Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА:
на тему: «Система автоматизованого керування безпекою застосунків у хмарному середовищі. Частина 1. Підсистема моніторингу застосунків»
08-20.МКР.025.00.000 ПЗ

Виконав: студент 2 курсу групи 1БС–22м
спеціальності 125 Кібербезпека
_____ Борис ЧЕРНОВОЛ

Керівник: к.ф.-м.н., доц. каф ЗІ
_____ Галина ШЕЛЕПАЛО
«11» 12 2023 р.

Опонент: д.т.н., проф., зав. каф. ПЗ
_____ Олександр РОМАНЮК
«11» 12 2023р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«12» 12 2023 р.

Вінниця - 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА:
на тему: «Система автоматизованого керування безпекою застосунків у хмарному середовищі. Частина 1. Підсистема моніторингу застосунків»
08-20.МКР.025.00.000 ПЗ

Виконав: студент 2 курсу групи 1БС–22м
спеціальності 125 Кібербезпека
_____ Борис ЧЕРНОВОЛ

Керівник: к.ф.-м.н., доц. каф ЗІ
_____ Галина ШЕЛЕПАЛО
«11» 12 2023 р.

Опонент: д.т.н., проф., зав. каф. ПЗ
_____ Олександр РОМАНЮК
«11» 12 2023р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«12» 12 2023 р.

Вінниця - 2023

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Галина ШЕЛЕПАЛО, к.ф.-м.н., доц. каф ЗІ	19.09	25.09
2	Галина ШЕЛЕПАЛО, к.ф.-м.н., доц. каф ЗІ	19.09	10.10
3	Галина ШЕЛЕПАЛО, к.ф.-м.н., доц. каф ЗІ	19.09	30.10
4	Ольга РАТУШНЯК, к.т.н., доц. каф ЕВПМ	19.09	27.11

7. Дата видачі завдання 01.09.2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Прим
1	Аналіз завдання. Вступ	01.09.2023 – 10.09.2023	
2	Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	10.09.2023 – 15.09.2023	
3	Науково-технічне обгрунтування	16.09.2023 – 22.09.2023	
4	Розробка технічного завдання	23.09.2023 – 29.09.2023	
5	Розробка рішень	30.09.2023 – 12.10.2023	
6	Практична реалізація, моделювання, експериментування, результати	14.10.2023 – 10.11.2023	
7	Розробка розділу економічного обгрунтування доцільності розробки	11.11.23 – 17.11.2023	
8	Аналіз виконання ТЗ, висновки	18.11.2023 – 24.11.2023	
9	Оформлення пояснювальної записки	25.11.2023 – 30.11.2023	
10	Попередній захист та доопрацювання МКР	28.11.2023 – 01.12.2023	
11	Перевірка магістерської роботи на наявність плагіату	02.12.2023 – 10.12.2023	
12	Представлення МКР до захисту	11.12.2023 – 14.12.2023	
13	Захист МКР	14.12.2023 – 21.12.2023	

Студент _____ Борис ЧЕРНОВОЛ
(підпис)

Керівник роботи _____ Галина ШЕЛЕП
(підпис)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Галина ШЕЛЕПАЛО, к.ф.-м.н., доц. каф ЗІ		
2	Галина ШЕЛЕПАЛО, к.ф.-м.н., доц. каф ЗІ		
3	Галина ШЕЛЕПАЛО, к.ф.-м.н., доц. каф ЗІ		
4	Ольга РАТУШНЯК, к.т.н., доц. каф ЕВПМ		

7. Дата видачі завдання 01.09.2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.2023 – 10.09.2023	
2	Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	10.09.2023 – 15.09.2023	
3	Науково-технічне обґрунтування	16.09.2023 – 22.09.2023	
4	Розробка технічного завдання	23.09.2023 – 29.09.2023	
5	Розробка рішень	30.09.2023 – 12.10.2023	
6	Практична реалізація, моделювання, експериментування, результати	14.10.2023 – 10.11.2023	
7	Розробка розділу економічного обґрунтування доцільності розробки	11.11.23 – 17.11.2023	
8	Аналіз виконання ТЗ, висновки	18.11.2023 – 24.11.2023	
9	Оформлення пояснювальної записки	25.11.2023 – 30.11.2023	
10	Попередній захист та доопрацювання МКР	28.11.2023 – 01.12.2023	
11	Перевірка магістерської роботи на наявність плагіату	02.12.2023 – 10.12.2023	
12	Представлення МКР до захисту	11.12.2023 – 14.12.2023	
13	Захист МКР	14.12.2023 – 21.12.2023	

Студент _____ Борис ЧЕРНОВОЛ
(підпис)

Керівник роботи _____ Галина ШЕЛЕПАЛО
(підпис)

АНОТАЦІЯ

Магістерська робота присвячена вивченню та розробці системи автоматизованого керування безпекою застосунків у хмарному середовищі. Сучасний хмарний комп'ютерний ландшафт надає користувачам можливість зберігати та обробляти дані в онлайн-режимі, що вимагає надзвичайної уваги до безпеки та моніторингу Вінниця: ВНТУ, 2023. 101 с.

Ця кваліфікаційна магістерська робота є комплексною, тому розглядає першу частину системи - підсистему моніторингу застосунків.

В першому розділі було детально розглянуто безпеку застосунків у хмарному середовищі та необхідність створення автоматизованої системи керування безпекою для розв'язання цієї проблеми. Також було сформульовано мету дослідження, яка полягає в розробці та впровадженні підсистеми моніторингу застосунків з метою виявлення та попередження можливих загроз безпеки у хмарному середовищі.

В другому розділі було проводено аналіз і дослідження різноманітних підходів та методів, які використовувались для моніторингу безпеки застосунків у хмарному середовищі. Особливий акцент був зроблений на зіставленні та порівнянні різних підходів та виокремлено їх переваги та недоліки.

У третьому розділі розроблено архітектурне рішення для для підсистеми, що відповідає за моніторинг безпеки застосунків у хмарному середовищі. Цей процес охоплював визначення важливих вимог до функціонала даної підсистеми та способів взаємодії з іншими компонентами системи керування безпекою.

В четвертому розділі описано процес проведення тестування підсистеми моніторингу безпеки застосунків у хмарному середовищі та аналіз її ефективності. Цей розділ є важливою частиною дослідження, оскільки він дозволяє визначити, наскільки успішно розроблена підсистема виконує свої функції. Результати дослідження корисні для організацій, що працюють з хмарними сервісами та бажають покращити їхню безпеку та надійність.

ABSTRACT

The master's thesis is dedicated to the study and development of an automated security management system for applications in a cloud environment. The modern cloud computing landscape provides users with the ability to store and process data online, emphasizing the need for exceptional attention to security and monitoring. Vinnytsia: VNTU, 2023. 101 p.

This qualification master's thesis is comprehensive, focusing on the first part of the system - the application monitoring subsystem.

The first chapter extensively explores application security in a cloud environment and the necessity of creating an automated security management system to address this issue. The research goal is formulated, which involves the development and implementation of an application monitoring subsystem to detect and prevent potential security threats in the cloud environment.

The second chapter conducts an analysis and investigation of various approaches and methods used for monitoring application security in a cloud environment. Special emphasis is placed on comparing different approaches and highlighting their advantages and disadvantages.

The third chapter develops an architectural solution for the subsystem responsible for monitoring application security in a cloud environment. This process involves defining important functional requirements for this subsystem and ways of interacting with other components of the security management system.

In the fourth chapter, the process of testing the security monitoring subsystem for applications in a cloud environment is described, along with an analysis of its effectiveness. This section is a crucial part of the research, as it helps determine how successfully the developed subsystem performs its functions. The research results are valuable for organizations working with cloud services aiming to enhance their security and reliability.

ЗМІСТ

ВСТУП

Актуальність теми. За останні роки спостерігається стрімкий розвиток хмарних технологій та послуг. Хмарні обчислення стали важливою складовою для багатьох організацій та користувачів. Збільшення обсягів даних та зростаюча кількість застосунків, які працюють в хмарних середовищах, створюють нові виклики у сфері безпеки. Зростання кількості та складності кіберзагроз створює серйозні загрози для організацій та користувачів, які використовують хмарні сервіси. Кібератаки можуть призвести до витоку конфіденційної інформації, втрати даних, порушення функціональності застосунків та інших серйозних проблем [1].

Захист великих обсягів даних та низькорівневий моніторинг застосунків у хмарних середовищах стають складним завданням для адміністраторів та безпеки IT. Автоматизована система керування безпекою може значно полегшити виявлення та відповідь на потенційні загрози [2].

Організації та постачальники хмарних послуг повинні забезпечити високий рівень безпеки та надійності для збереження репутації та довіри клієнтів. Передбачення та запобігання можливим кіберпорушенням стають насущною задачею.

Збільшення важливості інформації та її вплив на ділові процеси ставлять під загрозу приховання даних та можливості їх використання. В разі порушення безпеки інформації, наслідки можуть бути надзвичайно серйозними. У зв'язку з цими чинниками, розробка та впровадження систем автоматизованого керування безпекою застосунків у хмарному середовищі, зокрема підсистеми моніторингу, є надзвичайно актуальною та важливою темою для дослідження.

Метою цієї роботи є розробка та впровадження підсистеми моніторингу застосунків у хмарному середовищі для покращення безпеки та надійності роботи застосунків.

Завданнями дипломної роботи є:

- Аналіз існуючих рішень і технологій для моніторингу безпеки застосунків у хмарних обчисленнях;

- Розробка та реалізація підсистеми моніторингу, із урахуванням надійності застосунків у хмарному середовищі;
- Тестування та оцінка працездатності та ефективності розробленої підсистеми;

Об’єктом дослідження є процес моніторингу безпеки застосунків у хмарному середовищі.

Предметом дослідження підсистема автоматизованого керування безпекою застосунків у хмарному середовищі.

Новизна одержаних результатів полягає у впровадженні підсистеми моніторингу, яка враховує специфіку хмарних обчислень та відповідає потребам сучасного інформаційного середовища. Ця робота може сприяти покращенню безпеки та надійності застосунків у хмарному середовищі.

Практичне значення одержаних результатів. результати можуть значно покращити рівень безпеки застосунків, які функціонують у хмарних середовищах. Інформаційна безпека стає все більш актуальною, і ваші розробки можуть допомогти зменшити загрози та ризики.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих рішень та технологій в області безпеки хмарних застосунків

Сучасний світ інформаційних технологій і хмарних обчислень характеризується швидким розвитком застосунків та послуг, що базуються на хмарних середовищах. Ця динаміка призвела до зростаючого попиту на рішення, які б забезпечували високий рівень безпеки і надійності для цих застосунків. Захист і забезпечення надійності важливі не тільки для бізнес-клієнтів, але й для індивідуальних користувачів, які зберігають свої дані в хмарних середовищах.

Хмарні середовища надають багато переваг, такі як масштабованість, доступність та зручність, проте разом з цим вони призводять до нових викликів у сфері інформаційної безпеки [3]. Зловмисники використовують нові методи та загрози, і важливо мати системи моніторингу, які можуть вчасно виявляти і врегульовувати ці загрози.

Вивчення існуючих рішень та технологій є критично важливим етапом для розробки системи автоматизованого керування безпекою застосунків у хмарному середовищі. Це дозволяє нашому дослідженню побудувати на наукових підставах та використовувати найкращі практики, які існують на сьогоднішній день у цій області.

Терміном контейнер називають несекретну інформацію, яка використовується для маскування повідомлення. Як повідомлення в контейнері можна використовувати не тільки звичайний текст, а й файли іншого формату, наприклад мультимедійного.

Метою цього розділу є детальний аналіз і класифікація існуючих рішень та технологій, а також вибір тих, які найкраще відповідають потребам нашого дослідження. Цей аналіз стане фундаментом для подальшої розробки системи автоматизованого керування безпекою застосунків у хмарному середовищі та надасть можливість зробити обґрунтовані висновки щодо вибору оптимальних інструментів та методів моніторингу безпеки.

Системи журналювання відіграють критичну роль у виявленні та відстеженні подій, що стосуються безпеки в хмарному середовищі. Під час аналізу існуючих рішень, було виявлено, що різноманітні інструменти та підходи використовуються для журналювання подій та дій, таких як спроби доступу, зміни конфігурації та інші аспекти, що стосуються безпеки. Відповідний вибір системи журналювання визначає, наскільки ефективно можна виявляти аномалії та реагувати на можливі загрози. Системи журналювання використовуються для виявлення та аналізу помилок та аномалій в роботі застосунків та інфраструктури хмарних обчислень. Вони забезпечують інформацію для оперативного виправлення несправностей та аналізу причин їх виникнення. Коли невідомі помилки або аномалії виникають у системі, журнали подій стають цінним інструментом для знаходження відповідей та вирішення проблем.

Моніторинг використання ресурсів систем журналювання також забезпечують засіб для стеження за використанням ресурсів, таких як обсяг процесорного часу, величина використаної пам'яті та обсяг мережевого трафіку. Ця можливість сприяє ефективному управлінню ресурсами та виявленню незвичайних або аномальних показників їх використання [4]. Наприклад, істотне збільшення кількості запитів до сервера або неефективне використання доступної пам'яті можуть свідчити про можливі проблеми в роботі системи. Записи журналів подій допомагають виявити ці аномалії і сприяють вжиттю відповідних заходів.

У виявленні та відстеженні подій існує ряд систем журналювання, які можуть бути використані:

ELK Stack - це популярна платформа для обробки та аналізу журнальних даних. Elasticsearch використовується для зберігання індексованих даних, Logstash - для збору та структурування журнальних даних, а Kibana - для візуалізації та аналізу цих даних.

Splunk - це комерційна платформа для обробки та аналізу журнальних даних. Вона надає широкий спектр інструментів для виявлення аномалій, моніторингу безпеки та аналізу подій.

Graylog - це відкрите програмне забезпечення для обробки та аналізу журнальних даних. Воно надає інтерфейс для пошуку, фільтрації та аналізу подій, а також можливості візуалізації даних.

LogRhythm - це інтегрована платформа для моніторингу безпеки та обробки журнальних даних. Вона надає інструменти для виявлення загроз, аналізу подій та відстеження аномалій в реальному часі.

SolarWinds Log & Event Manager: Ця система спеціалізується на моніторингу безпеки та аналізі подій. Вона допомагає виявляти підозрілі активності та інциденти безпеки.

Apache Flume: Apache Flume - це інструмент для збору та передачі журнальних даних. Він дозволяє інтегрувати різні джерела даних та надсилати їх до систем журналювання для подальшого аналізу.

Rsyslog - це відкрите програмне забезпечення, яке забезпечує функціональність журналювання в Unix-подібних операційних системах. Воно дозволяє налаштовувати і пересилати журнальні дані з різних джерел.

Кожна з цих систем журналювання має свої переваги і може бути використана для виявлення та відстеження подій в залежності від конкретних потреб та вимог вашого проекту або організації.

Системи виявлення вторгнень (IDS) та системи запобігання вторгненням (IPS) є важливими інструментами для забезпечення безпеки у хмарному середовищі [5]. У цьому розділі ми розглядаємо ключові аспекти цих систем та їх роль у моніторингу та реагуванні на потенційні загрози. Важливим аспектом є вибір систем IDS/IPS, які найкращим чином відповідають потребам організації та ресурсам хмарного середовища. IDS відстежує та аналізує мережевий трафік та події з метою виявлення потенційних загроз та вторгнень. У свою чергу, IPS, окрім виявлення загроз, може активно намагатися блокувати або запобігати їх виникненню, застосовуючи дієві заходи безпеки. Системи виявлення вторгнень можуть бути мережевими IDS (NIDS), що аналізують мережевий трафік, або хостовими IDS (HIDS), встановлені на конкретних хостах для аналізу їх активності. Системи запобігання вторгненням можуть бути мережевими IPS

(NIPS), розташованими на рівні мережі, або хостовими IPS (HIPS), встановленими на окремих хостах.

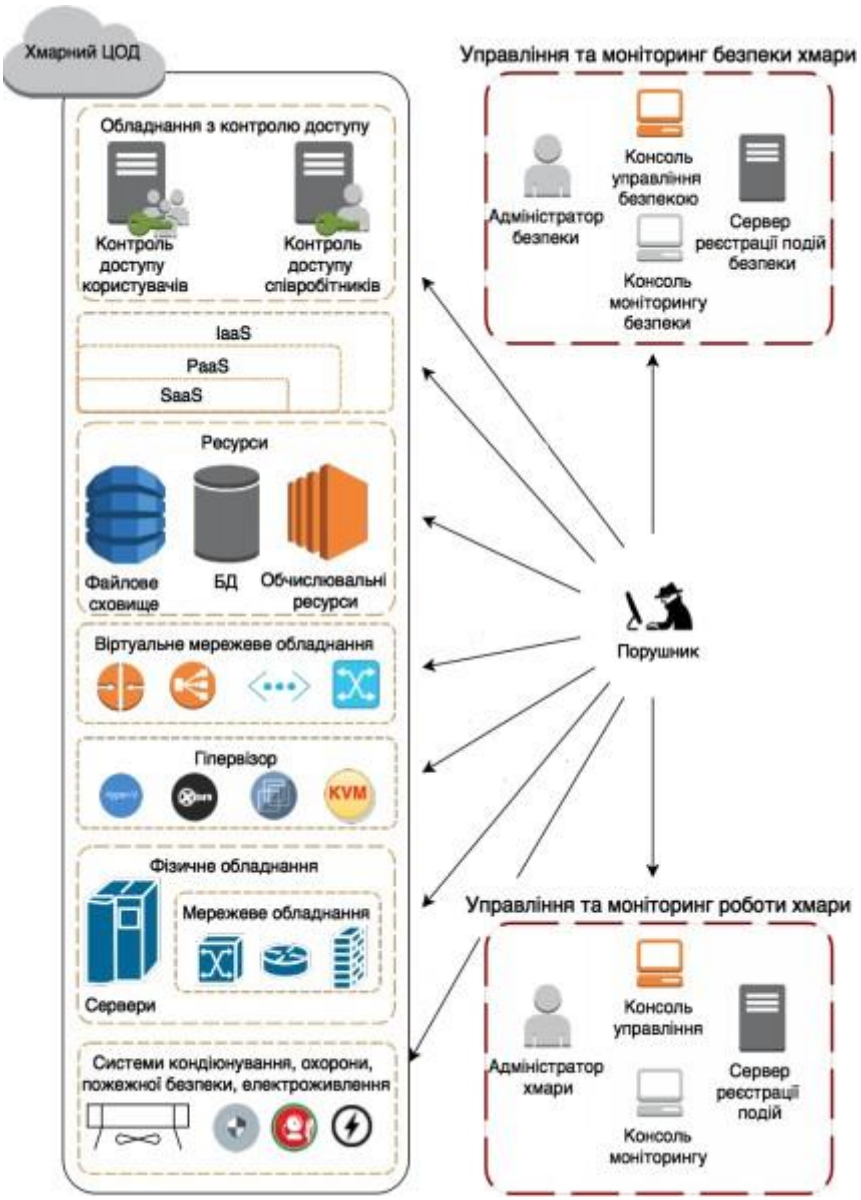


Рис 1.1 Модель загроз для хмарних обчислень

В нашому дослідженні ми розглядаємо різноманітні види вторгнень, включаючи атаки на рівні мережі, веб-додатків, використання вразливостей та внутрішні вторгнення. Підбір систем IDS/IPS у хмарних обчисленнях вимагає уважного розгляду різних критеріїв, включаючи продуктивність, сумісність з існуючою інфраструктурою, можливості адаптації до змінного середовища та вартість [6]. Важливо зрозуміти, як системи IDS/IPS можуть бути інтегровані з іншими системами моніторингу та безпеки в хмарних обчисленнях. Крім того, необхідно вивчити методи управління та налаштування цих систем для

досягнення найкращої ефективності. Розглядаючи виклики та переваги використання систем IDS/IPS в хмарних обчисленнях, ми враховуємо масштабованість, збереження безпеки даних та ефективність їхнього використання.

