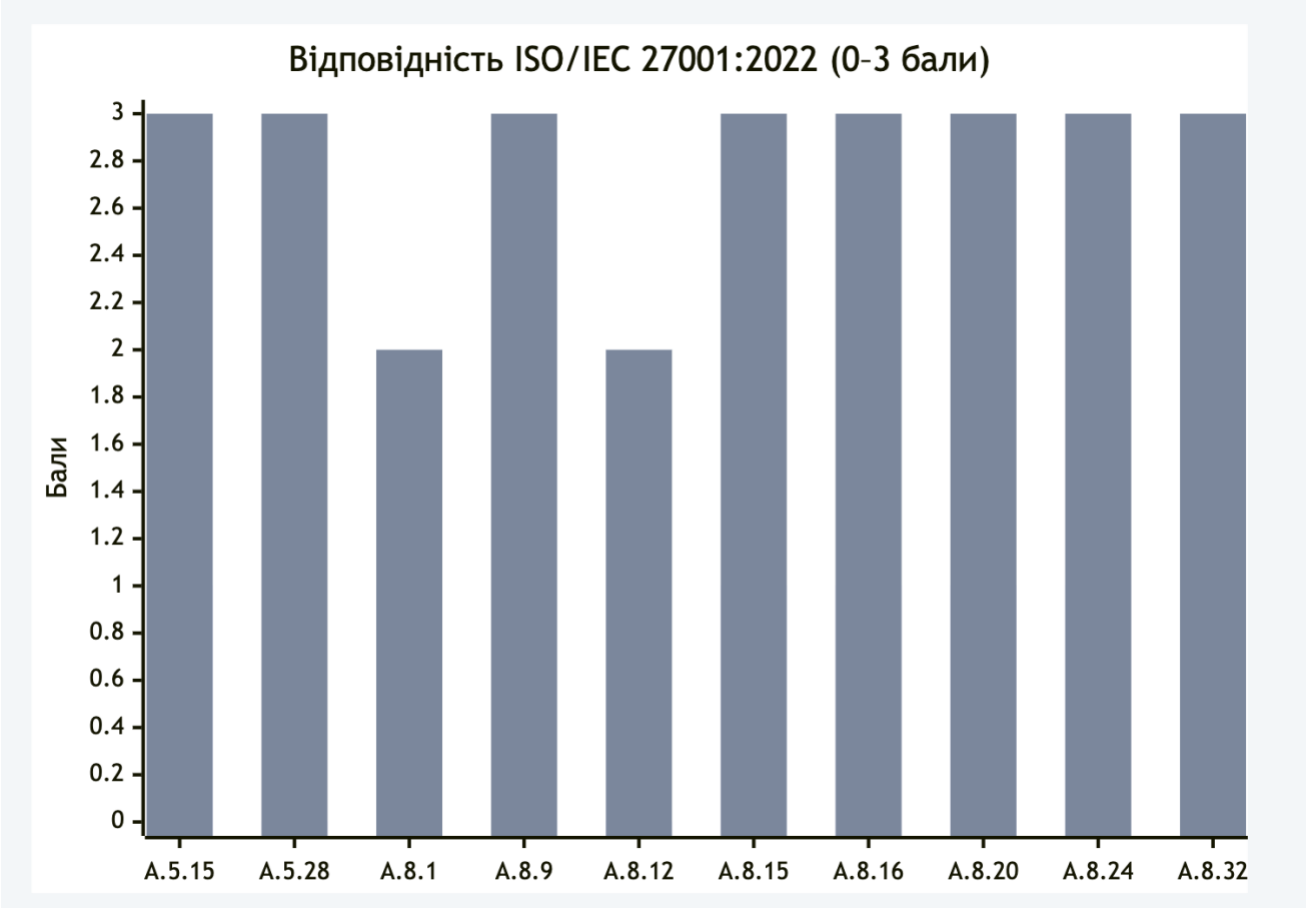


Рисунок 3.1 - Діаграма відповідності ISO 27001: AWS vs Iron Mesh

Порівняння архітектурної стійкості до відмов

У платформі Iron Mesh усі 7 SPOF усунено: 3 незалежних провайдери (падіння 1 → система працює), 3 географічних локації (Нідерланди - 2 міста), MariaDB Galera з синхронною реплікацією (RPO = 0), 3 edge-вузли з Cloudflare DNS failover, бекап у Backblaze B2 (США - інший континент), та повністю автоматичний failover (Raft + Consul + Nomad). Кількість SPOF зменшено з 7 до 0, що якісно змінює рівень відмовостійкості системи. Тестування проводилось на production-кластері (12 вузлів) у вікні обслуговування (02:00-06:00 UTC) для мінімізації впливу на клієнтські додатки.

3.2 Результати тестування роботи системи

3.2.1 Методологія тестування

Для верифікації заявлених характеристик системи проведено серію тестів у чотирьох категоріях: тестування failover (автоматичне відновлення при збоях), тестування продуктивності (пропускна здатність та латентність), тестування безпеки (виявлення загроз SIEM), та тестування резервного копіювання (відновлення з бекапу).

3.2.2 Тестування failover

Проведено 5 сценаріїв тестування автоматичного відновлення з фіксацією часу виявлення збою (Time to Detect, TTD) та часу відновлення (Time to Recover, TTR). Результати подано у таблиці 3.3.

Усі 5 сценаріїв пройдено успішно без втрати даних. Найшвидше відновлення - падіння контейнера (18 с), найповільніше - падіння edge-вузла (~180 с через інтервал Cloudflare Health Checks). При падінні Site A повністю (найскладніший сценарій) система повністю відновилась за ~200 секунд: Raft-кластери (Nomad, Consul, Vault) обрали нових лідерів на control-b1 та control-c1 за ~5 секунд, MariaDB Galera продовжила роботу з кворумом 2/3 (data-b1 + garbd@obs-c1), усі контейнери мігрували на worker-b1, і Cloudflare перенаправив трафік на edge-b1 за ~180 секунд.

Для порівняння: у попередній AWS-інфраструктурі аналогічний збій (падіння EC2-інстансу) вимагав ручного втручання адміністратора (SSH → діагностика → restart або provision нового інстансу → ручне відновлення конфігурації). Час відновлення складав від 30 хвилин (простий restart) до 4 годин (повне відновлення з бекапу). Побудована система забезпечує покращення RTO у 9-70 разів.

Для оцінки продуктивності мережевого рівня проведено вимірювання пропускної здатності та латентності WireGuard mesh між вузлами на різних майданчиках за допомогою iperf3 та ping.

