Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

Пояснювальна записка

до випускної кваліфікаційної роботи магістра
на тему:
«Дослідження методів захисту Linux кластеру Azure Kubernetes Service»

Виконав
студент II-го курсу, групи КБм-23-1
спеціальності 125 «Кібербезпека та захист інформації»
_____ В.В. Миколайчук

Керівник
Доцент кафедри комп'ютерної інженерії
та кібербезпеки,
кандидат технічних наук,
доцент
_____ Ю.М. Россінський

Консультант
Завідувач кафедри комп'ютерної інженерії
та кібербезпеки,
кандидат технічних наук,
доцент
_____ А.А. Єфіменко

Рецензент
Доцент кафедри комп'ютерної інженерії
та кібербезпеки, кандидат технічних наук,
доцент
_____ О.О. Шелуха

м. Житомир – 2024 рік

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ
СПЕЦІАЛЬНІСТЬ 125 «КІБЕРБЕЗПЕКА ТА ЗАХИСТ ІНФОРМАЦІЇ»
ОСВІТНЯ ПРОГРАМА «КІБЕРБЕЗПЕКА»

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерної інженерії
та кібербезпеки
А.А.Єфіменко
17 вересня 2024 р.

ЗАВДАННЯ

на випускну кваліфікаційну роботу

Студента **Миколайчука Вадима Володимировича**

Тема роботи: **Дослідження методів захисту Linux кластеру Azure Kubernetes Service**

Затверджена Наказом по університету від 17 вересня 2024 р. № 479/с

Термін здачі студентом закінченої роботи 31 грудня 2024 р.

Вихідні дані для роботи (зазначається предмет і об'єкт дослідження):

Предметом даного дослідження є методи та технології, спрямовані на підвищення захисту, що використовують кластери Kubernetes, а саме Azure Kubernetes Service.

Об'єктом дослідження є процес модернізації та оптимізації інформаційної системи на базі Linux кластера в Azure Kubernetes Service з метою підвищення її захисту. Цей процес включає в себе розробку та імплементацію стратегій, що базуються на кращих практиках із забезпечення захисту та зменшення вектору атак.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Єфіменко А.А.	17.09.2024	17.09.2024
2	Єфіменко А.А.	12.10.2024	12.10.2024
3	Єфіменко А.А.	17.12.2024	17.12.2024

Студент _____ В.В. Миколайчук

Керівник _____ Ю.М. Россінський

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з вимогами до виконання магістерської роботи	17.09.2024	Виконано
2	Розробка плану та завдання роботи	18.09.2024 – 26.09.2024	Виконано
3	Збір та аналіз літератури	27.09.2024 – 05.10.2024	Виконано
4	Дослідження принципів роботи віртуалізації та контейнеризації	06.10.2024 – 16.10.2024	Виконано
5	Дослідження хмарних провайдерів	17.10.2024 – 29.10.2024	Виконано
6	Дослідження технології Kubernetes та Azure Kubernetes Service	30.10.2024 – 09.11.2024	Виконано
7	Аналіз методів захисту кластеру Azure Kubernetes Service інструментами хмари	10.11.2024 – 17.11.2024	Виконано
8	Аналіз методів захисту кластеру Azure Kubernetes Service інструментами Kubernetes	18.11.2024 – 25.11.2024	Виконано
9	Розгортання середовища та перевірка роботи засобів покращення безпеки середовища	26.11.2024 – 13.12.2024	Виконано
10	Оформлення пояснювальної записки	14.12.2024 – 30.12.2024	Виконано
11	Захист	31.12.2024	Виконано

Студент _____ В.В. Миколайчук

Керівник _____ Ю.М. Россінський

АНОТАЦІЯ

Кваліфікаційна робота містить 98 сторінок, 44 ілюстрації, 7 таблиць, список використаних джерел із 31 найменувань.

Метою атестаційної магістерської роботи є аналіз та розробка методів захисту Linux кластеру Azure Kubernetes Service (AKS). Власне побудова захищеної інформаційної системи буде демонструватися на прикладі Linux кластеру в Azure Kubernetes Service (AKS) з інфраструктурою, наближеною до проду, що буде відображатись на виборі типу розміщення інфраструктури та технологій, що використовуються.

В ході дослідження проведено аналіз будови кластеру Kubernetes, а також Azure Kubernetes service та принципів розміщення конйнеризованих додатків в даних ситемах.

В роботі запропонований підхід для захисту інформаційної системи в даних середовищах.

Ключові слова: KUBERNETES, AZURE KUBERNETES SERVICE, КОНТЕЙНЕР, ЗАХИСТ

ANNOTATION

The qualification work contains 98 pages, 44 illustrations, 7 tables, a list of 31 references.

The purpose of the work is the analysis and development of methods for protecting the Linux cluster of the Azure Kubernetes Service (AKS). The actual construction of a secure information system will demonstated on the example of a Linux cluster in Azure Kubernetes Serivce (AKS) with an infrastructure close to the production, which will be reflected in the choise of the type of infrastructure placement and technologies used.

The study analyzes the structure of the Kubernetes cluster, as well as the Azure Kubernetes service and the principles of deploying containerized applications in these systems.

Keywords: KUBERNETES, AZURE KUBERNETES SERVICE, CONTAINER, SECURITY

ЗМІСТ

1. ТЕОРЕТИЧНІ ОСНОВИ ТА ОПИС МЕТОДІВ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ В ХМАРНИХ КОНТЕЙНЕРИЗОВАНИХ СЕРЕДОВИЩАХ 10
1.1. АНАЛІЗ ТЕХНОЛОГІЇ КОНТЕЙНЕРИЗАЦІЇ З ТОЧКИ ЗОРУ БЕЗПЕКИ 10
1.1.1. ПЕРЕВАГИ КОНТЕЙНЕРІВ ПЕРЕД ВІРТУАЛЬНИМИ МАШИНАМИ 10
1.1.2. СЕРЕДОВИЩЕ ВИКОНАННЯ КОНТЕЙНЕРІВ 15
1.1.3. ЖИТТЄВИЙ ЦИКЛ КОНТЕЙНЕРУ 19
1.2. ОГЛЯД ХМАРНИХ РІШЕНЬ 21
1.2.1. ХМАРНІ МОДЕЛІ 21
1.2.2. ХМАРНІ ПРОВАЙДЕРИ 23
1.3. ОГЛЯД ПЛАТФОРМИ KUBERNETES 31
1.3.1. ПРИЗНАЧЕННЯ KUBERNETES 31
1.3.2. СТРУКТУРА KUBERNETES 33
1.3.3. WINDOWS ТА LINUX КЛАСТЕРИ KUBERNETES 33
1.3.4. БАЗОВІ КОМПОНЕНТИ AZURE KUBERNETES SERVICE 35
ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ 37
2. АНАЛІЗ МЕТОДІВ ПІДВИЩЕННЯ ЗАХИСТУ LINUX КЛАСТЕРУ AKS 40
2.1. АНАЛІЗ МЕТОДІВ ЗАХИСТУ КЛАСТЕРУ AZURE KUBERNETES SERVICE ІНСТРУМЕНТАМИ ХМАРИ 40
2.1.1. AZURE WELL-ARCHITECTED FRAMEWORK 40
2.1.2. AZURE POLICY 45
2.2. ЗАХИСТ КЛАСТЕРУ AKS ІНСТРУМЕНТАМИ KUBERNETES 67
2.3. ВИКОРИСТАННЯ БАЗОВИХ НАЛАШТУВАНЬ МЕРЕЖІ ДЛЯ ПІДВИЩЕННЯ ЗАХИСТУ AKS 72
ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ 77
3. ДОСЛІДЖЕННЯ ПРАКТИК ПОКРАЩЕННЯ ЗАХИСТУ КЛАСТЕРУ AKS 78
3.1. АНАЛІЗ ПОТОЧНОГО СТАНУ КЛАСТЕРУ AKS З ВИКОРИСТАННЯМ ІНСТРУМЕНТІВ AZURE 78
3.1.1. ВПРОВАДЖЕН НЯ ПОКРАЩЕНЬ ЗГІДНО З РЕКОМЕНДАЦІЙ AZURE 83
3.2. ВПРОВАДЖЕННЯ ПРАКТИК БЕЗПЕКИ НАТИВНИМИ ЗАСОБАМИ KUBERNETES 87
3.3. ВПРОВАДЖЕННЯ ПРАКТИК БЕЗПЕКИ ДЛЯ KUBERNETES СТОРОННІМИ ЗАСОБАМИ 91
ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ 94
ВИСНОВКИ 95
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 96

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AKS – Azure Kubernetes Service

ІС – інформаційна система

ВМ – віртуальна машина

ОС – операційна система

CI – Continuous integration

CD – Continuous deployment

RPC – Remote Procedure Call

CLI – інтерфейс командного рядка

CNCF – Фонд хмарних обчислень

OCI – Ініціатива відкритих контейнерів

MAP – Migration Acceleration Program

SaaS – Software as a Service

PaaS – Platform as a Service

IaaS – Infrastructure as a Service

AWS – Amazon Web Services

MTEB – Massive Text Embedding Benchmark

GCP – Google Cloud Platform

ШІ – штучний інтелект

GDC – Google Distributed Cloud

LoRA – Low-Rank Adaptation

CSRB – Cyber Safety Review Board

SFI – Secure Future Initiative

PLEG – Pod Lifecycle Event Generator

API – application programming interface

KEDA – Kubernetes Event Drive Autoscaler

ВСТУП

В умовах стрімкого зростання використання хмарних технологій та контейнеризації для розгортання й масштабування сучасних інформаційних систем тема дослідження методів захисту тема даної роботи, а саме «Дослідження методів захисту Linux кластеру Azure Kubernetes Service» є досить актуальною. Kubernetes є де-факто стандартном оркестрації контейнерів, а Azure Kubernetes Service (AKS) пропонує інтегровану платформу для управління кластерами. Зростання популярності популярності цих технологій робить їх об’єктами цільових кібератак, що в свою чергу вимагає дослідження нових підходів до забезпечення їх безпеки. Спеціалісти, що працюють з інформаційними системами, які використовуються AKS, повинні володіти глибокими знаннями Kubernetes та інструментів, що пропонують Microsoft в хмарі Microsoft Azure, для того, щоб захиститсь від потенційних загроз та задовільнити стандарти безпеки.

Мета цього дослідження полягає в аналізі та розробці методів захисту інформаційних систем з використанням Kubernetes. Це буде демонструватися на прикладі Linux кластеру в Azure Kubernetes Service (AKS) з інфраструктурою, наближеної до проду, що буде відображатись на виборі типу розміщення інфраструктури та технологій, що використовуються.

Об'єктом дослідження є процес модернізації та оптимізації інформаційної системи на базі Linux кластера в Azure Kubernetes Service з метою підвищення її захисту. Цей процес включає в себе розробку та імплементацію стратегій, що базуються на кращих практиках із забезпечення захисту та зменшення вектору атак.

Предметом даного дослідження є методи та технології, спрямовані на підвищення захисту, що використовують кластери Kubernetes, а саме Azure Kubernetes Service.

Методологічний підхід базується на аналізі найкращих практик та використанні системного аналізу для створення ефективної та захищеної

інфраструктури Kubernetes, на приклад Linux кластеру Azure Kubernetes Service. Це включає моделювання, аналіз та побудову інфраструктури, оцінку поточного стану захисту систем, і планування удосконалень із зосередженням на практиках з підвищення безпеки інформаційної системи.

Наукова новизна полягає у:

розробці та обґрунтуванні унікального підходу до комбінації різних методів та рівнів захисту для Linux-кластерів AKS;

дослідженні реальних сценаріїв із залученням різних інструментів;

порівняльний аналіз обраних інструментів безпеки в межах хмарної платформи Azure;

пропозиція методики масштабування та підтримки безпеки кластеру AKS з врахуванням специфіки розподілених обчислень і динамічної інфраструктури;

формування рекомендацій для автоматизації реагування на ті чи інші інциденти безпеки, що можуть статись в досліджуваній інформаційній безпеці.

Практичне значення полягає у побудові фреймворку на основі аналізу методів захисту інформаційних систем, що використовують Kubernetes у своїй складовій, а саме Linux кластери Azure Kubernetes Service. Використовуючи дані результати інженери, що можуть бути залучені до аналогічного завдання, будуть мати змогу використати дані практики.

Апробація результатів:

1. Миколайчук В.В., Єфіменко А.А. Дослідження методів захисту Linux кластеру Azure Kubernetes Service // Тези доповідей VII Всеукраїнської науково-технічної конференції «Ком’ютерні технології: інновації, проблеми, рішення», м. Житомир, 02-03 грудня 2024 р. – Житомир: Житомирська політехніка, 2024.

1.

1.ТЕОРЕТИЧНІ ОСНОВИ ТА ОПИС МЕТОДІВ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ В ХМАРНИХ КОНТЕЙНЕРИЗОВАНИХ СЕРЕДОВИЩАХ

1.1. АНАЛІЗ ТЕХНОЛОГІЇ КОНТЕЙНЕРИЗАЦІЇ З ТОЧКИ ЗОРУ БЕЗПЕКИ

1.1.1.ПЕРЕВАГИ КОНТЕЙНЕРІВ ПЕРЕД ВІРТУАЛЬНИМИ МАШИНАМИ

На сьогодні при виборі технологій для побудови інфраструктури для розміщення інформаційних систем, стоїть вибір між застосування традиційних віртуальних машин (ВМ) та контейнеризованими рішеннями.

Віртуальні машини відіграють вирішальну роль у хмарних обчисленнях, емулюючи фізичні комп'ютери, запускаючи операційні системи в ізольованих інстансах. Кілька машин зазвичай розміщуються на одному сервері, а гіпервізор виконує роль легкого програмного забезпечення, розташованого між фізичним хостом і ВМ. Гіпервізор ефективно керує доступом до ресурсів, дозволяючи віртуальним машинам функціонувати як окремі сервери, пропонуючи при цьому підвищену гнучкість і маневреність.

Вперше набувши популярності у 2000-х роках завдяки ініціативам з консолідації та економії витрат, використання віртуальних машин з часом еволюціонувало. Організації вдосконалили розгортання віртуальних машин, вийшовши за рамки консолідації та охопивши різні сценарії використання. До них відносяться надання ресурсів на вимогу для додатків і оптимізація доступу до дорогих ресурсів, таких як графічні процесори.

Віртуальні машини також слугували основою багатьох ранніх середовищ хмарних обчислень, полегшуючи віртуалізацію ресурсів з підтримкою multi-tenancy та ізоляції – тобто, випадки коли кілька клієнтів використовують одні й ті ж самі ресурси.

Віртуальні машини містять власну операційну систему, що дозволяє їм виконувати кілька ресурсоємних функцій одночасно. Збільшені ресурси, доступні віртуальним машинам, дозволяють абстрагувати, розділяти, дублювати та емулювати цілі сервери, операційні системи, настільні комп'ютери, бази даних та мережі.

Програмне забезпечення, яке називається гіпервізором, відокремлює ресурси від фізичних машин, щоб їх можна було розділити для віртуальних машин. Коли користувач видає віртуальній машині інструкцію, яка вимагає додаткових ресурсів з фізичного середовища, гіпервізор передає запит до фізичної системи і кешує зміни. Віртуальні машини працюють як фізичні сервери, що є свого роду недоліком, тому що для запуску додатку на віртуальну машину потрібно встановлювати цілу операційну систему (ОС) та програмне забезпечення для запуску додатків користувача, що є занадто витратним з точки зору виділення дискових та обчислювальних ресурсів.

В той же час контейнер – це програмна одиниця, яка об'єднує всі компоненти та функції, необхідні для запуску програми. Більшість сучасних додатків складаються з декількох контейнерів, кожен з яких виконує певну функцію. Розмір контейнерів зазвичай вимірюються в мегабайтах, вони не використовують гіпервізор і, як правило, вважаються швидшим і гнучкішим способом ізоляції процесів.

Одним з важливих факторів, що сприяють успіху контейнерів, є портативність. Окремі контейнери можна легко замінювати та переміщувати в різні середовища. Після того, як додаток та його залежності упаковані в контейнер, його можна розгорнути де завгодно – на ноутбуці розробника, в центрі обробки даних, хмарі і т.д. – та бути впевненим, що він буде функціонувати точно так, як і раніше.

Все, що знаходиться в контейнері, пакується та постачається за допомогою образу контейнера – файлу, який включає всі бібліотеки та залежності. Файли образу контейнера схожі на інсталяційні пакунки програмного забезпечення (наприклад, RPM у Linux), але для запуску програми потрібно лише сумісне ядро і середовище виконання контейнера, незалежно від того, яку ОС було використано для створення контейнера і звідки були отримані бібліотеки всередині контейнера.

Тому перед вибором між контейнерами та віртуальними машинами можна сформувати наступне твердження: якщо необхідно використати легкі та

портативні інстанси для розгортання, то варто використати контейнери, а якщо потрібне напів-перманентне рішення для розгортання інфраструктури, то варто зупинитись на віртуальних машинах.

Окрім того, додаткивими факторами, які слід врахувати, включають архітектуру інформаційної системи, практику розробки, вимоги безпеки та регуляторні вимоги.

Контейнери можна розгортати їх на bare metal, а також у публічних, приватних та гібридних хмарах. Також контейнери можна запускати на існуючих віртуальних машинах, оскільки організації.

Контейнери також є ідеальним середовищем для розгортання сучасних cloud-native додатків, які являють собою набір мікросервісів, призначених для забезпечення послідовної розробки та автоматизованого управління в публічних, приватних, гібридних і мультихмарних середовищах. Cloud-native додатки допомагають пришвидшити створення нових додатків, оптимізацію існуючих та їхню взаємодію [1].

Порівняння контейнерів та традиційних віртуальних машин (ВМ) з точки зору безпеки починається з того, що контейнери, на відміну від віртуальних машин, не потребують окремого гостьового ядра операційної системи. Віртуальні машини працюють поверх гіпервізора (наприклад, Hyper-V чи VMware ESXi). Вони забезпечують апаратну ізоляцію, фактично розподіляючи між собою повноцінні операційні системи. Контейнери ж використовують ядро хостової системи й власні простори імен (namespaces) та cgroups (контроль груп), що відповідають за обмеження доступу до ресурсів процесора, пам’яті та файлової системи. З одного боку, це робить контейнери “легшими” порявняно з віртуальними машинами, бо вони швидко запускаються, споживають менше ресурсів і передбачають значно меншу кількість службових процесів, які необхідні для повноцінної роботи операційної системи. З іншого боку, таке спільне використання хостового ядра вносить додаткові ризики, оскільки вразливість у ядрі Linux може автоматично відкривати зловмисникові шлях до всіх контейнерів, що працюють на цьому хості. Ситуація дещо інша у випадку з

віртуальними машинами, де кожна ВМ має власне ядро та операційну систему, а отже, компрометація одного “гостьового” середовища не обов’язково означає злам решти ВМ на тому ж фізичному сервері.

Ще один аспект полягає в різниці рівня ізоляції: гіпервізор за замовчуванням створює більш “товсту” межу між віртуальними машинами, а контейнери залежать від механізмів ядра ОС та userland-утиліт для побудови “тонкої” межі. З точки зору безпеки, якщо команди безпеки правильно налаштовує політики безпеки з використанням SELinux, seccomp, AppArmor та ін., контейнери можуть бути достатньо добре ізольовані між собою. Проте реальні інциденти свідчать, що обходити контейнерну ізоляцію дещо легше, ніж намагатись “вирватися” на рівень гіпервізора з віртуальної машини. Утім, водночас легка вага контейнерів дозволяє застосовувати кращі практики “immutable infrastructure”: якщо контейнер має критичну вразливість у базовому образі, то є можливість швидко оновити образ, перезібрати та замінити контейнери в середовищі, в якому вони виконуються. Така мобільність є однією з сильних сторін контейнерної парадигми, чим суттєво спрощує виправлення вразливостей.

Для корпоративних середовищ, зокрема тих, що використовують хмари, важливим моментом є поєднання контейнерів із сервісами безпеки, наприклад, Microsoft Defender for Cloud або Azure Policy, які можуть автоматично аналізувати конфігурацію та виявляти вразливості в режимі реального часі. Проте, спільне ядро хост-машини, на якому запущено контейнер або декілька контейнерів, залишається єдиною точкою зламу (single point of failure) у порівнянні з окремим ядром для кожної віртуальної машини. Саме тому, плануючи архітектуру безпеки, важливо зважати на те, що контейнери ідеально підходять для мікросервісних застосунків, де швидке оновлення та стандартизована ізоляція є в пріоритеті, але водночас слід додатково забезпечувати багаторівневий моніторинг та контроль інформаційно системи, що використовує контейнеризовані рішення, наприклад, за допомогою RBAC у Kubernetes, мережевих політик та сканування базових образів, тощо.

Для прикладу варто також зазначити декілька реальних випадків експлуатації тих, чи інших вразливостей, що дозволяли проникнути за межі віртуальних машин.

1. CVE-2015-3456 (VENOM). Одна з найвідоміших вразливостей у QEMU, що пов’язана з емуляцією дисководу (floppy drive). Через помилку в обробці спеціально сформованих запитів можлива була ескалація привілеїв із рівня гостя (гостьової ОС у віртуальній машині) на хост-систему, фактично даючи змогу виконати код за межами ВМ. Уразливість стосувалася середовищ, де QEMU використовувався напряму чи в Xen, KVM тощо [2].

2. CVE-2021-26414 (Hyper-V). Ця вразливість стосувалася механізмів Remote Procedure Call (RPC) у Microsoft Hyper-V. Хоч безпосередньо не завжди згадувалась як “escape” у класичному вигляді, вона створювала передумови для потенційного виконання коду в контексті хосту. Вразливість була виправлена в оновленнях Windows Server, але слугувала прикладом того, що навіть Hyper-V підлягає ретельному аудиту на можливість переходу з гостьового середовища на хост [3].

3. CVE-2017-14316 (Xen Hypercall). Уразливість у Xen, що могла бути використана для ескалації рівня доступу, у випадку, коли зловмисник мав уже привелийований рівень доступу над гостьовою системою. У певних випадках це дозволяло з гостьової ОС “зайти глибше” й виконати небажані дії на рівні гіпервізора [4].

І ось декілька вразливостей, що дозволяли вийти за межі контейнерів:

1. CVE-2019-5736 (runC/Docker). Одна з найгучніших вразливостей, оскільки стосувалась популярний середовища виконання контейнерів runC, який використовується в Docker, containerd, CRI-O тощо. Вона дозволяла модифікувати runc із запущеного контейнера та виконати код на хості з привілеями root. Класичний приклад втечі з контейнеру, де зловмисник запускав шкідливий контейнер або мав можливість ін’єкції у контейнері, то міг через цю вразливість отримати контроль над хостовою системою [5].

2. CVE-2022-0185 (Linux Kernel cgroup). Ця вразливість безпосередньо стосувалася підсистеми cgroup в ядрі Linux. За певних умов зловмисник у контейнері міг скористатися недоліками валідації пам’яті й посилань у cgroup та отримати привілеї в просторі хост-ядра. Хоча технічно це була проблема ядра, а не самого контейнера, але саме сценарій використання контейнерів робив експлойт особливо небезпечним [6].

3. CVE-2021-22555 (Linux Kernel netfilter). Патч, що виправляв вразливість у netfilter, яка дозволяла локальному користувачеві (у т.ч. процесу всередині контейнера) отримати привілеї на рівні ядра. Як і в попередньому прикладі, основна проблема була в ядрі, однак експлуатувалася за допомогою контейнеру, що дозволяло проникнути в хост-систему [7].

Загалом, приклади CVE демонструють, що для успішної атаки з отриманням доступу до рівня гіпервізора чи системного ядра хосту потрібна критична вразливість у ключових компонентах (QEMU, Hyper-V, Xen, runC тощо). У випадку віртуальних машин складність розробки дієвих експлойтів вища, оскільки потрібно пройти крізь доволі низькорівневий прошарок гіпервізора. У контейнерах же ширша, оскільки ядро операційної системи на якій розміщені контейнери, фактично спільне для всіх контейнерів і неправильні налаштування безпеки чи наявність певних системних викликів (syscalls) спрощують пошук векторів атаки. Тим не менше, обидві платформи, за умови регулярного оновлення, належного патч-менеджменту та застосування рекомендованих політик (наприклад, SELinux, AppArmor, seccomp у контейнерах і суворих шаблонів безпеки для гіпервізора), дозволяють мінімізувати ризики “втечі” та суттєво ускладнюють завдання зловмисникам.

1.1.2.СЕРЕДОВИЩЕ ВИКОНАННЯ КОНТЕЙНЕРІВ

Середовище виконання контейнерів — це програмний компонент, який може запускати контейнери в операційній системі хоста. У контейнеризованій архітектурі середовище виконання контейнера відповідає за завантаження образів контейнера зі репозиторіїв, моніторинг локальних системних ресурсів,

ізоляцію системних ресурсів для використання контейнеру та керування його життєвим циклом.

Більшість середовищ виконання контейнерів зазвичай працюють разом із оркестраторами контейнерів. Оркестратор відповідає за керування кластерами контейнерів, відповідає за масштабування, мережу та безпеку. Механізм контейнерів бере на себе відповідальність за керування окремими контейнерами, що працюють на кожному обчислювальному вузлі в кластері.

Типовими прикладами середовищ виконання контейнерів є runC, containerd, Docker і контейнери Windows. Існує три основні типи середовищ виконання контейнерів — середовища виконання низького рівня, середовища виконання високого рівня та середовища виконання із ізольованим програмним середовищем або віртуалізовані середовища виконання.

Open Container Interface (OCI) — це проект Linux Foundation, започаткований компанією Docker, мета якого — забезпечити відкриті стандарти для контейнерів Linux. Основним проектом із відкритим кодом, розробленим OCI, є runC, випущений у 2015 році. runC — це низькорівневе середовище виконання контейнерів, яке реалізує специфікацію OCI. Він є основою для багатьох інших механізмів виконання контейнерів.

OCI надає специфікації середовища виконання контейнерів. Середовища виконання, реалізовані відповідно до специфікацій OCI, називаються низькорівневими середовищами виконання, тому що основна увага приділяється управлінню життєвим циклом контейнера.

Нативні низькорівневі середовища виконання контейнерів відповідають за створення та запуск контейнерів. Після запуску контейнерного процесу середовище виконання контейнерів не потрібна для виконання інших завдань. Це тому, що низькорівневі середовища виконання абстрагують примітиви Linux і не призначені для виконання додаткових завдань.

Найпопулярніші низькорівневі середовища виконання включають:

runC — створений Docker і OCI. Тепер це де-факто стандарт в класі низькорівневих середовищ виконання контейнерів. runC написано мовою Go. Він підтримується moby — проектом Docker з відкритим кодом.

crun — реалізація OCI під керівництвом Red Hat. crun написаний мовою C. Він розроблений, щоб бути легким і продуктивним, і був одним із перших середовищ виконання, які підтримували cgroups v2.

containerd — демон з відкритим вихідним кодом, який підтримується Linux і Windows, який полегшує керування життєвими циклами контейнера через запити API. API контейнера додає рівень абстракції та покращує портативність контейнера.