Аналіз мережевого трафіку полягає у моніторингу та інтерпретації пакетів даних, що передаються в мережі. Один зі способів цього аналізу - використання систем IDS/NIDS (систем виявлення вторгнень на рівні мережі). Ці системи аналізують трафік і використовують сигнатури або аналітичні методи для виявлення патернів, які можуть вказувати на вторгнення або аномальну активність. Такий аналіз включає в себе:

- **Виявлення вторгнень**: IDS/NIDS можуть виявити атаки, такі як SQL-ін'єкція, кросс-сайт скриптування, атаки на вразливості та інші спроби злому.
- **Моніторинг зловмисних активностей**: Ці системи аналізують пакети даних на предмет знаків аномалій, таких як надмірний обсяг запитів, підозріла поведінка користувачів чи спроби несанкціонованого доступу.
- **Виявлення атак з використанням патернів**: IDS використовують патерни атак, щоб виявити типові сценарії вторгнень, що дозволяє вчасно реагувати на відомі загрози.

Виявлення аномалій - це метод аналізу, який спрямований на виявлення незвичайної або підозрілої активності, що може свідчити про потенційні загрози без використання конкретних сигнатур. Цей підхід враховує "нормальну" поведінку системи та користувачів і сповіщає про будь-які відхилення від цієї норми. Важливими аспектами виявлення аномалій є:

- **Моделювання поведінки**: В системах виявлення аномалій використовуються алгоритми для створення моделей нормальної поведінки. Ці моделі порівнюються з актуальною активністю для виявлення відхилень.

- **Системи машинного навчання**: Використання методів машинного навчання, таких як нейронні мережі та класифікатори, для ідентифікації аномалій в реальному часі.
- **Підозрілість та контекст**: Аналіз аномалій враховує не тільки самі аномалії, але й контекст, у якому вони відбуваються, що дозволяє визначити, чи є вони загрозою.

Відстежування трафіку та виявлення аномалій в хмарних обчисленнях вимагає поєднання технічних рішень та аналітичних підходів для забезпечення максимальної безпеки в середовищі, де загрози постійно змінюються та еволюціонують [7]. Моніторинг та аналіз трафіку дозволяють виявляти загрози в реальному часі, забезпечуючи вчасну реакцію та захист хмарної інфраструктури та даних.

В рамках хмарних обчислень, забезпечення безпеки та виконання аудиту стають більш складними завданнями через розподіленість обчислень та інфраструктуру, що використовується. Автоматизовані системи аудиту безпеки відіграють ключову роль у виявленні, відстеженні та аналізі подій, які можуть вказувати на порушення безпеки. У цьому розділі розглянемо значення та функції таких систем, а також їх роль у забезпеченні безпеки в хмарних обчисленнях.

Аудит безпеки - це процес відстеження та аналізу активності, яка може вказувати на загрози та порушення безпеки. В хмарних обчисленнях важливо мати інструменти, які автоматизовано здійснюють цей аналіз, оскільки ручний аудит стає надто обтяжним та неможливим завданням через масштаби та розподіленість інфраструктури [8].

Розглядаючи впровадження автоматизованих систем аудиту безпеки, необхідно враховувати виклики, які можуть виникнути, але й переваги, які вони надають. Хмарні обчислення можуть бути більш розподіленими та гетерогенними, що ускладнює збір та аналіз даних безпеки. Обсяги журналів подій та даних можуть бути величезними, що потребує високоефективних рішень для зберігання та обробки таких обсягів. Хмарні обчислення можуть бути

вразливими до специфічних загроз, таких як атаки на мережевий периметр, атаки на ідентифікацію та контроль доступу [9]. Системи аудиту повинні бути налаштовані для виявлення цих загроз. Проте переваг набагато більше. Автоматизовані системи надають можливість в реальному часі реагувати на події та загрози безпеки, що дозволяє швидко приймати відповідні заходи. Також зменшення ручної роботи, вони автоматично здійснюють багато процесів аналізу та генерації звітів, що допомагає зменшити навантаження на персонал. Автоматизовані системи можуть аналізувати великі обсяги даних та виявляти навіть тонкі відхилення від норми. Аналіз подій дозволяє оцінити стан безпеки та вдосконалити стратегію безпеки в організації.

У підсумку, автоматизовані системи аудиту безпеки важливі для забезпечення безпеки в хмарних обчисленнях. Вони надають можливість виявляти та реагувати на загрози в реальному часі, спрощують аналіз та створення звітів, та допомагають підвищити рівень безпеки в організації, що використовує хмарні ресурси.

Іншою важливою перевагою є зменшення витрат. Організації можуть уникнути значних витрат на закупівлю та підтримку власної обчислювальної інфраструктури, оскільки хмарні сервіси надають обчислювальну потужність та зберігання даних на підписку.

Глобальний доступ, який надає хмарне обчислення, перетворює спосіб роботи для багатьох організацій та користувачів у всьому світі. Ця перевага відкриває безмежні можливості та перетворює традиційний підхід до роботи.

Забезпечуючи глобальний доступ, хмарні середовища дозволяють користувачам отримувати доступ до своїх даних, програмних засобів та ресурсів з будь-якого місця, де існує доступ до Інтернету. Це робить робочий процес більш гнучким та зручним, оскільки користувачі не обмежені місцем знаходження. Децентралізовані організації, які розподілені по всьому світу, отримують можливість спільно працювати, обмінюватися даними та доступ до спільних ресурсів без обмежень, що раніше ставило під загрозу продуктивність та ефективність.

Ця глобальна доступність також відкриває двері для розвитку та зростання бізнесів. Організації можуть розширювати свої операції на міжнародному рівні, не створюючи додаткової інфраструктури. Робіть бізнес з будь-якої точки світу та привертайте клієнтів, співробітників та партнерів з будь-якого куточка глобального ринку.

Ще однією важливою перевагою хмарних середовищ є їхній вплив на швидкість впровадження нових сервісів та застосунків. Завдяки відкритості та готовності хмарних середовищ, організації можуть швидко впроваджувати нові рішення та послуги. Ця швидкість реалізації дозволяє організаціям залишатися конкурентоспроможними у швидкозмінному інформаційному середовищі.

Прикладом може бути бізнес, який шукає спосіб покращити обслуговування клієнтів [10]. За використанням хмарних середовищ, компанія може швидко впроваджувати нові програми та засоби для взаємодії з клієнтами, реагуючи на їхні потреби та побажання. Це дає можливість залишатися конкурентоспроможним та реагувати на зміни в ринковому середовищі.

Отже, глобальний доступ та швидкість впровадження нових сервісів і застосунків завдають суттєвого впливу на сучасний бізнес та робочі процеси, роблячи їх більш гнучкими, зручними та швидкими. Використання хмарних середовищ дає можливість організаціям бути готовими до викликів і можливостей сучасного інформаційного світу.

Сучасний бізнес надзвичайно динамічний та конкурентний. Щоб вижити та процвітати в цьому середовищі, підприємства повинні бути готові швидко реагувати на зміни в попиті, технологіях та конкурентному середовищі. У цьому контексті хмарні середовища стають ключовим інструментом для бізнесу, оскільки вони надають підприємствам можливість бути більш гнучкими та ефективними.

Інфраструктура як послуга *(IaaS)* відкриває перед підприємствами можливості збільшувати або зменшувати потужність серверів в залежності від навантаження [11]. Це означає, що компанії можуть ефективно реагувати на коливання в обсязі роботи, не витрачаючи коштів на зайве обладнання чи його

обслуговування. Замість цього вони оплачують лише те, що реально використовують, знижуючи витрати та підвищуючи продуктивність.

Платформа як послуга *(PaaS)* надає підприємствам інструменти для розробки, тестування та впровадження програм. Це спрощує процес створення програмних продуктів та дозволяє бізнесу швидко реагувати на нові вимоги ринку. Розробники можуть концентруватися на створенні інноваційних рішень, не витрачаючи час на налаштування інфраструктури.

Програмне забезпечення як послуга *(SaaS)* надає доступ до програмних засобів через Інтернет. Це дозволяє підприємствам отримувати доступ до сучасних інструментів та додатків без необхідності їхнього власного обслуговування. Відкритість та готовність SaaS-рішень дозволяють бізнесу легко впроваджувати нові технології та розширювати свої можливості, що дозволяє підприємствам бути на передових рубежах та залишатися конкурентоспроможними в сучасному світі.

Загалом, хмарні середовища є ключовими для наукового прогресу та досліджень, допомагаючи вченим впроваджувати нові ідеї, вирішувати глобальні виклики та здійснювати науковий прорив.

Хмарні середовища є потужними інструментами, але вони не приходять без викликів та ризиків. Організації та користувачі повинні бути готові до розуміння та управління цими аспектами, щоб забезпечити безпеку та ефективність свого використання хмарних технологій. В цьому розділі ми розглянемо основні виклики та ризики, пов'язані з хмарними середовищами.

Загрози та вразливості: одним із найбільших викликів у використанні хмарних середовищ є забезпечення безпеки даних. Дані, які зберігаються та обробляються в хмарі, можуть піддаватися різноманітним загрозам та вразливостям. Кібератаки стають все більш винахідливими та агресивними, загрожуючи конфіденційності та цілісності даних. Витік інформації стає серйозним викликом для організацій, особливо тих, які обробляють конфіденційні дані клієнтів та партнерів. Крім того, технічні проблеми, такі як

відмови обладнання або послуг, можуть призвести до втрати даних та доступу до них.

Шифрування даних : одним із найважливіших кроків у забезпеченні безпеки даних в хмарних середовищах є шифрування даних. Шифрування даних перетворює їх в нечитабельний формат, який може бути розшифрований лише з використанням правильного ключа. Це допомагає захистити дані від несанкціонованого доступу, навіть якщо зловмисники отримають фізичний доступ до серверів або сховищ. Застосування сильного шифрування є обов'язковим для захисту конфіденційної інформації та особистих даних користувачів [12].

Захист від несанкціонованого доступу: окрім шифрування, важливо вживати заходів для захисту від несанкціонованого доступу до даних. Це включає в себе правильне налаштування прав доступу, аутентифікацію користувачів та моніторинг активності. Двофакторна аутентифікація, яка вимагає введення не лише пароля, але й додаткового підтвердження, може значно підвищити безпеку. Регулярний аудит безпеки та моніторинг дозволяють вчасно виявляти незвичайну активність та реагувати на потенційні загрози.

Запобігання втраті даних : важливим аспектом в забезпеченні безпеки даних є запобігання втраті даних через технічні проблеми. Резервне копіювання та реплікація даних можуть допомогти зменшити ризик втрати даних при виникненні несподіваних ситуацій. Також важливо мати план відновлення даних, який дозволить швидко відновити доступ до даних у випадку аварій.

Загалом, забезпечення безпеки даних в хмарних середовищах є критично важливою задачею. Використання шифрування, захисту від несанкціонованого доступу та запобігання втраті даних допомагають зменшити ризик і забезпечити конфіденційність та цілісність даних в хмарних середовищах.

Відмови та надійність : ще одним ключовим аспектом використання хмарних середовищ є надійність та доступність послуг. Оскільки багато організацій розраховують на хмарні сервіси для виконання своїх операцій, навіть

невеликі відмови серверів або мережеві перебої можуть призвести до серйозних втрат і простоїв.

Загалом, забезпечення надійності та доступності хмарних середовищ вимагає комплексного підходу та використання різних технологій та стратегій. Організації повинні бути готові до можливих технічних проблем та вміти ефективно реагувати на них для забезпечення безперервної роботи та захисту даних.

Залежність організацій від постачальників хмарних послуг може призвести до ризиків та втрати контролю над важливою інфраструктурою та даними [13]. Це особливо актуально, оскільки деякі постачальники можуть зазнавати перебоїв у наданні послуг або навіть закривати свої бізнеси. Організації повинні бути обачні при виборі постачальників та ретельно аналізувати їхню надійність та історію обслуговування клієнтів.

Однак організації також повинні мати плани та можливості для міграції даних у разі потреби. Міграція даних може стати необхідною, якщо вони вирішать змінити постачальника чи повернути певну частину інфраструктури під свій контроль. Проте, це може бути складним завданням, оскільки перенесення великої кількості даних може вимагати значних зусиль, часу та ресурсів.

Для забезпечення безпеки та надійності міграції даних, організації повинні розробити докладну стратегію, включаючи плани та процедури для переміщення даних та перевірки їхньої цілісності після міграції. Важливо також регулярно аудитувати та резервно копіювати дані, щоб запобігти можливим втратам чи пошкодженню під час міграції.

Загалом, залежність від постачальників хмарних послуг потребує обачності та готовності до різних сценаріїв, включаючи можливість міграції даних та відновлення контролю над інфраструктурою.

Одним із серйозних ризиків, пов'язаних з використанням хмарних середовищ, є можливість відмовки постачальника хмарних послуг. У разі, якщо постачальник припиняє свою діяльність або внесе зміни в умови обслуговування,

це може серйозно вплинути на доступність та надійність послуг для користувачів. Організації, які користуються хмарними середовищами, повинні бути готові до таких можливих сценаріїв. Вони повинні уважно аналізувати умови угоди з постачальником та розробляти плани реагування та міграції, щоб зменшити вплив можливих відмов.

Питання приватності даних може бути ще однією важливою аспектом, особливо коли дані зберігаються на серверах, розташованих за кордоном. Законодавство щодо захисту особистих даних може відрізнятися в різних країнах, і організації повинні бути готові до відповідності вимогам, які можуть містити шифрування даних та контроль над доступом.

Одним із ключових викликів є забезпечення надійності та доступності хмарних послуг. Відмови серверів, відключення Інтернету та інші технічні проблеми можуть призвести до призупинення роботи сервісів та втрати доступу до важливих даних. Організації повинні розробляти плани відновлення після аварій та забезпечувати резервне копіювання даних для зменшення впливу можливих негараздів.

Забезпечення Безпеки та Відновлення

Для забезпечення безпеки та відновлення важливо розробити стратегії та політики безпеки, які містити шифрування даних, сильну аутентифікацію та авторизацію. Також необхідно регулярно проводити аудит безпеки та моніторинг діяльності в хмарних середовищах. Важливо мати плани відновлення після кібератаки та регулярно їх тестувати, щоб забезпечити найшвидше відновлення послуг у разі потреби.

Управління Вартістю та Приватністю

Організації повинні внести в управління витратами та вартістю використання хмарних послуг. Це може включати в себе моніторинг ресурсів, оптимізацію витрат та регулярний аналіз тарифікації та ліцензування.

Щодо приватності даних, організації повинні бути готові до дотримання різних локальних та міжнародних стандартів та регуляції щодо захисту

особистих даних. Важливо розуміти обмеження щодо переміщення даних через кордон та мати відповідні заходи для їхнього захисту.

Надійність та Доступність

Забезпечення надійності та доступності послуг в хмарних середовищах вимагає розробки планів відновлення після аварій та забезпечення резервного копіювання даних. Організації повинні бути готові до швидкого відновлення роботи у разі виникнення негараздів та забезпечити постійний доступ до важливих даних.

В цілому, використання хмарних середовищ надає багато переваг, але також вимагає уваги до зазначених викликів та ризиків. Організації повинні бути готові до вирішення цих аспектів та розробляти стратегії для їх врегулювання з метою забезпечення успішного та безпечного використання хмарних технологій.

1.2 Оцінка безпеки і аналіз хмарного середовища AWS

Amazon Web Services (AWS) являє собою одного з головних гравців у галузі хмарних обчислень. Платформа AWS пропонує широкий спектр інфраструктурних та платформних послуг, які дозволяють клієнтам легко розгортати та масштабувати свої додатки і обчислення. AWS включає в себе такі послуги, як обчислювальні і ресурси для зберігання даних, машинне навчання, інтернет-речей, та багато інших. Така гнучкість та розширюваність робить AWS привабливим вибором для численних організацій, але при цьому виникають складності щодо забезпечення безпеки цієї інфраструктури та послуг.

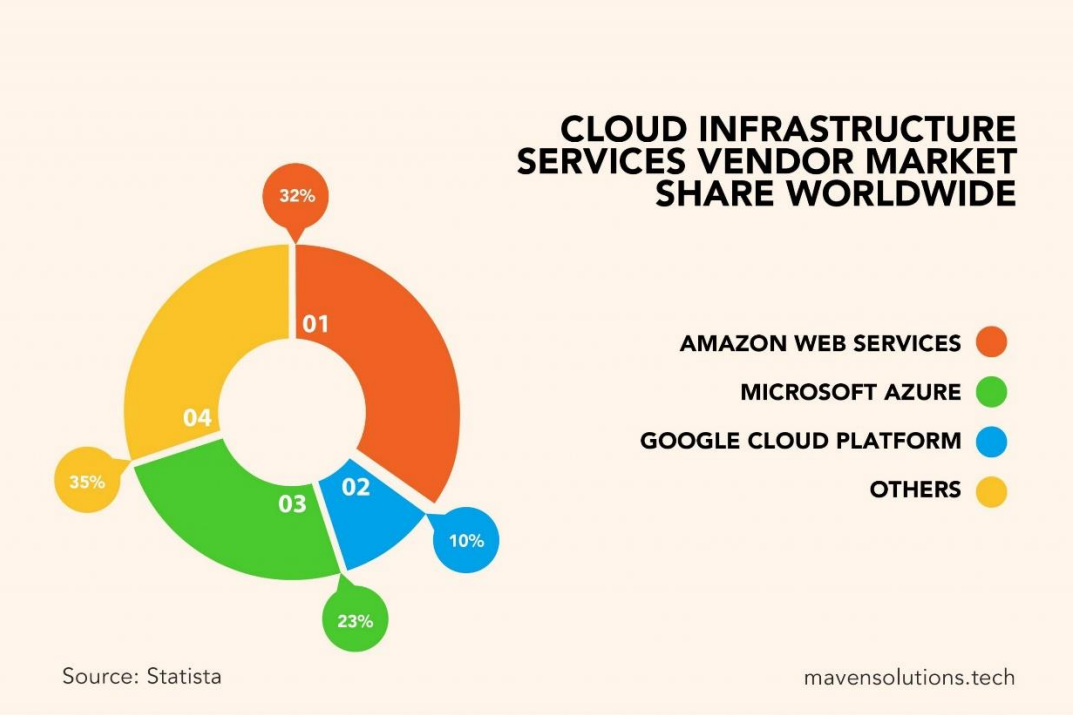


Рисунок 1.2 — Лідери на ринку AWS, Azure та Google Cloud

Існують сотні спеціалізованих хмарних провайдерів, але для більшості сценаріїв актуальним є вибір між трьома лідерами ринку - AWS, Azure та Google Cloud. Разом вони контролюють 65% частки ринку.

Безпека в хмарі AWS є складною темою, яка охоплює широкий спектр аспектів. Фізична безпека, технічні аспекти безпеки, адміністративні заходи безпеки та організаційні процедури безпеки – все це важливі фактори, які слід враховувати при забезпеченні безпеки в хмарі AWS [14].

Фізична безпека хмарної інфраструктури AWS є важливим аспектом її безпеки. Amazon має розгалужену мережу центрів обробки даних по всьому світу, які захищені за допомогою фізичних заходів безпеки, таких як охорона, відеоспостереження та контроль доступу. Компанія також має політику прозорості, яка дозволяє клієнтам знайомитися з фізичними заходами безпеки, що застосовуються в їхніх центрах обробки даних.

Технічні аспекти безпеки в хмарі AWS охоплюють широкий спектр заходів, таких як шифрування даних, аутентифікація користувачів та управління доступом, брандмауерінг та аналіз загроз. AWS пропонує широкий спектр технічних функцій безпеки, які клієнти можуть використовувати для захисту своїх даних і інфраструктури.

Адміністративні заходи безпеки в хмарі AWS охоплюють політику безпеки, процеси та процедури, які використовуються для захисту даних і інфраструктури. Amazon надає клієнтам широкий спектр ресурсів для розробки та впровадження політики безпеки, яка відповідає їхнім потребам.

Організаційні процедури безпеки в хмарі AWS охоплюють практики, які використовуються для забезпечення безпеки даних і інфраструктури. Amazon рекомендує клієнтам розробити та впровадити організаційні процедури безпеки, такі як управління ризиками, управління знаннями та навчання персоналу.