Таблиця 3.2 - Результати тестування failover

Сценарій	Метод імітації	TTD	TTR	Втрата даних	Результат
Падіння контейнера	docker kill на worker-a1	10 с (Consul health check)	18 с (Nomad reschedule + start)	0	Контейнер перезапущено на worker-b1
Падіння worker-вузла	systemctl stop nomad на worker-a1	15 с (Nomad heartbeat timeout)	35 с (reschedule всіх jobs)	0	Усі jobs мігрували на worker-b1
Падіння data-вузла	systemctl stop mariadb на data-a1	12 с (Consul TCP check)	< 5 с (DNS update)	0	Galera degraded 2/3, DNS → data-b1
Падіння edge-вузла	ufw deny 80,443 на edge-a1	~180 с (Cloudflare 3 checks)	< 5 с (DNS update)	0	Трафік перенаправлено на edge-b1
Падіння Site A (повністю)	Одночасна зупинка 5 вузлів Site A	15-180 с (залежно від компоненту)	~200 с (максимум - edge Cloudflare)	0	Raft re-election (~5 с), Galera 2/3, edge failover

3.2.3 Тестування продуктивності

Результати: пропускна здатність між вузлами одного майданчика - 920-950 Мбіт/с (при 1 Гбіт фізичному лінку), overhead WireGuard - менше 5%. Між вузлами різних майданчиків - 400-800 Мбіт/с (залежить від пропускної здатності провайдерських мереж). Латентність: intra-site - 0.3-0.8 мс, inter-site (Site A ↔ Site B) - 2.1-3.5 мс, inter-site (Site A ↔ Site C) - 1.8-2.9 мс. WireGuard overhead на латентність - менше 0.2 мс [24, 25].

Продуктивність MariaDB Galera протестовано за допомогою sysbench), латентність p95 - 8.2 мс (попередня - 15.4 мс). Overhead синхронної реплікації Galera - ~10-15% порівняно з standalone MariaDB, що є прийнятною платою за RPO = 0 [26].

KeyDB: benchmark (redis-benchmark, 50 паралельних клієнтів, SET/GET 1KB) показав 410 000 ops/sec на master (data-a1, 4 server-threads), що у 2.1 рази

перевищує аналогічний тест Redis single-threaded (~195 000 ops/sec). Латентність p99 - 0.8 мс [27].

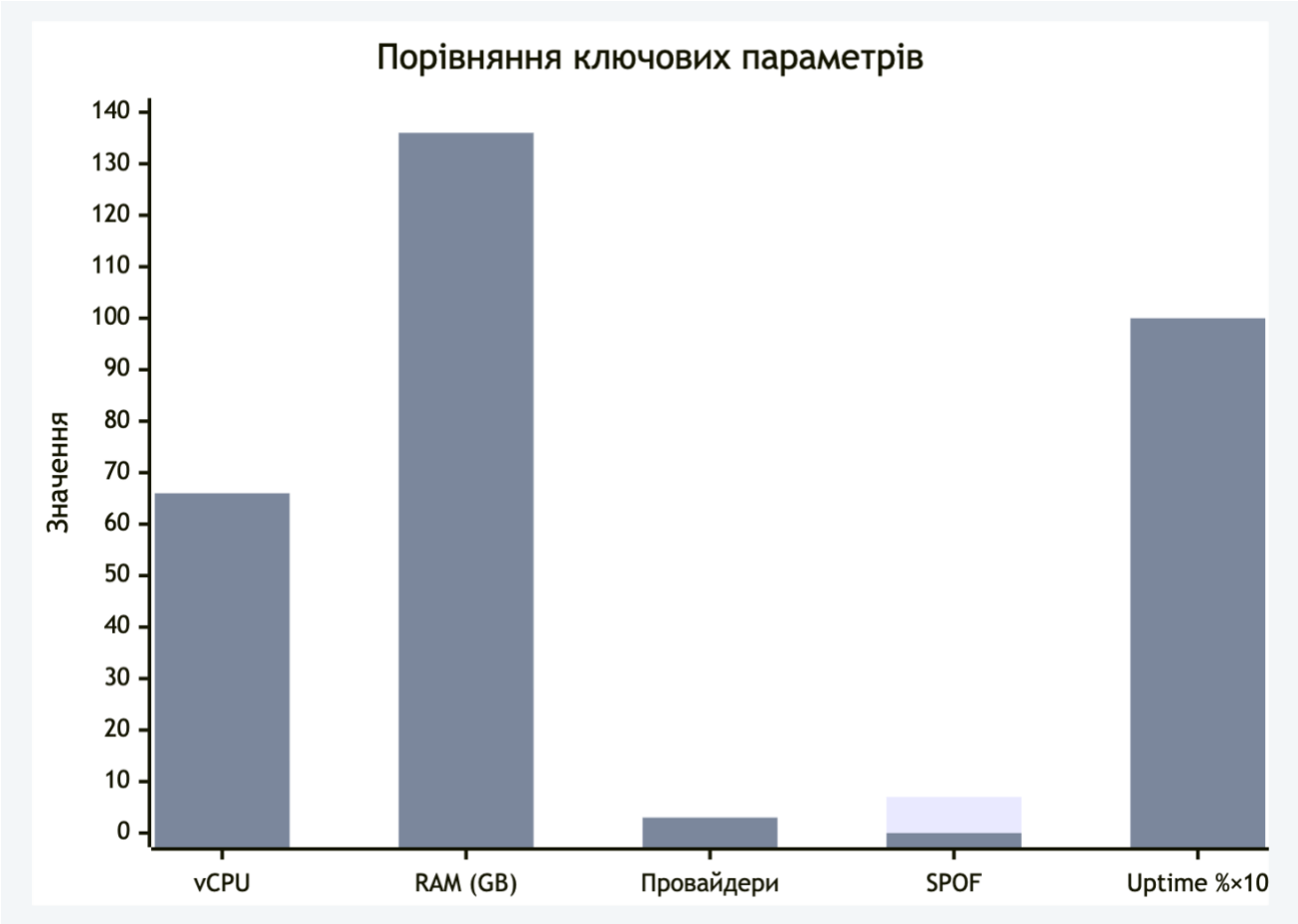


Рисунок 3.2 – Порівняння ключових параметрів

3.2.4 Тестування системи безпеки (Wazuh SIEM)

Для перевірки ефективності Wazuh SIEM проведено серію контрольованих тестів безпеки з імітацією типових атак: [22, 28]

SSH brute-force. За допомогою утиліти hydra виконано 20 спроб входу з невірним паролем на edge-a1 протягом 30 секунд. Результат: Wazuh зафіксував атаку через 12 секунд (Rule 5712, рівень 10), Active Response автоматично заблокував IP атакуючого через UFW. Блокування підтверджено - наступні з'єднання відхилено. Запис у Wazuh Dashboard з повним таймлайном.