Приклади високорівневих середовищ виконання контейнерів:

Docker (containerd) — провідна контейнерна система, яка пропонує повний набір функцій, як безкоштовних, так і платних. Це середовище виконання контейнерів за замовчуванням (мається на увазі containerd) в Kubernetes, що надає специфікації образів, інтерфейс командного рядку (CLI) і службу створення образу контейнеру.

CRI-O — реалізація середовища виконання контейнерів Kubernetes (CRI) з відкритим вихідним кодом, що пропонує легку альтернативу rkt та Docker. Дозволяє запускати контейнери за допомогою OCI-сумісних середовищ виконання, забезпечуючи підтримку в першу чергу runC і Kata (хоча можливо підключити будь-яке OCI-сумісне середовище виконання).

Контейнери Windows і контейнери Hyper-V — дві легкі альтернативи віртуальним машинам Windows, доступні у Windows Server. Контейнери Windows пропонують абстракцію (подібно до Docker), а Hyper-V — віртуалізацію. Контейнери Hyper-V легко переносяться, оскільки кожен з них має власне ядро, тож є можливість запускати несумісні програми у вашій хост-системі [8].

Важливо також додатково зазначити та пояснити зв'язок containerd (який являється середовищем виконання контейнерів за замовчуванням в Kubernetes)

та Docker (середовище виконання контейнерів, що часто використовується розробниками). Розглянемо ще раз, що вміє containerd та його історію.

containerd — це середовище виконання контейнерів, яке керує життєвим циклом контейнера на фізичній або віртуальній машині (хості). Це процес-демон, який створює, запускає, зупиняє та знищує контейнери. Він також може завантажувати образи контейнерів з реєстрів контейнерів, монтувати сховище і вмикати мережу для контейнера.

containerd був створений компанією Docker і переданий Cloud Native Computing Foundation (CNCF). Це підмножина оригінального Docker Engine, яка має більшість функціональних можливостей Docker для запуску контейнерів, роботи зі сховищем та керування образами, але не має багато функцій, орієнтованих на розробників, що робить її придатною для широкомасштабного використання у складі контейнерних оркестраторів, таких як Kubernetes. Docker Engine використовує containerd у своїй роботі.

Docker Engine є прикладом високорівневого контейнерного движка, що використовується переважно розробниками, в той час як containerd є прикладом низькорівневого контейнерного движка, що має лише базову функціональність, придатну для використання автоматизованими механізмами.

containerd повністю підтримує стандарти, встановлені OCI, організацією з відкритим управлінням, зосередженою на найкращих практиках у сфері контейнерів. Він також має підтримку container runtime interface (CRI) – специфікацію Kubernetes, розроблену для того, щоб дозволити декільком середовищам виконання контейнерів працювати у складі кластера. Kubernetes може використовувати containerd та інші низькорівневі середовища виконання контейнерів, що підтримують OCI, для запуску контейнерів на вузлах Kubernetes [9].

containerd є середовищем виконання контейнерів в Azure Kubernetes Service за замовчуванням також. Для нод на основі Linux він використовується за замовчуванням, починаючи з версії Kubernetes 1.19, а для нод на базі ОС Windows Server починаючи з версії Kubernetes 1.23 та вище [10]. containerd

оновлюється разом з іншим програмним забезпеченням під час оновлення образу операційної системи. Дана поведінка може додатково конфігуруватись.

1.1.3.ЖИТТЄВИЙ ЦИКЛ КОНТЕЙНЕРУ

Розглянемо життєвий цикл контейнеру на прикладі Docker-контейнерів (рис.1.1). Для ефективної роботи з контейнерами (в тому числі і Docker-контейнерами) та розуміння що може піти не так у безпекових питаннях, важливо розуміти різні етапи існування контейнера, які називаються його життєвим циклом. Життєвий цикл контейнера Docker складається з декількох станів, таких як created (створений), running (запущений), paused (зупинений), exited (завершений) і dead/deleted (видалений).

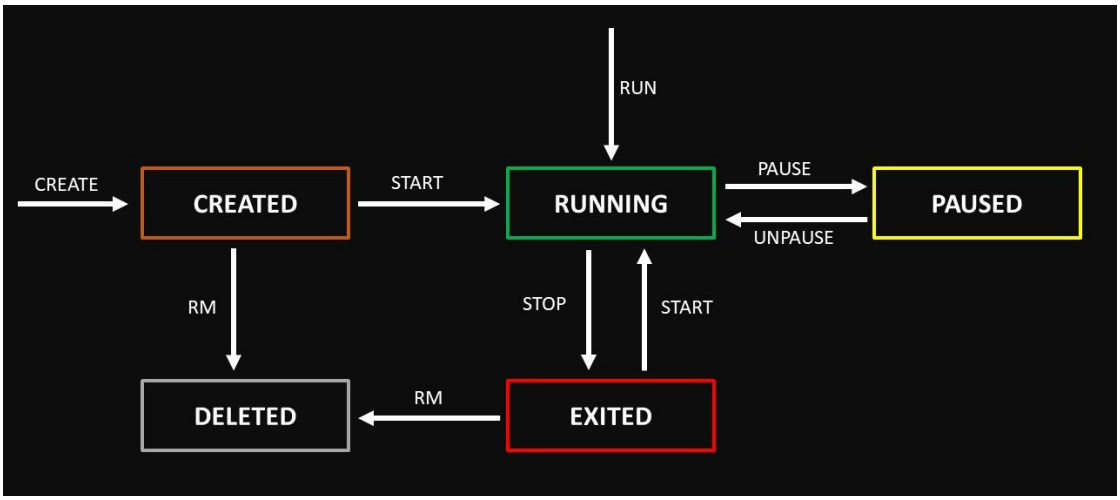


Рисунок 1.1 — Ілюстрація життєвого циклу контейнеру

Коротко опишемо кожен зі станів:

Created — це початковий стан Docker-контейнеру після його створення, але до того, як його було запущено.

Running — стан, у якому створений контейнер запущено. У цьому стані контейнер власне виконує свої процеси.

Paused — якщо контейнер призупинено, виконання його процесів тимчасово зупиняється, але він не завершується.

Exited — якщо основний процес контейнера завершується, контейнер зупиняється і переходить у стан exited.

Dead — якщо контейнер не вдається запустити, він переходить у стан "мертвий". Контейнери у цьому стані не можуть бути перезапущені і повинні бути створені заново.

Created — це перший стан у життєвому циклі контейнера. Цей стан означає, що контейнер створено, але не запущено. Створити контейнер можливо за допомогою команди docker create.

Коли ми створюємо контейнер Docker, до шарів тільки для читання (R/O) образу, що використовується для побудови свого, додається шар читання і запису(R/W). Це готує контейнер до запуску, наприклад, на цьому етапі завантажується образ, налаштовуються змінні оточення, встановлюється точка входу тощо.

Важливо зазначити, що створення контейнера не призводить до його автоматичного запуску. Однак, ми можемо встановити всі необхідні параметри конфігурації під час процесу створення, такі як ліміти процесора і пам'яті, образ контейнера і т.д.. Коли контейнер знаходиться у цьому стані, конфігурацію контейнера також можна оновити за допомогою команди docker update.

Це означає, що є можливість створити контейнер один раз з усіма необхідними параметрами і запустити його пізніше без необхідності вказувати їх знову. Також слід зазначити, що у цьому стані ресурси не виділяються.

Running — стан, при якому створений контейнер запущено. Команди, описані в маніфесті контейнеру виконуються одна за одною.

При запуску контейнеру Docker налаштовує виділення ресурсів, які необхідні контейнеру (мережеві ресурси, ресурси процесору, оперативну пам'ять). Потім Docker створює середовище для роботи контейнеру. Як тільки контейнер готовий до роботи, він починає виконувати інструкції, вказані в маніфесті.

Killed/exited — це стан, в якому контейнер не виконується. Є ряд причин, через які контейнер переходить у стан exited. Наприклад, якщо процес, що виконувався в контейнері завершився або коли процес завершився з помилкою.

Знаючи життєвий цикл функціонування контейнерів, можна передбачити де і на якому етапі зловмисник може скористатись тією чи іншою вразливістю та прийняти міру, що допоможе превентивно запобігти цьому. Також потрібно враховувати різні середовища запуску контейнерів. Наприклад, перед доставкою та запуском контейнеру в кластер Kubernetes, розробники або інші члени команд, задіяні в процесі циклу розробки контейнеризованої інформаційної системи, можуть мати на меті запустити контейнери локально, тому відповідні міри мають бути застосовані на цьому етапі також.

1.2.ОГЛЯД ХМАРНИХ РІШЕНЬ

1.2.1.ХМАРНІ МОДЕЛІ

Інфраструктуру для Kubernetes можливо розмістити різними способами. Розглянемо хмарні моделі, які можна використати для цього. Існують три хмарні моделі: private, public та hybrid.

Приватна хмара (private cloud) надає хмарні послуги, доступ до яких надається які пропонуються через Інтернет або приватну внутрішню мережу з обмеженим доступом (на сьогодні є велика кількість механізмів, що дозволяє це зробити). Приватні хмарні обчислення, які також називають внутрішньою або корпоративною хмарою, надають організаціям багато переваг загальнодоступної хмари, включаючи самообслуговування, масштабованість і еластичність. Крім того, приватні хмари забезпечують вищий рівень безпеки та конфіденційності при використанні брандмауеру та ізольований хостинг, що гарантує, що та конфіденційні дані недоступні для посторонніх. Одним із недоліків є те, що IT-відділ компанії несе відповідальність за обслуговування та витрати на підтримку приватної хмари. Таким чином, приватні хмари вимагають тих самих витрат на персонал, управління та технічне обслуговування, що й традиційний центр обробки даних [11].

Публічна хмара (public cloud) є найпопулярнішим типом послуги, що надають провайдери хмарних послуг. Хмарні ресурси (наприклад, сервери та сховища даних) належать і управляються власне постачальником хмарних послуг і надаються у користування через Інтернет. У публічній хмарі все

обладнання, програмне забезпечення та інша допоміжна інфраструктура належить і управляється постачальником хмари. Microsoft Azure є прикладом публічної хмари.

У загальнодоступній хмарі клієнти користуються серверами, сховищем і мережним обладнання, що може використовуватись одночасно для надання послуг одночасно кільком «орендарям», а доступ користувачі отримують через облікові записи з відповідного рівня доступом

Переваги публічних хмар:

- менші витрати — не потрібно купувати обладнання чи програмне забезпечення, і потрібно платити лише за послугу, якою користуєтеся;
- не потрібно виконувати оновлення апаратного забезпечення самостійно — постачальник послуг забезпечує обслуговування;
- майже необмежена масштабованість — доступні ресурси на вимогу для задоволення потреб;
- висока надійність — внутрішні механізми, що використовують хмарні провайдери для своєї інфраструктури забезпечують високу надійність;
- безпекові питання, за які відповідає хмарний провайдер, що відповідають багатьом стандартам безпеки (ISO, PCI DSS, SOC 1-3, тощо), що в свою чергу знімає частину відповідальності з кінцевих користувачів хмари.

Гібридна хмара (hybrid cloud) — це обчислювальне середовище, яке поєднує локальний дата-центр (типу приватної хмари) з публічною хмарою, дозволяючи розміщувати дані або робочі навантаження в будь-якій або в публічній, або в приватній площині. Багато організацій обирають гібридну хмару через вимоги, що диктує бізнес або для дотримання регуляторних вимог [12].

Розглядаючи та оцінюючи публічні хмару, дуже важливо розуміти модель розподіленої відповідальності, а також те, які завдання вирішує хмарний провайдер, а які користувач послуг. Відповідальність за робоче навантаження варіюється залежно від того, продуктом якого типу доводиться користуватись Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) або ж локальному центрі обробки даних (ЦОД).

Власник дата-центру повністю несе відповідальність за його інфраструктуру та підтримку (як програмну, так і апаратну). В той же час при використанні публічної хмари, провайдер хмарних послуг бере частину зобов'язань з підтримки на себе (див. табл.1.1).

1.1

Зони відповідальності у моделі розподіленої відповідальності

Відповідальність	SaaS	PaaS	IaaS	On-premise
Інфраструктура (фізичні сервери, мережа)	П	П	П	К
Фізична безпека дата-центрів	П	П	П	К
Управління мережею	П	П	С	К
Віртуалізація	П	П	С	К
Операційна система	П	П	К	К
Мережева безпека	П	С	К	К
Додатки	П	К	К	К
Ідентифікація та доступ	С	С	К	К
Захист даних	С	С	К	К

П – провайдер, К – клієнт (споживач послуг), С – спільна відповідальність

1.2.2.ХМАРНІ ПРОВАЙДЕРИ

Звіти "Магічний квадрант" є підсумком ретельних, заснованих на фактах досліджень в межах конкретних ринків. Звіти дають широке уявлення про позиції провайдерів на тих ринках, які мають високі темпи зростання і чітку диференціацію провайдерів. Так у звіті «Magic Quadrant for Strategic Cloud Platform Services» Gartner визначає стратегічні послуги хмарних платформ як такі, що є стандартизовані, автоматизовані, загальнодоступні хмарні пропозиції, що надають змогу інтегрувати інфраструктурні сервіси (IaaS) (наприклад, обчислювальні, мережеві та сховища зберігання даних), платформені сервіси (PaaS) (наприклад, керовані сервіси додатків і даних) та послуги трансформації (програми/ресурси, які допомагають клієнтам впроваджувати хмаро-орієнтовані моделі надання IT-послуг). Позиції хмарних провайдерів у рейтингу «Quartner Magic Quadrant for Stategic Cloud Platform Services» станом на серпень 2024 представлені на рис. 1.2.

Інфраструктурні та платформні сервіси можуть розміщуватися як на стороні постачальника хмарних послуг, так і в дата-центрах клієнтів; однак вони належать провайдеру послуг. Сервіси повинні бути еластично масштабованими,

їх використання має бути прозорим і управління сервісами має бути доступним через веб-інтерфейси та програмовані API. Програми трансформації можуть реалізовуватися за допомогою автоматизованих інтерфейсів самообслуговування та керованої взаємодії, яку забезпечують команди/партнери по роботі з клієнтами.



Рисунок 1.2 – Позиції провайдерів у «Magic Quadrant for Strategic Cloud Platform Services»

Розглянемо компанії трьох очевидних лідерів даного рейтингу (рис. 1.2) та їх продуктів – найпопулярніших хмарних провайдерів для подальшого аналізу у нашій роботі, а саме Amazon Web Services (AWS) з їх однойменним хмарним провайдером, Microsoft та їх Microsoft Azure та Google з їх Google Cloud Platform.

Amazon Web Services (AWS) є лідером у цьому магічному квадранті. AWS, дочірня компанія Amazon.com, пропонує широкий спектр послуг хмарної інфраструктури та платформ, що задовольняють повний спектр IT-вимог. Орієнтуючись на корпоративних клієнтів, AWS сприяє цифровій трансформації клієнтів за допомогою ряду програм, таких як Migration Acceleration Program (MAP) та AWS Enterprise Transformation Program. Хоча AWS не зосереджується на корпоративних SaaS-платформах або платформах цифрових робочих місць,

вона працює з цими бізнес-додатками через свою розгалужену партнерську екосистему.

Хоча AWS зберегла прибутковість і провідну глобальну частку ринку інфраструктури як послуги (IaaS) і платформи як послуги (PaaS), що перевищує 42%, темпи її зростання порівняно з ключовими конкурентами сповільнилися в 2023 році. Партнерські продажі через AWS Marketplace переломили цю тенденцію і прискорюються. Стратегія AWS полягає в тому, щоб продовжувати вдосконалювати свої провідні в галузі послуги IaaS і PaaS та партнерську екосистему, одночасно просуваючи свою флагманську платформу Amazon Bedrock для розробки додатків GenAI.

Сильні сторони:

Операційна досконалість: AWS має чудовий багаторічний досвід надійного надання послуг. Цьому сприяє її широко розподілена архітектура з публічно задокументованими межами ізоляції несправностей та щонайменше трьома фізично і логічно незалежними зонами доступності в кожному регіоні. Ретельне управління ланцюжком поставок дозволило AWS забезпечити надійні потужності по всьому світу, незважаючи на безпрецедентний попит, в тому числі для потреб, пов'язаних з GenAI.

Підтримка рішень: AWS підтримує своїх клієнтів, надаючи оптимальні рішення, що відповідають їхнім вимогам, шляхом спільного проектування з архітекторами рішень AWS, які спеціалізуються на конкретних технологічних стеках, галузевих вертикалях та корпоративних робочих навантаженнях. Крім того, партнери можуть скористатися послугами архітекторів партнерських рішень, які допоможуть їм у розробці індивідуальних рішень для своїх клієнтів.

Надійний підтримка інструментів розробки: AWS пропонує значно більше наборів для розробки програмного забезпечення (SDK), ніж її конкуренти, що охоплюють більшість сучасних і застарілих мов програмування. SDK постійно оновлюються та послідовно підтримуються, гарантуючи, що зміни та оновлення API сервісів AWS можуть бути швидко використані розробниками будь-якою мовою.

Застереження:

Складні та непослідовні інтерфейси сервісів: AWS надає автономність своїм командам розробників у створенні та вдосконаленні сервісів незалежно від інших команд. Хоча ця стратегія збільшує швидкість, з якою AWS може виводити функції на ринок, вона також призводить до того, що сервіси часто є ізольованими і не мають єдиного дизайну. Це може створювати проблеми для клієнтів при вивченні та використанні різноманітних сервісів AWS разом для створення рішень.

Обмежена підтримка власних моделей штучного інтелекту: власні базові моделі Titan від AWS наразі не можуть конкурувати з провідними альтернативами у стандартних тестах, таких як Massive Text Embedding Benchmark (MTEB), і досягли лише обмеженої популярності на ринку. Поточна стратегія AWS полягає в тому, щоб запропонувати ряд first-party моделей і third-party в рамках Amazon Bedrock, не виділяючи свої власні моделі.

Менше сервісів у sovereign cloud: Хоча AWS оголосила, що її European Sovereign Cloud буде запущена в четвертому кварталі 2025 року, не всі сервіси AWS будуть доступні при запуску в цьому новому регіоні. AWS Snow Family та AWS EKS Anywhere – єдині хмарні пропозиції AWS з можливістю роботи в режимі disconnected. Локальні зони не є стандартними в багатозонній архітектурі і пропонують менше послуг AWS, ніж типовий регіон AWS.

Google є лідером у цьому магічному квадранті. Google є дочірньою компанією Alphabet Inc. Google Cloud Platform (GCP) призначена для підприємств, які прагнуть перевести свої IT на повністю хмарний підхід, з акцентом на послуги з використанням штучного інтелекту. Вона пропонує потужний портфель можливостей IaaS і PaaS, а також широку глобальну мережу хмарних регіонів. За останні кілька років здатність Google задовольняти традиційні потреби підприємств постійно покращувалася, включаючи більш надійне управління корпоративними обліковими записами, більшу партнерську мережу та більш інтегровану підтримку робочих навантажень VMware та Oracle.

Основними сильними сторонами GCP є контейнерна архітектура, передовий дизайн центрів обробки даних та провідні в галузі моделі GenAI. У поєднанні з активною спільнотою хмарних розробників це допомогло Google стрімко вийти на ринок SCPS. У 2023 році дохід GCP зріс на 28%, що більш ніж на 50% швидше, ніж індустрія публічних хмар в цілому. Її загальна частка на ринку зараз становить 11,5%.

Сильні сторони:

Модернізація IT з використанням штучного інтелекту: на додаток до потужних послуг GCP з управління контейнерами, платформа Vertex AI пропонує уніфіковану розробку AI/ML, інструменти для створення агентів, повний спектр провідних галузевих моделей і вбудовану генерацію коду з використанням власних високо оцінених моделей Gemini. Можливості штучного інтелекту (ШІ), такі як вбудовування векторів, добре інтегровані в сервіси баз даних GCP. Крім того, його рішення для управління API, Apigee, є провідним у галузі.

Екологічна стійкість: рейтинг екологічності GCP вищий, ніж у інших постачальників у цьому магічному квадранті. GCP надає пріоритет циркулярним процесам, використанню відновлюваних джерел енергії та енергоефективності центрів обробки даних. Ці фактори важливі для клієнтів у регіонах з нестабільними цінами на енергоносії та для тих, хто займається енергоємною хмарною діяльністю, наприклад, навчанням моделей штучного інтелекту. GCP пропонує окремі інструменти, які допомагають організаціям керувати викидами парникових газів (ПГ).

Цифровий суверенітет: GCP пропонує рішення, що поєднують інфраструктуру, дані та операційний суверенітет для організацій у високо локалізованих та регульованих юрисдикціях. Google Assured Workloads пропонує автоматизований контроль для забезпечення конфіденційності даних. Портфоліо Google Distributed Cloud (GDC) пропонує клієнтам рішення для розгортання GDC в локальних середовищах центрів обробки даних, відключення

від батьківських хмарних регіонів і, в деяких випадках, роботу з локальним хмарним операційним партнером.

Застереження:

Неповне розуміння традиційних потреб підприємств: стратегія продажів GCP робить акцент на хмарних архітектурах і штучному інтелекті. Традиційні корпоративні клієнти продовжують повідомляти Gartner, що, незважаючи на поліпшену підтримку застарілих робочих навантажень, Google не повністю розуміє і не задовольняє їхні потреби. Клієнти на Gartner Peer Insights оцінюють GCP лише як середній рівень обслуговування та підтримки. GCP часто випускає нові функції в режимі попереднього перегляду без опублікованого графіку загальної доступності та чітко не вказує, які нові сервіси або типи обчислювальних екземплярів доступні в яких регіонах.

Нерівномірна відмовостійкість: хмарні регіони GCP історично складаються з декількох зон доступності (AZ) в межах одного фізичного центру обробки даних. Для підвищення відмовостійкості GCP працює над створенням щонайменше трьох фізично та логічно різних зон доступності в кожному регіоні. Прогрес у цьому процесі в кожному регіоні не розголошується. GCP не має інструментів оркестрування IaaS для аварійного відновлення, що змушує клієнтів впроваджувати власні рішення.

Невідповідності розподіленої хмари: пропозиції розподіленої хмари GCP містять інтерфейс командного рядка (CLI) та набір служб GCP, які не сумісні зі стандартними загальнодоступними регіонами Google. Клієнти, які переходять на GDC, повинні знати, що існуючі робочі навантаження GCP повинні бути модифіковані для коректної роботи в GDC, а додатки, написані з використанням API сервісів GDC, не будуть сумісні з іншими середовищами GCP.

Microsoft є лідером у цьому магічному квадранті. Microsoft Azure підтримує широкий спектр послуг IaaS та PaaS, які відповідають усім потребам корпоративних IT-компаній. Azure надає надійні гібридні хмарні можливості, що

дозволяють підприємствам інтегрувати свої локальні середовища на базі Windows у хмару. Широка мережа партнерів Microsoft по всьому світу робить Azure логічним вибором для багатьох клієнтів на регіональних ринках.

Корпорація Майкрософт активно конкурує в кожній сфері хмарних обчислень та інтегрує Azure з іншими провідними хмарними платформами, щоб просувати на ринку ідею «краще разом». Microsoft також стратегічно співпрацює з іншими лідерами ринку, такими як SAP, VMware, Oracle, NVIDIA та OpenAI, пропонуючи спільні рішення. Завдяки стрімкому впровадженню інфраструктури та послуг, пов'язаних зі штучним інтелектом, дохід Azure зріс на 24% у 2023 році, а її частка на ринку публічних хмар зараз становить 26%.

Сильні сторони:

Універсальні можливості Microsoft: широкий спектр хмарних і нехмарних можливостей Microsoft робить Azure привабливим для організацій, які стратегічно орієнтовані на Microsoft як постачальника. Ключовими факторами інтеграції є використання Microsoft 365, Power BI, Power Apps і локального програмного забезпечення Microsoft, яке можна розширити до Azure.

Промислові хмари: Azure також слугує основою для широкомасштабної галузевої хмарної стратегії корпорації Майкрософт. Галузеві хмари Microsoft складаються з низки галузевих пропозицій, які об'єднують служби та інструменти з усіх хмар Microsoft. Azure підтримує галузеві хмари Microsoft, розширюючи свої основні служби IaaS і PaaS галузевими роз'ємами, моделями даних, системами безпеки та службами штучного інтелекту.

Стратегічне партнерство з OpenAI: альянс корпорації Майкрософт із постачальником моделей GenAI - компанією OpenAI - позиціонує Azure як провідну корпоративну платформу GenAI. Необхідність підтримки OpenAI також підштовхнула Azure до розгортання низки вдосконалень, пов'язаних зі штучним інтелектом, які покращують економічність великомасштабного навчання, включаючи водяне охолодження, вдосконалені кабелі, мікроконтрольні точки та Low-Rank Adaptation (LoRA).

Застереження:

Постійні проблеми з безпекою: другий рік поспіль цей магічний квадрант містить застереження, пов'язане з проблемами безпеки Microsoft у її хмарних сервісах, включаючи Azure. У квітні 2024 року Cyber Safety Review Board (CSRB) Міністерства внутрішньої безпеки США виступила з різкою критикою недостатньої культури безпеки корпорації Microsoft у зв'язку зі зломів її системи Exchange Online у 2023 році. У травні 2024 року Microsoft відреагувала на це розширенням своєї Secure Future Initiative (SFI). Невідомо, скільки часу пройде, перш ніж поліпшення в рамках SFI стануть помітними.

Нестача потужностей: клієнти Gartner повідомляють про дефіцит потужностей у багатьох регіонах Azure в США та Європі. Дефіцит потужностей призвів до обмеження пропозицій у Західній Європі (Амстердам) і південно-центральному регіоні США (Техас). Нерівномірна доступність потужностей Azure може ускладнити розгортання нових робочих навантажень або реалізацію стратегій обходу відмов для наявних робочих навантажень. Клієнти, які не можуть надати ресурси, можуть співпрацювати з корпорацією Microsoft, щоб домовитися про кредити, обмін зарезервованими екземплярами або відшкодування, а також обговорити, як ресурси можуть вплинути на зобов'язання щодо витрат у їхніх угодах [13].

1.3.ОГЛЯД ПЛАТФОРМИ KUBERNETES

1.3.1.ПРИЗНАЧЕННЯ KUBERNETES

Kubernetes – це платформа з відкритим вихідним кодом для управління контейнеризованими робочими навантаженнями та супутніми службами. Її основні характеристики — кросплатформенність, розширюваність, успішне використання декларативної конфігурації та автоматизації. Вона має гігантську, швидкопрогресуючу екосистему. [14].