Оцінка безпеки в хмарному середовищі AWS вимагає глибокого аналізу потенційних ризиків та вразливостей, які можуть виникнути в контексті цієї платформи. Цей розділ розглядає різноманітні аспекти, пов'язані з безпекою, і включає в себе наступні елементи:

Аналіз загроз та потенційних ризиків

Аналіз безпеки в AWS розпочинається з ідентифікації можливих загроз та ризиків [15]. Це включає в себе визначення потенційних сценаріїв атак, а також загроз, що можуть бути специфічними для хмарного середовища, такі як атаки на API, атаки на служби аутентифікації та авторизації, а також атаки на дані.

Визначення слабких місць та потенційних вразливостей

Одним із ключових завдань в оцінці безпеки є ідентифікація слабких місць та потенційних вразливостей, які можуть бути використані зловмисниками для здійснення атак. Цей процес включає в себе аналіз конфігурацій систем, контроль доступу, шифрування та інші аспекти, які можуть створювати можливості для атак.

Оцінка впливу та ймовірності ризиків

Після ідентифікації ризиків та вразливостей, проводиться оцінка їхнього впливу на безпеку системи та ймовірності виникнення конкретних загроз. Це допомагає визначити, які ризики є найбільш критичними та потребують негайного усунення.

Моніторинг та оновлення безпеки

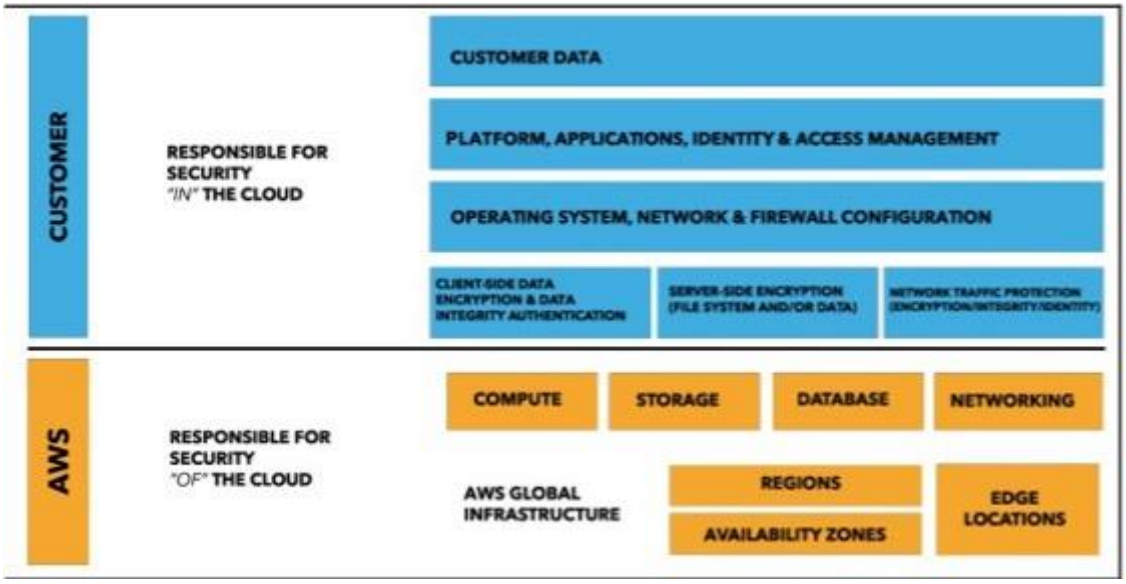


Рис 1.3 Модель спільної відповідальності за безпеку AWS.

Аспекти моніторингу та постійного оновлення безпеки. Це важливо для забезпечення захисту в хмарному середовищі, оскільки загрози постійно змінюються. Постійний моніторинг та адаптація стратегій безпеки допомагають зберегти високий рівень захисту в AWS.

Цей розділ служить основою для розробки та впровадження стратегій безпеки в хмарному середовищі AWS та допомагає забезпечити надійний захист від потенційних загроз та вразливостей**.**

Один із аспектів, який має вирішальне значення для забезпечення безпеки в хмарному середовищі Amazon Web Services (AWS), полягає в порівнянні стандартів безпеки, які визначають вимоги до безпеки інфраструктури, і які формулюються самою AWS, з міжнародними стандартами безпеки. Такий аналіз дозволяє нам краще зрозуміти, як AWS відповідає міжнародним стандартам безпеки, а також визначити ті області, де можуть існувати розходження або додаткові вимоги від AWS.

Спершу, важливо розглянути спільність вимог та рекомендацій між рекомендаціями AWS і міжнародними стандартами безпеки. Загальні принципи безпеки, такі як контроль доступу, аудит безпеки, шифрування та інші, є важливими елементами будь-якої стратегії безпеки.

З іншого боку, необхідно ретельно вивчити різницю між рекомендаціями AWS та міжнародними стандартами безпеки [16]. Деякі міжнародні стандарти

можуть встановлювати додаткові вимоги або надавати більше деталей щодо реалізації конкретних аспектів безпеки. Важливо ідентифікувати ті області, де можуть існувати розходження або додаткові вимоги, і розуміти, чому це важливо для безпеки в хмарному середовищі.

Після встановлення різниці і спільності вимог, можна розглянути, які конкретні заходи та процедури можуть бути необхідні для впровадження міжнародних стандартів безпеки в хмарному середовищі AWS. Це включає в себе питання конфігурації систем, моніторингу та забезпечення відповідності стандартам безпеки. Важливо розглянути, які зміни можуть бути необхідні для виконання міжнародних стандартів у контексті AWS.

Завершальний пункт цього розділу стосується переваг та обмежень конформності стандартів безпеки. Дотримання міжнародних стандартів безпеки може мати велику перевагу для організацій, оскільки це підтверджує високий рівень безпеки та довіру. Такий аналіз допомагає визначити, наскільки підходи AWS і міжнародних стандартів збігаються або відрізняються, і як можна покращити безпеку хмарного середовища.

1.3 Розробка системи управління безпекою застосунків

У сучасному світі, де все більше і більше застосунків переміщуються в хмарні середовища, забезпечення безпеки застосунків стає все більш важливим завданням. Хмарні середовища мають ряд особливостей, які роблять їх більш уразливими до атак, ніж традиційні середовища. Наприклад, у хмарних середовищах застосунки часто є віртуалізованими, що може ускладнити їх ідентифікацію та захист. Крім того, хмарні середовища часто є динамічними, що може ускладнити моніторинг безпеки.

Для забезпечення безпеки застосунків у хмарному середовищі AWS необхідно використовувати комплексний підхід, який включає в себе такі заходи:

- Розробка застосунків з урахуванням безпеки

- Впровадження процесів та процедур управління безпекою застосунків
- Використання інструментів та технологій безпеки

Одним із важливих елементів комплексного підходу до забезпечення безпеки застосунків у хмарному середовищі AWS є система автоматизованого керування безпекою застосунків вона дозволяє здійснювати безперервний моніторинг безпеки застосунків та оперативно реагувати на виявлення загроз.

Мета: забезпечення безперервного моніторингу безпеки застосунків та оперативного реагування на виявлення загроз.

Основні завдання:

1. **Моніторинг безпеки застосунків**
   - Виявлення вразливостей застосунків
   - Виявлення несанкціонованого доступу до застосунків
   - Виявлення аномалій у поведінці застосунків
2. **Реагування на виявлення загроз**
   - Сигналізація про виявлення загроз
   - Автоматичне усунення загроз
   - Ручне усунення загроз

Підсистема моніторингу застосунків

Підсистема моніторингу застосунків є однією з основних підсистем. Вона відповідає за виявлення вразливостей застосунків, несанкціонованого доступу до застосунків та аномалій у поведінці застосунків.

Така система реалізується за допомогою набору інструментів, які можуть бути згруповані за такими категоріями:

- Інструменти виявлення вразливостей та аналізу коду
- Інструменти динамічного аналізу коду та сканування мережі
- Інструменти виявлення несанкціонованого доступу
- Інструменти аудиту системного журналу та виявлення вторгнень
- Інструменти виявлення аномалій у поведінці застосунків
- Інструменти аналізу метрик продуктивності та аналізу трафіку

2. АРХІТЕКТУРА ТА ПРИНЦИПИ РОБОТИ ПІДСИСТЕМИ МОНІТОРИНГУ ЗАСТОСУНКІВ

2.1 Компоненти підсистеми моніторингу застосунків

Збирач даних виступає у ролі першого компонента в складі підсистеми моніторингу застосунків. Основною метою цього компонента є збір інформації про стан застосунків. Для досягнення цієї мети збирач даних може використовувати різноманітні джерела інформації, такі як журнали, метрики та дані, отримані в результаті подій [17].

Журнали є одним із ключових джерел інформації для збирача даних в підсистемі моніторингу застосунків. Ці журнали включають в себе записи подій, які відбуваються в межах конкретного застосунку. Їхнім основним завданням є документування різноманітних аспектів функціонування програми.

В контексті моніторингу застосунків, журнали можуть містити різноманітну інформацію, включаючи, але не обмежуючись:

1. **Вхідні підключення:** Інформація про події, пов'язані з встановленням з'єднань з іншими системами чи користувачами. Це може включати дані про успішні та невдалі підключення.
2. **Виконання команд:** Записи про виконані команди чи операції в межах застосунку. Це може включати виклики функцій, введені команди або інші важливі операції.
3. **Запуск скриптів:** Інформація про виконання скриптів чи автоматизованих процесів у застосунку. Це може бути важливим для виявлення аномалій чи несправностей у виконанні програмних компонентів.

Загальна мета використання журналів у контексті моніторингу полягає в тому, щоб мати повну та структуровану інформацію про функціонування застосунків. Це дозволяє аналізувати та виявляти проблеми, вдосконалювати продуктивність та забезпечувати ефективний моніторинг застосунків.

Метрики в підсистемі моніторингу застосунків [18] являють собою важливий інструмент для отримання детальної інформації про стан програмного

забезпечення. Їхня основна роль полягає в тому, щоб вимірювати та документувати різні аспекти функціонування застосунку, забезпечуючи велику кількість даних для подальшого аналізу та прийняття рішень.

В контексті моніторингу застосунків, метрики можуть включати:

1. **Доступність:** Ця метрика вказує на те, наскільки доступний є застосунок для користувачів. Вимірюється в часі безвідмовної роботи та може вказувати на можливі проблеми зі стабільністю.
2. **Продуктивність:** Метрики продуктивності надають інформацію про швидкодію та ефективність застосунку. Це може включати час відгуку на запити, час завантаження сторінок або ефективність виконання операцій.
3. **Використання ресурсів:** Ці метрики вказують на те, як ефективно використовуються ресурси системи застосунком. Це може охоплювати використання пам'яті, процесорного часу, мережевого трафіку та інших ресурсів.

Збір та аналіз метрик дозволяє виявляти тенденції, визначати патерни та передбачати можливі проблеми з ефективністю чи стабільністю застосунків. Це важливо для того, щоб оперативно реагувати на проблеми та оптимізувати функціонування системи. Метрики стають основою для моніторингу та управління ефективністю застосунків у реальному часі.

Дані з подієвості у контексті моніторингу хмарних рішень є джерелом важливої інформації, оскільки вони дозволяють відстежувати та аналізувати події, що відбуваються в хмарному середовищі [19]. Ці події можуть стосуватися різноманітних аспектів, включаючи конфігураційні зміни, використання ресурсів та мережевий трафік.

1. **Зміни в конфігурації:** Дані подієвості можуть містити інформацію про будь-які зміни в налаштуваннях хмарної інфраструктури. Це включає в себе зміни параметрів віртуальних машин, мережевих налаштувань, політик безпеки та інших конфігураційних параметрів.
2. **Використання ресурсів:** Дані подієвості можуть надавати інформацію про використання ресурсів, таких як процесорний час, обсяг оперативної

пам'яті, дисковий простір тощо. Це допомагає виявляти аномалії в споживанні ресурсів та оптимізувати їх використання.

3. **Мережевий трафік:** Дані з подієвості можуть також включати інформацію про мережевий трафік в хмарному середовищі. Це може включати дані про передачу даних між компонентами, взаємодію між сервісами та інші аспекти мережевої активності.

Ці дані дозволяють не лише виявляти проблеми та аномалії, але й забезпечують можливість проведення аналізу та прогнозування для оптимізації роботи хмарних рішень. Вони є ключовим елементом для забезпечення ефективного моніторингу, управління та підтримки оптимального функціонування хмарних середовищ.

Для ефективної реалізації збирача даних у хмарному середовищі Amazon Web Services (AWS), можна використовувати різні сервіси [20], кожен з яких спеціалізується на конкретному типі даних або джерела інформації. Нижче розглянуті три основні сервіси AWS, які можуть використовуватися для цієї мети:

1. **CloudWatch Logs:**

CloudWatch Logs - це сервіс, який забезпечує збір та зберігання журналів з різних джерел у хмарному середовищі AWS. Він може отримувати журнали від таких сервісів, як EC2 (Amazon Elastic Compute Cloud), ECS (Amazon Elastic Container Service), Lambda та CloudTrail. Основна функція цього сервісу полягає в можливості централізованого збору, моніторингу та аналізу журналів для виявлення проблем та трасування подій у розподілених середовищах.

2. **CloudWatch Metrics:**

CloudWatch Metrics - це сервіс, призначений для збору та зберігання метрик з різних джерел у хмарному середовищі AWS. Він здатен збирати метрики з різних сервісів, таких як EC2, ECS, Lambda та CloudTrail. Основна функція цього сервісу полягає в забезпеченні можливості вимірювання різних аспектів продуктивності та ефективності, таких як використання ресурсів, час відгуку та інші ключові показники.

3. **CloudTrail:**

CloudTrail - це сервіс, призначений для збору та зберігання даних про події, що відбуваються в хмарі AWS. Він записує події, які стосуються різних сервісів та ресурсів в AWS. Основна функція CloudTrail полягає в тому, щоб забезпечити детальну історію змін у конфігурації, доступ до ресурсів та інші важливі події, що відбуваються в хмарному середовищі.

Використання цих сервісів в поєднанні може допомогти створити повноцінний збирач даних, який охоплює різноманітні типи інформації та забезпечує повний обсяг моніторингу та аналізу для оптимізації та підтримки хмарних середовищ AWS.

Аналізатор даних, який представляє собою другий компонент підсистеми моніторингу застосунків, відіграє ключову роль у подальшому опрацюванні та розумінні даних, які були зібрані збирачем даних. Основна функція цього компонента полягає в проведенні аналізу отриманих інформаційних потоків для виявлення потенційних загроз та порушень безпеки в розглядуваному застосунку [21].

Для досягнення цієї мети, аналізатор даних використовує різноманітні алгоритми та моделі аналізу. Ці методи можуть включати в себе статистичний аналіз, машинне навчання, інтелектуальний аналіз даних та інші техніки, спрямовані на виявлення аномалій та несподіваних змін у поведінці застосунків.

Для реалізації аналізатора даних у хмарному середовищі Amazon Web Services (AWS) можна використовувати два ключові сервіси: Amazon SageMaker та Amazon Detective. Кожен з цих сервісів виконує визначену роль у підсистемі аналізу даних, спрямованій на виявлення потенційних загроз та порушень безпеки.

Amazon SageMaker - це сервіс для машинного навчання у хмарному середовищі AWS, який надає інструменти та сервіси для розробки, тренування та впровадження моделей машинного навчання. Однією з ключових його функцій є можливість використання алгоритмів, спеціально розроблених для виявлення аномалій та інцидентів безпеки. Для аналізу даних цей сервіс

користується інформацією, зібраною з інших сервісів, таких як CloudWatch Logs, CloudWatch Metrics або CloudTrail. SageMaker надає інструменти для ефективної роботи з цими даними, дозволяючи будувати та вдосконалювати моделі для виявлення аномалій.

Amazon Detective - це хмарна платформа, призначена для виявлення та розслідування інцидентів безпеки в середовищі AWS. Цей сервіс автоматично використовує дані з AWS CloudTrail, VPC Flow Logs та інших сервісів, щоб створити повне зображення подій, що відбуваються у вашому хмарному середовищі. Detective надає аналітичні можливості для визначення структурованих інформаційних зв'язків між різними подіями та об'єктами, спрощуючи розслідування інцидентів безпеки.

Застосування цих сервісів у підсистемі аналізу даних дозволяє використовувати передові методи та інструменти для ефективного виявлення та реагування на можливі загрози та порушення безпеки в хмарному середовищі AWS.

Система реагування, яка представляє собою третій компонент підсистеми моніторингу застосунків [22], відіграє невід'ємну роль у забезпеченні безпеки та стійкості роботи застосунків у хмарному середовищі. Її функції охоплюють виявлення, оцінку та реагування на потенційні загрози та порушення безпеки. Нижче розглянуті ключові аспекти роботи системи реагування:

1. **Виявлення загроз:** Система реагування починає свою діяльність з аналізу даних, які були зібрані попередніми компонентами, зокрема збирачем та аналізатором даних. Вона виявляє потенційні загрози та аномалії в системі, використовуючи різні алгоритми та моделі.
2. **Оцінка та класифікація загроз:** Після виявлення потенційних проблем система реагування проводить їх оцінку та класифікацію. Цей етап визначає серйозність та природу загрози для подальших дій.
3. **Реакція на загрози:** На основі оцінки система реагування вживає конкретних заходів для нейтралізації загроз та запобігання подальшим порушенням. До можливих заходів можуть входити сповіщення персоналу

безпеки, автоматичне блокування доступу до застосунку або відновлення застосунку до попереднього стану.

4. **Сповіщення та реєстрація подій:** Система реагування може генерувати сповіщення для відповідального персоналу або адміністраторів, повідомляючи про виявлені загрози та заходи, які вживаються для їх вирішення. Також може вести журнал подій для подальшого аналізу та аудиту.
5. **Адаптація до змін у середовищі:** Система реагування повинна бути здатна адаптуватися до змін у середовищі та вдосконалювати свої алгоритми та стратегії відповіді на основі навчання з інцидентів.

Ці етапи дозволяють системі реагування ефективно впоратися з потенційними загрозами та забезпечити швидке та адекватне реагування для збереження безпеки та продуктивності системи.

Реалізація системи реагування у хмарному середовищі AWS може використовувати низку сервісів для ефективного виявлення та реагування на потенційні загрози та порушення безпеки [23]. Нижче розглянуті ключові сервіси, які можна використовувати для цієї мети.

Amazon CloudWatch Events дозволяє налаштовувати правила для автоматичного виявлення подій в хмарі AWS. Це включає події від інших сервісів, таких як EC2, S3, Lambda та інші. Сервіс слідкує за різними подіями, такими як створення або видалення ресурсів, завершення запусків екземплярів або будь-які інші зміни у середовищі AWS. На основі налаштованих правил CloudWatch Events може запускати вказані дії, такі як виклик Lambda-функцій або надсилання повідомлень на різні канали сповіщень.

Amazon Simple Notification Service (SNS) надає масштабовану та гнучку інфраструктуру для надсилання повідомлень на різноманітні канали, такі як електронна пошта, текстові повідомлення або HTTP-запити. SNS може використовуватися як канал сповіщення для різних сценаріїв, наприклад, для інформування адміністраторів про виявлені загрози безпеки або інші події в

системі. Дозволяє відправляти сповіщення в реальному часі на різні канали залежно від події, яку слід взяти під контроль.

AWS Lambda дозволяє запускати код без необхідності управління серверами. Це може бути використано для автоматизації виконання конкретних дій у відповідь на виявлені події. Lambda може бути викликана з CloudWatch Events або інших сервісів для обробки та реагування на події в реальному часі. Забезпечує автоматичне виконання коду або скриптів для виконання конкретних завдань, таких як блокування доступу або відновлення застосунку до попереднього стану.

Ці сервіси взаємодіють між собою, створюючи механізм автоматизованого реагування на події в хмарному середовищі, забезпечуючи швидке та ефективне управління безпекою системи.

Збирач даних, аналізатор даних та система реагування є основними компонентами підсистеми моніторингу застосунків. Вони працюють разом для забезпечення безпеки застосунків у хмарі.