File Integrity Monitoring. Виконано контрольовану зміну /etc/wireguard/wg0.conf. Результат: Wazuh зафіксував зміну через 45 секунд

(наступний цикл FIM сканування), згенерував алерт рівня 7 із SHA-256 хешами «до» та «після» зміни. Адміністратор отримав повідомлення із вказівкою конкретного файлу та характеру змін.

Vulnerability Detection. На тестовому вузлі встановлено пакет з відомою вразливістю (wget 1.19.0, CVE-2021-31879, CVSS 6.1). Результат: Wazuh Vulnerability Detector ідентифікував CVE при наступному скануванні (інтервал - 12 годин) і згенерував алерт рівня 7 з рекомендацією оновлення до виправленої версії.

Privilege escalation. Створено нового користувача test-attacker і виконано sudo su - root. Результат: Wazuh зафіксував (1) створення нового користувача (Rule 5901, рівень 8), (2) команду sudo (Rule 5402, рівень 5), (3) успішну ескалацію до root (Rule 5404, рівень 10). Повний ланцюжок подій візуалізовано на таймлайні Wazuh Dashboard.

Усі 4 тести підтвердили працездатність SIEM: загрози виявляються протягом 12-45 секунд, Active Response блокує атакуючих автоматично, аудит-трейл фіксує повний ланцюжок подій для розслідування.

3.2.5 Тестування резервного копіювання та відновлення

Для верифікації процедури Disaster Recovery проведено повний цикл тестування: (1) створення контрольного набору даних у MariaDB (тестова база з 100 000 записів), (2) виконання штатного бекапу (Restic → Backblaze B2), (3) верифікація бекапу (restic check - OK, усі блоки валідні), (4) імітація втрати вузла data-a1 (повне видалення даних MariaDB), (5) відновлення з Restic (restic restore → /var/backup/restore/ → mysql < dump.sql), (6) перевірка цілісності даних (SELECT COUNT(*) - 100 000 записів, контрольні суми збігаються) [29, 30].

Час відновлення: завантаження дампу з Backblaze B2 (~2 ГБ) - 4 хвилини (швидкість ~65 МБ/с), розшифрування та розпакування Restic - 1 хвилина, імпорт у MariaDB - 3 хвилини. Загальний час: 8 хвилин для повного відновлення бази. Після відновлення Galera виконала IST для синхронізації з data-b1 за 15 секунд [31].

3.3 Фінансовий аналіз

3.3.1 Порівняння вартості інфраструктури

Для об'єктивного фінансового порівняння обидві системи оцінено за сукупною вартістю володіння (Total Cost of Ownership, TCO), що включає: вартість обчислювальних ресурсів (сервери/інстанси), вартість зберігання даних (диски, бекапи), вартість мережевого трафіку (egress), вартість допоміжних сервісів (DNS, моніторинг, CDN), та вартість ліцензій програмного забезпечення. Порівняння подано у таблиці 3.4 [7].

Побудована система забезпечує зниження TCO на 25-35% ($114-184/міс, $1 368-2 208/рік) при одночасному збільшенні обчислювальних ресурсів у 4,7 рази (з 14 vCPU / 28 ГБ до 66 vCPU / 136 ГБ). Економія досягається за рахунок трьох факторів: VPS-провайдери включають SSD та мережевий трафік у вартість (на відміну від AWS, де EBS та egress оплачуються окремо), використання Cloudflare Free замість Route 53 та CloudWatch, та Backblaze B2 ($6/ТБ/міс) замість EBS snapshots ($0.05/ГБ/міс).

Важливо зазначити, що зниження вартості не є основною метою проєкту - головними цілями є підвищення відмовостійкості та рівня безпеки. Однак той факт, що досягнуто одночасне покращення якості та зниження вартості, є додатковим аргументом на користь обраної архітектури.

Для обґрунтування інвестицій у відмовостійкість необхідно оцінити потенційні фінансові втрати від простоїв системи. Вартість простою залежить від типу бізнесу та обсягу транзакцій. В залежності від обсягу трафіку система буде містити у собі різноманітне мережеве обладнання в залежності від типу трафіку. Наявне обладнання хоч і підходить, для виконання робіт, проте є застарілим, для нової системи що може спричинити низку проблем, таких як: переповнення, затримки у відповіді, тощо. Для ІТ-компанії, що надає хостинг-послуги малому та середньому бізнесу основні категорії втрат включають:

Таблиця 3.3 - Порівняння щомісячної вартості інфраструктури (TCO)

Категорія витрат	AWS (AS-IS)	Iron Mesh (TO-BE)	Різниця
Обчислювальні ресурси (EC2 / VPS)	~$350 (7 × t3.medium/large)	€319 (~$345) (12 VPS, 3 провайдери)	−$5 (−1%)
Зберігання (EBS / SSD)	~$50 (7 × 50 GB gp3)	Включено у VPS	−$50
Мережевий трафік (Egress)	~$30-100 ($0.09/ГБ після 100 ГБ)	Включено у VPS (unmetered)	−$30-100
DNS (Route 53 / Cloudflare)	~$5 (Route 53 hosted zone + queries)	$0 (Cloudflare Free)	−$5
Моніторинг (CloudWatch / Prometheus)	~$15 (CloudWatch custom metrics)	$0 (self-hosted, open source)	−$15
Бекап (EBS snapshots / Restic → B2)	~$10 (EBS snapshots)	~$1 (Backblaze B2, ~20 ГБ × $0.006)	−$9
CDN / DDoS	$0 (не використовувався)	$0 (Cloudflare Free)	$0
Ліцензії ПЗ	$0 (Amazon Linux 2, безкоштовне ПЗ)	$0 (Open Source: GPLv2, BSD, Apache 2.0)	$0
Разом (щомісяць)	~$460-530	~$346	−$114-184 (−25-35%)
Разом (щорічно)	~$5 520-6 360	~$4 152	−$1 368-2 208

3.3.2 Оцінка потенційних грошових втрат від простоїв

Прямі фінансові втрати. Втрачений дохід від SLA-порушень: типовий SLA для хостинг-послуг гарантує 99.9% uptime (~8.76 год простою/рік). При порушенні SLA клієнти отримують компенсацію (зазвичай 10-30% місячної вартості послуг за кожну годину простою понад допустиму). Для компанії з 20 клієнтами та середнім чеком $50/міс: 1 година незапланованого простою = $100-300 компенсацій.