Власне самі контейнери, які використовуються в Kubernetes – це чудовий варіант для упакування та запуску необхідних для роботи системи, додатків. У наближених до продовських умовах, командам інженерів потрібно керувати інформаційною системою, що складається з багатьох контейнерів та відслідковувати їх життєвий цикл. Наприклад, якщо один контейнер припиняє свою роботу, інший має бути запущений замість нього – таку властивість називають самозціленням. Якщо таких контейнерів в системі багато, то раціонально було б автоматизувати цей процес та дозволити системі робити це для них. Цю проблему Kubernetes і дозволяє вирішити. Він надає його користувачам можливість для гнучкого запуску розподілених систем, а також має здатність вирішувати питання масштабування та аварійного відновлення додатків, що використовуються в системах, має доступні шаблони для розгортання, тощо. Як приклад, Kubernetes також дозволяє реалізувати популярний підхід розгортання за стратегією canary. Розгортання за стратегією canary дозволяє розгорнути нову версію додатку, поступово замінюючи стару. Цей метод дозволяє зменшити ризик впровадження нової версії додатку, забезпечуючи плавний перехід для користувачів.

Kubernetes дозволяє вирішувати наступні задачі з:

Виявлення сервісів та балансування навантаження: Kubernetes може надавати доступ до контейнера, використовуючи DNS-ім'я або його власну IP-адресу. Якщо контейнер зазнає завеликого мережевого навантаження, Kubernetes

здатний збалансувати та розподілити його таким чином, щоб якість обслуговування залишалась стабільною.

Оркестрація сховища інформації в Kubernetes дозволяє вам автоматично монтувати системи збереження інформації на ваш вибір: локальні сховища, рішення від хмарних провайдерів тощо.

Автоматичне розгортання та відкатування – за допомогою Kubernetes можнаа описати бажаний стан контейнерів, що розгортаються, і він регульовано простежить за виконанням цього стану. Наприклад, ви можете автоматизувати в Kubernetes процеси створення нових контейнерів для розгортання, видалення існуючих контейнерів і передачу їхніх ресурсів на новостворені контейнери.

Автоматичне розміщення задач. Kubernetes кластер надається для запуску контейнерізованих задач і вказуєте, скільки ресурсів процесору та оперативної пам'яті необхідно для роботи кожного контейнера. Kubernetes розподіляє контейнери по вузлах кластера для максимально ефективного використання ресурсів.

Самозцілення у Kubernetes працює наступним чином: Kubernetes перезапускає контейнери, що відмовили; заміняє контейнери; зупиняє роботу контейнерів, що не відповідають на задану користувачем перевірку стану, і не відповідає на клієнтські запити (у випадку веб-додатків, наприклад), допоки ці контейнери не будуть у стані робочої готовності.

Управління секретами та конфігурацією – Kubernetes дозволяє зберігати та керувати чутливою інформацією, такою як паролі, OAuth токени та SSH ключі. Також наявна можливість розгортати та оновлювати секрети та конфігурацію без перезбирання образів ваших контейнерів, не розкриваючи секрети в конфігурацію стека [15].

1.3.2.СТРУКТУРА KUBERNETES

Узагальнена структура кластеру Kubernetes зображена на рис. 1.3:

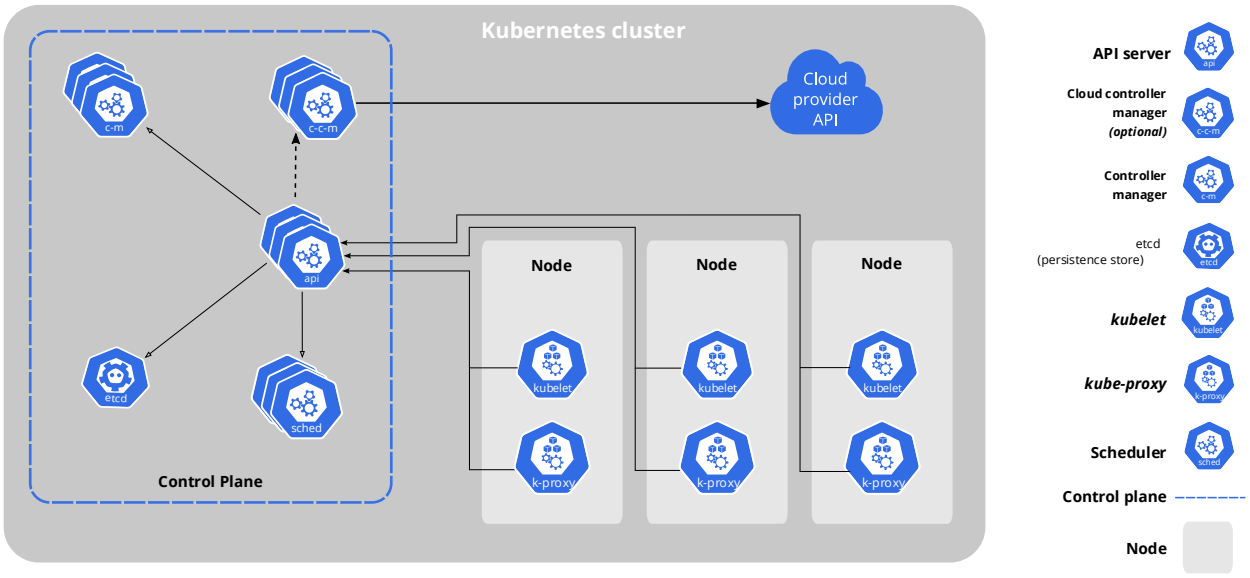


Рисунок 1.3 – Структуреа кластеру Kubernetes

Стандартного функціоналу, що надає середовище виконання контейнерів або ж самими контейнерами, зазвичай недостатньо для повноцінного журналювання.

Доступ до журналів може знадобитись у випадку наступних подій: container crash, Pod eviction, node death або ж для дослідження інцидентів безпеки для створення відповідних сповіщень для команд безпеки, або ж дослідження після того як проникнення в систему уже відбулося.

В кластерах Kubernetes журнали повинні мати окреме сховище і життєвий цикл їх зберігання, незалежний від вузлів, контейнерів або Pods. Даний концепт називається жруналювання на рівні кластеру.

Журналювання та його архітектура на рівні кластеру вимагає окремих ресурсів для зберігання та аналізу цих даних. У Kubernetes відсутнє рішення для зберігання даних журналів за замовчуванням. Натомість існує багато рішень для ведення журналів, які інтегруються з Kubernetes [21].

1.3.3.WINDOWS TA LINUX КЛАСТЕРИ KUBERNETES

Linux та Windows Kubernetes кластери представляють дві різні екосистеми, що використовуються для оркестрації контейнерів, кожна з яких має свої особливості та переваги. Розглянемо їх основні відмінності та поширення.

Linux Kubernetes кластери є найбільш традиційним і широко вживаним варіантом, адже Kubernetes було спочатку розроблено для операційних систем на базі Linux. Linux має велику спільноту, багато дистрибутивів і є відкритим рішенням з великою підтримкою незалежних розробників та великих компаній, типу Google, Microsoft, Red Hat, тощо. Велика гнучкість дозволяє операційних систем на базі Linux дозволяє користувачам легко модифікувати та налаштовувати систему під свої потреби. Він забезпечує велику гнучкість і контроль, широкі можливості для автоматизації та інтеграції з іншими інструментами DevOps. Також Linux підтримує широкий спектр мережевих плагінів, що робить його ідеальним для комплексних кластерних конфігурацій. В якості підтвердження цього є наявна велика кількість немалої кількості мережевих планів, як плагінів з відкритим кодом, типу kubenet та Calico, так і пропрієтарних, типу Azure CNI.

Windows Kubernetes кластери, з іншого боку, стали доступними пізніше і орієнтовані на організації, які використовують Windows-орієнтовані системи. Вони дозволяють запускати контейнери Windows в екосистемі Kubernetes, надаючи таким чином можливість інтеграції з існуючими Windows-орієнтованими програмами та сервісами. Це важливо для багатьох корпоративних середовищ, де Windows має сильні позиції. Однак, Windows кластери можуть мати певні обмеження в плані мережевої конфігурації та підтримки додаткового програмного забезпечення порівняно з Linux.

Що стосується популярності, Linux кластери мають більш широке використання в світі Kubernetes, особливо у великих та технологічно орієнтованих компаніях, які цінують його відкритість, гнучкість та масштабованість. Багато інструментів та розробок спільноти спочатку з'являються саме для Linux. Тим не менш, зі зростанням потреби у інтеграції з Windows-специфічними системами, Windows кластери також здобувають популярність, особливо в середовищах, де вже присутня значна інвестиція в технології Microsoft.

У виборі між Linux та Windows для Kubernetes кластерів часто визначальними є специфічні потреби проєкту, вже існуюча інфраструктура та вподобання команди. Linux залишається переважним вибором для більшості нових розгортань Kubernetes через свою зрілість, підтримку спільноти та гнучкість, але Windows кластери продовжують займати свою нішу, особливо в корпоративних середовищах, де потрібна інтеграція з іншими продуктами Microsoft.

1.3.4.БАЗОВІ КОМПОНЕНТИ AZURE KUBERNETES SERVICE

Azure Kubernetes Service (AKS) – це керована служба Kubernetes, яку можна використовувати для розгортання та керування контейнеризованими системами. Для використання AKS потрібні мінімальні знання з оркестрування контейнерів. AKS зменшує складність і операційні затрати на керування Kubernetes, перекладаючи більшу частину відповідальності на Azure. AKS є ідеальною платформою для розгортання та керування контейнеризованими системами, які потребують високої доступності, масштабованості та портативності, а також для розгортання програм у різних регіонах, використання інструментів з відкритим вихідним кодом та інтеграції з наявними інструментами DevOps.

AKS – є CNCF сертифікованим проєктом та відповідає стандартам SOC, ISO, PCI DSS та HIPAA.

Azure Kubernetes Service (AKS) спрощує розгортання керованого кластера Kubernetes в Azure, беручи частину операційної відповідальності на себе [22].

Розгорнутий в Azure кластер Kubernetes ряд має ряд рішень, які автоматично імплементовано для усіх користувачів, що спрощує управління кластером і також має рішення, які можна також імплементувати з Azure Portal (див. рис.1.3) або інших рішень, що використовують Azure Resource Manager у своїй роботі (наприклад, рішення для журналювання та збору метрик).

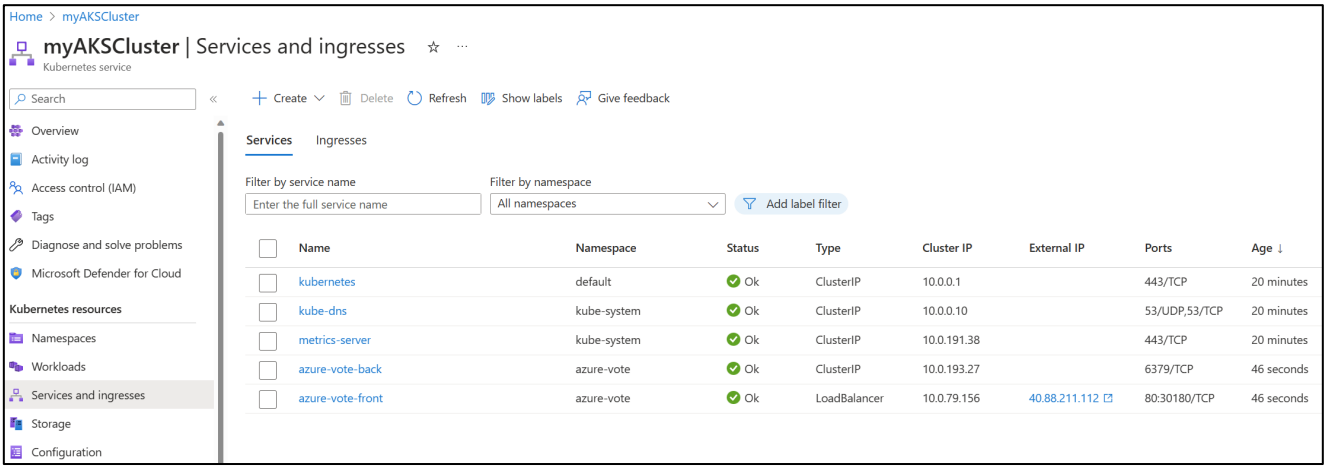


Рисунок 1.4 - Один з пунктів мені налаштування AKS в Azure Portal

Коли створюється кластер AKS, Azure автоматично створює та налаштовує площину керування, яка функціонує безкоштовно. Узагальнена структура кластеру AKS представлена на рис.1.5.

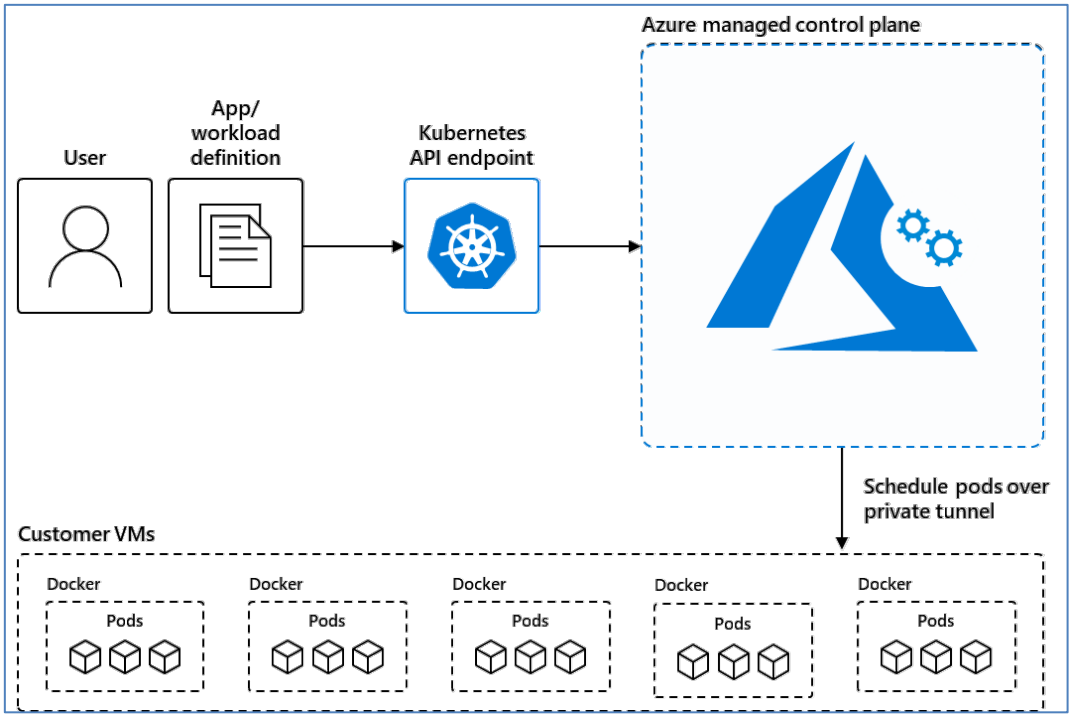


Рисунок 1.5 - Узагальнена архітектура кластеру AKS

Azure керує площиною керування AKS, яка відповідає за об'єкти Kubernetes і робочі вузли, що створюються кінцевими користувачами для запуску своїх програм. Azure піклується про критичні операції, такі як моніторинг стану та обслуговування, а плата здійснюється за вузли AKS, на яких працюють розгорнуті системи.

В таблиці 1.2 представлені основні можливості AKS:

Таблиця 1.2 – Ключовий функціонал AKS

Функція	**Опис**
Управління identeties та управління	1. Можливість впровадження засобів контролю відповідності нормативним вимогам за допомогою Azure Policy. 2. Інтеграція з Kubernetes RBAC для обмеження доступів. 3. Використання Entra ID для налаштування доступу до Kubernetes на основі наявних identities та груп.
Журналювання та моніторинг	1. Інтеграція з Container Insights, для моніторингу подій. 2. Налаштування розширеного моніторингу мережевого трафіку для його подальшого збору та аналізу.
Спрощення розгортання	1. Розгортання кластерів AKS з використанням наявних шаблонів для різних сценаріїв. 2. Автоматичне масштабування частин систем з Kubernetes Event Driven Autoscaler (KEDA). 3. Використання Draft для AKS для підготовки розгортання додатків в продакшн середовище.
Кластери та вузли	1. Підключення сервісів для зберігання даних, оновлення компонентів кластеру та використовувати графічні процесорів. 2. Створення кластерів з декількома пулами вузлів для підтримки змішаних систем. 3. Використання масштабування за допомогою cluster autoscaler та horizontal pod autoscaler. 4. Розгортання кластерів з конфіденційними вузлами.
Підтримка storage volume	1. Монтування статичних та динамічних дисків для даних. 2. Використання сервісу Azure Container Storage. 3. Використання Azure Disk CSI driver, коли потрібен доступ до диску для одного Pod та Azure Files CSI driver, коли він потрібен кільком подам.
Мережі	1. Гнучкі налаштування мережі. 2. Можливість встановлення сторонніх Container Network Interface (CNI). 3. Використання application routing доповнень, типу Nginx Ingress або Application Gateway Ingress Controller.
Інструменти для розробників	1. Інтрументи Azure для Helm. 2. Інтрументи Kubernetes для Visual Studio Code. 3. Доповнення Istion для AKS

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ

Після проведеного аналізу в рамках першого підрозділу були зроблені наступні дослідження та висновки на їх основі:

- Проведено порівняння віртуальних машин та контейнерів. Були розглянуті особливості кожної з технологій, їх переваги та особивості з різних аспектів,

в тому числі, аспектів безпеки. Контейнеризовані системи стають дедалі популярнішим рішенням для розробки інформаційних систем, проте ряд викликів для захисту даних систем збільшився, порівняно з системами, що використовують віртуальні машини. Тому вирішення даних проблем та захист цих систем потребує глибинних знань для забезпечення безпеки цих систем.

- Проаналізовано які бувають середовища виконання контейнерів, які з них можуть використовуватись в Kubernetes та визначено, що containerd (низькорівневе середовище виконання контейнерів) наразі використовується в Kubernetes за замовчуванням, в тому числів і в Azure Kubernetes Service. Окрім того, Docker все одно досі є популярним рішенням, що використовується розробниками для створення контейнеризованих додатків та систем. На основі образів, що створені за допомогою Docker можливо запукати контейнери в Kubernetes. Знання основних принципів функціонування середовищ виконання контейнерів, їх застосування, команди інтерфейсу командного рядку, інтрументів взаємодії з контейнерами є невід'ємною частиною для побудови безпечних контейнеризованих систем та дозволить уникнути потенційних конфігурацій, що містимимуть вразливості.
- Встановлено етапи життєвого циклу контейнерів. Це в свою чергу дозволяє глибше зрозуміти на якому з етапів циклу зловмисник може втрутитись та виконати потенційно шкідливі дії для системи, що будується чи експлуатується. Варто відзначити, що порівняно з ВМ, оновлення контейнерів та їх патчинг може здійснюватись досить швидко за рахунок портативності та легкості конфігурації певних частин контейнерів. Наприклад, якщо вразливість знаходиться в базовому образі контейнеру.
- Здійснено аналіз хмарних провайдерів на основі «Magic Quadrant for Strategic Cloud Platform Services» та було вирішено, що через наявні переваги, порівняно з іншими хмарними провайдерами, варто проводити аналіз методів підвищення відмовостійкості на прикладі провайдера хмарних рішень

Microsoft Azure. Хоча користувачі Microsoft Azure та незалежні аналітики відзначають, що питання безпеки в компанії Microsoft потрібно поліпшувати.

- Було проаналізовано структуру кластеру Kubernetes. Знання принципів функціонування кластерів є надзвичайно важливими для розуміння як взаємодіють компоненти кластеру та що може статись потенційно, якщо один з компонентів буде скомпроментовано, який буде радіус враження, тощо.
- Встановлено різницю між Linux та Windows кластерами Kubernetes, та встановлено, що перші є більш популярними за рахунок відкритості та підтримки. Окрім того, так склалось історично, що першими були створені саме Linux кластери Kubernetes і саме кластери Linux значно випереджають в поширеності на сьогодні, а, відповідно, й у підпримці спільнотою та корпоратиціями, накшталт Microsoft.
- Проаналізовано архітектурні особливості кластеру AKS, встановлено які компоненти Kubernetes встановлюються в кластері AKS, який функціонал доступний в AKS, в тому числі які функції безпеки можливо застосувати
- Таким чином було здійснений комплексний аналіз технологій, що використовуються в контейнеризованих рішеннях, проаналізовано як вони пов’язані між собою, як вони функціонують, що дозволить продовжити дослідження методів підвищення відмовостійкості даних систем.

2. АНАЛІЗ МЕТОДІВ ПІДВИЩЕННЯ ЗАХИСТУ LINUX КЛАСТЕРУ AKS

2.1. АНАЛІЗ МЕТОДІВ ЗАХИСТУ КЛАСТЕРУ AZURE KUBERNETES SERVICE ІНСТРУМЕНТАМИ ХМАРИ

2.1.1. AZURE WELL-ARCHITECTED FRAMEWORK

У цьому підрозділі розглянемо рекомендації та найкращі практики для конфігурації кластерів AKS.

Azure Well-Architected Framework допомагає архітекторам у створенні та аналізі хмарних робочих навантажень за допомогою рекомендацій, що охоплюють п'ять стовпів – *надійність*, *безпеку*, *оптимізацію витрат*, *операційну досконалість* та *ефективність роботи*. Вони є наскрізними і охоплюють весь обсяг робочого навантаження. Крім того, архітектору часто корисно зосередитися на ключовому компоненті робочого навантаження, а Well-Architected Framework також надає цілеспрямовані, орієнтовані на конкретну послугу рекомендації щодо ключових компонентів робочого навантаження. Як і всі посібники з обслуговування Well-Architected Framework, посібник Azure Kubernetes Service (AKS) містить контрольні списки та рекомендації за тими самими п'ятьма основними принципами, але з акцентом на архітектурах робочих навантажень, які включають AKS. Для тих, хто тільки починає створювати архітектуру з AKS або вже використовуєте її у виробництві, контрольні списки та рекомендації з цього посібника допоможуть вам визначити можливості для оптимізації вашого рішення та переконатися, що ви перебуваєте на правильному шляху.

Розглянемо детально рекомендації безпеки в Well-architected framework. Безпека є одним з найважливіших аспектів будь-якої архітектури. Щоб дізнатись, як використання AKS може посилити безпеку інформаційних систем, Microsoft рекомендують дотримуватись принципу безпечного дизайну системи (security design principles).

Security design principles в контексті хмарної інфраструктури Azure базуються на фундаментальних засадах захисту даних та робочих навантажень. По-перше, в основі будь-якої архітектури лежить підхід до безпеки за

замовчуванням, тобто ми намагаємося передбачити всі ймовірні сценарії загроз і зробити налаштування так, аби система була захищеною “з коробки”. Важливо будувати наші рішення так, аби навіть при несподіваному компрометуванні одного з елементів мережі або додатка всі критичні ресурси все одно залишались недоступними для зловмисника. Це тісно пов’язано з принципом мінімальних привілеїв, коли кожен користувач, служба чи компонент мають лише той обсяг доступу, який справді потрібен для виконання їхніх завдань.

Другим критичним принципом є “defense in depth”: архітектура розробляється з таким підходом, щоб кожен рівень захисту дублювався й зміцнювався наступним, створюючи ускладнення для потенційного зловмисника на всіх етапах. Навіть у випадку подолання одного рівня не можна відразу отримати доступ до найцінніших даних або системних ресурсів, адже ще лишаються додаткові захисні бар’єри. Не менш важливо забезпечувати контроль identities та управління доступом на найвищому рівні, уникаючи загального спільного облікового запису й унеможливлюючи несанкціоновані спроби використати пропущені в налаштуваннях “дірки”.

Загалом суть security design principles полягає в тому, щоб зробити рішення керованими, стійкими, здатними до своєчасного виявлення загроз і адаптації під нові виклики. Це охоплює планування стратегії моніторингу й журналювання, автоматизацію перевірок безпеки та покриття максимальної кількості сценаріїв потенційних вразливостей. Якщо все реалізувати грамотно, то ваші сервіси чи додатки будуть стійкими до найпоширеніших атак і витримають навіть висококваліфіковані спроби вторгнення [23].

Якщо ваш кластер Azure Kubernetes Service потрібно спроектувати для виконання чутливого робочого навантаження, яке відповідає нормативним вимогам Payment Card Industry Data Security Standard (PCI-DSS 3.2.1), то варто звернути увагу на відповідну статтю документації Microsoft.

Основна ідея даних нормативів полягає у створенні моделі, яка враховує всі вимоги безпеки, контроль доступу, а також необхідну сегментацію мережі, аби гарантувати збереження даних про платіжні картки. Microsoft пропонують

архітектуру, де AKS розглядається як ключова компонента, а в поєднанні з усіма підтримувальними сервісами Microsoft Azure – від вбудованих засобів моніторингу й аудитів до спеціалізованих інструментів для виявлення загроз – досягається додержання вимог PCI DSS.

Основний акцент робиться на захисті контейнеризованих робочих навантажень: дотримується принцип “захисту в глибині” (defense in depth), тому робочі вузли AKS розміщуються в окремих сегментах, ізольованих за допомогою мережевих політик і механізмів Kubernetes. Для керування доступом застосовуються обов’язкові правила найменших привілеїв: кожен компонент чи служба отримує тільки ті права, що необхідні для його роботи. Крім того, рекомендується інтегрувати сервіси типу Entra ID для централізованого управління обліковими записами та групами.

Ще одна ключова частина – це моніторинг і аудит. Для відповідності PCI DSS завжди потрібно забезпечувати детальне журналювання всіх критичних подій і спроб підключення, з подальшим аналізом цих журналів. Azure пропонує низку сервісів для цього (зокрема, Azure Monitor та Azure Log Analytics), що допомагають одразу помічати й досліджувати підозрілі активності. Microsoft також радять застосовувати Azure Policy, щоб автоматизувати процеси впровадження стандартів безпеки й уніфікувати політики доступу та налаштувань.

Загалом же, якщо реалізувати рекомендації, наведені в архітектурі, то можна створити гнучке й захищене оточення в AKS, яке не лише відповідає PCI DSS, а й має резерви для масштабування та швидкої адаптації до майбутніх вимог індустрії платіжних карт.

Обговорюючи безпеку в Azure Kubernetes Service, важливо розрізняти *безпеку кластера* та *безпеку робочих навантажень*. За безпеку кластера несуть спільну відповідальність адміністратор кластера та постачальник ресурсів, тоді як за безпеку робочих навантажень відповідає розробник. Microsoft надає рекомендації для Azure Kubernetes Service для обох цих ролей.

У наведеній нижче таб. 2.1 розглянуто рекомендації, що стосуються або безпеки кластеру, або безпеки робочих навантажень.