У цьому підрозділі докладно розглянуто перший компонент підсистеми моніторингу застосунків - збирач даних. Розглянуто різні джерела даних, які може використовувати збирач даних, а також сервіси AWS, які можна використовувати для реалізації збирача даних.

2.2 Функції підсистеми моніторингу застосунків

Вибір хмарного середовища AWS для реалізації підсистеми моніторингу застосунків є стратегічно обґрунтованим, оскільки це надає гнучкість, масштабованість та широкий набір сервісів для забезпечення ефективного моніторингу та управління застосунками. У цьому контексті підсистема моніторингу виконує ряд ключових функцій [24].

Збір та агрегація даних

Цей етап підсистеми моніторингу застосунків у хмарному середовищі AWS спрямований на ефективний та повний збір різноманітних даних, які визначають стан та роботу застосунків. Важливі аспекти цього етапу. Постійний

збір даних – це невпинний та систематичний процес з отримання інформації про різноманітні аспекти роботи застосунків у реальному часі. Основною метою цього процесу є створення вичерпного та актуального образу стану застосунків у хмарному середовищі AWS. Збір даних відбувається безперервно, а це означає постійний моніторинг та реєстрацію різноманітних подій та параметрів, що відображають роботу застосунків.

Цей процес включає в себе широкий спектр інформації, яка охоплює різні аспекти роботи застосунків. Серед найважливіших елементів інформації, що збирається, є:

1. **Інформація про виконання коду:** Збираються дані щодо виконання програмного коду застосунків. Це може включати в себе інформацію про час виконання, результати виконання функцій, та інші аспекти виконання коду.
2. **Зміни в конфігурації:** Фіксуються зміни в конфігурації застосунків, такі як параметри, налаштування та інші конфігураційні зміни, які можуть впливати на його роботу.
3. **Користувацькі дії:** Реєструються дії користувачів у межах застосунку, такі як вхід, вихід, взаємодія з функціоналом та інші користувацькі події.
4. **Інші фактори, що впливають на роботу застосунку:** Збираються дані про різноманітні фактори, які можуть впливати на роботу застосунку, такі як завантаження ресурсів, мережевий трафік, та інші параметри.

Постійний та систематичний характер збору даних дозволяє підсистемі моніторингу застосунків завжди мати актуальну інформацію про їхню діяльність та стан, що, в свою чергу, сприяє ефективному аналізу та реагуванню на події та виклики, що можуть виникнути у хмарному середовищі AWS.

Джерела даних

Для повного збору інформації про стан застосунків використовуються різноманітні джерела, такі як журнали (logs), метрики та дані подій. Журнали містять текстову інформацію про події, метрики надають кількісний аналіз роботи застосунків, а дані подій фіксують конкретні дії в системі [25].

CloudWatch Logs:

Сервіс CloudWatch Logs відповідає за централізований збір та зберігання журналів з різних джерел, таких як EC2, ECS, Lambda та CloudTrail. Це створює єдиний інтерфейс для моніторингу та аналізу текстової інформації, що міститься в журналах застосунків.

CloudWatch Metrics:

Сервіс CloudWatch Metrics використовується для збору та зберігання метрик з різних джерел, таких як EC2, ECS, Lambda та CloudTrail. Метрики надають кількісний погляд на різні аспекти роботи застосунків, такі як завантаження CPU, використання пам'яті та інші ключові параметри.

CloudTrail:

Сервіс CloudTrail фіксує дані про події, які відбуваються в хмарі AWS. З його допомогою можна отримати детальну історію змін в конфігурації, використання ресурсів та інші події, пов'язані з безпекою та аудитом застосунків. Використання цих джерел дозволяє побудувати повний образ роботи застосунків та забезпечити їхній ефективний моніторинг та управління в хмарному середовищі AWS.

Агрегація даних

Після неперервного збору інформації з різноманітних джерел, дані піддаються процесу агрегації для створення централізованої бази даних [26]. Цей етап є критичним для подальшого ефективного аналізу та управління застосунками в хмарному середовищі AWS.

Збір та Об'єднання: Зібрані дані з сервісів CloudWatch Logs, CloudWatch Metrics та CloudTrail об'єднуються в єдиний пул інформації. Цей пул включає текстову інформацію з журналів, кількісні показники з метрик і дані про події з CloudTrail. Цей процес створює централізовану збірку даних, яка служить основою для подальшого аналізу та виявлення взаємозв'язків.

Структурування: Після збору дані структуруються для створення логічних зв'язків та категорій. Цей етап включає групування подій за типами, визначення відомостей про використання ресурсів та визначення ключових

параметрів продуктивності. Це полегшує подальший аналіз та виявлення патернів у великому обсязі даних.

Поповнення Бази Даних: Агреговані дані, після структурування, додаються до централізованої бази даних. Цей процес дозволяє створити одноманітний доступ та управління інформацією. Централізована база даних слугує як основа для подальшого моніторингу, аналізу та реагування на стан застосунків в хмарному середовищі AWS.

Переваги агрегації даних:

Централізація: Процедура агрегації забезпечує централізований доступ до інформації з різних джерел. Це спрощує управління та моніторинг застосунків, оскільки зібрані дані стають доступними в єдиному місці.

Аналіз: Агреговані дані створюють сприятливі умови для подальшого аналізу. Вони дозволяють виявляти взаємозв'язки, патерни та аномалії в роботі застосунків. Аналітичний процес може виявити ключові фактори, що впливають на продуктивність та безпеку застосунків.

Ефективність: Агрегація даних є ключовим етапом для ефективного виявлення проблем, трасування подій та прийняття рішень в реальному часі. Це дозволяє оперативно реагувати на події, що відбуваються в хмарному середовищі, забезпечуючи безпеку та надійність застосунків.

Агрегація даних є необхідною складовою для створення повного та вичерпного образу стану застосунків, що дозволяє ефективно вирішувати завдання моніторингу та безпеки в хмарному середовищі AWS.

Моніторинг доступності та продуктивності

Підсистема моніторингу доступності та продуктивності забезпечує постійний нагляд за роботою застосунків, оцінюючи їхню доступність і продуктивність. Для досягнення цього використовуються різноманітні метрики, які вимірюють різні аспекти роботи системи [27].

Один з ключових аспектів моніторингу - це час відгуку застосунків. Відслідковуючи час, який система витрачає на обробку запитів, можна

визначити, наскільки ефективно працює застосунок і як швидко відповідає на користувацькі запити.

Додатково, використовуючи метрики, доступні в CloudWatch Metrics, можна аналізувати використання ресурсів застосунків. Наприклад, метрики, такі як завантаження CPU і обсяг вільної пам'яті, надають інформацію про те, наскільки завантажена система і чи є достатньо ресурсів для ефективної роботи. Це дозволяє вчасно виявляти можливі проблеми з продуктивністю та реагувати на них. Такий моніторинг сприяє підтримці високої продуктивності та доступності застосунків у хмарному середовищі AWS.

Виявлення та аналіз аномалій:

Підсистема виявлення та аналізу аномалій відіграє ключову роль у забезпеченні безпеки та ефективності застосунків. Цей процес базується на використанні аналізатора даних, який визначає непередбачувані відхилення від звичайних патернів та ідентифікує потенційні загрози для безпеки системи [28].

Одним із інструментів, що може бути використаний для виявлення аномалій, є сервіс Amazon SageMaker. Використовуючи алгоритми машинного навчання, SageMaker дозволяє створювати моделі, спроектовані для виявлення ненормальних патернів та аномалій у великому обсязі зібраних даних. Це може включати незвичайні взаємозв'язки між подіями, несподівані зміни в конфігурації або виняткові використання ресурсів.

Процес аналізу аномалій дозволяє оперативно виявляти потенційно шкідливі дії або інциденти, що можуть виникнути в системі. Це створює можливість для реагування на загрози та впровадження заходів безпеки ще до того, як вони стануть критичними. Такий підхід допомагає забезпечити високий рівень безпеки застосунків у хмарному середовищі AWS.

Сповіщення та автоматична реакція:

Підсистема сповіщень та автоматичної реакції в системі моніторингу застосунків грає важливу роль у забезпеченні оперативного реагування на виявлені проблеми чи потенційні загрози. Механізми цієї підсистеми дозволяють

операторам чи адміністраторам отримувати сповіщення про критичні ситуації, що виникають у хмарному середовищі AWS [29].

Для цього можуть використовуватися різні сервіси, але одним із ключових є Amazon Simple Notification Service (SNS). SNS надає гнучкий інструментарій для надсилання сповіщень на різні канали, такі як електронна пошта, текстові повідомлення, HTTP-запити тощо. Це дозволяє оперативно і ефективно інформувати відповідальних осіб про події, що вимагають уваги.

За допомогою AWS Lambda можливо використовувати автоматичні реакції на конкретні події чи умови, що виникають в системі. Наприклад, можна налаштовувати автоматичне відновлення застосунків до попереднього стану або блокування доступу в разі виявлення потенційно небезпечних дій.

Забезпечення аудиту та звітності:

Підсистема аудиту та звітності в системі моніторингу застосунків відіграє роль в забезпеченні повної спостереженості за діями та подіями в хмарному середовищі AWS. Детальний аудит дозволяє збирати історію змін, використання ресурсів та інші ключові події для подальшого аналізу та формування звітності.

Один з таких сервісів, що забезпечує аудит подій, - це AWS CloudTrail. CloudTrail фіксує всі події, що відбуваються в середовищі AWS, створюючи журнал для аналізу та забезпечення відповідності. Це дозволяє проводити детальний аудит та виявлення невідомих або підозрілих дій, що сприяє безпеці та відповідності в управлінні застосунками в хмарному середовищі [30].

Забезпечення аудиту та звітності:

Забезпечення аудиту та звітності в системі моніторингу застосунків представляє собою ключовий аспект для забезпечення безпеки, виявлення аномалій та відповідності в хмарному середовищі AWS.

Підсистема аудиту розроблена для збирання та реєстрації детальної інформації про всі події та зміни, що відбуваються в середовищі AWS. Це охоплює широкий спектр дій, від створення чи видалення ресурсів до змін конфігурацій та використання ресурсів. Такий широкий спектр аудитованих

подій дозволяє забезпечити повний моніторинг за взаємодією користувачів та системних компонентів.

Одним із сервісів, що забезпечує цей аспект, є AWS CloudTrail. Цей сервіс реєструє події на рівні облікового запису та рівні ресурсу, надаючи повний звіт про всі взаємодії з ресурсами та обліковим записом. CloudTrail генерує журнали подій, які можна використовувати для аналізу та виявлення нестандартних чи підозрілих дій, а також для дотримання вимог відповідності та аудиту.

Забезпечення аудиту та звітності дозволяє не лише реагувати на поточні події, але й створювати детальні звіти для подальшого аналізу та вивчення патернів у поведінці системи. Це стає важливою складовою для управління безпекою та відповідності в хмарному середовищі.

Ці функції взаємодіють для створення ефективної та надійної підсистеми моніторингу застосунків у хмарному середовищі AWS, забезпечуючи високий рівень безпеки та продуктивності системи.

2.3 Принципи роботи підсистеми моніторингу застосунків

. Принципи роботи підсистеми моніторингу застосунків в хмарному середовищі AWS визначаються рядом ключових аспектів, спрямованих на забезпечення ефективності, надійності та безпеки моніторингового процесу.

Використання інструментів машинного навчання:

Використання інструментів машинного навчання, таких як Amazon SageMaker, в підсистемі моніторингу застосунків в хмарному середовищі AWS є ключовим для ефективного виявлення аномалій та потенційних загроз безпеки. Amazon SageMaker володіє рядом алгоритмів, які дозволяють аналізувати динамічні патерни та виявляти несподівані зміни у поведінці застосунків [31].

Один із основних аспектів використання інструментів машинного навчання полягає у їх здатності адаптуватися до нових сценаріїв та виявлення ризикових аномалій, які можуть бути складні для виявлення за допомогою традиційних методів моніторингу. Це особливо важливо в хмарних середовищах, де застосунки можуть бути піддані постійним змінам та оновленням.

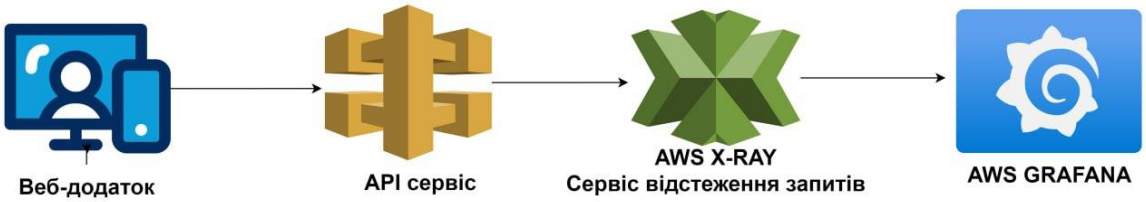


Рис 2.1 Модуль моніторингу API.

Інтелектуальні аналітичні методи, використовувані в Amazon SageMaker, дозволяють підсистемі автоматично визначати аномалії та ризикові події. Це включає в себе не лише виявлення незвичайних активностей, але й аналіз контексту та взаємозв'язків між різними подіями. Такий підхід дозволяє зменшити кількість помилкових спрацьовувань та надає більш точні результати при виявленні потенційних загроз.

Загалом, інтеграція інструментів машинного навчання в підсистему моніторингу застосунків створює ефективний механізм виявлення та реагування на загрози безпеки в хмарних обчислювальних середовищах.

Сповіщення та автоматична реакція:

Сповіщення та автоматична реакція в підсистемі моніторингу застосунків в хмарному середовищі AWS представляють собою важливий етап в забезпеченні безпеки та ефективного управління виявленими проблемами.

У випадку виявлення проблем чи загроз, підсистема надає механізми для сповіщення операторів або адміністраторів. Використання сервісів, таких як Amazon SNS та AWS Lambda, дозволяє забезпечити негайне інформування ключових учасників. Amazon Simple Notification Service (SNS) використовується для надсилання сповіщень на різні канали, такі як електронна пошта, текстові повідомлення чи HTTP-запити. Це надає гнучкість у виборі комунікаційних каналів залежно від потреб команди безпеки та управління.

Крім того, AWS Lambda використовується для автоматичних реакцій на виявлені проблеми. Це означає, що можна налаштовувати конкретні дії, які автоматично виконуються у відповідь на певні події. Наприклад, у разі виявлення аномалій у використанні ресурсів чи інших критичних сценаріїв, Lambda може запускати автоматизовані процедури, такі як блокування доступу

до системи, відновлення до попереднього стану або інші конфігуровані заходи безпеки [32].

Загалом, сповіщення та автоматична реакція грають ключову роль у швидкому виявленні, інформуванні та вирішенні проблем у хмарному середовищі, забезпечуючи надійний та ефективний механізм управління безпекою застосунків.

Забезпечення аудиту та звітності:

Забезпечення аудиту та звітності в підсистемі моніторингу застосунків у хмарному середовищі AWS є ключовим компонентом для забезпечення відповідності, аналізу та детального вивчення історії змін.

Сервіс CloudTrail використовується для реєстрації всіх подій та змін, які відбуваються в середовищі AWS. Це включає в себе дії, такі як створення чи видалення ресурсів, входи в систему, зміни конфігурації та інші активності користувачів. Детальний аудит зафіксованих подій надає можливість отримати повний звіт про використання ресурсів та дії користувачів у хмарному середовищі.

Цей аудит є важливим для забезпечення відповідності з регуляторними вимогами, де необхідно зберігати та представляти вичерпну інформацію про всі зміни та події. Більше того, детальний аналіз записів CloudTrail дозволяє вдосконалювати стратегії моніторингу та безпеки, виявляти аномалії, ідентифікувати потенційні загрози та реагувати на їх виявлення.

Загалом, забезпечення аудиту та звітності через CloudTrail є ефективним засобом для контролю за подіями та змінами у хмарному середовищі, покращення безпеки та забезпечення відповідності з вимогами стандартів та регуляцій.

Масштабованість та гнучкість:

Масштабованість та гнучкість є ключовими аспектами підсистеми моніторингу застосунків у хмарному середовищі AWS, забезпечуючи ефективну роботу в умовах зростання обсягу даних та розширюваності інфраструктури. Основний вибір хмарної інфраструктури AWS відображається в здатності

системи автоматично масштабувати ресурси, що забезпечує необхідну гнучкість та оптимальну продуктивність [33].

Масштабованість в системі означає, що вона може ефективно обробляти збільшення обсягу даних, витрат чи завдань без втрати продуктивності. Завдяки хмарному середовищу AWS, система може автоматично адаптувати кількість ресурсів, необхідних для обробки завдань, що гарантує оптимальний рівень продуктивності.

Гнучкість системи полягає в її здатності легко адаптуватися до різноманітних умов та вимог. Підсистема моніторингу застосунків може змінювати свою конфігурацію, реагуючи на зміни в обсязі даних, вимогах до продуктивності чи інших факторах. Це важливо для підтримки динамічних середовищ, де умови можуть швидко змінюватися.

Загалом, масштабованість та гнучкість в підсистемі моніторингу застосунків на платформі AWS дозволяють ефективно впоратися з ростом обсягу даних, забезпечують оптимальну продуктивність та надають можливість легко адаптуватися до змінних умов експлуатації.

Відслідковування та аналіз трендів:

Відслідковування та аналіз трендів в роботі застосунків є важливим елементом підсистеми моніторингу, призначеним для отримання інсайтів у довгострокові зміни та покращення ефективності. Ця функціональність забезпечує систему інструментами для обробки історичних даних та метрик, які визначають розвиток застосунків у тривалі періоди.

Використання історичних даних дозволяє підсистемі виявляти тренди, що описують довгострокові патерни та зміни в роботі застосунків. Аналіз цих трендів може включати в себе виявлення сталої зміни в завантаженні ресурсів, ефективності алгоритмів або інших параметрах, які впливають на продуктивність системи.

Прогнозування можливих ризиків та покращень у майбутньому є ще однією важливою функцією. Використовуючи отримані дані та аналіз трендів, система може розробляти прогнози щодо можливих проблем або можливостей

для вдосконалення. Це дозволяє операторам та адміністраторам приймати обґрунтовані рішення та планувати стратегії для оптимізації роботи застосунків у майбутньому.

Отже, функціонал відслідковування та аналізу трендів в підсистемі моніторингу застосунків на платформі AWS грає ключову роль у забезпеченні системи інформацією про довгострокові динаміки та сприяє ухваленню обґрунтованих рішень для оптимізації роботи застосунків.

Інтеграція з іншими хмарними сервісами:

Інтеграція з іншими хмарними сервісами в підсистемі моніторингу застосунків на платформі AWS є важливим аспектом, спрямованим на максимальне використання можливостей хмарного середовища. Ця інтеграція дозволяє підсистемі співпрацювати з різними сервісами та інструментами для оптимізації функціоналу та покращення загальної ефективності [34].