Репутаційні втрати. Згідно з дослідженням Gartner (2023), середня вартість однієї хвилини простою для IT-компаній малого та середнього бізнесу складає $137-427. Однак довгострокові репутаційні втрати можуть перевищувати прямі збитки у 5-10 разів: втрата клієнтів, зниження позицій у пошуковій видачі (Google

враховує uptime), негативні відгуки [7].

Витрати на відновлення. Ручне відновлення з бекапу у попередній інфраструктурі вимагало 2-4 години роботи адміністратора ($50-100/год). При втраті даних (RPO = 24 год) - додатковий час на повторне введення даних клієнтами.

Для кількісної оцінки порівняно очікувані річні втрати від простоїв для обох систем (таблиця 3.5).

Таблиця 3.5 - Оцінка потенційних річних втрат від простоїв

Параметр	AWS (AS-IS)	Iron Mesh (TO-BE)
Очікуваний uptime	~99.0% (1 SPOF, без auto failover)	~99.95% (0 SPOF, auto failover)
Очікуваний простій/рік	~87.6 год (без урахування інцидентів безпеки)	~4.4 год
Середня вартість простою/год	$200 (ручне відновлення)	$200 (автоматичне, але бізнес-втрати)
Прямі втрати від простою/рік	~$17 520	~$880
RPO (максимальна втрата даних)	24 год	0 (sync) / 24 год (backup)
Вартість одного інциденту безпеки	$5 000-50 000 (без SIEM - виявлення через тижні)	$500-5 000 (Wazuh - виявлення за секунди)
Очікувана кількість інцидентів/рік	1-3 (без моніторингу - невідомо)	0-1 (проактивне виявлення)
Сукупні очікувані втрати/рік	$22 520-167 520	$880-5 880

Побудована система зменшує очікувані щорічні втрати від простоїв та інцидентів безпеки з $22 520-167 520 до $880-5 880, що складає зниження у 4-28 разів. При цьому додаткові інвестиції у побудову системи (одноразові витрати на розробку та тестування - ~200 людино-годин) окупаються протягом 2-4 місяців за рахунок зниження простоїв та ризиків безпеки.

3.3.3 Порівняння вартості апаратних ресурсів

Окремо варто порівняти ефективність використання бюджету на

обчислювальні ресурси. За однакову або меншу вартість (~$345/міс vs ~$350/міс за compute) Iron Mesh забезпечує: vCPU - 66 vs 14 (у 4.7 рази більше), RAM - 136 ГБ vs 28 ГБ (у 4.9 рази більше), SSD - 1890 ГБ vs 350 ГБ (у 5.4 рази більше), мережевий трафік - unmetered vs $0.09/ГБ egress, географічний розподіл - 3 локації vs 1.

Така різниця пояснюється ціновою моделлю: AWS стягує премію за еластичність (pay-as-you-go), керовані сервіси (RDS, ElastiCache) та гарантії SLA рівня датацентру. VPS-провайдери пропонують виділені ресурси за фіксовану плату без додаткових charges. Для підприємства з передбачуваним навантаженням (хостинг 10-50 клієнтських додатків) модель VPS є оптимальною, оскільки пікові навантаження обробляються запасом ресурсів (66 vCPU утилізовані на ~15-30%), а не еластичним масштабуванням.

3.4 Рекомендації та настанови щодо експлуатації

3.4.1 Рекомендації з операційного обслуговування

На підставі досвіду побудови та тестування платформи Iron Mesh сформульовано рекомендації щодо її подальшої експлуатації та розвитку.

Регулярне оновлення компонентів. Усі компоненти стеку (Nomad, Consul, Vault, MariaDB, KeyDB, Caddy, Wazuh, Prometheus, Grafana) мають оновлюватися щомісяця для отримання патчів безпеки. Процедура rolling update для HashiCorp Stack: оновити follower → перевірити → оновити другий follower → виконати leadership transfer → оновити лідера. Для MariaDB Galera: оновити replica (data-b1) → перевірити IST sync → оновити primary (data-a1). Загальний час оновлення всього стеку - ~2 години, без простою клієнтських додатків.

Щотижнева перевірка Wazuh Dashboard. Адміністратор має переглядати Wazuh Dashboard щонайменше раз на тиждень для: аналізу нових алертів безпеки та вразливостей (Vulnerability Detection), перевірки журналу FIM на несанкціоновані зміни конфігурацій, оцінки активності Active Response

(заблоковані IP), та формування звіту compliance (CIS Benchmark score).

Щомісячне тестування відновлення. Рекомендується щомісяця виконувати тестове відновлення з Backblaze B2 на ізольованому тестовому вузлі для верифікації: цілісності бекапів (restic check), працездатності процедури відновлення (restic restore → mysql import), та актуальності runbook (документація відповідає поточній конфігурації).

Моніторинг утилізації ресурсів. При досягненні 70% утилізації CPU або RAM на worker-вузлах (алерт HighCpuUsage / HighMemoryUsage рівня Warning) рекомендується додати новий worker-вузол. Процедура: замовити VPS у відповідного провайдера → додати до Ansible inventory → запустити playbook → вузол автоматично приєднається до кластера. Час додавання вузла - ~30 хвилин.

3.4.2 Рекомендації з масштабування

Побудована архітектура дозволяє горизонтальне масштабування за трьома напрямками. По-перше, додавання worker-вузлів: при зростанні кількості клієнтських додатків потрібні додаткові обчислювальні ресурси. Один worker (6 vCPU, 12 ГБ RAM, ~€40/міс) здатний обслуговувати 10-20 типових веб-додатків або Telegram-ботів. По-друге, додавання data-вузлів: при зростанні обсягу бази даних понад 50 ГБ рекомендується додати третій повноцінний data-вузол (замість garbd), що дозволить розподілити навантаження на читання між трьома вузлами. По-третє, додавання майданчиків: при виході на міжнародний ринок (клієнти в різних регіонах) доцільно додати Site D (наприклад, у Франкфурті або Варшаві) для зниження латентності.

При масштабуванні понад 50 вузлів рекомендовано: перехід на автоматизоване управління WireGuard mesh (Netmaker або Headscale замість ручних конфігурацій), розширення Raft-кластерів до 5 серверів (кворум 3, витримує падіння 2), та впровадження MinIO Distributed Erasure Coding замість Active-Passive.