Таблиця 2.1 – Рекомендації захисту кластеру або його робочих навантажень

Категорія	**Рекомендація**
Архітектура кластеру	Використовуйте Managed Identities, щоб уникнути необхідності керувати та регулярно змінювати облікові дані (service principals).
Архітектура кластера	Застосовуйте Kubernetes RBAC (role-based access control) разом із Microsoft Entra ID, щоби надати доступ із найменшими привілеями та мінімізувати надання прав адміністратора, захищаючи конфігурацію та секрети.
Архітектура кластера	Використовуйте Microsoft Defender for Containers разом із Azure Sentinel, аби виявляти загрози й оперативно реагувати на них у кластері та на робочих навантаженнях, що там виконуються.
Архітектура кластера	Розгортайте приватний кластер AKS, щоб увесь керуючий трафік до вашого API-сервера залишався в межах приватної мережі. Для неприватних кластерів можна налаштувати список дозволених IP-адрес (allow list) для API-сервера.
Архітектура робочих навантажень	Використовуйте Web Application Firewall для захисту HTTP(S)-трафіку.
Архітектура робочих навантажень	Переконайтеся, що ваш конвеєр CI/CD достатньо захищений, зокрема має сканування, орієнтоване на контейнери.

Окрім того Microsoft доповнюють ці рекомендації наступним матеріалом, що подано в табл. 2.2.

Таблиця 2.2 – Рекомендації для захисту інфрастукрури, що використовує AKS

Рекомендація	**Перевага**
Архітектура кластера: Використовуйте інтеграцію з Microsoft Entra.	Використання Microsoft Entra ID централізує керування ідентифікацією. Будь-яка зміна у статусі облікового запису чи групи автоматично оновлюється в доступі до кластера AKS. Розробники та власники застосунків вашого кластера Kubernetes потребують доступу до різноманітних ресурсів.

Архітектура кластера: Автентифікуйтеся в Azure Container Registry за допомогою Microsoft Entra ID.	AKS та Microsoft Entra ID дають змогу здійснювати автентифікацію в Azure Container Registry без використання secrets imagePullSecrets.
Архітектура кластера: Захищайте мережевий трафік до вашого API-сервера за допомогою приватного кластера AKS.	За замовчуванням, мережевий трафік між пулами вузлів і API-сервером проходить через магістральну мережу Microsoft. Використання приватного кластера гарантує, що трафік до API-сервера залишатиметься лише в межах приватної мережі.
Архітектура кластера: Для публічних кластерів AKS варто використовувати дозволені IP-діапазони (authorized IP ranges) для API-сервера.	При використанні публічних кластерів ви все одно можете обмежити трафік, який досягає API-сервера кластера, за допомогою функції дозволених IP-діапазонів. Варто врахувати джерела на кшталт публічних IP-адрес агентів збірки, інструментів керування та вихідних точок пулів вузлів (наприклад, Azure Firewall).
Архітектура кластера: Захищайте API-сервер за допомогою Microsoft Entra RBAC.	Забезпечення безпеки доступу до Kubernetes API-сервера – один із найважливіших кроків у захисті кластера. Інтегруйте Kubernetes RBAC із Microsoft Entra ID для контролю доступу до API-сервера. Вимкніть локальні облікові записи, аби доступ до кластера здійснювався тільки через ідентичності Entra ID.
Архітектура кластера: Використання політики мережі Azure або Calico.	Захист та контроль мережевого трафіку між подами в кластері.
Архітектура кластера: Захист кластери та поди за допомогою Azure Policy.	Azure Policy допомагає централізовано й масштабовано впроваджувати необхідні політики й правила безпеки в кластерах. Також він контролює, які операції дозволені для подів і чи виконується щось усупереч політиці компанії.
Архітектура кластера: Захистіть доступ контейнерів до ресурсів.	Обмеження доступ до дій, які можуть виконувати контейнери. Надання мінімально необхідних дозволів й уникання використання привілейованих або root-прав.

Продовження таблиці 2.2

Архітектура робочих навантажень: Використовуйте Web Application Firewall для захисту HTTP(S)-трафіку.	Щоб перевіряти вхідний трафік на потенційні атаки, використовуйте веб-екран (Web Application Firewall), наприклад Azure Web Application Firewall (WAF) у зв'язці з Azure Application Gateway або Azure Front Door.
Архітектура кластера: Контролюйте вихідний трафік кластера.	Переконайтеся, що весь вихідний трафік кластера проходить через мережевий безпековий шлюз (Azure Firewall) або HTTP-проксі.
Архітектура кластера: Використовуйте відкритий Microsoft Entra Workload ID та Secrets Store CSI Driver разом із Azure Key Vault.	Захищайте і регулярно ротуйте секрети, сертифікати та рядки підключення в Azure Key Vault завдяки надійному шифруванню. Це забезпечує аудит доступу та допомагає уникнути зберігання ключових секретів у конвеєрі розгортання.
Архітектура кластера: Використовуйте Microsoft Defender for Containers.	Архітектура кластера: Використовуйте Microsoft Defender for Containers.

Додатково в Azure наявний такий інструмент, як Azure Advisor. Azure Advisor допомагає забезпечити та покращити роботу служби Azure Kubernetes. Він надає рекомендації щодо підмножини елементів, перелічених у розділі політики нижче, таких як кластери без налаштованого RBAC, відсутність конфігурації Microsoft Defender, відстуність обмежень мережного доступу до API-серверу. Аналогічно, він надає рекомендації щодо робочого навантаження для деяких інших елементів кластеру [24].

2.1.2.AZURE POLICY

Azure Policy дає змогу забезпечувати дотримання організаційних стандартів і здійснювати оцінку відповідності в масштабі всієї інфраструктури. Завдяки інтегрованій панелі моніторингу відповідності створюється зведений огляд загального стану середовища з можливістю деталізувати інформацію до конкретного ресурсу або полісі. Вважається доречним застосовувати Azure Policy у різних випадках, зокрема для впровадження єдиних правил керування ресурсами, відповідності вимогам регуляторних органів, підвищення безпеки, оптимізації витрат і поліпшення керованості.

В Azure Policy існують різні компоненти та можливості. Політики допомагають визначити бажану конфігурацію, якій мають відповідати ресурси. Існує можливість встановити певні параметри, за якими перевіряються ресурси і, в разі потреби, до них застосовується автоматичне виправлення. Поняття ініціатив об'єднує декілька політик у один набір, що спрощує управління та повторне використання. За допомогою цих інструментів можливо охопити широкий спектр вимог, починаючи з обмежень конкретних SKU для віртуальних машин і завершуючи сценаріями контролю мережевих параметрів.

Важливо розуміти, що Azure Policy працює в режимі оцінки та примусу. Під час оцінки відбувається аналіз уже наявних ресурсів: визначається, чи відповідають вони зазначеним критеріям. Завдяки панелі моніторингу відповідності можна отримати зведене уявлення про відхилення в конфігурації або потенційні проблеми. Одночасно діють і механізми примусу: якщо ресурс не відповідає визначеним правилам, передбачено можливість виконувати автокорекцію або забороняти створення чи оновлення невідповідних ресурсів.

Управління політиками може відбуватися на різних рівнях: від певної підписки й ресурсної групи до окремих екземплярів ресурсів. Також можливо інтегрувати Azure Policy з іншими службами, зокрема використовувати його разом із Azure Blueprints. Це дає змогу охопити політики, шаблони ARM та ініціативи в єдиному рішенні для впровадження Azure-архітектури в комплексі.

Практика показує, що ефективне використання Azure Policy може значно зменшити навантаження на команди, які відповідають за безпеку й відповідність стандартам. Завдяки централізованому керуванню політиками вдається уникнути помилок чи відхилень у конфігураціях, що особливо важливо в складних середовищах із великою кількістю ресурсів. Окрім того, завдяки докладному журналюванню дій і стану політик надається змога вдосконалювати підхід до безпеки та дотримання внутрішніх або зовнішніх нормативних вимог.

Загалом, Azure Policy пропонує зручний спосіб упорядкувати керування ресурсами й забезпечити відповідність корпоративним чи регуляторним правилам. Рекомендується звернути увагу на планування правил і ініціатив перед

масовими розгортаннями, щоб проактивно забезпечувати відповідність нових ресурсів установленим вимогам. Такий підхід допомагає впорядкувати життєвий цикл ресурсів в Azure, унеможливлюючи застосування небажаних конфігурацій, і робить середовище більш надійним і керованим.

Azure Policy також допомагає встановлювати глобальні правила в середовищі Azure та контролювати їх виконання, що особливо корисно при дотриманні вимог різноманітних стандартів і сертифікацій, як-от ISO 27001, PCI DSS, SOC 2 тощо. Ці стандарти вимагають, аби у хмарній інфраструктурі існував узгоджений і контрольований набір правил і процесів, спрямованих на захист конфіденційних даних та забезпечення їх цілісності й доступності. Використовуючи Azure Policy, можливо визначити єдину політику – або навіть цілу групу політик (ініціативу) – яка накладає конкретні вимоги на всі ресурси в хмарі, автоматично перевіряє, чи вони їм відповідають, і запобігає створенню або зміні ресурсів, що не дотримуються встановлених правил.

Завдяки панелі моніторингу відповідності в Azure Policy існує можливість у реальному часі відстежувати рівень відповідності різноманітним нормам і вимогам. До прикладу, якщо організація прагне підтримувати відповідність PCI DSS, можна створити набір політик, що регулюють, як зберігаються й обробляються дані власників платіжних карт. Аналогічно в рамках ISO 27001 важливо мати контроль над мережевою інфраструктурою та доступом до даних. Налаштувавши набір обов’язкових правил, вдається обмежити, наприклад, доступ до деяких сервісів або забезпечити, що всі ресурси розміщуються тільки у вибраних регіонах Azure для запобігання несанкціонованій міграції даних. А у випадку дотримання SOC 2 варто зробити акцент на безпеці, надійності й процесах обробки даних, тож Azure Policy дозволяє систематично перевіряти виконання цих вимог і повідомляти про порушення ще до того, як воно вплине на сервіси.

Окрім того, з Azure Policy вдається інтегрувати захист у конвеєри CI/CD через механізми policy as code. Це означає, що будь-які нові ресурси або конфігурації, які не відповідають діючим нормативам, не можуть бути

впроваджені. Таким чином, команди, які відповідають за DevOps-процеси, отримують автоматичне блокування або виправлення порушень політик до того, як вони потраплять у production. Це значно знижує ризик ненавмисних помилок конфігурації, які часто є слабким місцем у дотриманні стандартів ISO, PCI, SOC та інших подібних регуляцій.

Підсумовуючи, Azure Policy забезпечує централізоване й автоматизоване керування вимогами безпеки та відповідності, що має вирішальне значення при проходженні аудитів або сертифікації на відповідність різним стандартам. Такий підхід дозволяє не лише зберігати прозорість і контроль за конфігураціями, а й дає змогу оперативно реагувати на виявлені проблеми, роблячи хмарну інфраструктуру надійною, керованою й повністю відповідною вимогам індустрії.

Azure Policy для Azure Kubernetes Service (AKS) допомагає централізовано керувати станом кластерів і робочих навантажень, запроваджуючи набір правил, яким мають відповідати всі ресурси у кластері. Це робиться за допомогою ініціатив і політик, що накладають обмеження на різні аспекти функціонування AKS, починаючи від мережевих політик і завершуючи конкретними версіями Kubernetes, які дозволено використовувати. Завдяки цьому вдається автоматично відстежувати порушення і, в разі потреби, блокувати або виправляти невідповідні конфігурації. Результати оцінок можна переглядати в панелі моніторингу відповідності (Compliance), яка дає змогу швидко визначати, де саме виникли відхилення від вимог. З інтеграцією підходу “policy as code” можна вбудувати політики в процеси CI/CD, отже ще до розгортання контейнерів у кластері перевіряється, чи вони відповідають вимогам безпеки й корпоративним стандартам. Такий підхід забезпечує контроль на всьому життєвому циклі розгортання, допомагає гарантувати, що AKS-кластери залишаються безпечними, керованими та відповідають регуляторним вимогам або внутрішнім нормам компанії.

Таблиця 2.3 – Ініціативи для AKS

Ім'я	**Опис**	**Кі-ть політик**	**Версія**
[Попередній перегляд]:Використовуйте цілісні образи, щоб забезпечити розгортання лише довірених образів	Використовуйте цілісність образів, щоб переконатися, що кластери AKS розгортають лише надійні образи, увімкнувши надбудови цілісності образів і політики Azure на кластерах AKS. Надбудови Image Integrity Add-On і Azure Policy Add-On є необхідними умовами для використання Image Integrity для перевірки того, чи підписаний образ під час розгортання.	3	1.1.0-preview
[Попередній перегляд]:Засоби захисту при розгортанні повинні допомогти розробникам орієнтуватися на найкращі практики, рекомендовані AKS	Збірка найкращих практик Kubernetes, рекомендованих Azure Kubernetes Service (AKS). Щоб отримати найкращий досвід, використовуйте засоби захисту розгортання, щоб призначити цю ініціативу політики: https://aka.ms/aks/deployment-safeguards. Надбудова політики Azure для AKS є необхідною умовою для застосування цих найкращих практик до ваших кластерів	20	1.9.0-preview
Базові правила безпеки подів для робочих навантажень на базів Linux	Ця ініціатива включає політики для базових стандартів безпеки кластерних модулів Kubernetes. Ця політика доступна для Kubernetes Service (AKS), а також у попередній версії для Kubernetes з підтримкою Azure Arc.	5	1.4.0
Обмежені стандарти безпеки кластерних модулів Kubernetes для робочих навантажень на базі Linux	Ця ініціатива включає політики для обмежених стандартів безпеки кластерних блоків Kubernetes. Ця політика є загальнодоступною для Kubernetes Service (AKS), а також у попередньому перегляді для Kubernetes з підтримкою Azure Arc.	8	2.5.0

В Azure Policy існує кілька основних дій, які називають ефектами (effects). Ці ефекти визначають, як політика поводитиметься при оцінюванні відповідності ресурсу вимогам. Найпростішим для розуміння є ефект Deny, який одразу блокує створення або зміну ресурсу, що не відповідає визначеній

політиці. Це означає, що ви ніколи не зможете успішно розгорнути або модифікувати ресурси, які порушують умови політики, адже такі спроби будуть автоматично відхилені.

Існують також ефекти, що не блокують дії, а дозволяють здійснювати аналітику або автоматичні виправлення. До прикладу, Audit просто фіксує порушення для подальшого аналізу, але не перешкоджає створенню чи зміні ресурсу. Це зручно, коли ви хочете спочатку виявити наявні відхилення від стандартів, не заважаючи щоденній роботі колег. AuditIfNotExists поглиблює цю логіку, оскільки він відслідковує конфігурації та подає сигнал лише в тому випадку, якщо певний пов'язаний ресурс або тег відсутній. DeployIfNotExists, зі свого боку, не тільки виявляє відсутність конкретного ресурсу чи параметра, а й намагається автоматично розгорнути все необхідне, щоб привести середовище у відповідність до вимог. Це корисно для компаній, які прагнуть максимально зменшити кількість ручних рутинних завдань і зробити інфраструктуру цілісною та безпечною.

Важливою і цікавою дією є ефект Append, який додає або змінює певні поля чи теги у конфігурації, коли ви створюєте або змінюєте ресурси. По суті, воно "дописує" потрібні властивості, щоб ресурс гарантовано відповідав політиці без явного блокування процесу. Ще розширенішим сценарієм є ефект Modify. Він працює подібно до Append, але може робити складніші коригування у конфігурації ресурсу, змінюючи вже існуючі властивості або навіть проводячи пост-обробку безпосередньо після створення чи оновлення. Це відкриває широкий простір для реальних інженерних сценаріїв, коли потрібно впливати на мережеві параметри, безпекові налаштування або будь-які інші критичні атрибути.

Окремо варто згадати ефект Disabled, який повністю відключає перевірку політики. Застосовується рідко, але іноді корисний у виняткових ситуаціях, коли вам тимчасово потрібно вимкнути політику без її видалення. У більшості випадків довготривале використання Disabled не рекомендується, бо це потенційно може призвести до зниження рівня безпеки та створення різнорідних

конфігурацій у вашому середовищі. Утім, комбінація з інших ефектів, особливо з такими, як Audit та DeployIfNotExists, зазвичай дозволяє досягти балансу між жорсткою забороною та зручністю автоматичного виправлення конфігурацій.

Політика [Preview]: [Image Integrity] Kubernetes clusters should only use images signed by notation наголошує, що варто розгортати лише образи, які підписані за допомогою інструмента notation. Такий підхід гарантує надходження образів із надійних джерел і запобігає їх зловмисному модифікуванню. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика [Preview]: Azure Backup Extension should be installed in AKS clusters описує необхідність встановити розширення Azure Backup в кластери AKS, щоб скористатися перевагами Azure Backup. Це безпечне хмарне рішення для захисту даних дає змогу відновлювати робочі навантаження у разі непередбачуваних подій. Дана політика підтримує такі ефекти: AuditIfNotExists, Disabled.

Політика [Preview]: Azure Backup should be enabled for AKS clusters зазначає, що потрібно налаштувати Azure Backup для захисту ваших кластерів AKS. Azure Backup для AKS є безпечним та хмарно-орієнтованим рішенням для збереження даних у кластерах. Дана політика підтримує такі ефекти: AuditIfNotExists, Disabled.

Політика [Preview]: Azure Kubernetes Service Managed Clusters should be Zone Redundant рекомендує налаштовувати кластери AKS у режимі «Zone Redundant». Вона перевіряє вузли в пулі та переконується, що для кожного використано зони доступності. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Cannot Edit Individual Nodes забороняє вносити зміни безпосередньо в окремі вузли кластера, закликаючи редагувати конфігурацію лише на рівні пулів вузлів. Це допомагає уникнути суперечливих налаштувань та підвищує безпеку. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Deploy Image Integrity on Azure Kubernetes Service передбачає розгортання надбудов Image Integrity та Policy Add-Ons у кластерах AKS. Вона дає змогу перевіряти підпис образу під час розгортання, запобігаючи

використанню неперевірених контейнерних образів. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика [Preview]: Install Azure Backup Extension in AKS clusters (Managed Cluster) with a given tag. наголошує на потребі встановлювати розширення Azure Backup у всі кластери AKS із певним тегом. Таким чином забезпечується централізоване резервне копіювання у масштабі. Дана політика підтримує такі ефекти: AuditIfNotExists, DeployIfNotExists, Disabled.

Політика [Preview]: Install Azure Backup Extension in AKS clusters (Managed Cluster) without a given tag. діє аналогічно, але охоплює кластери, в яких відсутній зазначений тег. Це дає змогу впровадити Azure Backup скрізь, де він ще не налаштований. Дана політика підтримує такі ефекти: AuditIfNotExists, DeployIfNotExists, Disabled.

Політика [Preview]: Kubernetes cluster container images must include the preStop hook запроваджує вимогу, щоб контейнери у кластері мали preStop-хук для коректного завершення процесів під час зупинки подів. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Kubernetes cluster container images should not include latest image tag вимагає використовувати конкретні версії образів замість тегу latest, оскільки це полегшує відтворення середовища, відкат і відстеження оновлень. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Kubernetes cluster containers should only pull images when image pull secrets are present вказує, що завантаження образів контейнерів має відбуватись лише при наявності ImagePullSecrets. Це запобігає несанкціонованому або небезпечному завантаженню образів. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Kubernetes cluster services should use unique selectors уточнює, що всі служби у межах одного namespace мають унікальні селектори. Це допомагає однозначно спрямовувати трафік на потрібні поди. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Kubernetes cluster should implement accurate Pod Disruption Budgets забороняє невірні параметри Pod Disruption Budgets, аби забезпечити стабільну кількість працюючих подів під час оновлень чи збоїв. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Kubernetes clusters should restrict creation of given resource type обмежує розгортання певного типу ресурсу Kubernetes у визначеному просторі імен, якщо це вимагають корпоративні чи безпекові стандарти. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Must Have Anti Affinity Rules Set гарантує, що поди не розміщуватимуться всі разом на одному вузлі. Правила anti-affinity допомагають залишатися сервісам доступними, якщо певний вузол стає недоступним. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Mutate K8s Container to drop all capabilities вносить зміни (мутацію) до securityContext контейнерів, додаючи “ALL” у capabilities.drop. Це відключає всі можливості Linux-контейнера і знижує ризики підвищення привілеїв. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Mutate K8s Init Container to drop all capabilities накладає аналогічне обмеження на init-контейнери, щоб унеможливити небезпечні дії з надмірними правами. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: No AKS Specific Labels забороняє застосовувати мітки з префіксом kubernetes.azure.com, які зарезервовані під компоненти, що використовуються самою службою AKS. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Prevents containers from being ran as root by setting runAsNotRoot to true. блокує можливість запускати контейнери з правами root, підвищуючи загальний рівень безпеки. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Prevents init containers from being ran as root by setting runAsNotRoot to true. розповсюджує таку ж заборону на init-контейнери,

уникаючи випадків виконання від імені root. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Prints a message if a mutation is applied перевіряє, чи є анотації від мутації, і виводить службове повідомлення за їх наявності, що допомагає відстежувати факти зміни ресурсів. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика [Preview]: Reserved System Pool Taints обмежує використання taint CriticalAddonsOnly лише системним пулам вузлів, ізолюючи ними керовані компоненти від робочих навантажень користувача. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика [Preview]: Restricts the CriticalAddonsOnly taint to just the system pool. виконує мутацію, запобігаючи застосуванню taint CriticalAddonsOnly у призначених для користувача пулах, щоб уникнути конфліктів між системними та користувацькими подами. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets automountServiceAccountToken in the Pod spec in containers to false. відключає автоматичне монтування токена сервісного облікового запису, щоб зменшити ризик несанкціонованого використання прав доступу. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets Kubernetes cluster container securityContext.runAsUser fields to 1000, a non-root user id змінює параметри контейнерів, щоб вони працювали під користувачем із UID 1000, знижуючи поверхню для атак. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets Kubernetes cluster containers CPU limits to default values in case not present. автоматично задає ліміти CPU за замовчуванням, щоб не допустити сценаріїв, де контейнер безконтрольно споживає ресурси. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets Kubernetes cluster containers memory limits to default values in case not present. встановлює ліміти пам’яті за замовчуванням для

контейнерів, якщо вони не були явно задані, захищаючи кластер від переповнення пам’яті. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets Kubernetes cluster containers' secure computing mode profile type to RuntimeDefault if not present. вмикає профіль seccomp “RuntimeDefault” для контейнерів, щоб заборонити небезпечні системні виклики. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets Kubernetes cluster init containers securityContext.runAsUser fields to 1000, a non-root user id забезпечує, щоб init-контейнери також працювали під UID 1000, що зменшує шанси експлуатації вразливостей. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets Kubernetes cluster init containers' secure computing mode profile type to RuntimeDefault if not present. аналогічно встановлює seccomp-профіль RuntimeDefault для init-контейнерів, посилюючи безпеку. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets Kubernetes cluster Pod securityContext.runAsUser fields to 1000, a non-root user id зобов’язує всі поди працювати від імені UID 1000, уникаючи привілейованого запуску. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets maxUnavailable pods to 1 for PodDisruptionBudget resources встановлює обмеження максимум одного недоступного пода (maxUnavailable=1), щоб підтримувати стабільність сервісу під час оновлень. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets Privilege escalation in the Pod spec in init containers to false. забороняє підвищення привілеїв у init-контейнерах і знижує ризик несанкціонованих дій. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets Privilege escalation in the Pod spec to false. блокує ескалацію привілеїв у звичайних контейнерах, запобігаючи можливим атакам із підвищення прав. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets readOnlyRootFileSystem in the Pod spec in init containers to true if it is not set. переводить кореневу файлову систему init-

контейнерів у режим лише для читання, що ускладнює внесення небажаних змін. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика [Preview]: Sets readOnlyRootFileSystem in the Pod spec to true if it is not set. робить те саме, але для звичайних контейнерів, запобігаючи зловмисному модифікуванню файлової системи. Дана політика підтримує такі ефекти: Mutate, Disabled.

Політика Authorized IP ranges should be defined on Kubernetes Services радить обмежувати доступ до Kubernetes Service Management API конкретними діапазонами IP-адрес, знижуючи ймовірність несанкціонованих підключень. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Clusters should disable SSH передбачає відключення SSH на вузлах кластера AKS, що робить середовище менш уразливим до зламів і зменшує поверхню атаки. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Clusters should enable Container Storage Interface(CSI) зобов'язує використовувати CSI як стандарт для підключення до файлових і блокових сховищ, підвищуючи сумісність і гнучкість у керуванні даними. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Clusters should enable Key Management Service (KMS) рекомендує ввімкнути KMS, щоб додатково зашифрувати секрети, що зберігаються в etcd, і краще захищати конфіденційні дані. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Clusters should use Azure CNI є вимогою для деяких функцій AKS, зокрема Windows node pools і вбудованих політик мережі, тому закликає застосовувати Azure CNI для мережевих налаштувань. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Service Clusters should disable Command Invoke пропонує заборонити можливість безпосередньо викликати команди всередині кластера, щоб уникнути потенційних ризиків безпеки і оминання мережевих або RBAC-обмежень. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Service Clusters should enable cluster auto-upgrade заохочує автовстановлення оновлень в AKS, щоб кластери вчасно отримували виправлення безпеки та нові функції. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Service Clusters should enable Image Cleaner закликає ввімкнути Image Cleaner, який автоматично видаляє невикористані або вразливі образи, таким чином покращуючи безпеку та керованість. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Service Clusters should enable Microsoft Entra ID integration наголошує на доцільності інтегрувати Microsoft Entra ID для централізованого керування доступом і використання Kubernetes RBAC на основі ідентичностей. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Service Clusters should enable node os auto-upgrade пропонує ввімкнути автоматичні оновлення операційної системи вузлів, щоб швидко впроваджувати важливі патчі безпеки. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Service Clusters should enable workload identity додає можливість призначати окремим подам унікальні ідентичності, що дає їм безпечний доступ до захищених ресурсів, як-от Azure Key Vault. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Service clusters should have Defender profile enabled закликає активувати Microsoft Defender for Containers, аби отримувати інструменти аналізу загроз і захисту під час виконання в кластерах AKS. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Service Clusters should have local authentication methods disabled підсилює безпеку, вимикаючи локальні способи автентифікації й примушуючи використовувати виключно Microsoft Entra ID. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Azure Kubernetes Service Clusters should use managed identities радить переходити на керовані ідентичності (managed identities) замість

використання облікових даних service principal, щоб спростити керування доступом. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure Kubernetes Service Private Clusters should be enabled рекомендує створювати приватні кластери AKS, при яких увесь трафік між API-сервером та вузлами лишається у внутрішній мережі. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Azure Policy Add-on for Kubernetes service (AKS) should be installed and enabled on your clusters заохочує встановлювати надбудову Azure Policy в кластери AKS, що розширює функціонал Gatekeeper v3 для централізованого контролю політик і відповідності. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Azure running container images should have vulnerabilities resolved (powered by Microsoft Defender Vulnerability Management) перевіряє образи контейнерів на поширені вразливості (CVE) й відображає, які з них запущені у кластерах, щоб своєчасно усунути ризики. Дана політика підтримує такі ефекти: AuditIfNotExists, Disabled.