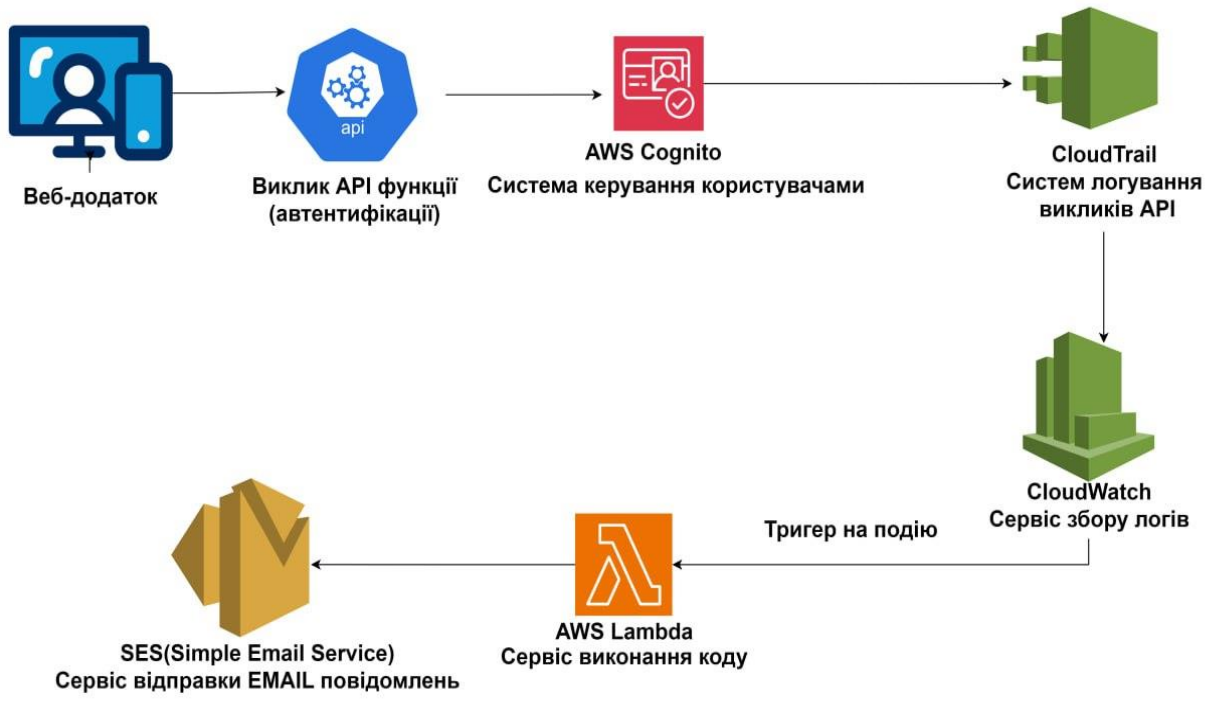


Рис 2.1 Модуль моніторингу системи автентифікації.

Переваги інтеграції виявляються в здатності комбінувати функціонал моніторингу з іншими хмарними послугами, такими як обчислювальні ресурси (EC2), контейнери (ECS), сервіси без серверних обчислень (Lambda) та інші. Це створює можливості для розширення функціональності підсистеми та забезпечення комплексних рішень для управління та безпеки.

Наприклад, інтеграція з Amazon CloudWatch дозволяє використовувати метрики та журнали для більш точного моніторингу роботи застосунків. Сервіс Amazon Simple Notification Service (SNS) може бути використаний для сповіщення про виявлені проблеми, а AWS Lambda - для автоматизованої реакції на події.

Така інтеграція забезпечує гнучкість та розширюваність підсистеми, оскільки вона може адаптуватися до різних потреб та умов експлуатації застосунків у хмарному середовищі AWS. Такий підхід до інтеграції дозволяє максимально використовувати можливості хмарних сервісів для забезпечення надійного та ефективного моніторингу та управління застосунками.

Захист конфіденційності та відновлення після інцидентів:

Захист конфіденційності та відновлення після інцидентів є критичними аспектами підсистеми моніторингу застосунків в хмарному середовищі AWS. Щоб забезпечити безпеку конфіденційних даних, підсистема використовує комплексний підхід, включаючи механізми шифрування та управління доступом.

1. **Шифрування даних:**
    - Використання механізмів шифрування для захисту конфіденційної інформації в CloudWatch Logs та інших хмарних сервісах.
    - Застосування ключів шифрування та інфраструктури керування ключами (KMS) AWS для ефективного управління процесом шифрування та розшифрування.
2. **Управління доступом:**
    - Впровадження механізмів управління доступом для забезпечення тільки авторизованого доступу до конфіденційних даних.
    - Використання ідентифікації та автентифікації заснованої на ролях для точного контролю над доступом до ресурсів та функціоналу.
3. **Відновлення після інцидентів:**

- Регулярне резервування та зберігання даних в безпечних місцях для забезпечення можливості швидкого відновлення роботи застосунків після виявлення та розв'язання проблем.
- Використання інструментів моніторингу та журналювання для аналізу інцидентів, їх причин та запобігання повторенню у майбутньому.

Таким чином, принципи роботи підсистеми моніторингу застосунків в хмарному середовищі AWS охоплюють комплексний підхід до забезпечення ефективності, безпеки та надійності застосунків у розподіленому та динамічному середовищі хмарних обчислень.

3 РОЗРОБКА ПІДСИСТЕМИ МОНІТОРИНГУ ДОДАТКІВ В ХМАРНОМУ СЕРЕДОВИЩІ

3.1 Компоненти підсистеми моніторингу застосунків

У процесі визначення технічних аспектів розробки підсистеми моніторингу в хмарному середовищі було прийнято рішення використовувати дві мови програмування: Python і JavaScript. Це обрання виникло з урахуванням різноманітних факторів, що визначають вдалу інтеграцію та оптимальне виконання функціонала підсистеми.

Обрання мови програмування для розробки системи — це важливе рішення, яке має значний вплив на її успіх. Існує безліч мов програмування, кожна з яких має свої переваги та недоліки. При виборі мови програмування для конкретної системи необхідно враховувати такі фактори, як:

Сфера застосування системи. Для різних сфер застосування підходять різні мови програмування. Наприклад, для розробки вебдодатків часто використовуються такі мови, як JavaScript, PHP або Python. Для розробки мобільних додатків — Java, Kotlin або Swift. Для розробки десктопних додатків — C++, C# або Java.

Характеристики системи. Якщо система має високі вимоги до продуктивності, то необхідно вибрати мову програмування, яка забезпечує високу швидкість виконання коду. Якщо система має складну структуру, то необхідно вибрати мову програмування, яка підтримує об'єктноорієнтоване програмування.

Доступність ресурсів. Необхідно враховувати, чи є у команди розробників досвід роботи з обраною мовою програмування. Якщо досвіду немає, то необхідно буде витратити час на навчання.

Python — це універсальна мова програмування, яка використовується в різних сферах, включаючи веброзробку, науку про дані, машинне навчання та штучний інтелект. Python має ряд переваг, які роблять її ідеальною для розробки нашої підсистеми:

- Python є універсальною мовою програмування, яка підтримує широкий спектр завдань, включаючи моніторинг.
- Python має численні бібліотеки та фреймворки, які спрощують розробку програм для моніторингу.
- Python є мультиплатформною мовою, що дозволяє використовувати її для моніторингу ресурсів AWS на різних платформах.
- Велика спільнота розробників**.** Python має велику та активну спільноту розробників, яка постійно створює нові бібліотеки, фреймворки та інструменти. Це забезпечує підтримку та допомогу в разі виникнення проблем.
- Безкоштовне поширення. Python є безкоштовною мовою.

Використання мови програмування Python надає можливість розробки високоефективної підсистеми моніторингу, спеціально адаптованої до умов та вимог хмарного середовища AWS. Python володіє низкою характеристик та інструментів, що роблять його ідеальним вибором для створення таких систем.

Для реалізації підсистеми моніторингу в хмарних застосунках було вибрано ще одну мову програмування JavaScript.

JavaScript є високорівневою та інтерпретованою мовою з нестрогою динамічною типізацією. Це надає розробникам можливість зосереджуватися на високорівневих концепціях та завданнях програмування, полегшуючи розуміння та розробку коду. Вона широко використовується як на клієнтському (у веб-браузерах), так і на серверному рівнях за допомогою платформи Node.js. Це робить мову універсальною та дозволяє єднати розробку клієнтських та серверних частин додатка.

JavaScript також використовується для розробки хмарних додатків, забезпечуючи клієнт-серверну однорідність. Це означає, що JavaScript може бути використаний як для клієнтської, так і для серверної частини додатка, полегшуючи взаємодію між ними. Завдяки платформі Node.js, JavaScript може виконувати серверні функції, що полегшує інтеграцію з іншими серверними технологіями, такими як AWS Lambda або AWS EC2.

Щодо інтеграції з AWS, JavaScript використовує AWS SDK, що дозволяє легко взаємодіяти з різними послугами AWS. Також він підтримує написання функцій AWS Lambda, що забезпечує гнучкість та масштабованість в моніторингу та управлінні подіями. JavaScript є ключовим інструментом для взаємодії з веб-інтерфейсом, особливо в контексті моніторингу, де він може взаємодіяти з DOM та створювати динамічні елементи. Багата екосистема JavaScript, що включає різноманітні бібліотеки та фреймворки, такі як React, Angular, чи Vue.js, полегшує розробку та підтримку коду підсистеми моніторингу.

За допомогою інструментів Python та JavaScript, а також використання переваг хмарної інфраструктури AWS, формується надійний та високотехнологічний веб-додаток. Це комплексне рішення відкриває перед нами широкий спектр можливостей, які можуть змінити парадигму розробки веб-додатків. Сполучення сил Python та JavaScript, а також використання хмарних ресурсів AWS, надає системі не лише потужність, але і гнучкість, необхідну для відповіді на вимоги та очікування сучасного веб-середовища.

Використання готових бібліотек у розробці проєктів має велике значення та пропонує ряд переваг:

1. **Економія часу і ресурсів:**
    - Готові бібліотеки забезпечують готові до використання рішення для типових завдань, що вимагають менше часу та зусиль для їх інтеграції. Розробники можуть скористатися функціональністю, яка вже пройшла випробування та оптимізацію.
2. **Висока якість та стандартизація:**
    - Багато готових бібліотек створені професіоналами та підтримуються спільнотою розробників, що гарантує високу якість коду та відповідність стандартам. Це сприяє створенню надійного та стабільного програмного продукту.
3. **Зменшення ймовірності помилок:**

- Використання готових бібліотек дозволяє уникнути повторного винахідництва та зменшити ймовірність виникнення помилок. Розробники можуть користуватися вже перевіреними та відлагодженими рішеннями.

4. **Розширені можливості:**
    - Багато бібліотек надають розширені можливості та функції, які дозволяють розробникам працювати на вищому рівні абстракції, концентруючись на основних завданнях проєкту.
5. **Активна підтримка та оновлення:**
    - Готові бібліотеки часто підтримуються та оновлюються спільнотою розробників, що гарантує актуальність та безпеку використання.
6. **Спільнота розробників:**
    - Використання популярних бібліотек означає входження в широку спільноту розробників, яка може надати підтримку, поради та рішення для виникаючих завдань.

Як висновок, використання готових бібліотек робить процес розробки значно простішим, підвищує якість фінальних продуктів і дозволяє розробникам фокусуватися на унікальних аспектах своїх проектів, замість витрачання часу на створення базових компонентів.

Під час створення функції Lambda, важливого елемента хмарної інфраструктури, були використані наступні бібліотеки та модулі Python, що сприяли розширенню функціональності та надали доступ до різноманітних інструментів для більш ефективної обробки даних:

NumPy (Numerical Python) є однією з ключових бібліотек для наукових обчислень та роботи з числовими даними в середовищі програмування Python. Вона надає потужні структури даних, основними з яких є масиви, та високопродуктивні функції для математичних операцій. Можливості бібліотеки:

1. **Масиви та Матриці:** NumPy надає об'єкт **numpy.array**, який є ефективною структурою даних для зберігання та маніпулювання масивами та матрицями. Це особливо корисно для векторизованих операцій.

2. **Операції Лінійної Алгебри:** Бібліотека дозволяє виконувати різні операції лінійної алгебри, такі як добуток матриць, знаходження власних значень та векторів, розв'язання систем лінійних рівнянь тощо.
3. **Функції Математичного Обчислення:** NumPy має велику кількість вбудованих функцій для роботи з числами, включаючи тригонометричні, логарифмічні, степеневі та інші функції.
4. **Операції Трансформації та Статистики:** Забезпечує зручні функції для обробки та трансформації даних, такі як згладжування, фільтрація, обчислення середнього, дисперсії та інші.
5. **Ефективність та Оптимізація:** NumPy використовує мову C для реалізації багатьох своїх операцій, що забезпечує високу швидкодію та оптимізацію в порівнянні з чистим Python.

Загалом, NumPy став необхідним інструментом для роботи з числовими даними та наукових обчислень в Python, забезпечуючи високу продуктивність та гнучкість в операціях з великими обсягами даних.

Boto3 є офіційною бібліотекою для взаємодії з Amazon Web Services (AWS) на мові програмування Python. Ця бібліотека розроблена AWS і надає програмістам зручний інтерфейс для використання широкого спектру AWS-сервісів. Вона включає у себе інструменти для роботи з обчислювальними, зберігальними, базами даних, мережевими та іншими сервісами, що пропонуються AWS. Boto3 виявляється особливо корисним для розробників, які створюють функції AWS Lambda на мові програмування Python. Основні можливості використання Boto3 в Lambda включають:

1. **Створення та Конфігурація Функцій Lambda:**
    - З Boto3 можна створювати нові функції Lambda та конфігурувати їх параметри, такі як розмір пам'яті, тригери та інші налаштування.
2. **Виклик Функцій Lambda:**
    - Boto3 надає можливість викликати функції Lambda за допомогою Python-коду. Це може бути корисно в інтегрованих сценаріях, де

функції Lambda викликаються іншими AWS-сервісами чи за подіями.

3. **Взаємодія з Іншими AWS-Сервісами:**
    - Boto3 дозволяє вам взаємодіяти з різними AWS-сервісами зсередини функцій Lambda. Наприклад, ви можете використовувати його для завантаження файлів в Amazon S3, отримання даних з DynamoDB, або роботи з іншими обчислювальними ресурсами.
4. **Управління IAM-роллю:**
    - Boto3 дозволяє налаштовувати та управляти ролями Identity and Access Management (IAM), які пов'язані з функціями Lambda, забезпечуючи правильні дозволи для взаємодії з іншими AWS-ресурсами.
5. **Логування та Відлагодження:**
    - Boto3 допомагає виводити логи та відлагоджувати ваші Lambda-функції, щоб спростити виявлення помилок та аналіз їхнього виконання.

Загалом, Boto3 робить взаємодію з AWS-сервісами, зокрема з функціями Lambda, зручною та ефективною завдяки його дружелюбному API та гнучкій функціональності.

Requests є простою та елегантною бібліотекою для виконання HTTP-запитів в мові програмування Python. Ця бібліотека надає зручний інтерфейс для роботи з HTTP-протоколом, дозволяючи здійснювати HTTP-запити, обробляти відповіді та передавати дані між клієнтом і сервером. Requests володіє простим синтаксисом, що полегшує використання та розуміння для розробників.

Використання в Lambda: Requests може бути корисною бібліотекою в контексті функцій AWS Lambda, особливо коли виникає потреба взаємодії зі зовнішніми веб-сервісами або API, які не є специфічними для AWS. Деякі можливості використання Requests в Lambda включають:

1. **Запити до Зовнішніх Ресурсів:**

- Requests дозволяє здійснювати HTTP-запити до зовнішніх веб-ресурсів. Це може бути корисно, наприклад, при інтеграції з іншими веб-сервісами чи отриманні даних з зовнішніх джерел.

2. **Взаємодія з Зовнішніми API:**
   - Якщо потрібно взаємодіяти з API, що не є частиною AWS, Requests надає можливість легко виконувати HTTP-запити до цих API, передаючи необхідні параметри та отримуючи відповіді.
3. **Отримання Даних з Інтернет-ресурсів:**
   - Requests дозволяє здійснювати HTTP GET-запити для отримання даних з інтернет-ресурсів. Це може бути використано, наприклад, для отримання інформації для подальшого оброблення в Lambda-функціях.
4. **Відправлення Даних на Зовнішні Ресурси:**
   - Requests також підтримує відправлення HTTP POST-запитів, що може бути використано для взаємодії з зовнішніми сервісами та відправлення даних на сервери.
5. **Оброблення Відповідей:**
   - Зручний інтерфейс Requests дозволяє легко обробляти відповіді, отримані від серверів, розпаковувати JSON та інше.

Requests стає універсальним інструментом для роботи з HTTP в контексті Lambda-функцій, допомагаючи розробникам легко взаємодіяти з різноманітними веб-ресурсами.

Zappa є фреймворком для Python, який спрощує розгортання та управління Python-додатками на Amazon Web Services (AWS) Lambda та API Gateway. Розроблений з урахуванням особливостей серверних функцій (serverless), Zappa дозволяє розробникам ефективно використовувати веб-додатки, сервіси та мікросервіси, перетворюючи їх у функції AWS Lambda. Zappa розширює можливості використання Python у середовищі AWS Lambda, забезпечуючи зручний спосіб розгортання та масштабування додатків. Деякі ключові аспекти використання Zappa в AWS Lambda включають:

1. **Просте Розгортання:**
    - Zappa дозволяє легко розгортати Python-додатки на AWS Lambda. Розробникам потрібно всього лише викликати команду для розгортання, і фреймворк автоматично створить та налаштує всі необхідні ресурси.
2. **Інтеграція з API Gateway:**
    - Автоматично налаштовує інтеграцію з API Gateway, що дозволяє звертатися до функцій Lambda через HTTP-запити. Забезпечує легкий доступ до ваших додатків через веб-інтерфейс.
3. **Автоматичне Управління Залежностями:**
    - Zappa автоматично керує залежностями вашого проекту, упаковуючи їх разом з функцією Lambda. Це дозволяє уникнути проблем із середовищем та забезпечити повну ізоляцію.
4. **Масштабування:**
    - Забезпечує простий механізм масштабування для функцій Lambda, що дозволяє автоматично розгортати додаток на багато серверів для обробки збільшеного навантаження.
5. **Підтримка для Застосунків Flask та Django:**
    - Zappa підтримує популярні фреймворки Python, такі як Flask та Django, що робить його відмінним вибором для розгортання веб-додатків.

Zappa дозволяє розробникам легко переходити в середовище serverless, спрощуючи і ускладнюючи розгортання та управління Python-додатками на AWS Lambda та API Gateway.

В процесі розробки програмного засобу було використано ряд модулів та бібліотек, що дозволило створити програмний продукт з різноманітним функціоналом та широким спектром можливостей. Обрані інструменти відіграють ключову роль у реалізації проєкту, забезпечуючи ефективність, надійність та гнучкість.

Важливість вибору середовища розробки виявляється у визначенні якості кінцевого продукту, швидкості розробки та комфорту для програміста. Це особливо актуально для мови програмування Python, яка знаходить застосування у великому спектрі областей, від веб-розробки до наукових досліджень.

При виборі середовища розробки слід враховувати різні фактори. Для новачків може бути важливим інтуїтивний інтерфейс та наявність підказок для швидкого навчання. Досвідчені розробники, навпаки, можуть шукати розширені можливості налаштування та оптимізації для максимально ефективної роботи.

Сумісність середовища розробки з іншими інструментами, такими як системи управління версіями, тести коду, бази даних та хмарні сервіси, також є ключовою. Вибір правильного середовища може значно покращити продуктивність та якість проектів, забезпечуючи комфорт у процесі програмування.

Виділимо найпопулярніші середовища розробки для Python. Вони відзначаються не лише своєю популярністю, але і високою функціональністю та зручністю використання.