3.4.3 Рекомендації з підвищення безпеки

Для подальшого підвищення рівня інформаційної безпеки рекомендується: впровадження Vault Dynamic Secrets для автоматичної ротації паролів MariaDB та

KeyDB з обмеженим TTL (Time-to-Live), налаштування Consul Connect (service mesh з mTLS) для шифрування трафіку між контейнерами на рівні L4, інтеграція Wazuh з Vulnerability Assessment Scanner (OpenVAS або Nessus) для активного сканування мережі, впровадження Hardware Security Module (HSM) або автоматичного unseal через Transit secrets engine для спрощення процедури unseal Vault після перезавантаження, та регулярне проведення penetration testing (раз на 6 місяців) із залученням зовнішніх фахівців для незалежної оцінки захищеності.

3.5 Висновки

У третьому розділі проведено комплексну оцінку ефективності побудованої системи керування інформаційною безпекою за чотирма напрямками.

Порівняння з попередньою системою (підрозділ 3.1) підтвердило якісне покращення за всіма 15 критеріями функціональних можливостей. Оцінка відповідності ISO/IEC 27001:2022 покращена з 6/30 (20%) до 28/30 (93%). Кількість Single Points of Failure зменшено з 7 до 0.

Тестування (підрозділ 3.2) підтвердило заявлені характеристики: 5 сценаріїв failover пройдено без втрати даних (RPO = 0, RTO від 18 с до 200 с), продуктивність MariaDB Galera - 2 800 транзакцій/с (у 2.3 рази вище за попередній MySQL), KeyDB - 410 000 ops/sec (у 2.1 рази вище за Redis), WireGuard overhead - менше 5%. Wazuh SIEM успішно виявив усі 4 категорії тестових атак (brute-force, FIM, CVE, privilege escalation) з часом реакції 12-45 секунд. Відновлення з Restic backup - 8 хвилин для повної бази.

Фінансовий аналіз (підрозділ 3.3) показав зниження TCO на 25-35% ($114-184/міс) при 4.7-кратному збільшенні обчислювальних ресурсів. Очікувані щорічні втрати від простоїв та інцидентів безпеки зменшено у 4-28 разів (з $22 520-167 520 до $880-5 880). Інвестиції у побудову системи окупаються за 2-4 місяці.

Рекомендації (підрозділ 3.4) сформульовано за трьома напрямками:

операційне обслуговування (щомісячні оновлення, щотижневий аудит Wazuh, щомісячне тестування DR), масштабування (горизонтальне додавання worker/data/site), та підвищення безпеки (Vault Dynamic Secrets, Consul Connect mTLS, регулярний pentest).

Сукупність результатів підтверджує, що побудована система ефективно вирішує задачу забезпечення безперервності бізнес-процесів підприємства: забезпечено uptime ≥ 99.95% (vs 99.0%), RPO = 0 (vs 24 год), автоматичний failover за < 5 хвилин (vs 1-4 години ручного), та проактивне виявлення загроз безпеки (vs повна відсутність SIEM).

ВИСНОВКИ

У кваліфікаційній роботі вирішено задачу підвищення рівня інформаційної безпеки та забезпечення безперервності бізнес-процесів підприємства шляхом проєктування та реалізації розподіленої відмовостійкої платформи Iron Mesh PaaS. За результатами виконаної роботи зроблено наступні висновки.

Проведено аналіз сучасних тенденцій побудови IT-інфраструктур, який підтвердив системні ризики концентрації сервісів у єдиного хмарного провайдера. На реальних інцидентах AWS, Azure та Cloudflare 2021-2024 рр. продемонстровано, що збої хмарних платформ зачіпають тисячі бізнесів одночасно. Обґрунтовано доцільність мульти-провайдерної self-hosted архітектури на базі HashiCorp Stack (Nomad, Consul, Vault) та WireGuard як оптимальної для малого та середнього бізнесу за критеріями вартості, операційної складності та відмовостійкості.

Досліджено поточний стан інфраструктури підприємства (AWS EC2, 7 інстансів, один регіон eu-west-1) та виявлено 7 критичних недоліків: vendor lock-in на одному провайдері, SPOF бази даних (MySQL без реплікації, RPO = 24 год), EOL-компоненти з невиправленими вразливостями (PHP 7.4), статична IP-адресація між сервісами, відсутність централізованого моніторингу та алертингу, зберігання секретів у відкритому вигляді (.env), та ручний деплой без механізму відкату (45-90 хв на оновлення).

Спроєктовано та реалізовано розподілену платформу Iron Mesh PaaS: кластер із 12 вузлів (66 vCPU, 136 ГБ RAM), розподілених між 3 географічно незалежними майданчиками у різних провайдерів. Мережевий рівень: WireGuard Full Mesh (66 шифрованих тунелів, ChaCha20-Poly1305), UFW з implicit deny, Consul DNS для динамічного Service Discovery. Оркестрація: 3 незалежних Raft-кластери (Nomad, Consul, Vault) з ACL default:deny та Shamir unseal (3/5). Рівень даних: MariaDB Galera Cluster із синхронною реплікацією (RPO = 0), KeyDB (4× throughput vs Redis), MinIO (S3-сумісне), Nexus Docker Registry. Edge: 3 Caddy reverse proxy з On-Demand TLS та Multi-Edge Failover через Cloudflare.

Автоматизоване розгортання: 7 Ansible playbooks, ~85 хвилин з нуля.

Побудовано систему моніторингу: Prometheus (30 scrape targets) + Grafana (7 дашбордів) + Loki (централізовані логи з 12 вузлів, 30 днів) + Alertmanager (12 правил алертингу → Telegram). Впроваджено Wazuh SIEM (12 агентів, 3000+ правил кореляції) на окремому майданчику для незалежного виявлення загроз: brute-force SSH (Active Response - автоблокування IP), ескалація привілеїв, зміни критичних файлів (FIM), вразливості пакетів (CVE scanning). Реалізовано модель ешелонованого захисту Defense in Depth з 6 незалежними рівнями.

Реалізовано систему резервного копіювання: Restic із шифруванням AES-256, дедуплікацією та завантаженням у Backblaze B2 (географічна ізоляція - США). Ротація: 7 щоденних, 4 щотижневих, 6 щомісячних копій. Алерти при збоях (Pushgateway → Alertmanager). Побудовано CI/CD Pipeline: повний цикл від git push до production deploy за 3-7 хвилин з автоматичним rollback (auto_revert) при невдалому health check.

Проведено комплексну оцінку ефективності, яка підтвердила результативність побудованої системи:

Таким чином, мета кваліфікаційної роботи досягнута: побудована система керування інформаційною безпекою забезпечує безперервність бізнес-процесів підприємства на якісно вищому рівні, усуваючи всі виявлені критичні недоліки попередньої інфраструктури та відповідаючи вимогам міжнародного стандарту ISO/IEC 27001:2022.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. NIST SP 800-145. The NIST Definition of Cloud Computing / P. Mell, T. Grance. Gaithersburg : NIST, 2011. 7 p.