Політика Both operating systems and data disks in Azure Kubernetes Service clusters should be encrypted by customer-managed keys вимагає шифрувати ОС і диски із даними клієнтськими ключами, забезпечуючи більшу гнучкість і контроль у керуванні ключами. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Configure Azure Kubernetes Service clusters to enable Defender profile пропонує автоматично розгортати Microsoft Defender for Containers у кластері AKS, що дає повноцінний захист робочих навантажень і збір подій безпеки. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure installation of Flux extension on Kubernetes cluster передбачає встановлення розширення Flux у кластер, що дозволяє застосовувати fluxconfigurations для автоматичного впровадження змін у робочому середовищі. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure Kubernetes clusters with Flux v2 configuration using Bucket source and secrets in KeyVault забезпечує розгортання fluxConfiguration з вказаного Bucket, використовуючи секрети, збережені у Key Vault. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure Kubernetes clusters with Flux v2 configuration using Git repository and HTTPS CA Certificate керує отриманням конфігурацій із Git-репозиторію через HTTPS із використанням сертифіката CA, покращуючи безпеку з'єднання. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure Kubernetes clusters with Flux v2 configuration using Git repository and HTTPS secrets передбачає, що секрет ключа (HTTPS key secret) зберігатиметься у Key Vault, а кластери забиратимуть потрібні конфігурації із Git. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure Kubernetes clusters with Flux v2 configuration using Git repository and local secrets вимагає наявності локальних секретів у кластері для підключення до Git-репозиторію, дозволяючи автономне налаштування без зовнішніх сховищ. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure Kubernetes clusters with Flux v2 configuration using Git repository and SSH secrets зобов'язує тримати SSH-приватний ключ у Key Vault, щоб Flux міг безпечно отримувати конфігурації з Git. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure Kubernetes clusters with Flux v2 configuration using public Git repository дозволяє взагалі не використовувати секретів, якщо застосунки чи конфігурації зберігаються у відкритому публічному Git-репозиторії. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure Kubernetes clusters with specified Flux v2 Bucket source using local secrets вимагає підключення кластера до вказаного Bucket і застосування локальних секретів у самому кластері для аутентифікації. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure Kubernetes clusters with specified GitOps configuration using HTTPS secrets передбачає розгортання sourceControlConfiguration з обліковими даними (HTTPS user і key secrets) у Key Vault. Це дає змогу безпечно отримувати налаштування та оновлення з Git. Дана політика підтримує такі ефекти: auditIfNotExists, AuditIfNotExists, deployIfNotExists, DeployIfNotExists, disabled, Disabled.

Політика Configure Kubernetes clusters with specified GitOps configuration using no secrets дає змогу підключитися до Git-репозиторію без жодних секретів, якщо він є публічним. Таке рішення зручне для відкритих проєктів. Дана політика підтримує такі ефекти: auditIfNotExists, AuditIfNotExists, deployIfNotExists, DeployIfNotExists, disabled, Disabled.

Політика Configure Kubernetes clusters with specified GitOps configuration using SSH secrets розгортає sourceControlConfiguration, що дозволяє під’єднатися до Git по SSH, зберігаючи SSH-приватний ключ у Key Vault. Дана політика підтримує такі ефекти: auditIfNotExists, AuditIfNotExists, deployIfNotExists, DeployIfNotExists, disabled, Disabled.

Політика Configure Microsoft Entra ID integrated Azure Kubernetes Service Clusters with required Admin Group Access посилює безпеку кластера, централізуючи керування групами адміністраторів у середовищі AKS з інтегрованим Microsoft Entra ID. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Configure Node OS Auto upgrade on Azure Kubernetes Cluster орієнтована на автоматичне оновлення операційних систем вузлів, щоб без затримок отримувати критичні латки безпеки. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Deploy - Configure diagnostic settings for Azure Kubernetes Service to Log Analytics workspace пропонує автоматично налаштовувати переспрямування журнальних даних з AKS до Log Analytics, що спрощує моніторинг і аудит. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Deploy Azure Policy Add-on to Azure Kubernetes Service clusters акцентує на потребі встановлювати Azure Policy Add-on для централізованого контролю стану відповідності ваших кластерів AKS. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Deploy Image Cleaner on Azure Kubernetes Service рекомендує розгортати Image Cleaner у кластерах AKS, щоб регулярно та автоматично видаляти застарілі та потенційно небезпечні образи. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Deploy Planned Maintenance to schedule and control upgrades for your Azure Kubernetes Service (AKS) cluster робить можливим планувати та контролювати оновлення кластера в конкретні часові вікна, зменшуючи ризик збоїв у пікові години. Дана політика підтримує такі ефекти: DeployIfNotExists, AuditIfNotExists, Disabled.

Політика Disable Command Invoke on Azure Kubernetes Service clusters повністю вимикає можливість виконання команд у кластері поза межами встановлених мережевих чи RBAC-обмежень. Дана політика підтримує такі ефекти: DeployIfNotExists, Disabled.

Політика Ensure cluster containers have readiness or liveness probes configured примушує всі поди мати readiness та/або liveness probes, що гарантує своєчасне виявлення збоїв і підвищує надійність сервісів. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Kubernetes cluster containers CPU and memory resource limits should not exceed the specified limits примушує контейнери дотримуватися заданих граничних ресурсів, запобігаючи перевитраті й потенційним відмовам. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster containers should not share host process ID or host IPC namespace виключає спільне використання host process ID та host IPC namespace, що узгоджується з вимогами CIS і підвищує безпеку середовища. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster containers should not use forbidden sysctl interfaces обмежує використання недозволених параметрів sysctl у контейнерах. Це зменшує ризик конфігурацій, що можуть нашкодити стабільності та безпеці кластера. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster containers should only use allowed AppArmor profiles зобов'язує контейнери застосовувати лише дозволені профілі AppArmor, підсилюючи ізоляцію та безпеку процесів. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster containers should only use allowed capabilities передбачає обмеження підмножини можливостей (capabilities), які може отримати контейнер, зменшуючи потенційну поверхню атаки. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster containers should only use allowed images наголошує, що контейнери повинні братися виключно з надійних реєстрів, зменшуючи ризики використання шкідливих або скомпрометованих образів. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster containers should only use allowed ProcMountType унеможливлює використання заборонених ProcMountType у контейнерах, дотримуючись кращих практик безпеки. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster containers should only use allowed pull policy дає змогу обмежити pull policy для контейнерів, змушуючи їх завантажувати лише дозволені образи під час розгортання. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Kubernetes cluster containers should only use allowed seccomp profiles стосується використання лише дозволених seccomp-профілів, щоб захищати ядро від небажаних системних викликів. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster containers should run with a read only root file system встановлює режим лише для читання кореневої файлової системи контейнерів, запобігаючи несанкціонованим змінам файлів під час їх виконання. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster pod FlexVolume volumes should only use allowed drivers змушує FlexVolume використовувати лише перелік затверджених драйверів, зменшуючи ризики експлуатації невідомих чи небезпечних модулів. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster pod hostPath volumes should only use allowed host paths регулює, які саме шляхи на вузлі можуть бути змонтовані в под, запобігаючи доступу до критичних каталогів. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster pods and containers should only run with approved user and group IDs обмежує користувача, групи та додаткові групи, які можуть бути використані в подах, аби уникнути виконання шкідливого коду з підвищеними привілеями. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster pods and containers should only use allowed SELinux options контролює SELinux-опції, дозволені для подів і контейнерів, підвищуючи безпеку через суворішу політику доступу. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster pods should only use allowed volume types запобігає застосуванню небажаних або ризикованих типів томів, чим захищає кластер від потенційних уразливостей. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster pods should only use approved host network and port range встановлює, що поди можуть використовувати лише дозволену хост-мережу та діапазон портів. Це входить до вимог CIS і допомагає зменшити можливі вектори атаки. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster pods should use specified labels вимагає наявності певних міток на подах, спрощуючи керування й моніторинг у складних середовищах. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster services should listen only on allowed ports обмежує порти, на яких можуть працювати служби, зменшуючи ризик випадкових витоків даних або несанкціонованих доступів. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster services should only use allowed external IPs стежить, аби служби не були доступними з несанкціонованих IP-адрес, уникаючи згаданої вразливості CVE-2020-8554. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster should not allow privileged containers забороняє створювати контейнери з привілеями, що прописано у CIS 5.2.1. Такий підхід значно звужує поверхню атак. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes cluster should not use naked pods блокує запуск подів без контролерів на кшталт Deployment або ReplicaSet, щоб поди могли автоматично відновлюватись у разі збою вузла. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Kubernetes cluster Windows containers should not overcommit cpu and memory запобігає перевищенню лімітів CPU і пам’яті у Windows-контейнерів, уникаючи сповільнень через використання сторінки на диск замість завершення процесу за відсутності ресурсів. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Kubernetes cluster Windows containers should not run as ContainerAdministrator забороняє запуск Windows-контейнерів із правами ContainerAdministrator, що знижує ризик компрометації. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Kubernetes cluster Windows containers should only run with approved user and domain user group дозволяє Windows-контейнерам виконуватися лише від передбачених облікових записів і доменних груп, посилюючи безпеку. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Kubernetes cluster Windows pods should not run HostProcess containers унеможливлює запуск HostProcess-контейнерів у Windows-подах, зменшуючи ризик отримання привілейованого доступу безпосередньо на вузол. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Kubernetes clusters should be accessible only over HTTPS вимагає використовувати HTTPS для доступу до кластерів, щоб запобігти перехопленню даних і несанкціонованим діям у мережевому трафіку. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes clusters should disable automounting API credentials забороняє контейнерам автоматично монтувати облікові дані API, блокуючи виконання потенційно шкідливих команд від імені кластера. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes clusters should ensure that the cluster-admin role is only used where required підкреслює, що роль cluster-admin повинна застосовуватися виключно за суворої потреби, оскільки вона дає надмірні права. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Kubernetes clusters should minimize wildcard use in role and cluster role вказує на небезпечність використання символу ‘*’, оскільки він може надавати зайві дозволи, які створюють передумови для несанкціонованого доступу. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Kubernetes clusters should not allow container privilege escalation посилює безпеку, забороняючи процесам у контейнерах ескалувати права до root (CIS 5.2.5). Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes clusters should not allow endpoint edit permissions of ClusterRole/system:aggregate-to-edit закриває можливість редагувати endpoint у

ролі ClusterRole/system:aggregate-to-edit, запобігаючи перенаправленню запитів між namespace (CVE-2021-25740). Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Kubernetes clusters should not grant CAP_SYS_ADMIN security capabilities обмежує виділення контейнерам небезпечної можливості CAP_SYS_ADMIN, яка є однією з найпотужніших у Linux. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes clusters should not use specific security capabilities пропонує блокувати певні загрозливі security capabilities для запобігання виконанню зайвих привілейованих операцій. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes clusters should not use the default namespace виключає використання default namespace для ключових ресурсів (Pod, Secret, Service, тощо), скорочуючи ймовірність конфліктів чи несанкціонованого доступу. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes clusters should use Container Storage Interface(CSI) driver StorageClass наполягає на використанні CSI-драйвера StorageClass замість застарілого in-tree provisioner, відповідаючи сучасним рекомендаціям з керування сховищем. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Kubernetes clusters should use internal load balancers передбачає застосування внутрішніх балансувальників навантаження, щоб обмежити доступ до сервісів виключно в межах віртуальної мережі кластера. Дана політика підтримує такі ефекти: audit, Audit, deny, Deny, disabled, Disabled.

Політика Kubernetes resources should have required annotations змушує забезпечувати наявність обов’язкових анотацій на певних ресурсах Kubernetes, спрощуючи керування, документування й аудит. Дана політика підтримує такі ефекти: Audit, Deny, Disabled.

Політика Kubernetes Services should be upgraded to a non-vulnerable Kubernetes version пропонує оновлювати кластер до безпечнішої версії

Kubernetes, оскільки деякі старі релізи містять відомі вразливості (наприклад, CVE-2019-9946). Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Resource logs in Azure Kubernetes Service should be enabled рекомендує увімкнути журнали ресурсів (resource logs) в AKS, щоб створювати історію дій для розслідування інцидентів безпеки. Дана політика підтримує такі ефекти: AuditIfNotExists, Disabled.

Політика Role-Based Access Control (RBAC) should be used on Kubernetes Services зобов'язує використовувати ролі та привілеї на рівні кластера й простору імен (RBAC), щоб накладати гнучкі обмеження та правила доступу. Дана політика підтримує такі ефекти: Audit, Disabled.

Політика Temp disks and cache for agent node pools in Azure Kubernetes Service clusters should be encrypted at host спрямована на шифрування тимчасових дисків і кешу на вузлах, унеможливлюючи несанкціонований доступ до даних на рівні хосту. Дана політика підтримує такі ефекти: Audit, Deny, Disabled [25].

2.2.ЗАХИСТ КЛАСТЕРУ AKS ІНСТРУМЕНТАМИ KUBERNETES

Безпека Kubernetes охоплює кілька компонентів, серед яких кластер, образи контейнерів, програмний код, середовище виконання контейнерів, мережі та базова інфраструктура вузлів. Також важливим є сам процес розробки програмного забезпечення, що охоплює створення, поширення й розгортання контейнерів, оскільки саме він безпосередньо впливає на загальну безпеку платформи. У цій секції представлено інформацію про захист кластера, безпеку процесу розробки програмного забезпечення, а також інші теми, пов'язані з контейнерами й Kubernetes.

Рекомендується розглядати безпеку Kubernetes з огляду на кілька шарів. На найнижчому рівні звертається увага на забезпечення безпеки базової операційної системи та середовища, у якому розгорнуто кластер, включно з налаштуванням вузлів і роботою мережі. Далі необхідно забезпечувати безпеку кластера Kubernetes, зокрема контролерів, планувальників та інших компонентів, з яких складається контрольна площина (control plane). Слід подбати про правильні

політики доступу, конфігурації тощо, щоб унеможливити несанкціоновані операції з кластером. Окрему увагу слід приділити безпеці на рівні контейнерів і застосунків, а також організувати надійні процеси та інструменти для побудови, перевірки та публікації образів. Важливо стежити за тим, аби процес розробки програмного забезпечення, починаючи від вихідного коду й закінчуючи реєстром образів контейнерів, залишався надійним і захищеним.

З метою забезпечення безпеки платформи рекомендується дотримуватися найкращих практик керування доступом і застосовувати принцип найменших доступів. Слід призначати кожній сутності (користувачеві, службі чи процесу) стільки прав, скільки потрібно для виконання конкретних завдань, і не більше. Це стосується як контролю доступу всередині Kubernetes (через механізми RBAC), так і взаємодії з іншими системами та сервісами, наприклад, хмарними API чи сховищами даних. Також радиться керувати доступом до вузлів, обмежувати SSH-з'єднання та застосовувати автоматизовані підходи до виправлення вразливостей і оновлення.

До концептуальних питань безпеки Kubernetes належить тема мережевих політик. Мережеві політики дають змогу контролювати потоки даних між подами, просторами імен і сервісами всередині кластера. Рекомендовано дуже ретельно налаштовувати такі політики з урахуванням потреб додатків в системах, щоб мінімізувати поверхню атаки та потенційно унеможливити небажані з'єднання або прослуховування. Коректно реалізована ізоляція між додатками знижує ризики, пов'язані з компрометацією одного поду й поширенням атаки на інші ресурси.

Одним із критично важливих завдань є сканування вразливостей в образах контейнерів. Образи нерідко містять операційні системи, пакунки або бібліотеки, що можуть мати відомі вразливості. Тому радиться регулярно перевіряти образи, використовуючи спеціалізовані інструменти (наприклад, Clair, Trivy або інші), і здійснювати моніторинг, щоб вчасно виявити й усунути небезпечні компоненти. Також рекомендується підписувати контейнерні образи та налаштовувати політики перевірки таких підписів перед розгортанням у

кластері. Такі механізми підвищують рівень довіри до походження та цілісності образів.

Важливим аспектом є подальше дотримання принципів “shifting left”. Ідеться про якнайраніше виявлення проблем безпеки, поки зміни не встигли розповсюдитися в середовищі. Це передбачає інтеграцію тестів і статичного аналізу коду на етапах розробки й створення контейнерних образів, щоб не пропустити потенційні загрози чи неправильні конфігурації. Слід зазначити, що безпека не обмежується суто технічними інструментами; важливу роль відіграють командні процеси, навчання й сформовані політики. Доцільно застосовувати комплексний підхід, який поєднує підготовку персоналу, правильну організацію внутрішніх процесів і використання належних інструментів.

Для кращої безпеки в середовищі виконання корисно налаштовувати моніторинг і сповіщення про підозрілі дії або відхилення від нормальної поведінки. Існують різні рішення для відстеження поведінки процесів у контейнерах, збирання журналів, аналізу показників та виявлення аномалій. Слід аналізувати події, пов’язані з мережевим трафіком і діяльністю в самих контейнерах, щоб виявити ризиковані чи неочікувані операції. Можна впровадити механізми автоматичного карантину або ізоляції подів у разі підозрілого трафіку. Такі можливості знижують ймовірність поширення атаки та дають час на реакцію.

У межах Kubernetes варто зважати на те, які саме дані зберігаються або обробляються в кластері. Секрети Kubernetes (Kubernetes Secrets) дають змогу безпечніше керувати конфіденційними даними, такими як паролі або ключі доступу. Рекомендується зберігати секрети у зашифрованому вигляді в тої чи іншої системи (наприклад, в KMS-провайдері) і встановлювати чіткі обмеження на те, хто й коли може їх змінювати чи переглядати. Корисно налаштовувати ротацію секретів, особливо для облікових даних сервісів і користувачів, що забезпечують доступ до критичних систем.

Окрему увагу слід приділяти політикам безпеки для подів. У версіях Kubernetes до 1.25 існувала функція PodSecurityPolicy (PSP), яка використовувалася для визначення обмежень на те, як контейнери розгортаються й запускаються в кластері. У новіших версіях рекомендується використовувати політики безпеки на основі механізму Pod Security Admission, а також зовнішні інструменти на кшталт Gatekeeper або Kyverno, аби запроваджувати політики на рівні кластера. Але у випадку Azure Kubernetes Service вже є можливість використовувати Azure Policies, який в свою чергу для роботи деяких політик використовує Gatekeeper. Коли мова йде про політики, то мова йде про те, щоб блокувати чи допускати підозрілі конфігурації контейнерів (наприклад, запуск у привілейованому режимі, доступ до hostPID або hostNetwork). Безособове формулювання цих правил дає змогу централізовано контролювати можливості, які надаються подам, і тим самим покращувати загальну безпеку.

Також варто зазначити, що різні хмарні провайдери часто пропонують вбудовані засоби безпеки, сумісні з Kubernetes. Ідеться про інтеграцію з інструментами керування ідентифікацією, таємницями або шифруванням дисків. Рекомендується ознайомлюватися з офіційними документами хмарного провайдера, щоб налаштувати всі відповідні служби безпеки й інтегрувати їх із Kubernetes-кластером. На додачу, корисно відстежувати оновлення основних компонентів Kubernetes і регулярно застосовувати патчі, оскільки вони часто містять покращення безпеки та виправлення вразливостей.

Загалом концепція безпеки Kubernetes зводиться до дотримання низки найкращих практик, поєднуючи їх із глибоким розумінням того, як працюють контейнери й кластерні сервіси. Безпеку рекомендується розглядати як неперервний процес, що починається з розробки, продовжується під час збирання образів, охоплює процес розгортання й забезпечує контроль роботи контейнерів на етапі експлуатації. Кожен із цих кроків можна додатково посилювати за допомогою інструментів аналізу, моніторингу та керування конфігураціями. Врешті, безпеку Kubernetes найзручніше підтримувати, коли

вона вплетена в загальну культуру DevSecOps, а учасники проєкту усвідомлюють свою роль у підтримці надійного та стійкого середовища [26].

Окрім стандартних методів Kubernetes та рішень від хмарного провайдера Azure є ряд утиліт, які можуть доповонити фунціонал того ж Defender for Cloud, і таким чином команди безпеки або інші відповідальні команди зможуть додатково перевіряти свої контейнеризовані системи всередині кластерів.

Kubescape – це інструмент для сканування кластерів Kubernetes на відповідність політикам безпеки й кращим практикам. Він аналізує конфігураційні файли та виявляє потенційні загрози або відхилення від стандартів безпеки, включно з налаштуваннями подів, контейнерів і ролей доступу [27].

Kube-bench – це програма, що порівнює конфігурацію Kubernetes із заздалегідь визначеними рекомендаціями та вимогами безпеки (наприклад, з CIS Benchmarks). Вона допомагає виявити, чи не залишені якісь параметри за замовчуванням, які можуть бути потенційними точками уразливості [28].

Kubesec – це легковаговий засіб, що аналізує файли манифестів Kubernetes з погляду безпеки. Його завдання – допомогти розробникам та адміністраторам контейнерів виявляти ненадійні конфігурації, що можуть спричинити витоки даних чи ескалацію привілеїв [29].

Kubeaudit – інструмент, призначений для аудиту кластерів Kubernetes з точки зору відповідності стандартам безпеки. Він виконує серію перевірок (audit checks) і видає рекомендації, що саме варто виправити в налаштуваннях ролей, мережевих політиках та інших аспектах безпеки [30].

Trivy – багатоцільовий сканер безпеки, що перевіряє Docker-образи й Kubernetes-ресурси на предмет відомих вразливостей у бібліотеках і залежностях, а також аналізує можливі похибки в конфігурації. Це зручний спосіб виявити CVE ще до розгортання контейнерів у кластері [31].

Falco – це система виявлення вторгнень (IDS) та моніторингу “runtime”-загроз для Kubernetes і контейнерів. Вона використовується для збору інформації

про системні виклики (syscalls) та відстеження підозрілих чи небажаних дій у робочому середовищі контейнерів [31].

2.3.ВИКОРИСТАННЯ БАЗОВИХ НАЛАШТУВАНЬ МЕРЕЖІ ДЛЯ ПІДВИЩЕННЯ ЗАХИСТУ AKS

Як було згадано раніше, Kubernetes має велику кількість плагінів, як відкритих, так і пропрієтарних. В Azure є можливість вибирати між ними, але Microsoft надають підтримку тільки для прорієтарних плагінів і у випадку вибору відкритих рішень не варто розраховувати на підтримку. Лише спільнота з розробників даних плагінів та інших користувачів зможуть допомогти організаціям в такому разі, але вони не надають жодних гарантій чи гарантованих термінів відповіді. Тому якщо в проді трапиться нестандартна ситуація, що буде критичною, то краще розраховувати на рішення з підтримкою фахівців, тобто на пропрієтарні.

Розглянемо перший тип плагінів, а саме мережеві плагіни в AKS. Мережевий плагін в AKS відіграє важливу роль у забезпеченні комунікації між контейнерами та між контейнерами та зовнішнім світом. Основна задача цього плагіну полягає в управлінні IP-адресами для подів, сервісів та інших ресурсів в Kubernetes. Він визначає, як мережевий трафік буде направлятися всередині кластера та як кластер спілкується з зовнішньою мережею. таблицю 2.4 з доступними в AKS мережевими плагінами.

Таблиця 2.4

Порівняльна таблиця Networking plugins для AKS

Функціонал	**Kubenet**	**Azure CNI**	**Azure CNI Overlay**	**Azure CNI Powered by Cilium**
Розгортання кластера в існуючій або новій віртуальній мережі	Підтримується - UDRs застосовуються вручну	Підтримується	Підтримується	Підтримується
З’єднання Pod-pod	Підтримується	Підтримується	Підтримується	Підтримується

Продовження таблиці 2.4

З'єднання Pod-ВМ; ВМ в тому ж Vnet	Працює, коли ініціалізовано від Pod	Підтримується в обидва шляхи	Працює, коли ініціалізовано від Pod	Працює, коли ініціалізовано від Pod
З'єднання Pod-ВМ; ВМ в peered Vnet	Працює, коли ініціалізовано від Pod	Підтримується в обидва шляхи	Працює, коли ініціалізовано від Pod	Працює, коли ініціалізовано від Pod
Доступ до on-premise через VPN чи Exress Route	Працює, коли ініціалізовано від Pod	Підтримується в обидва шляхи	Працює, коли ініціалізовано від Pod	Працює, коли ініціалізовано від Pod
Надається доступ до K8s, через Load balancer, App Gateway чи ingress controller	Підтримується	Підтримується	Нема підтримки Application Gateway Ingress Controller (AGIC)	Ті ж обмеження, що і в Overlay
Підтримка Windows пулів вузлів	Не підтримується	Підтримується	Підтримується	Доступний тільки для Linux, не для Windows

Власне Azure CNI є пропрієтарним рішення з підтримкою від Microsoft і відповідно до таблиці має найкращу підтримку для багатьох сценаріїв. Порівняємо тепер Azure Network Policy Manager та Calico Network Policy (див. табл.2.2).

Azure Network Policy Manager має підтримку від Microsoft, підтримує Azure CNI та справляється зі своєю головною функцією: створення правил для вхідного та вихідного трафіку. За замовчуванням в Kubernetes Pods весь вхідний і вихідний трафік дозволений і Pods можуть між собою встановлювати мережевий зв'язок.

Таблиця 2.2

Порівняльна таблиця Network Policy

Функціонал	**Azure Network Policy Manager**	**Calico Network Policy**
Підтримувані платформи	Linux, Windows Server 2022	Linux, Windows Server 2019 and 2022
Підтримувані мережеві параметри	Azure CNI	Azure CNI (Linux, Windows Server 2019 and 2022) and kubenet (Linux)
Відповідність специфікації Kubernetes	Підтримуються всі типи політик	Підтримуються всі типи політик
Додаткові функції	Відсутні	Розширена модель політики, що складається з глобальної мережевої політики, глобального мережевого набору і кінцевого вузла.
Підтримка	Підтримується командою підтримки та інженерів Azure	Підтримка спільнотою Calico. Наявна додаткова платна підписка
Логування	Логи доступні за допомогою команди kubectl log -n kube-system <network-policy-pod	Інформація доступна на офіційному сайті

Розглянемо наступний важливий компонент Kubernetes кластеру, як ingress controller. Ingress Controller в Kubernetes – це компонент для управління зовнішнім доступом до сервісів у кластері через HTTP та HTTPS маршрутизацію. Він функціонує як вхідна точка, що направляє трафік із зовнішнього світу до відповідних сервісів в кластері здатен до маршрутизації трафіку, забезпечення безпеки через SSL/TLS та ефективного управління зовнішніми IP-адресами. Вибір і налаштування Ingress Controller залежать від конкретних потреб. На сьогодні вибір Ingress controller’ів досить великий (див. рис. 2.3), але сконцентруємо увагу на пропрієтарному рішенні від Microsoft – Application Gateway Ingress Controller (AGIC).