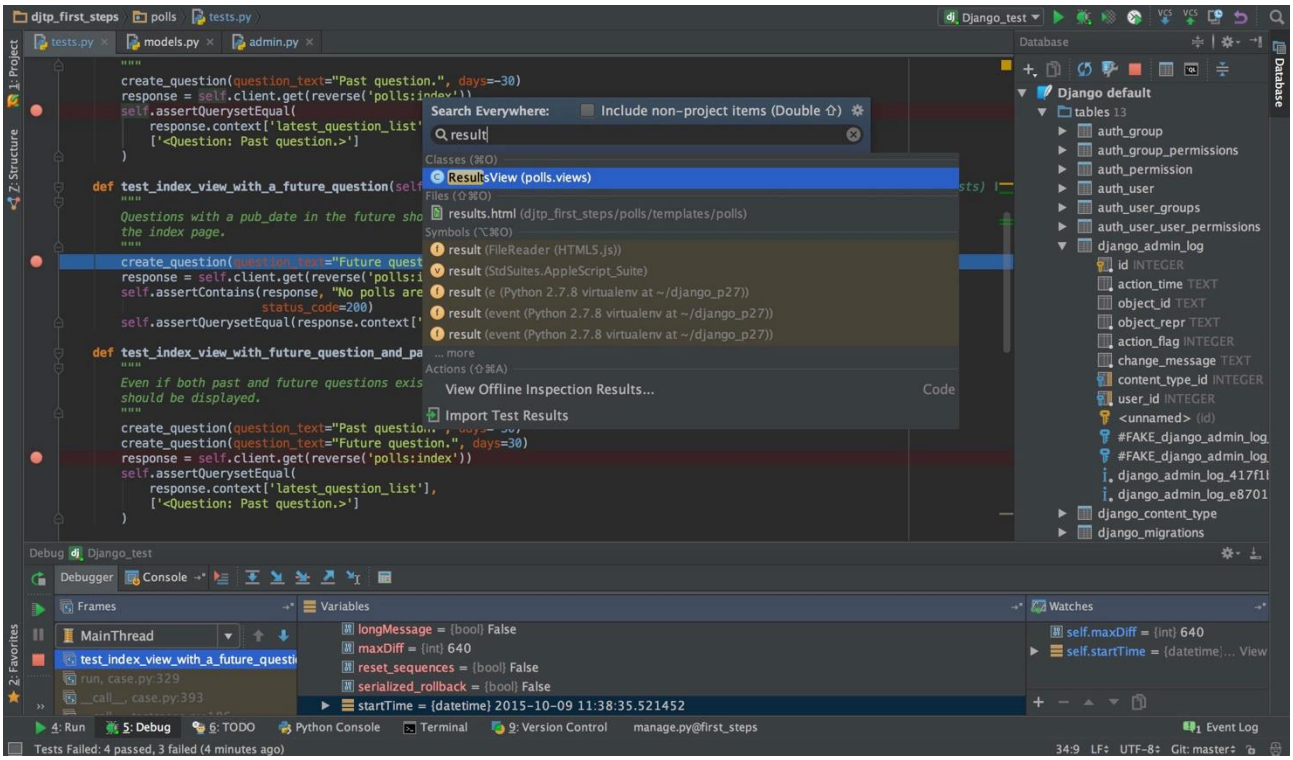


Рисунок 3.1 – Середовище PyCharm

1) PyCharm, розроблене відомою компанією JetBrains, є одним із найпопулярніших та найфункціональніших середовищ розробки для мови програмування Python. Завдяки свому потужному наборові функцій, PyCharm визначається як ідеальний інструмент для розробників, які працюють з Python.

Основні характеристики PyCharm включають автоматичне завершення коду, що значно прискорює написання коду та зменшує кількість помилок. Вбудована система відладки спрощує виявлення та усунення помилок у вашому програмному коді. Підтримка аналізу коду дозволяє розробникам писати більш надійний та ефективний код.

Окрім цього, PyCharm володіє інтегрованою системою керування версіями, яка дозволяє зручно працювати з кодовою базою та відслідковувати зміни. Широкий спектр розширень та плагінів дозволяє налаштовувати середовище розробки під власні потреби та предметну область проекту.

PyCharm встановлює високі стандарти для роботи з Python та надає розробникам зручне та продуктивне середовище для творення високоякісного програмного забезпечення.

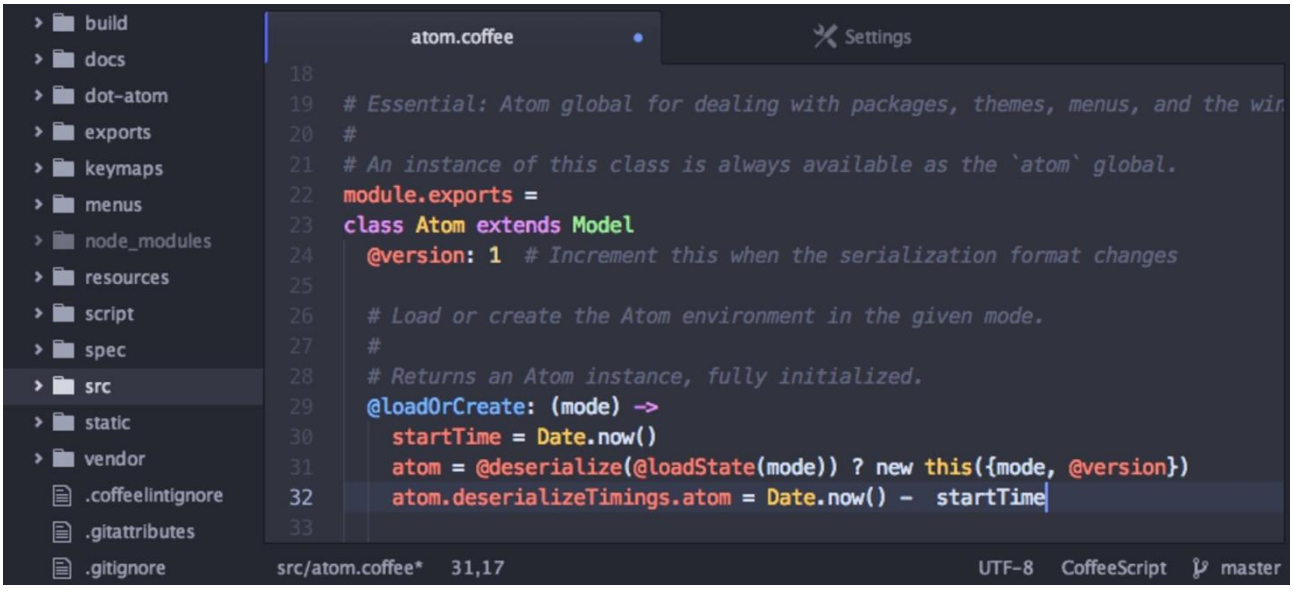


Рисунок 3.2 – Середовище Atom

2) Atom, розроблений компанією GitHub, представляє собою розширюваний текстовий редактор, спроектований з урахуванням потреб розробників. Його основна перевага полягає в гнучкості та можливості

налаштувань завдяки великій кількості розширень та плагінів, що дозволяє створювати потужне середовище для розробки на мові Python.

Один з ключових аспектів Atom - це його відкритість для спільноти. Розширення можуть бути створені та вдосконалені спільнотою розробників, що робить Atom ідеальним вибором для тих, хто цінує відкритість та активну участь у розвитку інструментів.

Atom підтримує основні функції, такі як автоматичне завершення коду, відладка та керування версіями, і створює комфортне середовище для написання та редагування коду. Його легкість використання та можливість розширення робить його популярним вибором серед багатьох розробників Python, які цінують простоту та гнучкість.

PyCharm встановлює високі стандарти для роботи з Python та надає розробникам зручне та продуктивне середовище для творення високоякісного програмного забезпечення.

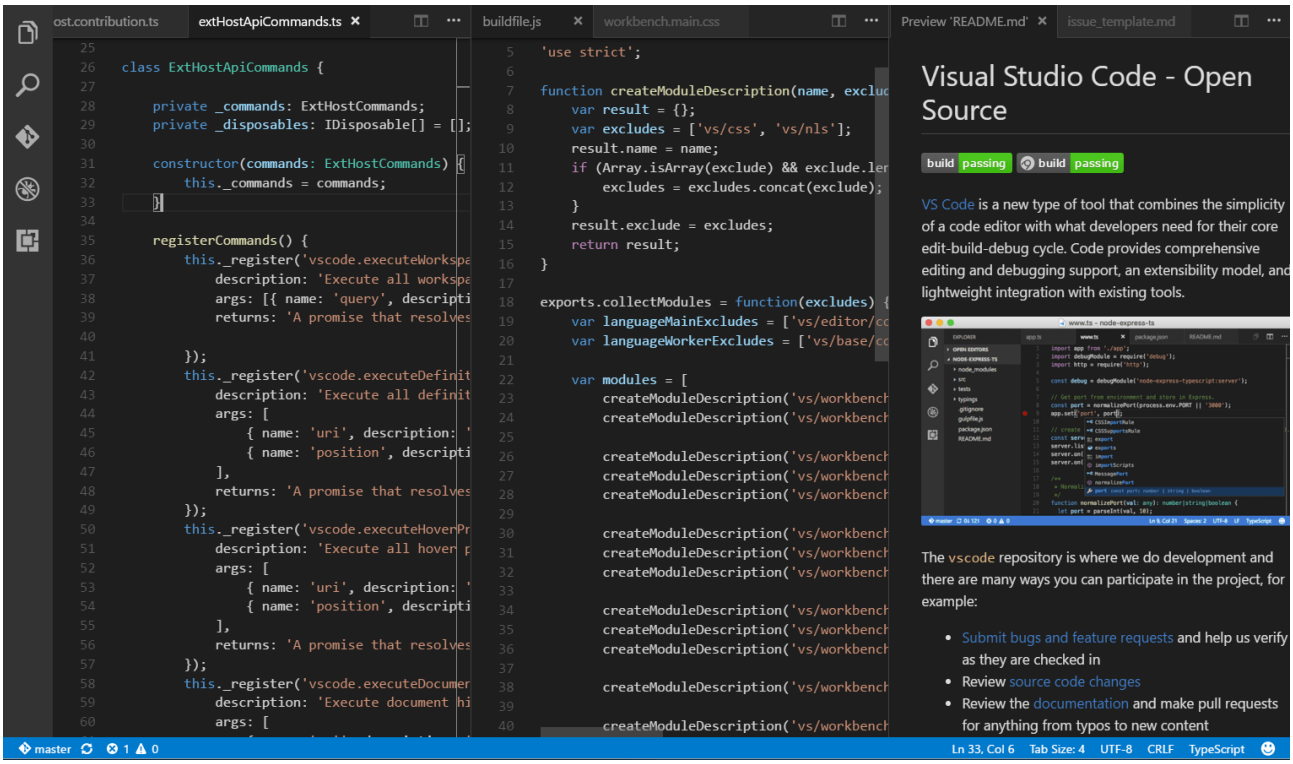


Рисунок 3.3 – Середовище Visual Studio Code

3) Visual Studio Code, часто скорочуване до VS Code, є відкритим та легким текстовим редактором, розробленим компанією Microsoft. Цей редактор став

дуже популярним серед розробників Python завдяки своїм потужним можливостям та великому співтовариству користувачів.

Основні переваги VS Code включають розширюваність та широкий вибір розширень для роботи з мовами програмування, включаючи Python. Завдяки інтегрованим інструментам для відладки, автоматичного завершення коду та підтримки керування версіями, VS Code забезпечує продуктивне середовище для розробки.

Окрім цього, VS Code володіє зручним інтерфейсом користувача та можливістю роботи з різними типами проектів. Велика кількість налаштувань та можливість роботи з Git робить його улюбленим інструментом для багатьох розробників, які працюють з Python.

PyCharm встановлює високі стандарти для роботи з Python та надає розробникам зручне та продуктивне середовище для творення високоякісного програмного забезпечення.

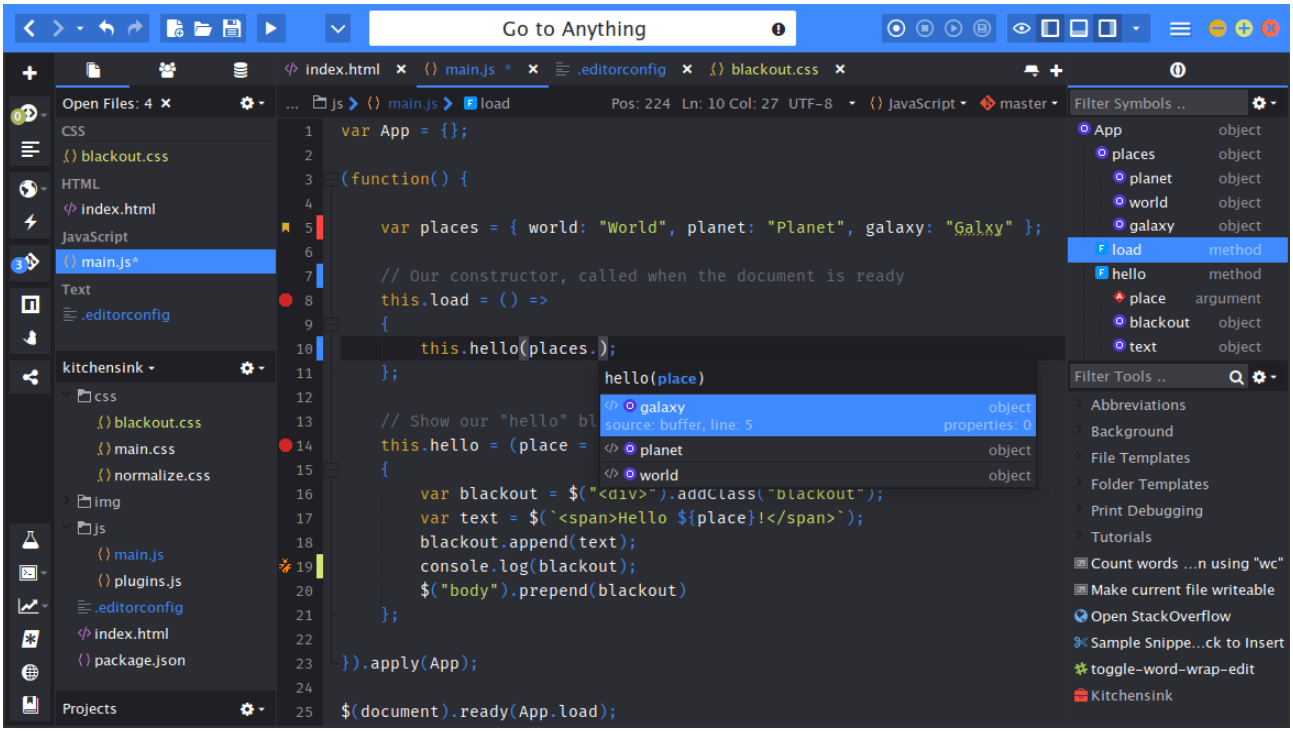


Рисунок 3.4 – Середовище Comodo IDE

4) Komodo IDE - інтегроване середовище розробки (IDE), що підтримує різні мови програмування, включаючи Python. Забезпечує автоматичне завершення коду, відлагодження, підтримку віртуальних середовищ,

розширюваність та підтримку різних технологій. Відмінний вибір для розробників, які працюють з декількома мовами програмування. Komodo IDE підтримує різні мови програмування, включаючи Python, JavaScript, HTML, CSS, PHP, Ruby та інші, що розширює можливості для проєктів, що використовують різні технології. Інтегрована система автоматичного завершення коду полегшує написання та редагування коду, пропонуючи підказки та автоматичні вставки. Вбудований засіб відлагодження дозволяє ефективно виявляти та виправляти помилки у програмному коді. Підтримка віртуальних середовищ розуміє важливість ізоляції проектів та забезпечення чистоти робочого середовища. Розширюваність через підтримку розширень дозволяє розробникам додавати новий функціонал та адаптувати його до своїх потреб.

Обґрунтування вибору компонентів для підсистеми моніторингу застосунків виявилося фундаментальним кроком у розробці ефективної та надійної системи. Аналіз виправдав вибір конкретних компонентів, орієнтованих на забезпечення широкого охоплення моніторингу, включаючи аспекти продуктивності, надійності та безпеки. Впровадження високопродуктивних інструментів виявиться ключовим фактором для отримання точних та повних даних з моніторингу, сприяючи ефективному управлінню та прийняттю інформованих рішень. Такий підхід до вибору компонентів підсистеми моніторингу покладає тверді засади для подальшого розвитку та оптимізації системи управління та контролю застосунків.

3.2 Розробка програмного засобу

Розробка підсистеми моніторингу в хмарному середовищі AWS включає в себе кілька ключових етапів та модулів, спрямованих на забезпечення ефективного, надійного та безпечного контролю за різними аспектами застосунків та інфраструктури

Аналіз кроків реалізації підсистеми моніторингу застосунків в хмарному середовищі AWS вказує на високий рівень вдосконалення та надійності моніторингового процесу. Використання різних сервісів та інструментів AWS

дозволяє створити комплексну систему, яка охоплює різні аспекти функціонування застосунку.

Спочатку, процес стартує зі створення облікового запису AWS та ролі IAM, надаючи необхідні права доступу. Це забезпечує безпечний та контрольований доступ до сервісів AWS. Подальша конфігурація CloudWatch Metrics і CloudWatch Alarms дозволяє ефективно відстежувати та аналізувати метрики продуктивності застосунку. Після успішної реєстрації, новий користувач перенаправляється на головну сторінку AWS (рис. 3.5), де він може створити обліковий запис з адміністративними правами.

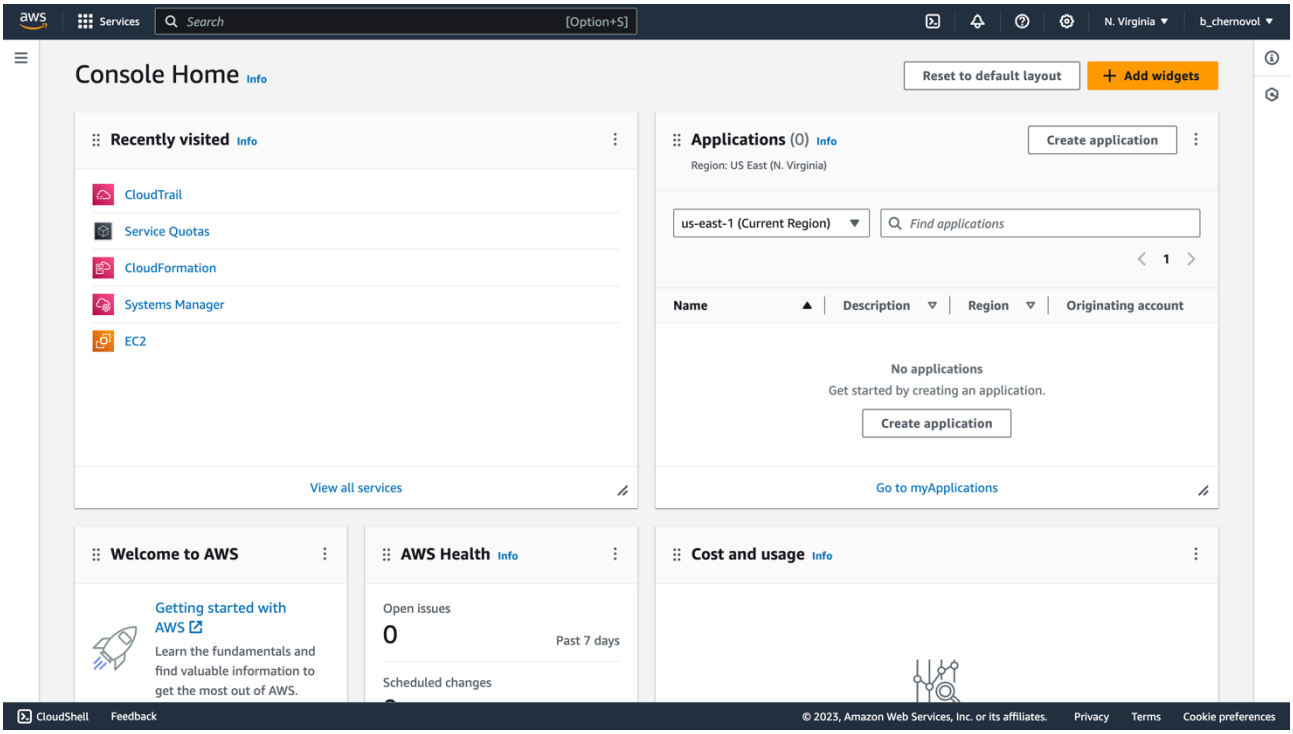


Рисунок 3.5 – Головна сторінка AWS консолі

Налаштування CloudTrail в хмарному середовищі AWS включає кілька ключових етапів для ефективного моніторингу та аудиту подій. Перш за все, потрібно визначити шлях для збереження журналів подій, вказавши бакет Amazon S3 для надійного зберігання даних. Далі, важливо вибрати регіон, для якого буде активований CloudTrail (рис. 3.6), та визначити конкретні події та активності, які слід відстежувати, такі як вхід та вихід з облікових записів, зміни конфігурації ресурсів, доступ до API тощо.