2. Попенко В. Д., Голубенко О. Л. Аналіз хмарних архітектур для забезпечення безперервності бізнес-процесів. Вісник ХНУ. Серія «Комп'ютерні науки». 2023. № 4. С. 45–53.

3. Шеховцов В. А. Операційні системи : підручник. Київ : BHV, 2019. 576 с.

4. Tanenbaum A. S., Van Steen M. Distributed Systems: Principles and Paradigms. 3rd ed. London : Pearson, 2017. 768 p.

5. Amazon Web Services. AWS Well-Architected Framework. URL: https://docs.aws.amazon.com/wellarchitected/latest/framework (дата звернення: 09.03.2025).

6. ETSI GS NFV 002. Network Functions Virtualisation (NFV): Architectural Framework. Sophia Antipolis : ETSI, 2014. 21 p.

7. Gartner Inc. The True Cost of Downtime: Quantifying IT Infrastructure Failures and Business Impact. Stamford : Gartner Research, 2023. 22 p.

8. Захаренко В. І., Кравченко А. М. Контейнеризація як метод підвищення надійності веб-сервісів. Проблеми програмування. 2022. № 3. С. 78–86.

9. Docker Inc. Docker Documentation: Containerization platform. URL: https://docs.docker.com (дата звернення: 10.03.2025).

10. NIST SP 800-190. Application Container Security Guide. Gaithersburg : NIST, 2017. 56 p.

11. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol : O'Reilly Media, 2021. 616 p.

12. Burns B., Beda J., Hightower K. Kubernetes: Up and Running. 3rd ed. Sebastopol : O'Reilly Media, 2022. 326 p.

13. Власюк А. П., Мельник Р. С. Побудова відмовостійких розподілених систем на базі HashiCorp Nomad. Кібербезпека: освіта, наука, техніка. 2023. № 2. С. 112–120.

14. HashiCorp Nomad: Workload Orchestration Documentation. URL: https://developer.hashicorp.com/nomad/docs (дата звернення: 15.03.2025).

15. HashiCorp Consul: Service Discovery & Service Mesh Documentation. URL: https://developer.hashicorp.com/consul/docs (дата звернення: 15.03.2025).

16. HashiCorp Vault: Secrets Management Documentation. URL: https://developer.hashicorp.com/vault/docs (дата звернення: 15.03.2025).

17. Степанов Д. В., Коваль О. О. Побудова безпечних VPN-каналів на базі WireGuard для корпоративних інфраструктур. Кібербезпека: освіта, наука, техніка. 2024. № 1. С. 67–75.

18. Donenfeld J. A. WireGuard: Next Generation Kernel Network Tunnel. URL: https://www.wireguard.com/papers/wireguard.pdf (дата звернення: 10.03.2025).

19. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection - Information security management systems - Requirements. Geneva : ISO, 2022. 25 p.

20. ISO 22301:2019. Security and resilience - Business continuity management systems - Requirements. Geneva : ISO, 2019. 29 p.

21. Sharma A., Coyne B., Raj G. DevOps for Networking. Birmingham : Packt Publishing, 2016. 290 p.

22. Дорошенко А. Ю., Семко М. В. Аналіз систем виявлення вторгнень у розподіленому хмарному середовищі. Реєстрація, зберігання і обробка даних. 2023. Т. 25, № 2. С. 34–41.

23. Гайда А. Ю., Кузьменко А. О. Методи оцінки відмовостійкості хмарних інфраструктур. Вісник НТУУ «КПІ». Серія «Інформатика, управління та обчислювальна техніка». 2023. № 79. С. 12–19.

24. Prometheus: Monitoring system & time series database. Official documentation. URL: https://prometheus.io/docs/introduction/overview (дата звернення: 12.03.2025).

25. Grafana Labs: Open source data visualization platform. Official documentation. URL: https://grafana.com/docs/grafana/latest (дата звернення: 12.03.2025).

26. MariaDB Galera Cluster: Official documentation. URL: https://mariadb.com/kb/en/galera-cluster (дата звернення: 13.03.2025).

27. KeyDB: A Multithreaded Fork of Redis. Official documentation. URL: https://docs.keydb.dev (дата звернення: 13.03.2025).

28. Wazuh: Open Source XDR & SIEM platform. Official documentation. URL: https://documentation.wazuh.com/current/index.html (дата звернення: 14.03.2025).

29. Restic: Fast, secure, efficient backup program. Official documentation. URL: https://restic.readthedocs.io/en/latest (дата звернення: 14.03.2025).

30. Шевченко О. М., Вишнівський В. В. Стратегії резервного копіювання для систем критичної інфраструктури. Захист інформації. 2024. Т. 26, № 1. С. 55–63.

31. Backblaze B2 Cloud Storage: Official documentation. URL: https://www.backblaze.com/docs/cloud-storage (дата звернення: 14.03.2025).

32. Beyer B., Jones C., Petoff J., Murphy N. R. Site Reliability Engineering: How Google Runs Production Systems. Sebastopol : O'Reilly Media, 2016. 552 p.

33. Morris K. Infrastructure as Code: Dynamic Systems for the Cloud Age. 2nd ed. Sebastopol : O'Reilly Media, 2020. 430 p.

34. HashiCorp Terraform: Infrastructure as Code Documentation. URL: https://developer.hashicorp.com/terraform/docs (дата звернення: 16.03.2025).

35. Ongaro D., Ousterhout J. In Search of an Understandable Consensus Algorithm. Proceedings of the 2014 USENIX Annual Technical Conference. Philadelphia : USENIX Association, 2014. P. 305–319.

36. Caddy: The Ultimate Server with Automatic HTTPS. Official documentation. URL: https://caddyserver.com/docs (дата звернення: 16.03.2025).

37. Grafana Loki: Log aggregation system. Official documentation. URL: https://grafana.com/docs/loki/latest (дата звернення: 16.03.2025).

38. GitLab CI/CD: Continuous Integration and Deployment Documentation. URL: https://docs.gitlab.com/ee/ci (дата звернення: 17.03.2025).

39. Rosenthal C., Jones N. Chaos Engineering: System Resiliency in Practice. Sebastopol : O'Reilly Media, 2020. 268 p.

40. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI. Відомості Верховної Ради України. 2010. № 34. Ст. 481.