AGIC допомагає усунути необхідність мати ще один Load balancer/Public IP address перед кластером AKS і уникає багатократних переходів у вашому каналі передачі даних, перш ніж запити досягнуть кластера AKS. Application Gateway зв'язується з Pods, використовуючи їхні приватні IP-адреси напряму, і не потребує NodePort або KubeProxy. Ця можливість також забезпечує кращу продуктивність розгорнутих додатків.. Ingress Controller підтримує Web

Application Firewall наступних SKU Standard_v2 і WAF_v2, які також забезпечують переваги автомасштабування та безпеки. Application Gateway може реагувати на збільшення або зменшення навантаження трафіку і відповідно масштабуватися, не споживаючи жодних ресурсів вашого кластера AKS

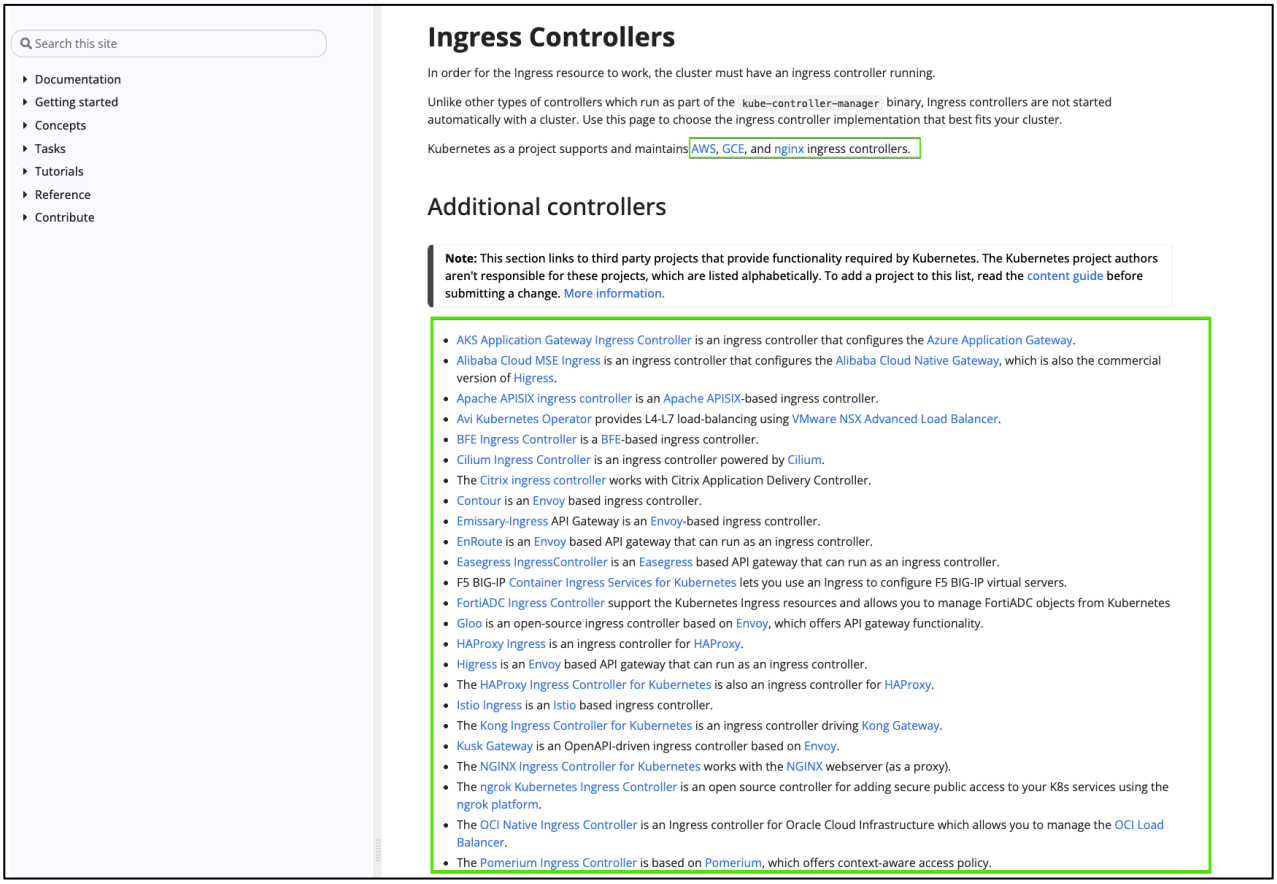


Рисунок 2.3 - Список Ingress controller'ів на офіційному сайті Kubernetes

AGIC налаштовується за допомогою ресурсу Kubernetes Ingress, а також Service і Deployments/Pods. Він надає багато функцій, використовуючи нативний Application Gateway L7 load balancer від Azure. Ось деякі з них:

- Маршрутизація URL-адрес;
- Cookie-based affinity;
- TLS termination;
- End-to-end TLS;
- Підтримка публічних, приватних і гібридних веб-сайтів;
- Інтегрований Web Application Firewall [24].

AGIC можливо встановити в AKS кластер, в якості доповнення. Це доповнення повністю підтримується та оновлює через Azure, тому даний тип

встановлення є досить легким. Розміщується в кластерві він все одно, як Pod. Не дивлячись на це, всі користувачі AGIC завжди будуть отримувати останню версію доповнення.

Щоб уникнути проблем із надійністю, пов'язаних з обмеженнями Azure Load Balancer при інтенсивному використанні вихідного трафіку для підключення інформаційної системи до зовнішніх сервісів можна використовувати NAT Gateway.

Azure NAT Gateway – це повністю керована та високостійка служба трансляції мережевих адрес (NAT). Azure NAT Gateway можна використовувати в якості шлюзу за замовчуванням для об'єктів приватної підмережі для виходу в Інтернет (в чому числі і AKS), залишаючись при цьому повністю приватними. Через NAT Gateway можуть проходити лише пакети, що надходять у відповідь на вихідне з'єднання. NAT Gateway є рекомендованим методом для систем з великою кількістю вихідного трафіку, де 65534 портів може бути недостатньо для вихідного трафіку і його використання також може підвищити відмовостійкість, надійність роботи нашого кластеру, при спробі ініціалізувати SNAT port exhaustion, NAT Gateway допоможе уникнути такої ситуації.

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ

В ході проведених досліджень у другому розділі було проаналізовано ряд підходів та інстрментів, що допоможуть підвищити рівень захисту Linux кластерів Azure Kubernetes Service.

Фреймворк можна поділити на дві частини:

Технології та методи покращення захисту хмарної інфраструктури з використання Azure Kubernetes service.

Загальні принципи та підходи для підвищення захисту інформаційних систем на базі Kubernetes (незалежно від того, де вони розміщені) та які інструменти для цього можна виконати. Очевидно, виходячи з аналізу можна стверджувати, що методи захисту саме Azure Kubernetes Service є більш вичерпними та дозволяють покрити більшу кількість задач з захисту, які будуть відповідати багатьом стандартам індустрії, наприклад ISO, SOC, PCI, тощо.

В першій частині було визначено, що безпекові рекомендації Well Architected Framework для AKS є досить вичерпними і його рекомендації поділяються на дві ключові частини: захист кластеру та захист самих інформаційних систем.

В другій частині було визначено які специфікації об’єктів Kubernetes можуть доповнити або замінити деякі рішення від Azure, у випадку, якщо останні з тієї чи іншої причини не задовільняють вимоги.

Отже, проведене дослідження дозволяє продовжити ознайомлення з наступним дослідженням попередньо розглянутих практик.

3.ДОСЛІДЖЕННЯ ПРАКТИК ПОКРАЩЕННЯ ЗАХИСТУ КЛАСТЕРУ AKS

3.1. АНАЛІЗ ПОТОЧНОГО СТАНУ КЛАСТЕРУ AKS З ВИКОРИСТАННЯМ ІНСТРУМЕНТІВ AZURE

Кластер AKS, що буде створено з мінімальною кількістю необхідних параметрів з використаням інструменту, типу Azure CLI, буде неможливо оцінити ступінь його захисту, зі стандартним функціоналом (див. рис.3.1).

```
vadym [ ~ ]$ az group create -n MyResourceGroup -l polandcentral
{
  "id": "/subscriptions/9c60189e-e6cb-4f1a-b907-a802f2823db1/resourceGroups/MyResourceGroup",
  "location": "polandcentral",
  "managedBy": null,
  "name": "MyResourceGroup",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
vadym [ ~ ]$ az aks create -g MyResourceGroup -n MyManagedCluster -l polandcentral --generate-ssh-keys | tee clusterinfo.txt
| Running ..
```

Рисунок 3.1 – Створення кластеру AKS з мінімальним набором параметрів

Лише після того, як на рівні підписки, де було розміщено даний кластер, було активовано Microsoft Defender for Cloud, можна було побачити оцінку Security Score та рекоменлації, які появились (див. рис.3.2-3.4).

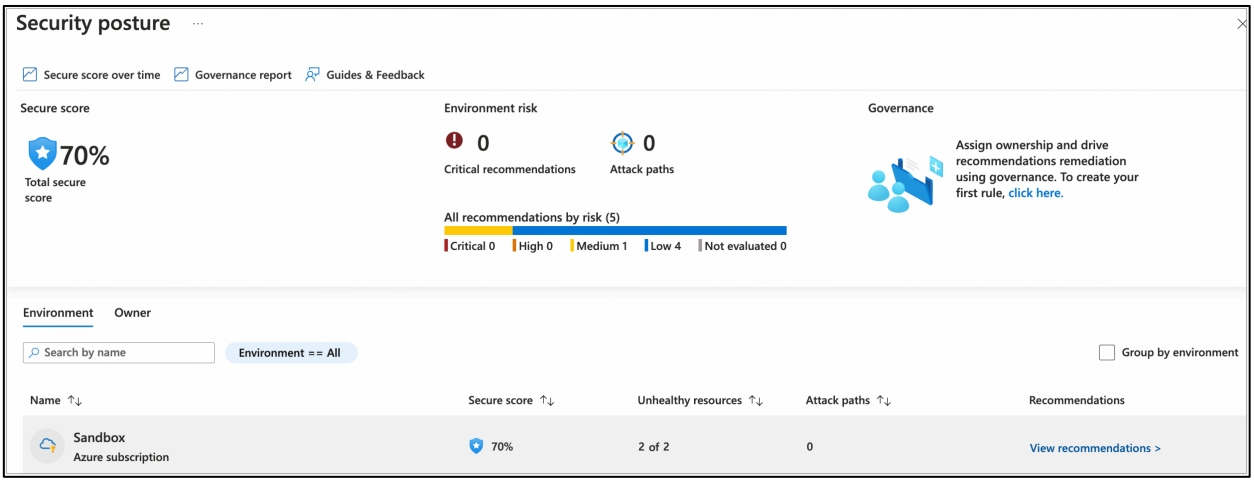


Рисунок 3.2 – Оцінка Security Score (70%)

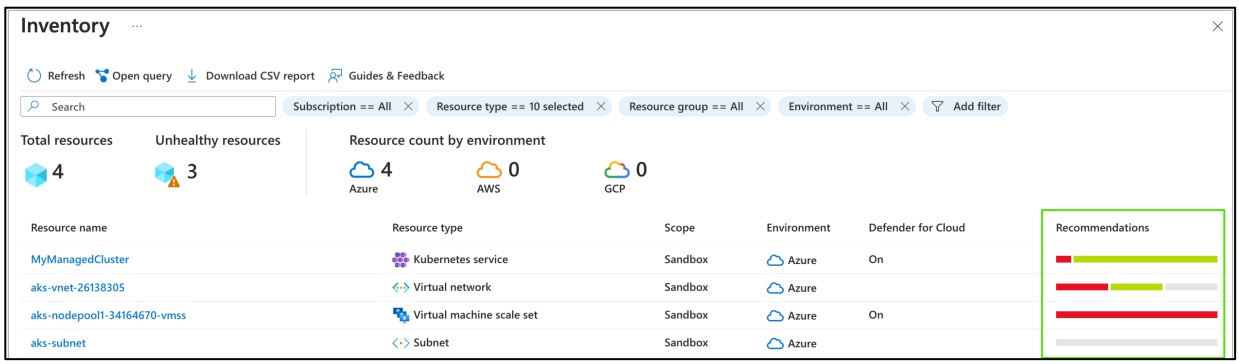


Рисунок 3.3 – Список ресурсів, що містять рекомендації

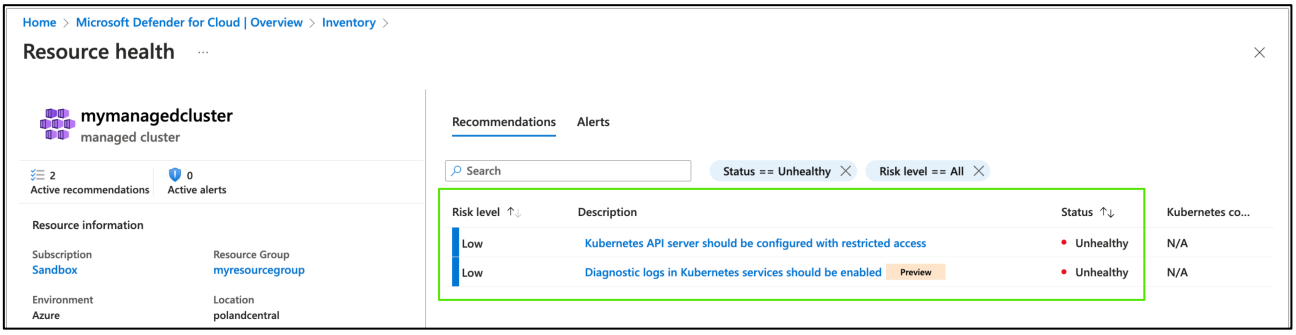


Рисунок 3.4 – Рекомендації для кластеру AKS

Також Defender for Cloud дозволяє бачити оцінку стану захисту відповідно до Microsofot Security Benchmark (див. рис. 3.5-3.11) та Azure CSPM (див. рис. 3.9) та їх відповідні рекомендації. Варто зазначити, що певні рекомендації появляються навіть після виправлення попередніх.

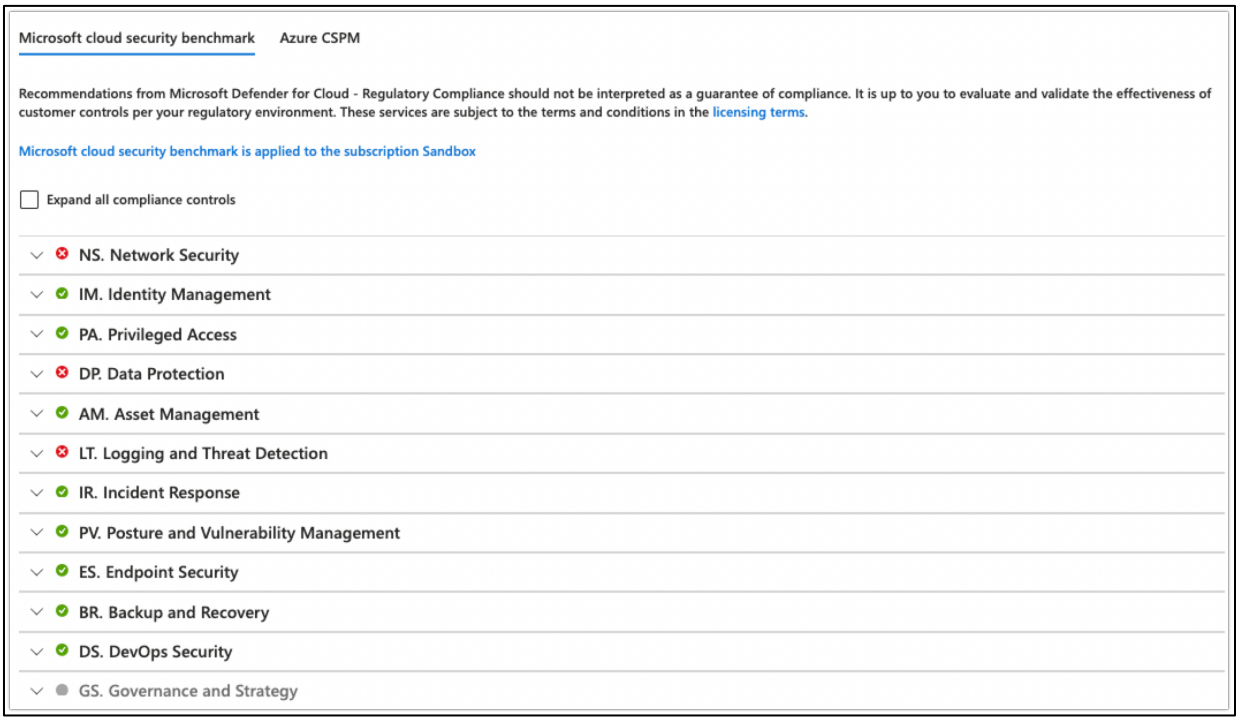


Рисунок 3.5 – Стан Microsoft Security Benchmark

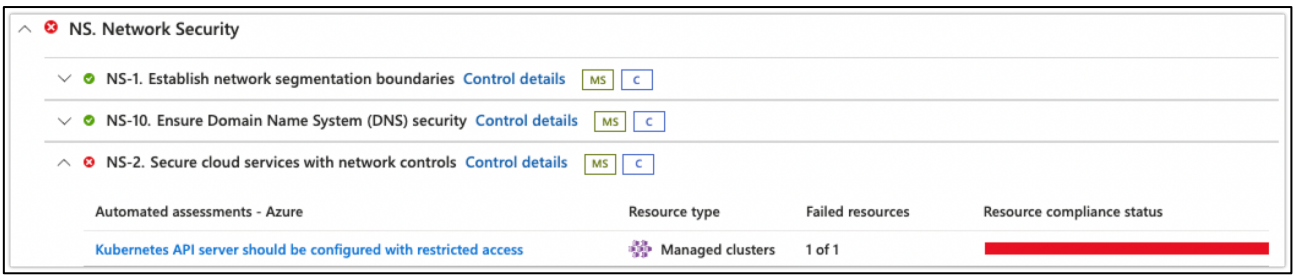


Рисунок 3.6 – Рекомендоване для виправлення для увімкнення обмежень доступу до Kubernetes API server

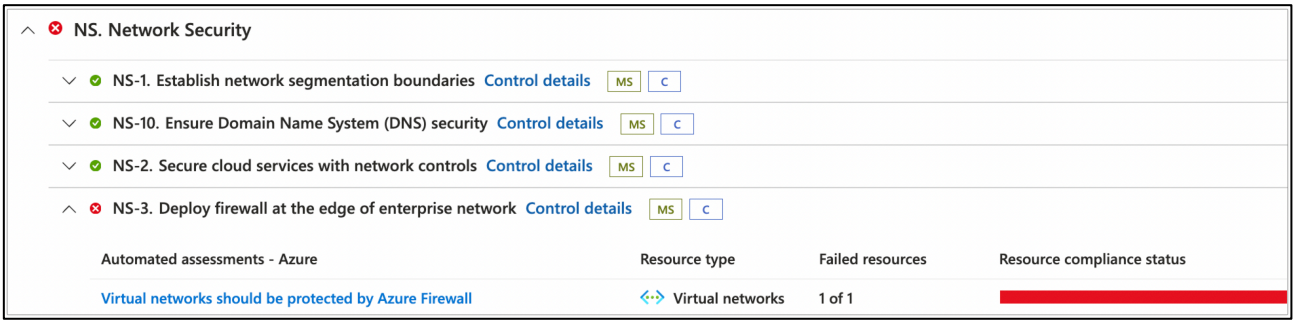


Рисунок 3.7 – Рекомендація для увімкнення Azure Firewall для віртуальної мережі кластеру

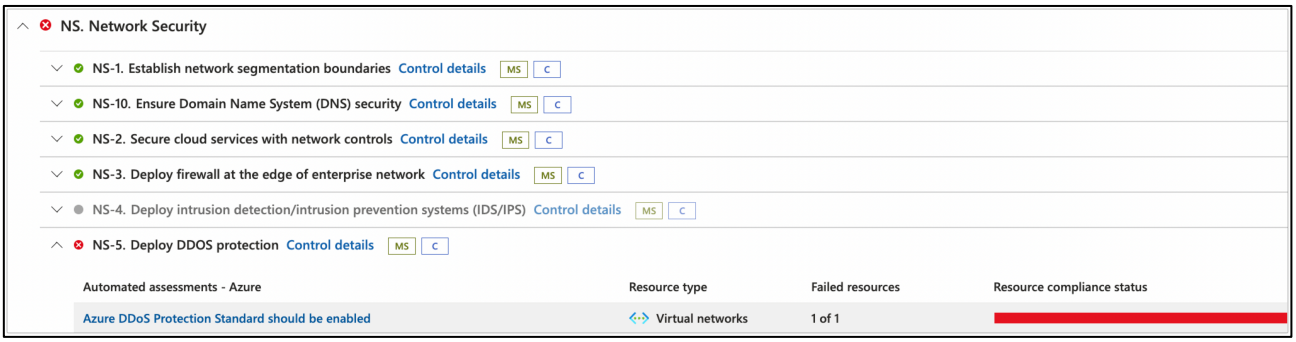


Рисунок 3.8 – Рекомендація для увімкнення Azure DDoS Protection Standard

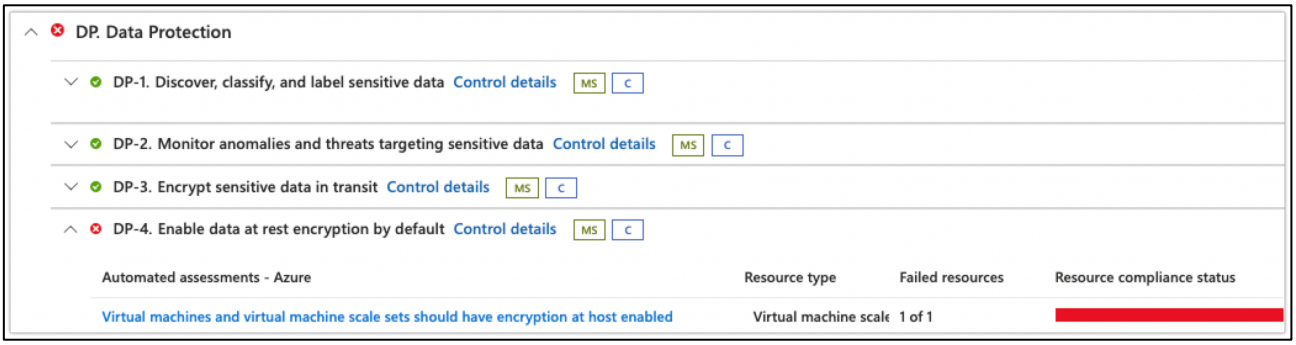


Рисунок 3.9 – Рекомендоване для виправлення налаштування для увімкнення шифрування для virtual machine scale sets, що входять до складу кластеру

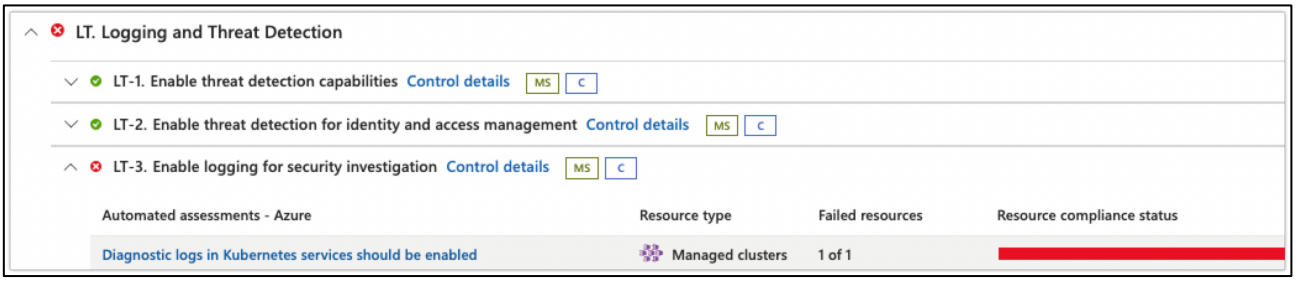


Рисунок 3.10 – Рекомендоване для виправлення налаштування для увімкнення діагностичних журналів кластеру



Рисунок 3.11 – Стан Azure CSPM та рекомендація для зниження рівня доступу для певних identities

Azure Policies містять ініціативи, згадані в другому розділі (див.рис. 3.12-3.13) і є змогу їх застосувати до тестового кластеру. Переглянемо вміст ініціативи Kubernetes cluster pod security baseline standards for Linux-based workloads.

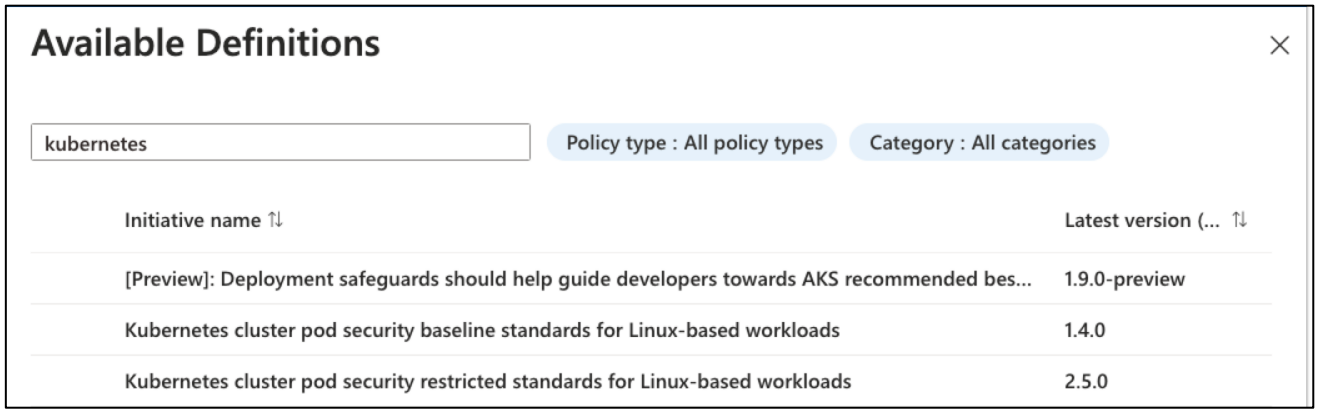


Рисунок 3.12 – Ініціативи для AKS

Список ініціатив, що політик, що містяться в одній з ініціатив, а саме Kuberbetes cluster should not allow privileged containers, Kubernetes cluster pods should ony use approved host network and port range, Kubernetes cluster containers should now share host process ID or host IPC namespace, Kubernetes cluster should only use allowed capabilities, Kubernetes cluster pods hostPath volumes should only use allowed host paths.