Додатково, слід включити аудит для відстеження дій користувачів та налаштувати управління доступом для забезпечення безпеки. Також, важливо визначити частоту збереження журналів у бакеті S3 та налаштувати можливість надсилання сповіщень про певні події через Amazon SNS для оперативного реагування на важливі сценарії(рис. 3.6). Правильне налаштування CloudTrail є ключовим елементом для забезпечення безпеки, відповідності та відстеження активності в хмарному середовищі AWS.

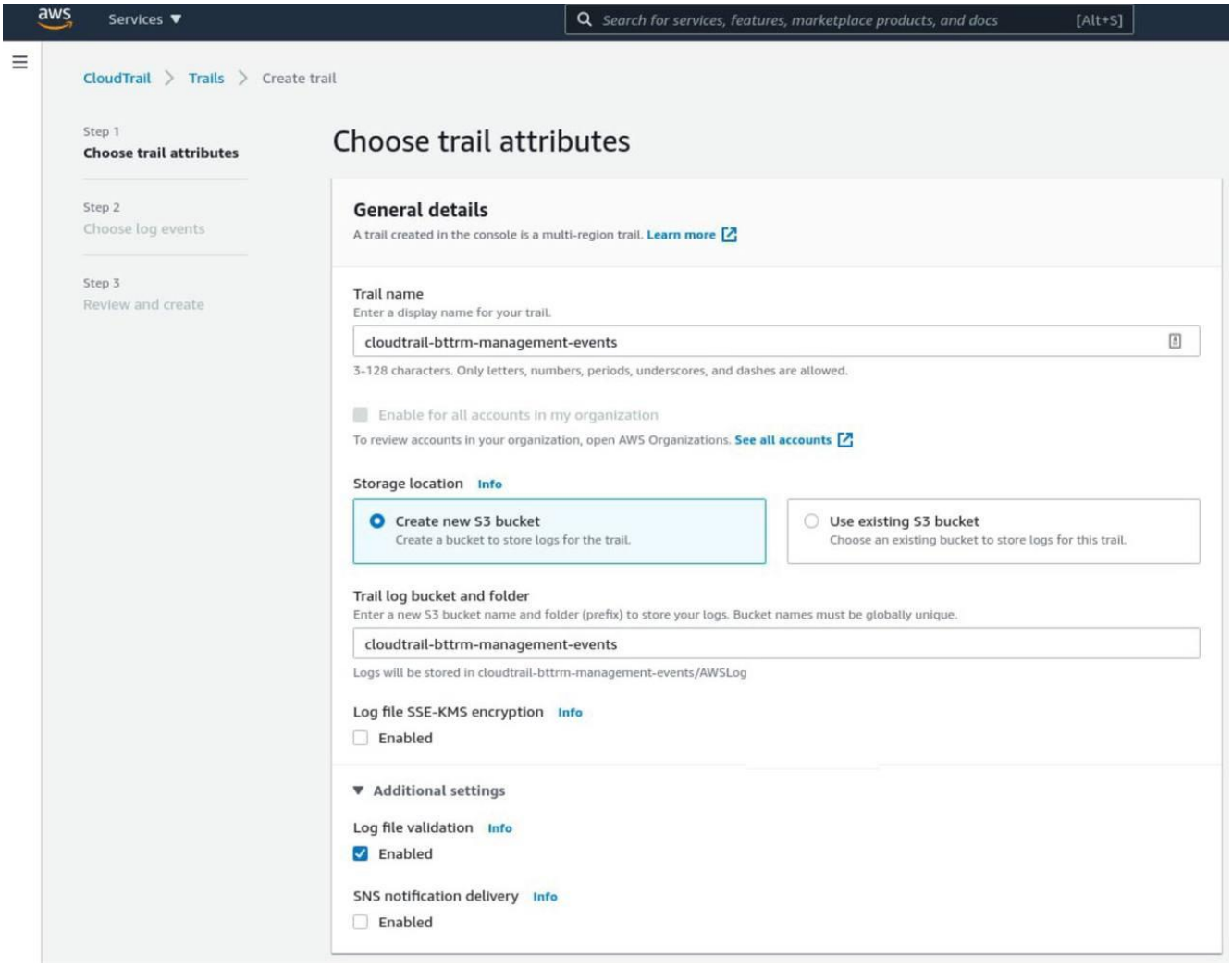


Рисунок 3.6 –Сторінка керування CloudTrail

В бакетах Amazon S3 зберігаються логи в чистому вигляді, це корисно для аналізу та моніторингу різних операцій та подій в середовищі AWS (рис. 3.7). Логи містять інформацію про запити до об'єктів, доступ до бакетів, помилки, аудит-інформацію та інше.

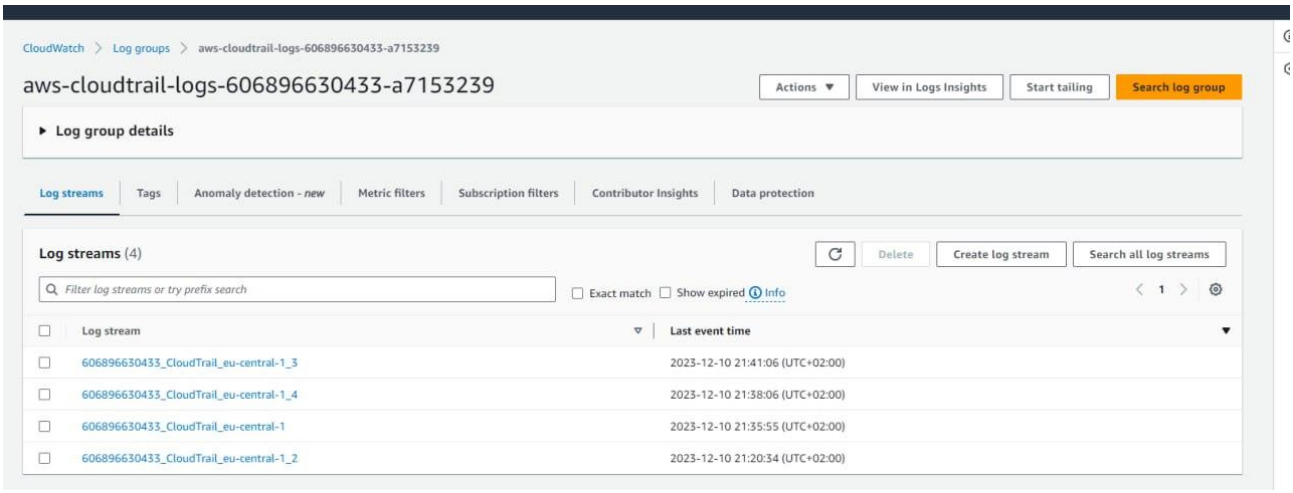


Рисунок 3.7 – Бакет де зберігаються логи

Логи передаються з AWS CloudTrail в Amazon CloudWatch, використаємо CloudWatch для моніторингу та налаштування тривог (рис. 3.8). Це дозволить отримувати сповіщення про певні події або умови, що виникають в ході виконання операцій у AWS-середовищі.

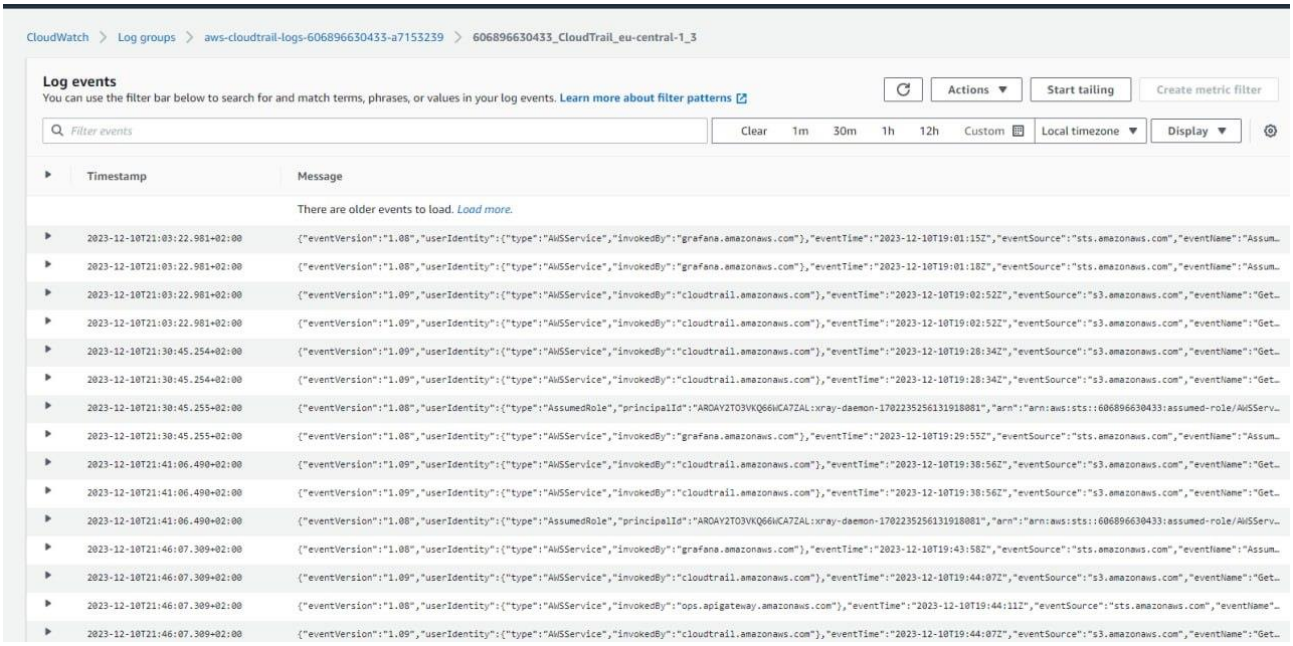


Рисунок 3.8 – Бакет де зберігаються логи

Налаштування Amazon CloudWatch для ефективної підсистеми моніторингу застосунків в середовищі AWS включає кілька ключових кроків. По-перше, необхідно визначити метрики, які ви хочете відстежувати. Це може включати продуктивність, доступність, використання ресурсів тощо. Створення відповідних метрик дозволяє отримати уявлення про функціонування застосунку.

Далі, слід створити спостерігачі (alarms) в CloudWatch для кожної метрики, які будуть активовані при досягненні або перевищенні певних порогових значень. Це дозволяє оперативно виявляти потенційні проблеми та запускати автоматизовані сценарії в разі необхідності (рис. 3.9).

Також, важливо використовувати CloudWatch Logs для централізованого збору та аналізу журналів з застосунків. Це надає можливість відстежувати подробиці роботи застосунків та виявляти проблеми у вигляді журналів подій.

Забезпечення належного конфігурування CloudWatch дозволяє створити ефективну систему моніторингу, що допомагає вчасно виявляти та розв'язувати можливі проблеми з застосунками в середовищі AWS.

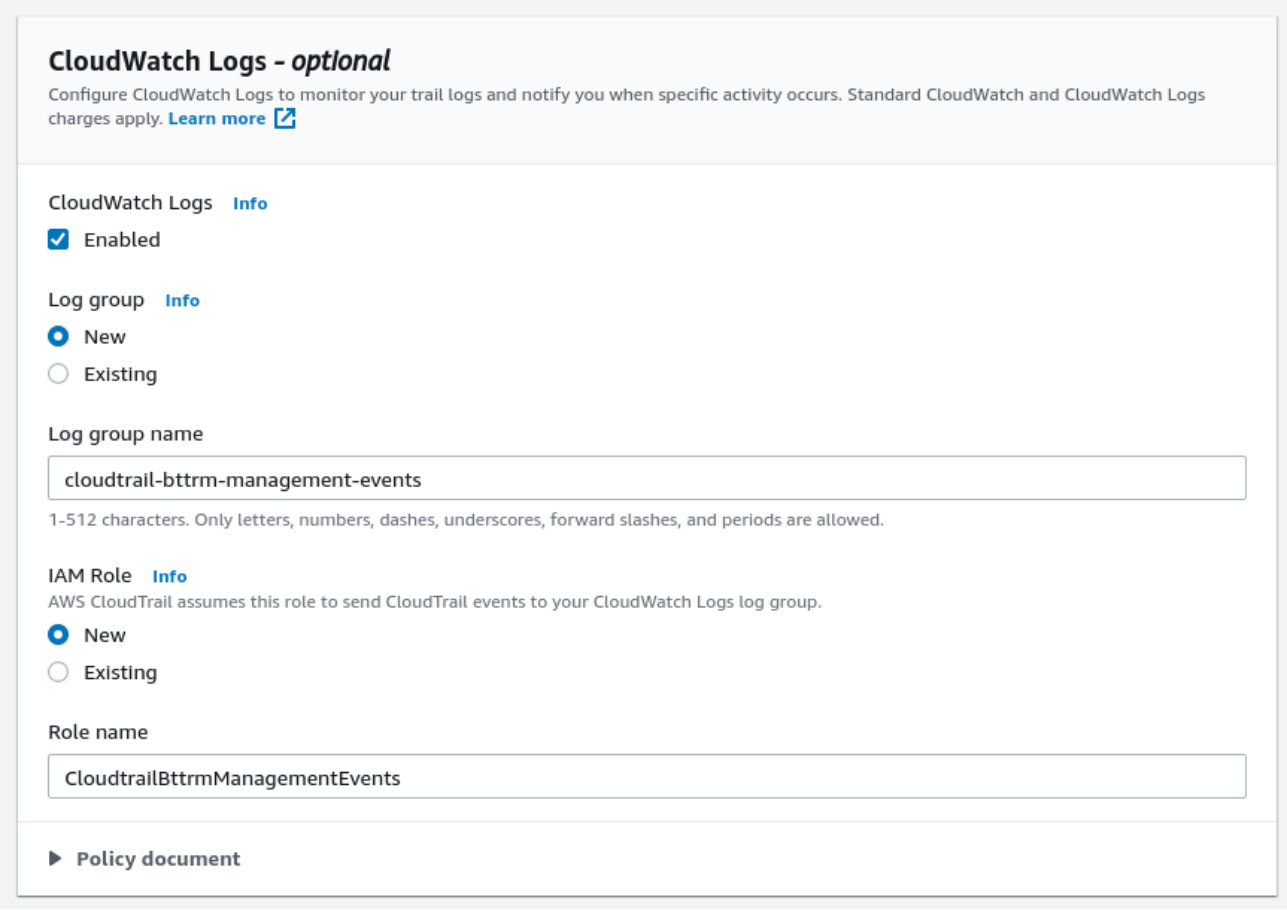


Рисунок 3.9 – Сторінка налаштувань сповіщень в CloudWatch

В розділі "Події керування" (Management events) активуємо опції "Читання" і "Запис", а для "Подій інсайтів" (Insights events) активуємо "Швидкість виклику API".

Це налаштування дозволяє включити моніторинг інформації про події управління, такі як читання та запис, а також визначити швидкість викликів API для отримання інсайтів з використання ресурсів (рис. 3.10).

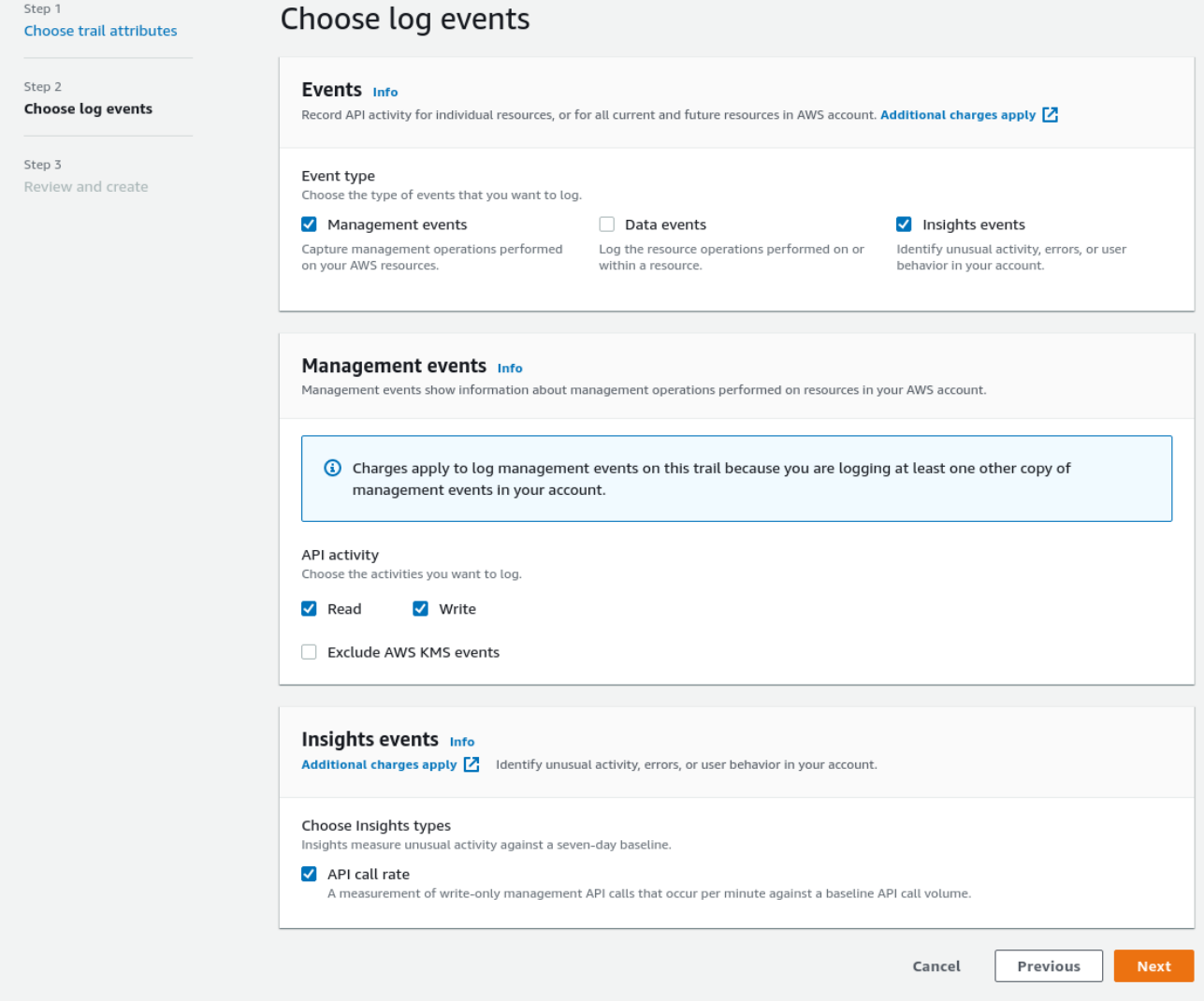


Рисунок 3.10 – Сторінка налаштувань сповіщень в CloudWatch

AWS Lambda відповідає за автоматизацію реакції на події, запускаючи сценарії при виявленні проблем. Це спрощує вирішення проблем та забезпечує оперативну відповідь на негативні ситуації. Використання Simple Email Service (SES) для надсилання сповіщень через електронну пошту робить процес інформування користувачів більш гнучким та універсальним. Для налаштування AWS Lambda слід спочатку створити нову функцію в консолі AWS (рис. 3.11). При цьому ви визначаєте назву функції, обираєте роль, яка має права доступу до інших ресурсів AWS, і обираєте виконавчу роль.

Далі слід додати тригери, які запускатимуть вашу функцію Lambda. Це може бути подія, така як завершення завдання в Amazon S3 або виклик API Gateway. Тригери визначають, коли саме має викликатися функція.

Після цього слід налаштувати середовище функції, визначивши змінні середовища, параметри пам'яті, тайм-аут та інші конфігурації. Налаштування захисту та безпеки, такі як правила управління доступом (IAM) та конфігурації віртуальних приватних хмар (VPC), також є важливою частиною налаштування для забезпечення безпеки вашої функції.

Після налаштування служби AWS Lambda важливо включити моніторинг та журналювання, використовуючи Amazon CloudWatch. Це дозволяє вам відстежувати метрики та журнали для вашої функції Lambda, щоб забезпечити її ефективну роботу та вчасно виявляти проблеми. Завершується процес розгортання та тестування функції Lambda для забезпечення її коректної роботи в середовищі AWS.