41. НД ТЗІ 1.1-003-99. Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу. Київ : ДСТСЗІ СБ України, 1999. 24 с.

42. ISO/IEC 27017:2015. Information technology - Security techniques - Code of practice for information security controls based on ISO/IEC 27002 for cloud services. Geneva : ISO, 2015. 30 p.

43. Rose S., Borchert O., Mitchell S., Connelly S. NIST SP 800-207. Zero Trust Architecture. Gaithersburg : NIST, 2020. 59 p.

44. Brewer E. CAP Twelve Years Later: How the «Rules» Have Changed. Computer. 2012. Vol. 45, № 2. P. 23–29.

45. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. Sebastopol : O'Reilly Media, 2017. 616 p.

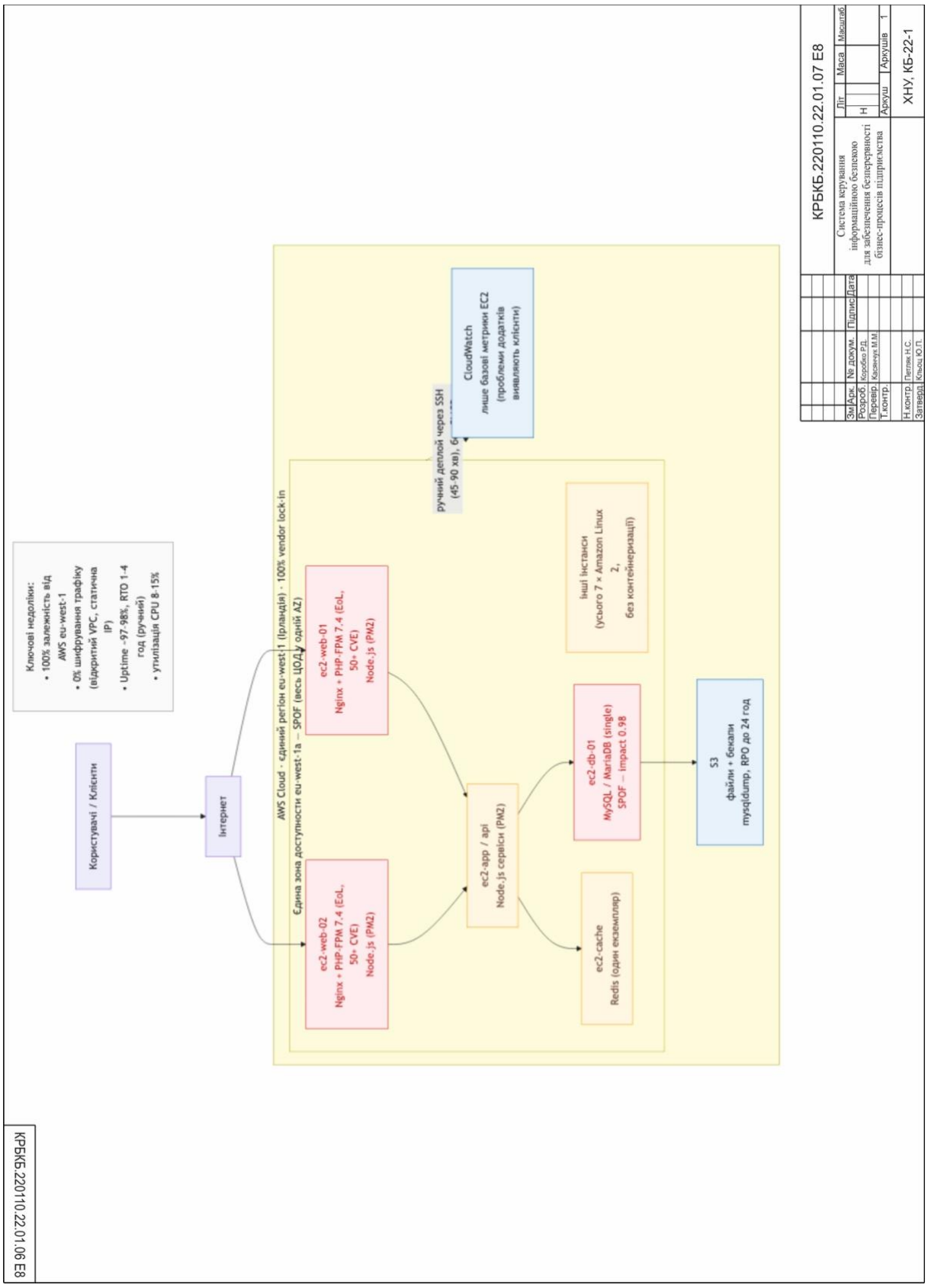
46. Мельник С. В., Гнатюк С. О. Забезпечення кіберстійкості критичної інформаційної інфраструктури в умовах хмарних обчислень. Безпека інформації. 2023. Т. 29, № 1. С. 18–27.