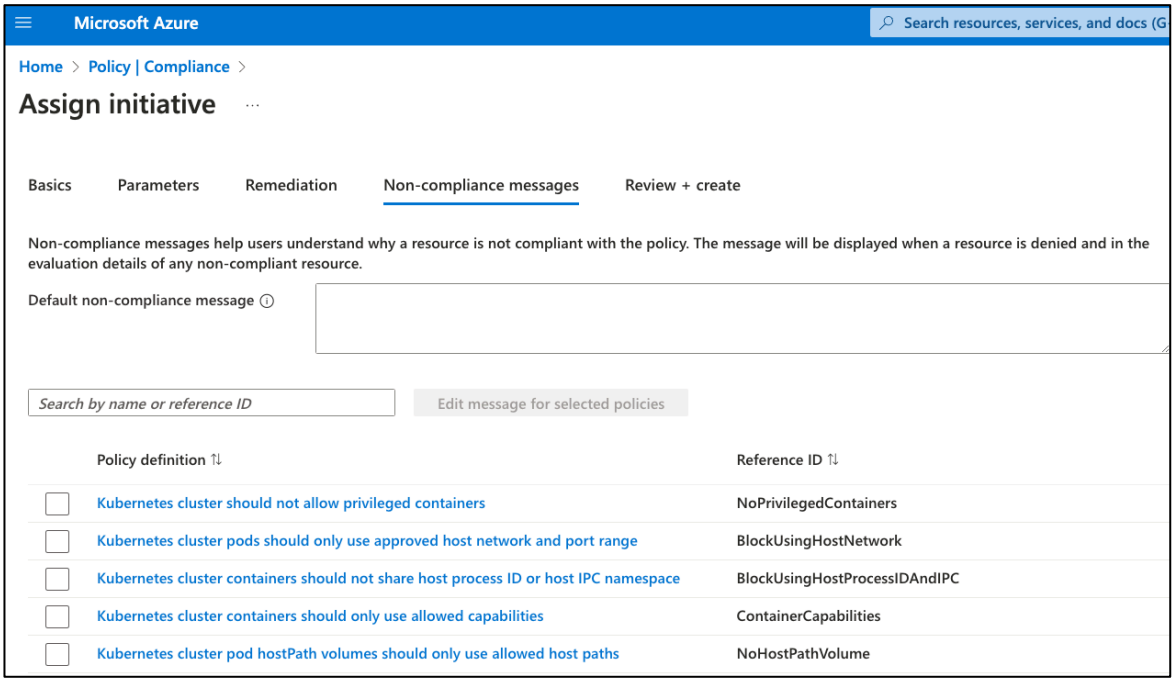


Рисунок 3.13 – Політики всередині ініціативи AKS

Тож застосуємо наступні ініативи до нашого кластеру (див. рис. 3.14), також заповнивши необхідні параметри для політик та увімкнувши для політик функцію автоматичного виправлення невідповідностей політики від імені system-assigned managed identity:

- [Preview]: Deployment safeguards should help guide developers towards AKS recommended best practices
- Kubernetes cluster pod security baseline standards for Linux-based workloads
- Kubernetes cluster pod security restricted standards for Linux-based workloads

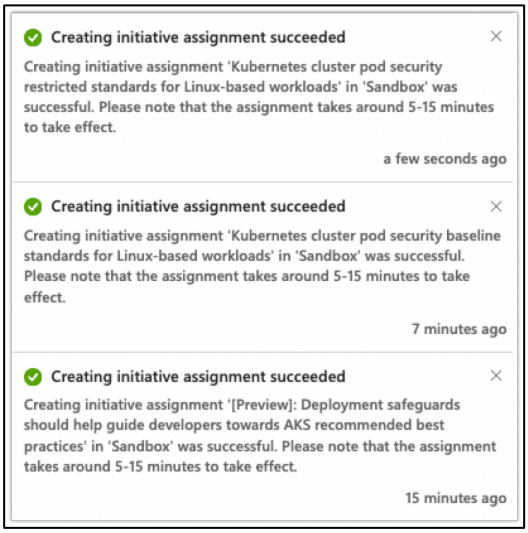


Рисунок 3.14 – Увімкнення політик

3.1.1.ВПРОВАДЖЕННЯ ПОКРАЩЕНЬ ЗГІДНО З РЕКОМЕНДАЦІЙ AZURE

Застосуємо рекомендації, що згадані в Network Security, Data Protect та Logging and Threat Detection в попередніх пунктах. В пункті Network security наявна рекомендація по застосуванню обмеження доступу до Kubernetes API server, тож скористаємось нею (див. рис.3.15) та протестуємо її (див. рис.3.16).

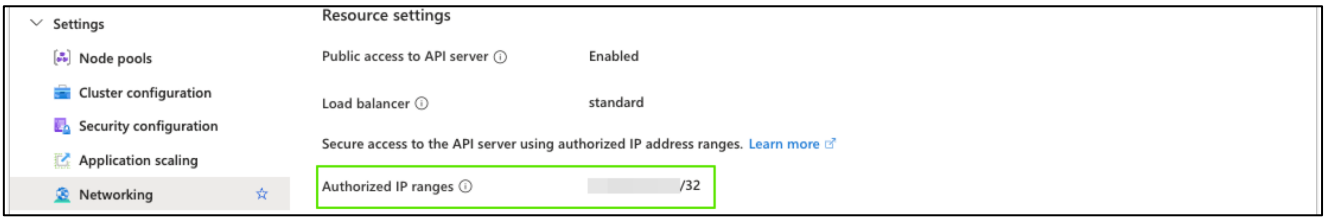


Рисунок 3.15 – Обмеження доступу до Kubernetes API server кластеру тільки з авторизованих IP-адрес

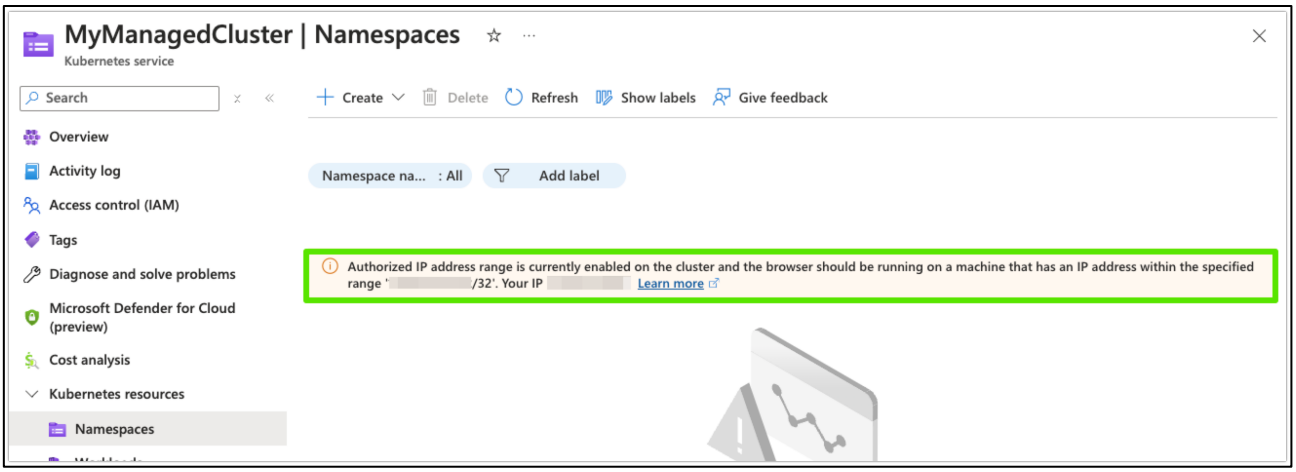


Рисунок 3.16 – Спроба отримання інформації про namespaces кластеру з неавторизованої адреси та повідомлення про відмову у доступі

Рекомендацію «Virtual machines and virtual machine scale sets should have encryption at host enabled» буде дещо важче імпелментувати, оскільки її застосування можливе тільки під час створення нового кластеру або під час створення нового node pool. Тому створимо новий node pool, де буде увімкнено encryption at host. За замовчування у VMSS дані шифруються у стані спокою з використання server-side шифрування з platform-managed keys. Host-based шифрування відрізняється за своїм принципом: віртуальні диски використовують Azure Storage для автоматичного шифрування у стані спокою під час збереження даних, а host-based шифрування використовує хост самої

віртуальної машини для обробки і ширфування перед передачею даних в Azure Storage.

Власне додамо ще один node pool до кластеру та замінимо існуючий, використовуючи Azure CLI (див. рис.3.17-3.18) та перевіримо результат, використовуючи Azure Graph Explorer (див. рис.3.19).

```
az aks nodepool add \
    --name hostencrypt \
    --cluster-name MyManagedCluster \
    --resource-group MyResourceGroup \
    -s Standard_B2s \
    --enable-encryption-at-host \
    --mode System
```

Рисунок 3.17 – Додавання node pool з host-based шифруванням

```
az aks nodepool delete \
    --cluster-name MyManagedCluster \
    --name nodepool1 \
    --resource-group MyResourceGroup
```

Рисунок 3.18 – Видалення node pool без host-based шифрування



Рисунок 3.19 – Підтвердження що Encryption at host увімкнено для новоствореного node pool

Увімкнемо журналювання для кластеру відповідно до вимоги «Enable logging for security investigation». Для цього всередині самої рекомендації знайдемо опцію «Fix» (див.рис.3.20). У свою чергу це увімкне журналювання для нашого кластеру та дані будуть зберігатись у Log Analytics Workspace. Перевіримо як

пишуться журнали подій у Log Analytics Workspace, виконавши запит в ньому (див. рис.3.21).

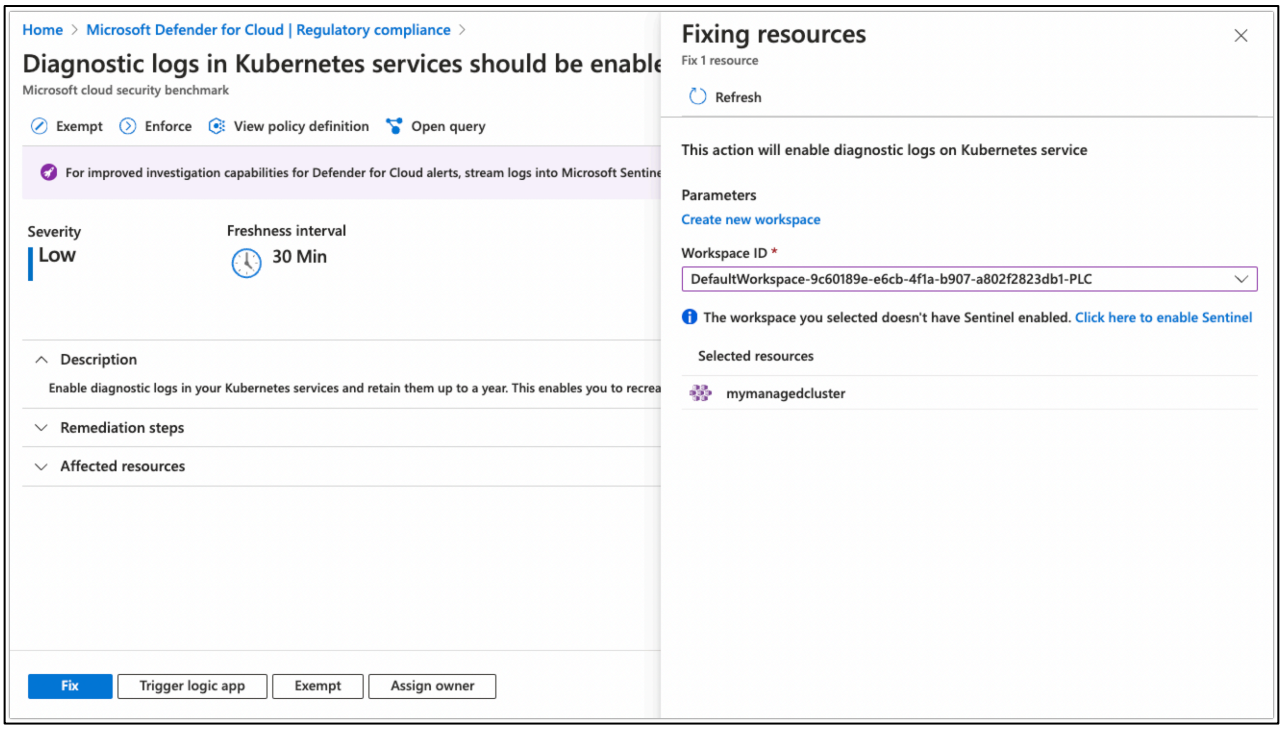


Рисунок 3.20 – Увімкнення журналювання подій для кластеру

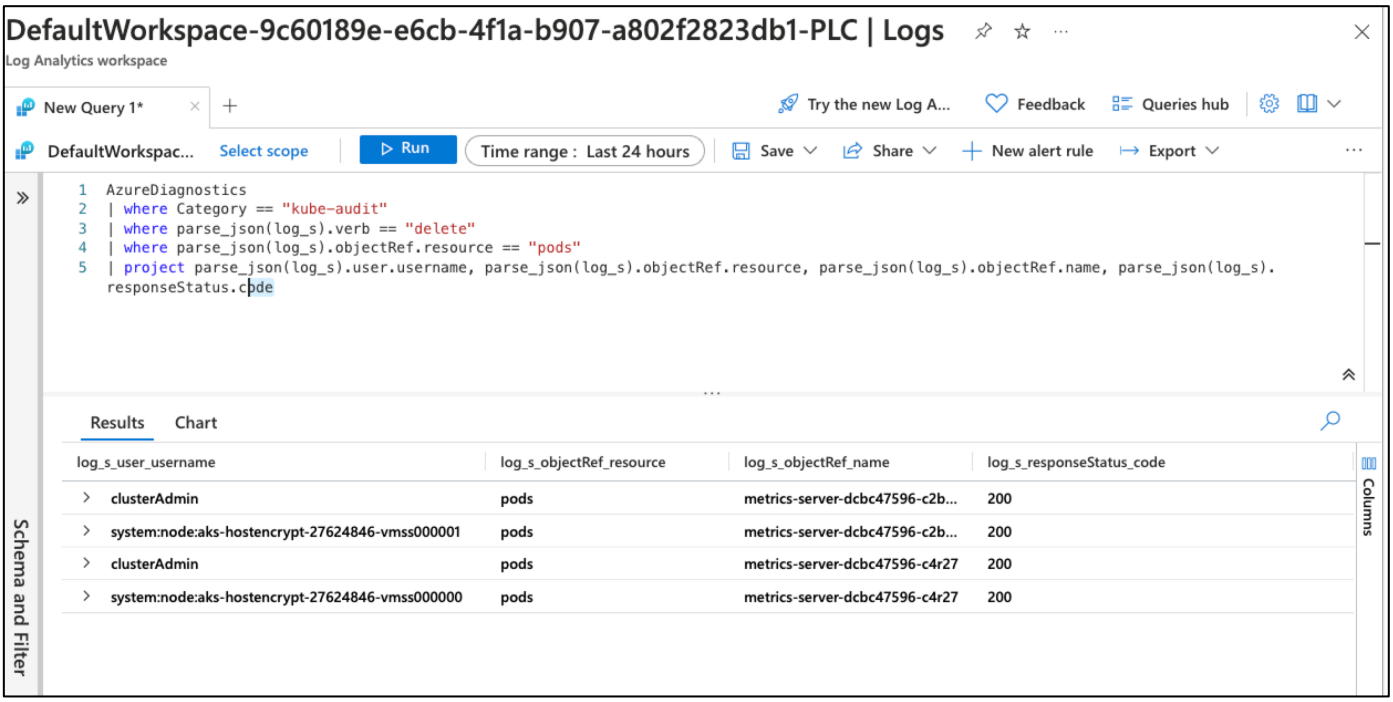


Рисунок 3.21 – Перевірка роботи Log Analytics Workspace

За допомогою Azure CLI виконаємо необхідні налаштування для додавання Azure Firewall до віртуальної мережі кластеру (див. рис. 3.22).

```
az network vnet subnet create \
    --name AzureFirewallSubnet \
    --resource-group MC_MyResourceGroup_MyManagedCluster_polandcentral \
    --vnet-name aks-vnet-26138305 \
    --address-prefix 10.239.0.0/24

az network public-ip create \
    --name fw-pip \
    --resource-group MC_MyResourceGroup_MyManagedCluster_polandcentral \
    --sku Standard

az network firewall create \
    -g MC_MyResourceGroup_MyManagedCluster_polandcentral \
    -n fw-basic \
    -l polandcentral \
    --enable-dns-proxy true

az network firewall ip-config create \
    --resource-group MC_MyResourceGroup_MyManagedCluster_polandcentral \
    --firewall-name fw-basic \
    --name fw-config \
    --public-ip-address fw-pip \
    --vnet-name aks-vnet-26138305
```

Рисунок 3.22 – Конфігурація роботи Azure Firewall у віртуальній мережі кластеру

Після увімкнення кластеру можемо переконатися, що умова виконана (див. рис.3.23).

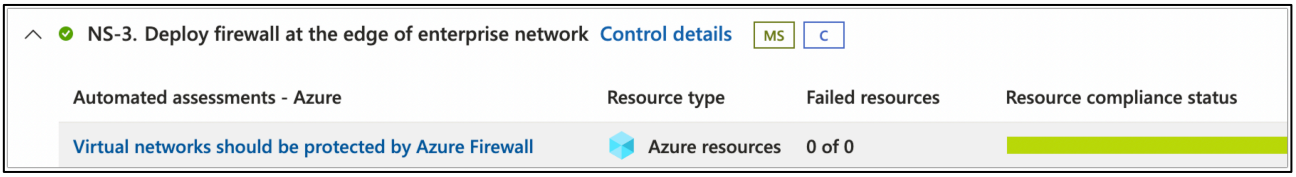


Рисунок 3.23 – Підтвердження того, що Azure Firewall увімкнено

Наступним кроком увімкнено DDoS Network protection для нашої віртуальної мережі кластеру (див. рис. 3.24).

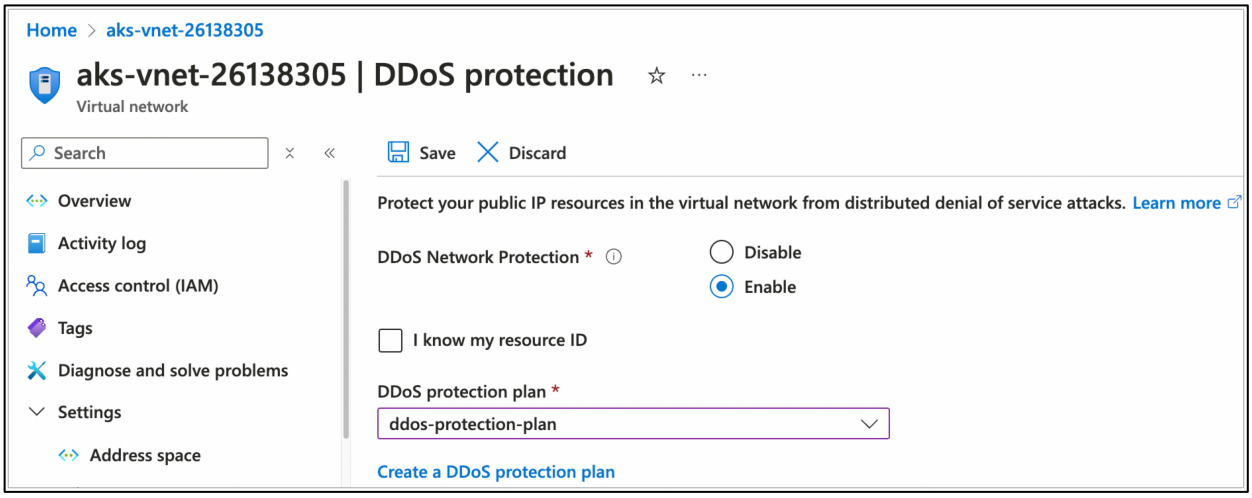


Рисунок 3.24 – Увімкення DDoS Network protection

Далі виправимо помилкову невідовідність політиці Azure CSPM, Azure overprovisioned identities should have only the necessary permissions. Вона помічає Managed Identities, які було створено в момент розгортання кластеру та увімкнення Defender for Cloud, як. такі, що мають занадто багато дозволів і їх було створено самим Azure. Тому ми можемо додати дану політику у виключення (див. рис.3.25).

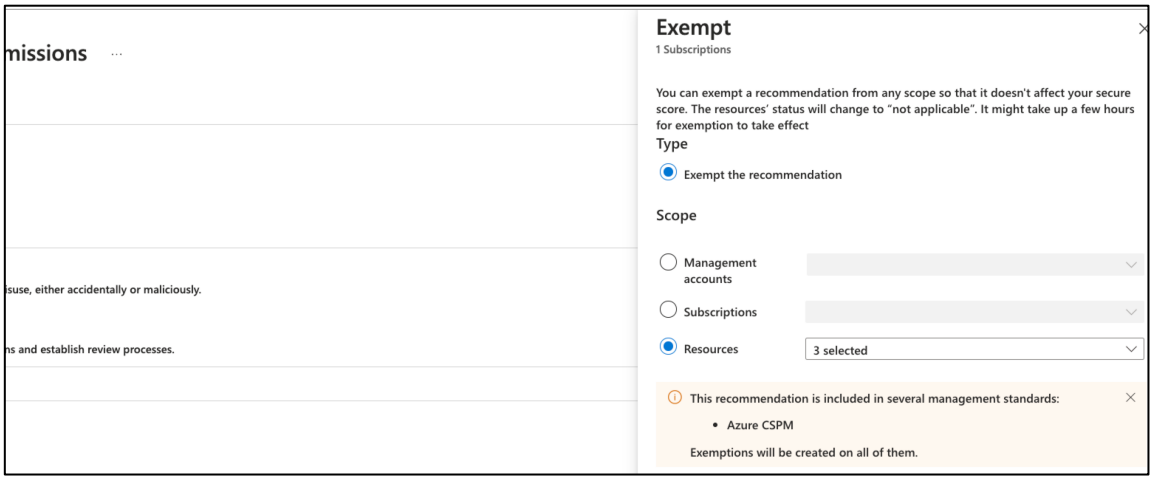


Рисунок 3.25 – Виправлення невідповідності політики Azure overprovisioned identities should have only the necessary permissions

3.2.ВПРОВАДЖЕННЯ ПРАКТИК БЕЗПЕКИ НАТИВНИМИ ЗАСОБАМИ KUBERNETES

До нативних засобів, що прийнято використовувати для підвищення безпеки кластерів Kubernetes, в тому числі і Azure Kubernetes Service можна віднести впровадження Kubernetes Role-based Access (RBAC), використання мережевих політик (network policisies), використання Kubernetes secrets. Розглянемо кожен з них окремо на прикладах.

Класично в Kubernetes немає такого поняття, як «користувач», як такого. Він містить тільки так звані службові облікові записи (service account). Тому розглянемо життєвий цикл створення сервісного облікового запису, який буде мати змогу виконувати будь-які дії, але тільки в рамках одного namespace (див. рис. 3.26-3.30). Даний цикл включає в себе наступні кроки:

- створення службового облікового запису;
- створення ролі, в якій будуть описані дозволетні операції для заданого скопу;
- прив’язка службового облікового запису до ролі.

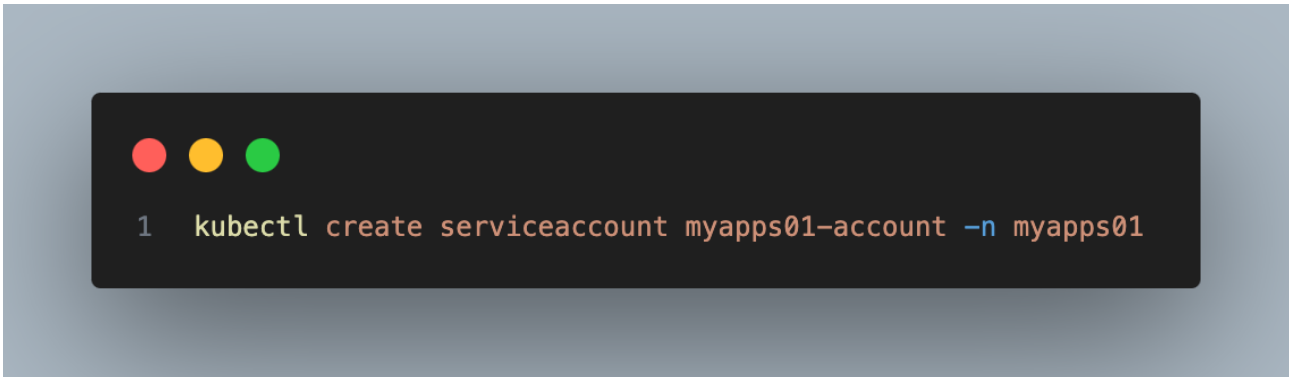


Рисунок 3.26 – Створення службового облікового запису myapps01-account

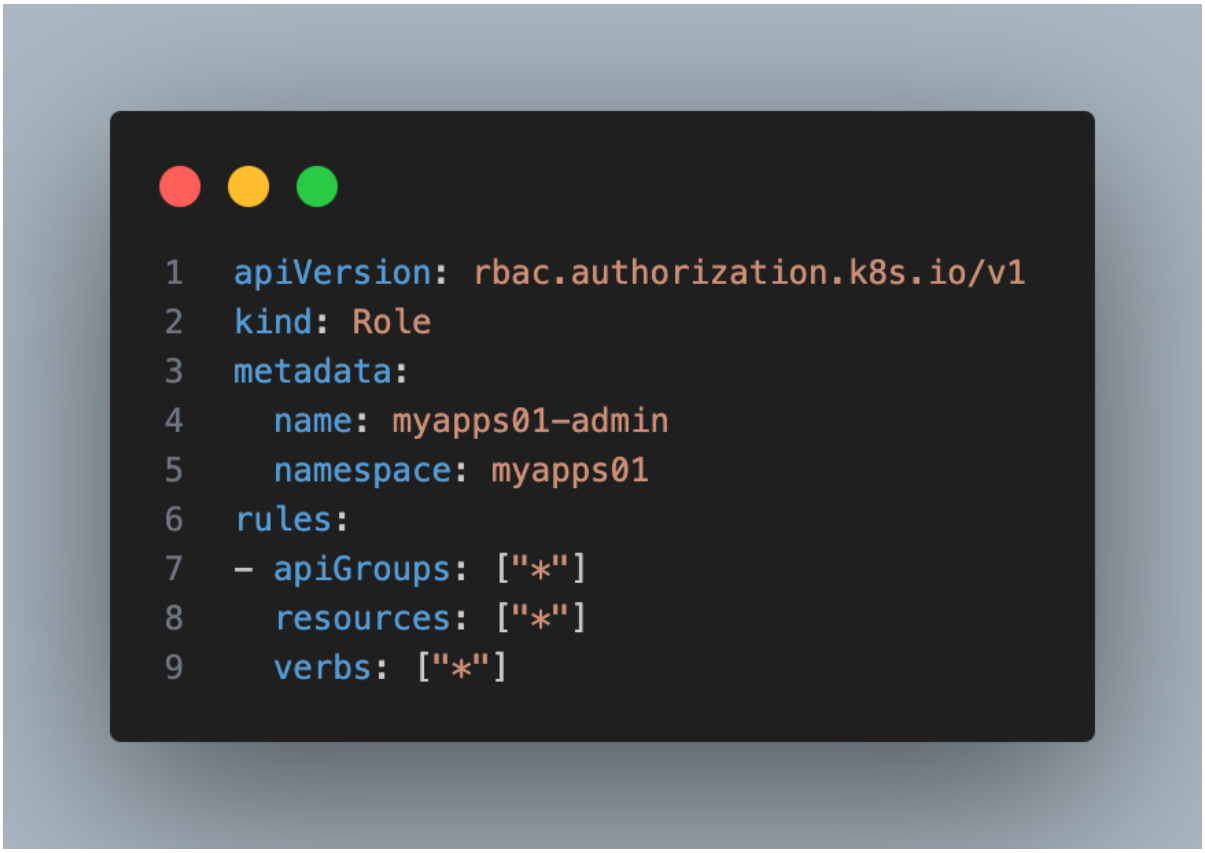


Рисунок 3.27 – Створення role.yml для налаштування ролі myapps02-admin

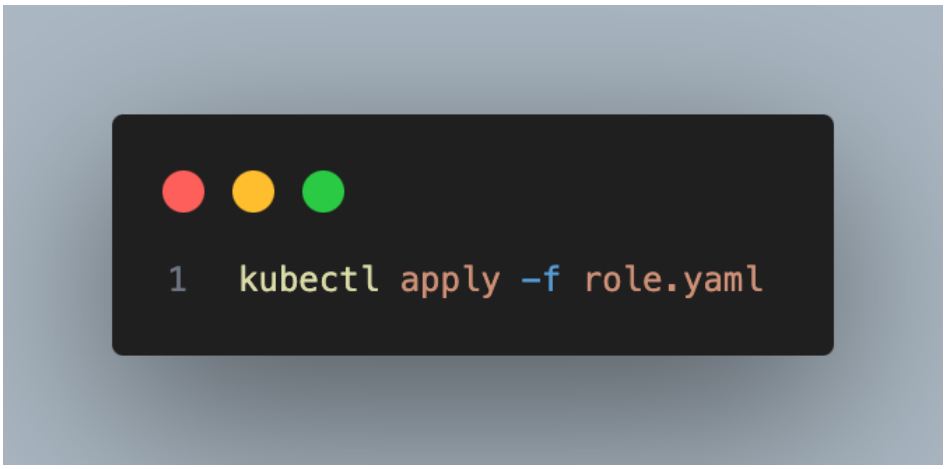
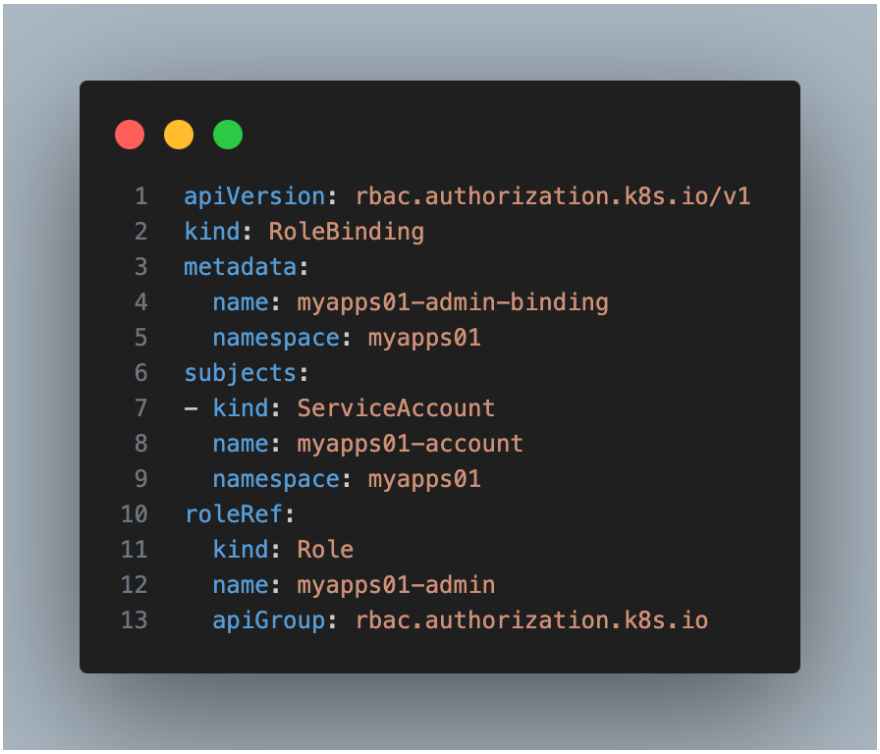


Рисунок 3.28 – Застосування конфігурації role.yml



```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: myapps01-admin-binding
  namespace: myapps01
subjects:
- kind: ServiceAccount
  name: myapps01-account
  namespace: myapps01
roleRef:
  kind: Role
  name: myapps01-admin
  apiGroup: rbac.authorization.k8s.io
```

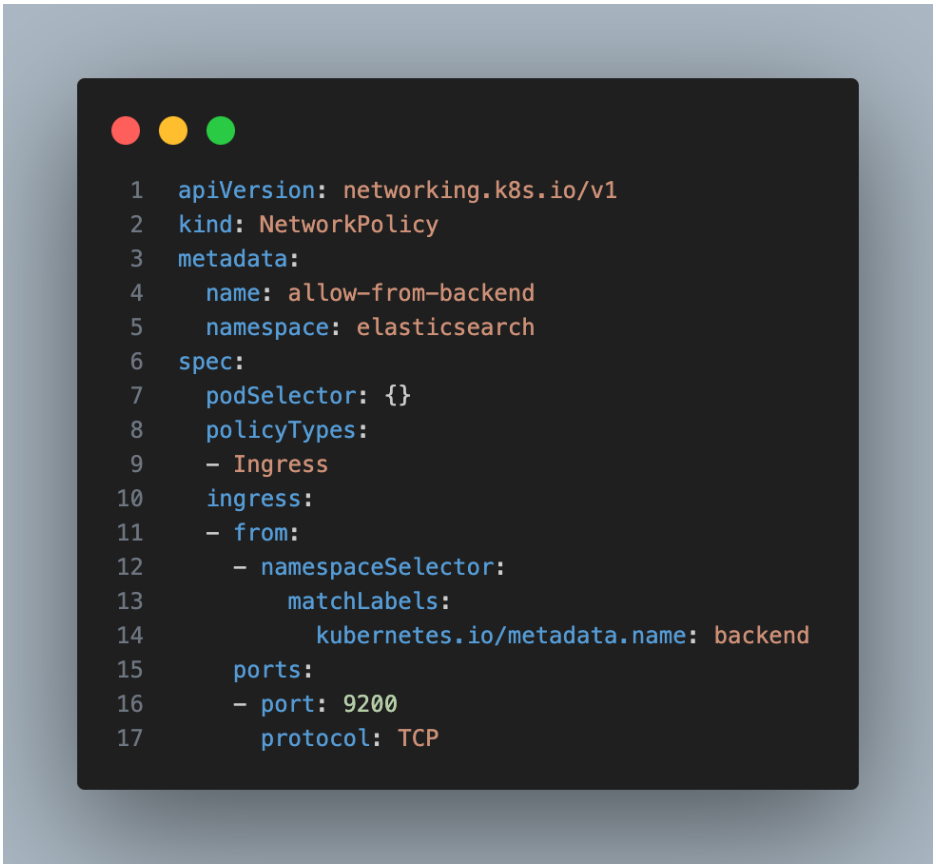
Рисунок 3.29 – Створення rolebinding.yml для прив'язки службового облікового запису до ролі



```
kubectl apply -f rolebinding.yaml
```

Рисунок 3.30 – Застосування конфігурації rolebinding.yml

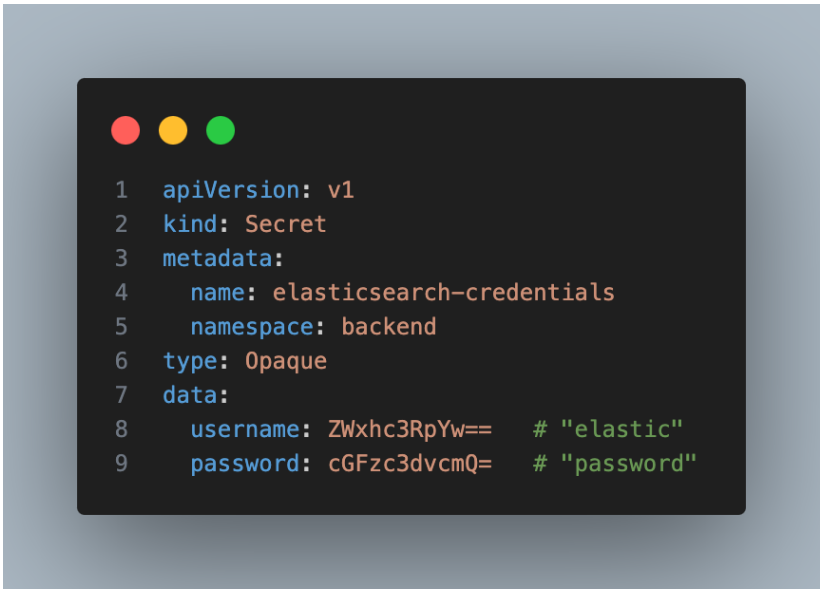
Розглянемо приклад, коли в кластері Kubernetes є два namespace: в одному з них розгорнуті поди з серевісами бекенду, а в іншому використовується сервіс ElasticSearch. Ці сервіси розгорнуті у відповідних namespace, які називаються backend та elasticsearch. В даній системі тільки сервіс backend може комунікувати з сервісом elasticsearch, який буде використовувати порт 9200 TCP. Власне обмежити мережевий доступ до namespace elasticsearch може network policy (див. рис. 3.31).



```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-backend
  namespace: elasticsearch
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: backend
    ports:
    - port: 9200
      protocol: TCP
```

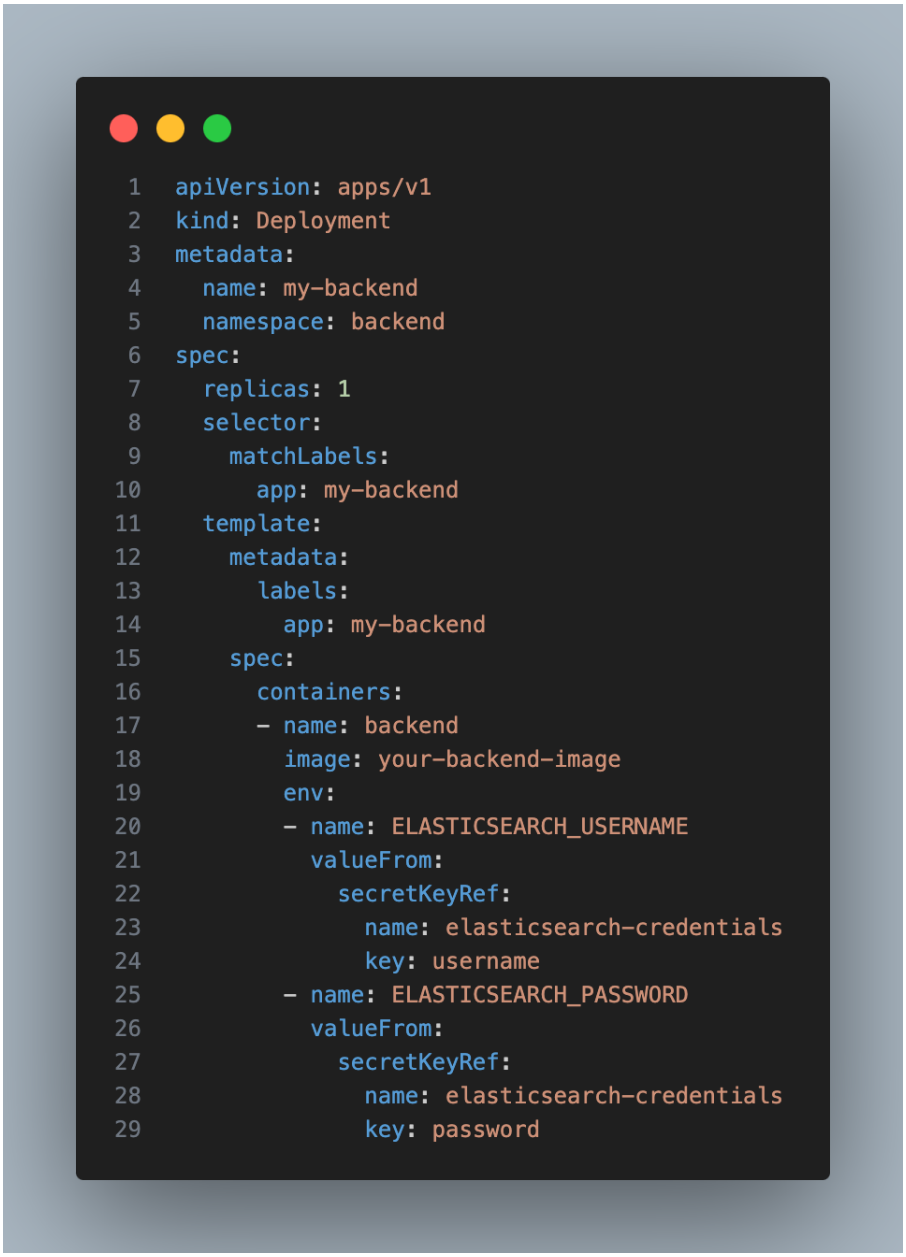
Рисунок 3.31 – Network policy для namespace elasticsearch

Kubernetes secret дозволяє зберігати такі об’єкти, як паролі, конфігурації підключення до баз даних, API-токени, тощо. Розглянемо приклад, в якому backend сервіс буде підключатись сервісу elasticsearch, де спочатку буде створено сам secret, а потім він буде використовуватись у маніфесті деплойменту (див. рис.3.32-3.33).



```yaml
apiVersion: v1
kind: Secret
metadata:
  name: elasticsearch-credentials
  namespace: backend
type: Opaque
data:
  username: ZWxhc3RpYw==   # "elastic"
  password: cGFzc3dvcmQ=   # "password"
```

Рисунок 3.32– Маніфест Kubernetes secret



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-backend
  namespace: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-backend
  template:
    metadata:
      labels:
        app: my-backend
    spec:
      containers:
      - name: backend
        image: your-backend-image
        env:
        - name: ELASTICSEARCH_USERNAME
          valueFrom:
            secretKeyRef:
              name: elasticsearch-credentials
              key: username
        - name: ELASTICSEARCH_PASSWORD
          valueFrom:
            secretKeyRef:
              name: elasticsearch-credentials
              key: password
```

Рисунок 3.33 – Маніфест Kubernetes deployment, що використовує secret

3.3.ВПРОВАДЖЕННЯ ПРАКТИК БЕЗПЕКИ ДЛЯ KUBERNETES СТОРОННІМИ ЗАСОБАМИ

Такий інструмент, як trivy дозволяє сканувати на вразливості образи контейнерів та самі кластери, що дозволяє адміністраторам кластеру проводити аудит конфігурації та приймати відповідні рішення по усуненню актуальних вразливостей. Дуже схожий функціонал надають такі інструменти, як kubescape, kube-bench, kubesec, kubeaudit. Тому практично розглянуто приклад використання тільки trivy, але на практиці можуть використовуватись і вищезгадані альтернативи (див .рис. 3.34-3.35).

```
❯ trivy image python:3.4-alpine
2022-05-30T15:47:04.096+0300    INFO    Detected OS: alpine
2022-05-30T15:47:04.096+0300    INFO    Detecting Alpine vulnerabilities...
2022-05-30T15:47:04.098+0300    INFO    Number of language-specific files: 1
2022-05-30T15:47:04.098+0300    INFO    Detecting python-pkg vulnerabilities...
2022-05-30T15:47:04.099+0300    WARN    This OS version is no longer supported by the distribution: alpine 3.9.2
2022-05-30T15:47:04.099+0300    WARN    The vulnerability detection may be insufficient because security updates are not provided

python:3.4-alpine (alpine 3.9.2)

Total: 37 (UNKNOWN: 0, LOW: 4, MEDIUM: 16, HIGH: 13, CRITICAL: 4)
```

Library	Vulnerability	Severity	Installed Version	Fixed Version	Title
expat	CVE-2018-20843	HIGH	2.2.6-r0	2.2.7-r0	expat: large number of colons in input makes parser consume high amount... https://avd.aquasec.com/nvd/cve-2018-20843
expat	CVE-2019-15903	HIGH	2.2.6-r0	2.2.7-r1	expat: heap-based buffer over-read via crafted XML input https://avd.aquasec.com/nvd/cve-2019-15903
libbz2	CVE-2019-12900	CRITICAL	1.0.6-r6	1.0.6-r7	bzip2: out-of-bounds write in function BZ2_decompress https://avd.aquasec.com/nvd/cve-2019-12900

Рисунок 3.34 – Сканування на вразливості образу python:3.4-alpine з trivy

```
☁  trivy [main] ⚡  trivy k8s --report summary
204 / 204 [----------------------------------------------------------------------------------------------------------------------------------------------------------------]

Summary Report for kind-kind

Workload Assessment
```

Namespace	Resource	Vulnerabilities					Misconfigurations					Secrets				
		C	H	M	L	U	C	H	M	L	U	C	H	M	L	U
local-path-storage	Deployment/local-path-provisioner	4	32	10	2			1	3	9						

```
Severities: C=CRITICAL H=HIGH M=MEDIUM L=LOW U=UNKNOWN

Infra Assessment
```

Namespace	Resource	Vulnerabilities					Misconfigurations					Secrets				
		C	H	M	L	U	C	H	M	L	U	C	H	M	L	U
kube-system	Pod/etcd-kind-control-plane					6		2	4	7						
kube-system	Pod/kube-apiserver-kind-control-plane					5		2	5	17						
kube-system	Deployment/coredns		13	11	3			1	3	4						

Рисунок 3.35 – Сканування на вразливості кластеру з trivy

Застосовувати політики для кластеру Kubernetes дозволяє такий інструмент, який Gatekeeper. Gatekeeper встановлюється використовується як інструмент для застовування політик в кластері Azure Kubernetes Service, коли використовуються Azure Policies. Тому його можна використовувати радше, як доповнення, коли стоїть необхідність застосування політик, що відсутні в Azure Policies. Наприклад, можна додати політику, яка буде знаходити підозрілі змінні середовища, які передаються у base64, та блокувати потенційне створення ресурсів, де будуть наявні маніфести подів з таким вмістом (див. рис.3.36-3.37).

```yaml
apiVersion: templates.gatekeeper.sh/v1
kind: ConstraintTemplate
metadata:
  name: k8snoenvbase64
spec:
  crd:
    spec:
      names:
        kind: K8sNoEnvBase64
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8snoenvbase64

        violation[{"msg": msg}] {
          input.review.object.kind == "Pod"
          some i
          envvar := input.review.object.spec.containers[i].env[_]
          re_match("^[A-Za-z0-9+/]{20,}=*$", envvar.value)
          msg := sprintf("Environment variable '%v' looks like a base64-encoded secret: '%v'", [envvar.name, envvar.value])
        }
```

Рисунок 3.36 – Створення Gatekeeper ConstraintTemplate

```yaml
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sNoEnvBase64
metadata:
  name: no-env-base64-constraint
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters: {}
```

Рисунок 3.37 – Створення Gatekeeper Constraint

Для встановлення інструменту Falco, який забезпечує виявлення загроз для хмарних, контейнерних і Kubernetes робочих навантажень майже в реальному часі, використовуючи дані під час виконання, можна скористатись таким інструментом, як helm (див. рис.3.38).

```
helm repo add falcosecurity https://falcosecurity.github.io/charts
helm repo update
helm install --replace falco --namespace falco --create-namespace --set tty=true falcosecurity/falco
```

Рисунок 3.38 – Встановлення Falco

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ

В ході виконаних в даному розділі досліджень було перевірено практично рекомендовані підходи для захисту Linux кластеру Azure Kubernetes Service на окремих його складових. Було проаналізовано початковий стан кластеру, який не містить додних застосованих практик безпеки. Згодом була зроблена базова оцінка з використання Defender for Cloud та Azure Policies. Дані інструменти дозволяють далі глибше аналізувати та нарощувати пошарову складову захисту системи.

Також було розглянуто використання практик Kubernetes, які можливо використовувати у кластерах, без застосування додаткових інструментів, таких, як Kubernetes RBAC, Network Policy, Kubernetes Secrets, а також використання популярних рішень, таких як trivy, Gatekeeper та Falco, які дозволяють захистити інфрастукруру кластеру, виявляти вразливості на різних етапах розробки та розгортання інформаційних систем.

Подальшою ціллю команди безпеки може бути призначення існуючих ініціатив, аналіз Well-Architected Framework, а також застосування open-source інструментів для сканування та виявлення хибних конфігурацій, що містять вразливості, реагування на спроби проникнення в систему, тощо.

ВИСНОВКИ

В даній кваліфікаційні роботі було досліджено методи підвищення захисту Linux кластеру Azure Kubernetes Service.

У першому розділі було детально розглянуто теоретичні відомості про особливості технології контейнеризації, розглянуто складові типового кластеру Kubernetes, та розглянуто особливості кластерів Azure Kubernetes service. Дані теоретичні відомості дали підґрунтя для подальшого аналізу проблематики даної кваліфікаційної роботи у другому розділі.

Власне у другому розділі розглянуто рекомендації для покращення захисту інформаційних систем у кластері Kubernetes, на прикладі кластеру Azure Kubernetes Service, які можна застосувати як виключно в середовищі Azure, так і деякі з них можна використати для кластерів розміщених у інших середовищах та хмарах.

В третьому розділі було перевірено практично рекомендовані підходи для підвищення захисту кластеру, розглянуті у другому розділі рекомендації, як для класичного кластеру Kubernetes, так і його реалізації в Azure Kubernetes Service, на прикладі Defender for Cloud та Azure Policies.

Таким чином основна ціль кваліфікаційної роботи з побудови фреймворку, що дозволить підвищити рівень захисту кластеру Azure Kubernetes Service методами та інструментами хмарного провайдера Azure та open-source інстрментами була досягнута. Слід також зауважити, що дані список не є вичерпним і залишає місце для подальших досліджень, оскільки як мінімум станом на сьогодні є ще досить велика кількість open-source інструментів, які розвиваються та можуть бути використані для поставленої задачі. Можливо, навіть частину цих інструментів згодом Microsoft інтегрує в Azure, як це сталось з багатьма іншими (Nginx, Gatekeeper, Eraser, тощо) Це зумовлено стрімким розвитком технологій, зміною вимог до інформаційних систем, їх архітектурою та випуском все нових і нових рішень від технологічних гігантів, типу Microsoft та незалежних розробників продуктів з відкритим вихідним кодом, що набувають все більшої і більшої популярності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1.Red Hat – Containers vs. VMs [Електронний ресурс] – Режим доступу до ресурсу: https://www.redhat.com/en/topics/containers/containers-vs-vms

2.MITRE Corporation – CVE-2015-3456 [Електронний ресурс] – Режим доступу до ресурсу: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2015-3456

3.MITRE Corporation – CVE-2021-26414 [Електронний ресурс] – Режим доступу до ресурсу: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2021-26414

4.MITRE Corporation – CVE-2017-14316 [Електронний ресурс] – Режим доступу до ресурсу: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2017-14316

5.MITRE Corporation – CVE-2019-5736 [Електронний ресурс] – Режим доступу до ресурсу: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2019-5736

6.MITRE Corporation – CVE-2022-0185 [Електронний ресурс] – Режим доступу до ресурсу: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2022-0185

7.MITRE Corporation – CVE-2021-22555 [Електронний ресурс] – Режим доступу до ресурсу: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2021-22555

8.Cloud Native Wiki by Aqua – containerd: What You Should Know [Електронний ресурс] – Режим доступу до ресурсу: https://www.aquasec.com/cloud-native-academy/container-platforms/containerd/

9.Microsoft Learn – Containers vs. virtual machines [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm

10.Microsoft Learn – Core AKS concepts (container runtime) [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/aks/core-aks-concepts#container-runtime

11.Microsoft Azure – What is a private cloud? [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-private-cloud

12.Microsoft Azure – What are public, private, and hybrid clouds? [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-are-private-public-hybrid-clouds/
13.Gartner – Репринт [Електронний ресурс] – Режим доступу до ресурсу: https://www.gartner.com/doc/reprints?id=1-2J4WMCRK&ct=241021
14.Kubernetes Documentation – What is Kubernetes? [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/uk/docs/concepts/overview/what-is-kubernetes/
15.Kubernetes Documentation – Чому вам потрібен Kubernetes і що він може робити [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/uk/docs/concepts/overview/what-is-kubernetes/#чому-вам-потрібен-kubernetes-і-що-він-може-робити
16.Kubernetes Documentation – Площина управління Kubernetes [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/uk/docs/concepts/#площина-управління-kubernetes
17.Kubernetes Documentation – Загальна інформація [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/uk/docs/concepts/#загальна-інформація
18.Kubernetes Documentation – Kubernetes Master [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/uk/docs/concepts/#kubernetes-master
19.Kubernetes Documentation – Вузли Kubernetes [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/uk/docs/concepts/#вузли-kubernetes
20.Kubernetes Documentation – Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/docs/concepts/architecture/
21.Kubernetes Documentation – Logging [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/docs/concepts/cluster-administration/logging/
22.Microsoft Learn – What is AKS? [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/aks/what-is-aks, Microsoft Learn – Policy reference for AKS [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/aks/policy-reference

23.Kubernetes Components [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/docs/concepts/overview/components/

24.Microsoft Learn – Service guide: Azure Kubernetes Service (recommendations) [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/well-architected/service-guides/azure-kubernetes-ervice?wt.mc_id=techcomm_wablog_blog_azuremktg#recommendations

25.Microsoft Learn – Policy reference for AKS [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/aks/policy-reference

26.Kubescape Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://kubescape.io/docs/.

27.kube-bench Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/aquasecurity/kube-bench.

28.kubesec Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://kubesec.io/.

29.kubeaudit Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Shopify/kubeaudit.

30.Trivy Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/aquasecurity/trivy.

31.[Електронний ресурс] – Режим доступу до ресурсу: https://falco.org/docs/.