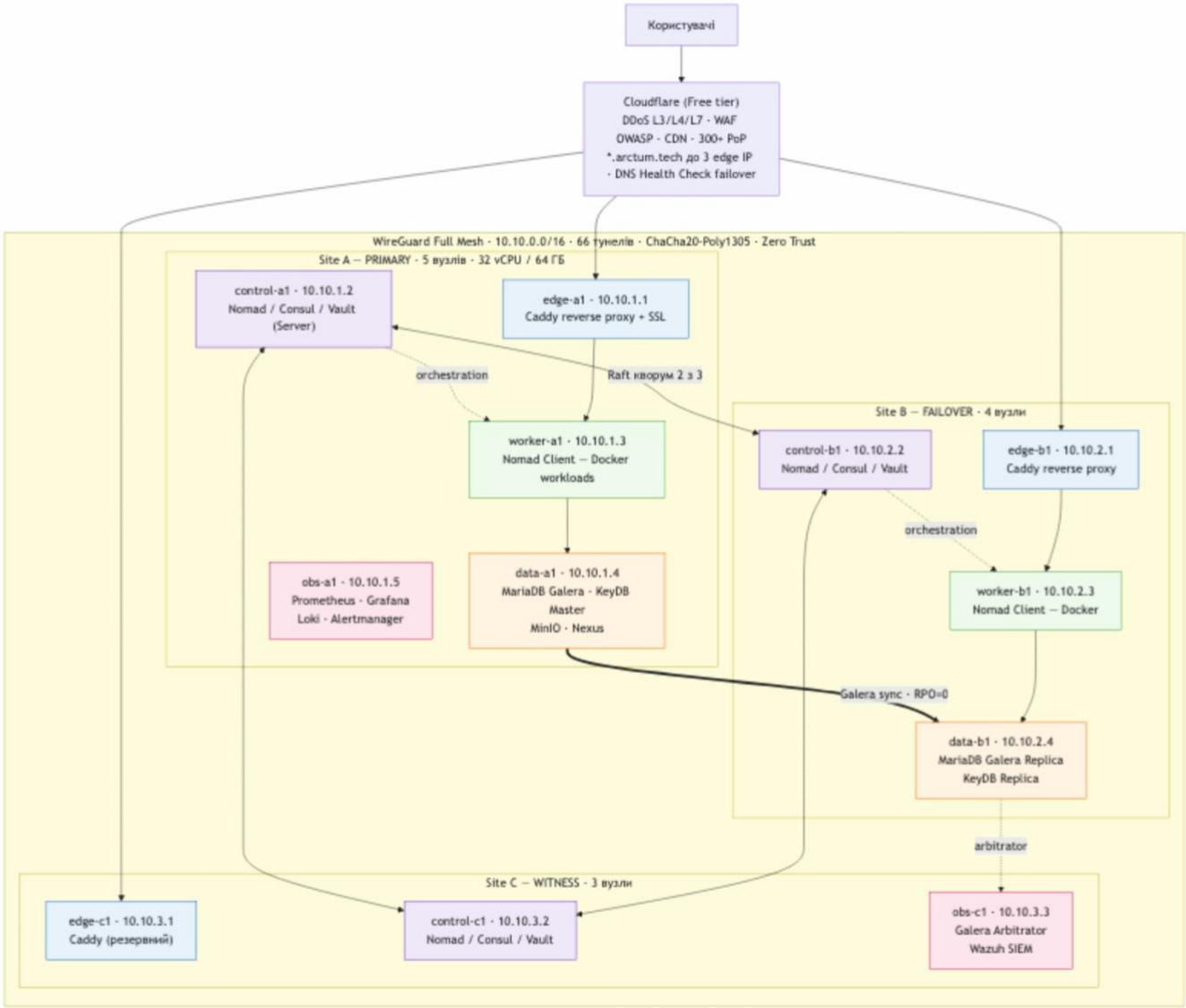
47. Гончаренко Т. А., Литвиненко О. В. Автоматизація процесів безперервного розгортання програмного забезпечення засобами CI/CD. Сучасні інформаційні технології та системи. 2024. № 2. С. 91–99.

Додаток А

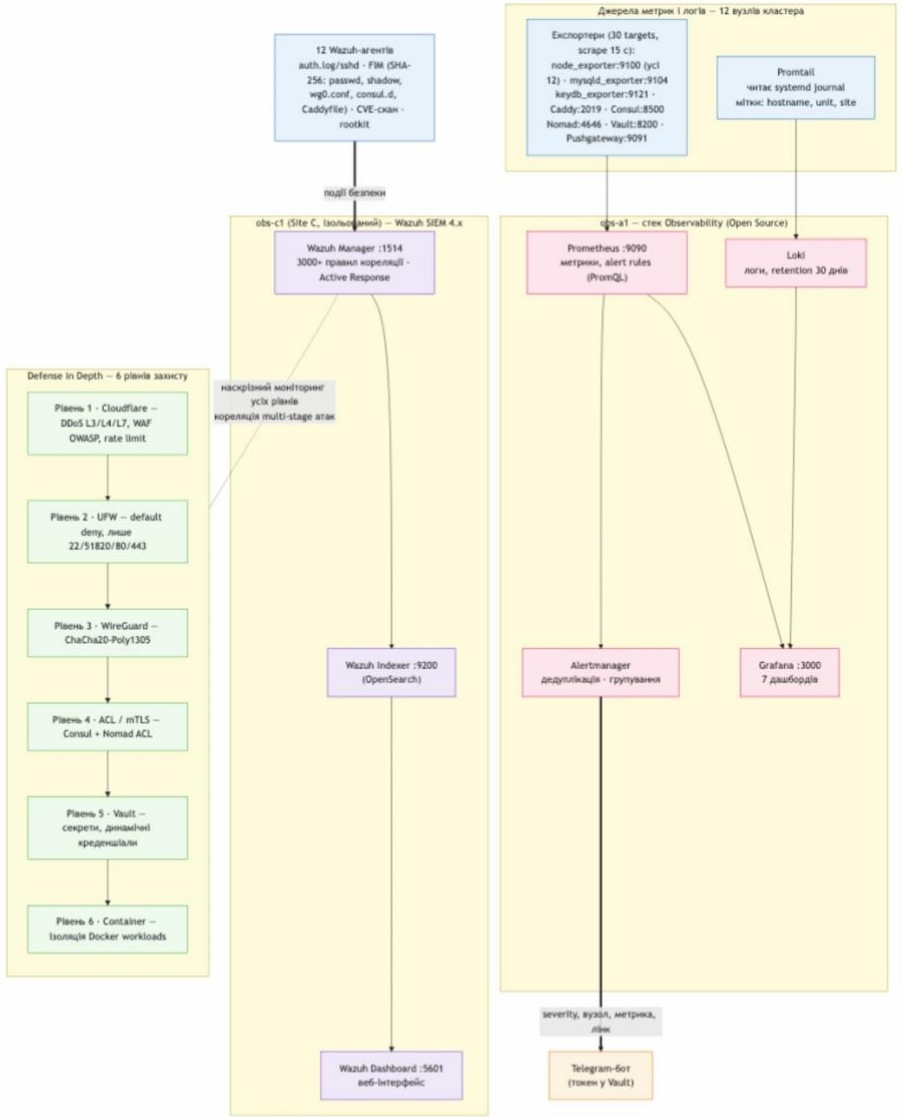
(обов’язковий)

Копії графічної частини





					КРБКБ.220110.22.01.07 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Система керування інформаційною безпекою для забезпечення безперервності бізнес-процесів підприємства	Літ / Маса / Масштаб
Розроб.		Коробко Р.Д.				Н
Перевір.		Касянчук М.М.				Аркуш / Аркушів 1
Т.контр.						
Н.контр.		Петляк Н.С.				ХНУ, КБ-22-1
Затверд.		Кльоц Ю.П.				



					КРБКБ.220110.22.01.07 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Система керування інформаційною безпекою для забезпечення безперервності бізнес-процесів підприємства	Літ. / Маса / Масштаб
Розроб.		Коробко Р.Д.				Н
Перевір.		Касянчук М.М.				Аркуш / Аркушів 1
Т.контр.						
Н.контр.		Петляк Н.С.				ХНУ, КБ-22-1
Затверд.		Кльоц Ю.П.				