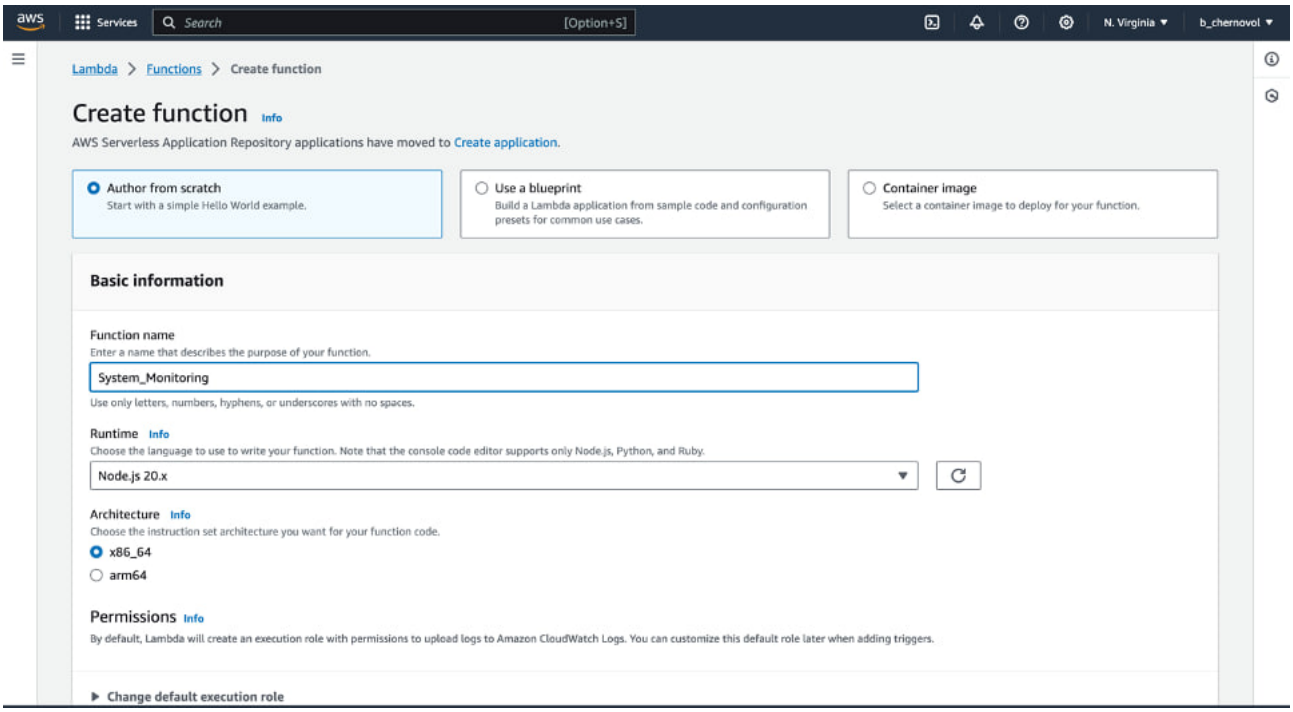


Рисунок 3.11 – Сторінка налаштувань AWS Lambda

Для відправки аларму з AWS Lambda використовується бібліотека boto3 для взаємодії з AWS-сервісами. Нижче подано базовий код на Python для створення аларму та його відправлення з функції Lambda.

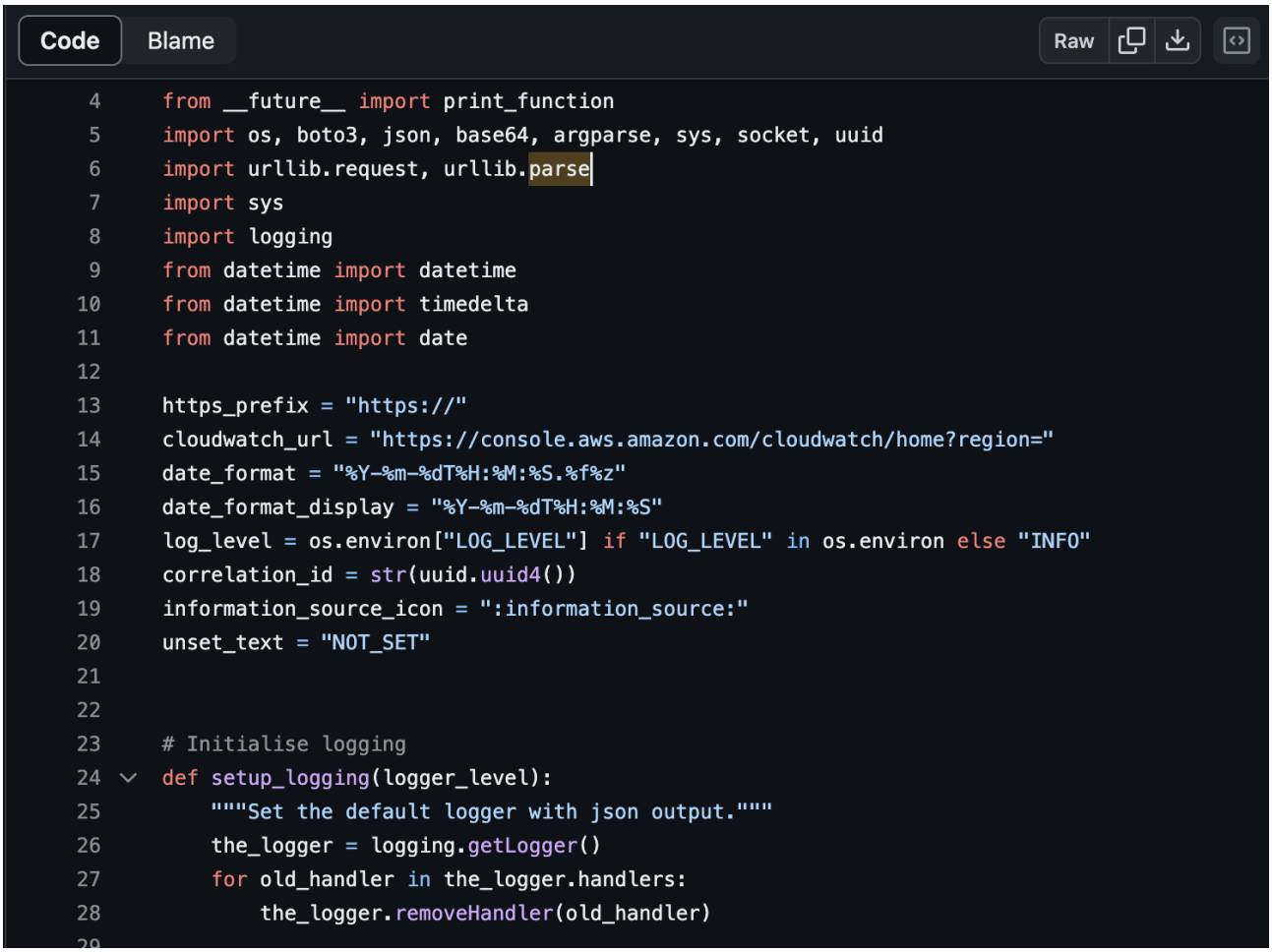


Рисунок 3.12 – Редагування коду Lambda функції

Функція, **setup_logging**, встановлює конфігурацію системи логування в програмі. Вона отримує рівень логування як параметр та налаштовує логер так, щоб виводити логи у форматі JSON із різноманітною інформацією, такою як час, рівень логування, повідомлення, середовище, ім'я застосунку, модуль, ID процесу, ID потоку та ім'я хоста. Функція видаляє всі наявні обробники логера та додає новий обробник, який виводить логи у стандартний вивід (stdout)..

```
def setup_logging(logger_level):
    """Set the default logger with json output."""
    the_logger = logging.getLogger()
    for old_handler in the_logger.handlers:
        the_logger.removeHandler(old_handler)

    new_handler = logging.StreamHandler(sys.stdout)
    hostname = socket.gethostname()
    json_format = (
        '{ "timestamp": "%(asctime)s", "log_level": "%(levelname)s", "message":
"%(message)s", '
        f'"environment": "{args.environment}","application":
"{args.application}", '
        f'"module": "%(module)s", "process":"%(process)s", '
        f'"thread": "[%(thread)s]", "hostname": "{hostname}" }}'
    )
```

Метод **config_notification**, використовується для обробки повідомлень про конфігурацію. Вона отримує повідомлення про конфігурацію, AWS-регіон та, можливо, певну конфігурацію (**payload**). Функція виводить інформаційне повідомлення з екранованим рядком JSON, областю AWS та ідентифікатором кореляції. Також визначає два словники для відповідності статусів (COMPLIANT, NOT_APPLICABLE, NON_COMPLIANT) їх текстовим або емодзі-представленням.

```
      def config_notification(message, region, payload):
          dumped_message = get_escaped_json_string(message)
          logger.info(
              f'Processing config notification", "dumped_message":
{dumped_message}, "region": "{region}", "correlation_id": "{correlation_id}'
          )
          no_emojis = {
              "COMPLIANT": "Compliant",
              "NOT_APPLICABLE": "N/A",
              "NON_COMPLIANT": "Non-Compliant",
          }
          emojis = {
              "COMPLIANT": ":white_check_mark:*Compliant*:white_check_mark:",
              "NOT_APPLICABLE": ":white_check_mark:*N/A*:white_check_mark:",
              "NON_COMPLIANT": ":x:*Non-Compliant*:x:",
          }
```

Функція, **config_cloudwatch_alarm_notification**, обробляє сповіщення про спрацювання тривоги у сервісі AWS CloudWatch та надає конфігурацію для відправки сповіщення на email.

```
def config_cloudwatch_alarm_notification(
    message, region, prowler_slack_channel, payload
):
    dumped_message = get_escaped_json_string(message)
    logger.info(
        f'Processing cloudwatch notification", "dumped_message":
{dumped_message}, "region": "{region}", "prowler_email":
{prowler_slack_channel}, "correlation_id": "{correlation_id}'
    )

    trigger_object = message["Trigger"] if "Trigger" in message else None

    if (
        trigger_object is not None
        and "Namespace" in trigger_object
        and trigger_object["Namespace"] == "Prowler/Monitoring"
    ):
```

```
        payload = config_prowler_cloudwatch_alarm_notification(message, region,
payload)
        payload["channel"] = prowler_email
    else:
        payload = config_custom_cloudwatch_alarm_notification(message, region,
payload)

    return payload
```

Після налаштування підсистеми можна переглянемо історію алармів. Налаштування підсистеми для моніторингу та обробки алармів зазвичай дозволяє збирати та зберігати інформацію про події, які призводять до спрацювання алармів (рис. 3.13). Ця історія включає деталі про те, коли аларми спрацьовують, який був тип нотифікації, опис та які параметри були порушені, і інші відомості.

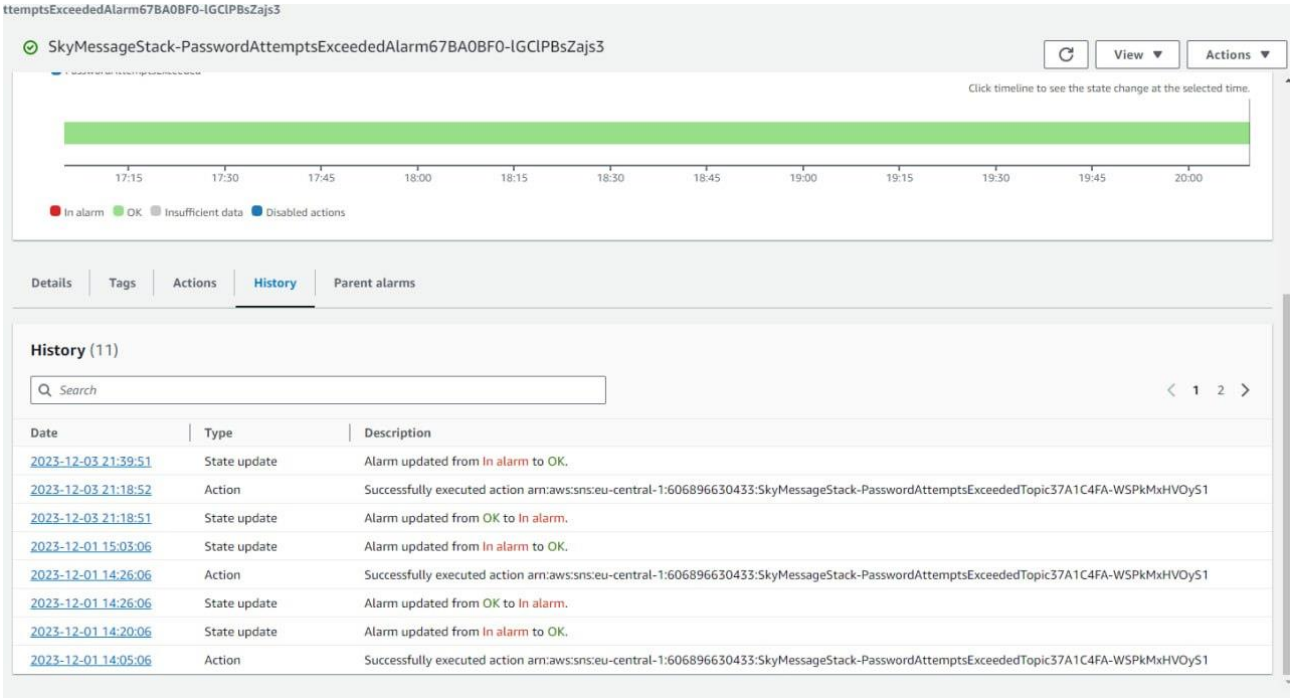


Рисунок 3.13 – Історія виклику Lambda функції.

Налаштовано тригер для функції Lambda, функція готова реагувати на вказані події чи зміни в системі. Тригер дозволяє функції відповідати на події в реальному часі та автоматично викликати функцію Lambda при необхідних умовах. Таким чином, функція відповідає на зміни в S3-бакеті безпосередньо під час їх виникнення. Налаштування тригера для функції Lambda надає можливість створювати динамічний та реактивний застосунок.

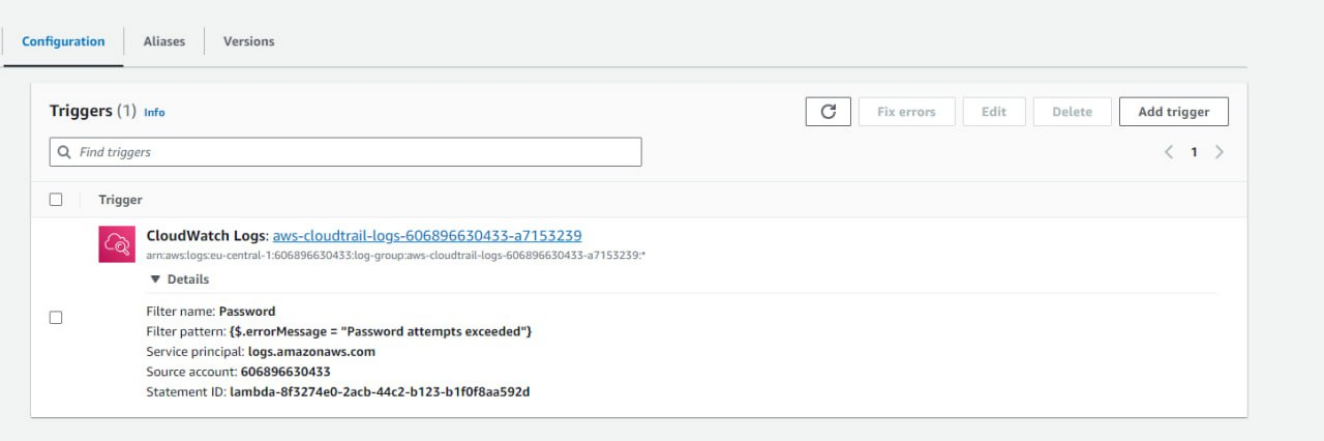


Рисунок 3.14 – Історія виклику Lambda функції.

У підсумку, використання AWS CloudTrail, AWS Lambda, Amazon CloudWatch, Simple Email Service, виявилося надзвичайно важливим для успішної розробки та підсистеми моніторингу в середовищі Amazon Web Services (AWS). Загальне використання цих сервісів дозволило не лише забезпечити ефективність та надійність підсистеми, але й забезпечити її готовність до впровадження в середовищі AWS. Ці інструменти стали основою для побудови стійкого та динамічного середовища, готового відповідати на вимоги сучасних хмарних технологій.

3.3 Тестування підсистеми моніторингу

Тестування підсистеми моніторингу важливо для забезпечення стабільності та ефективності. Воно дозволяє виявляти та виправляти потенційні проблеми перед впровадженням в реальному виробничому середовищі, забезпечуючи високий рівень якості та надійності моніторингової системи.

На поданому графіку відображена повна статистика викликів до нашого API, яка забезпечує всебічний огляд його функціонування. Перш за все, вказана загальна кількість викликів, яка відображає обсяг запитів до API за певний період часу, включаючи як вдачі, так і невдачі.

Додатково вказана кількість вдалих спроб, що вказує на кількість успішних викликів, та кількість помилок 401 і 500 (рис. 3.15). Останні вказують на проблеми, пов'язані з аутентифікацією та внутрішніми помилками API, і важливі для вчасної ідентифікації та виправлення.

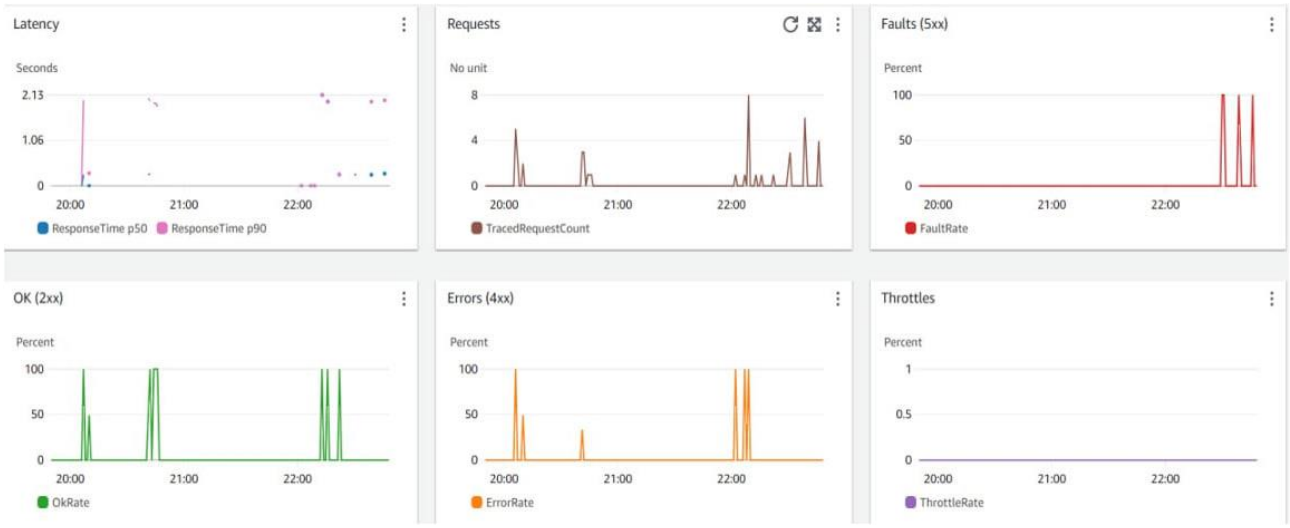


Рисунок 3.15 – Статистика викликів API.

Крім того, на графіку представлена інформація про швидкість відповіді від API, яка вказує на час, який необхідний для виконання кожного запиту (рис. 3.16). Цей параметр є ключовим для оцінки продуктивності та ефективності API.

Загальною метою аналізу цих показників є не лише виявлення проблем, а й постійне вдосконалення якості обслуговування та реакції на зміни у функціонуванні API. Такий моніторинг допомагає забезпечити стабільність та надійність API у реальному часі.

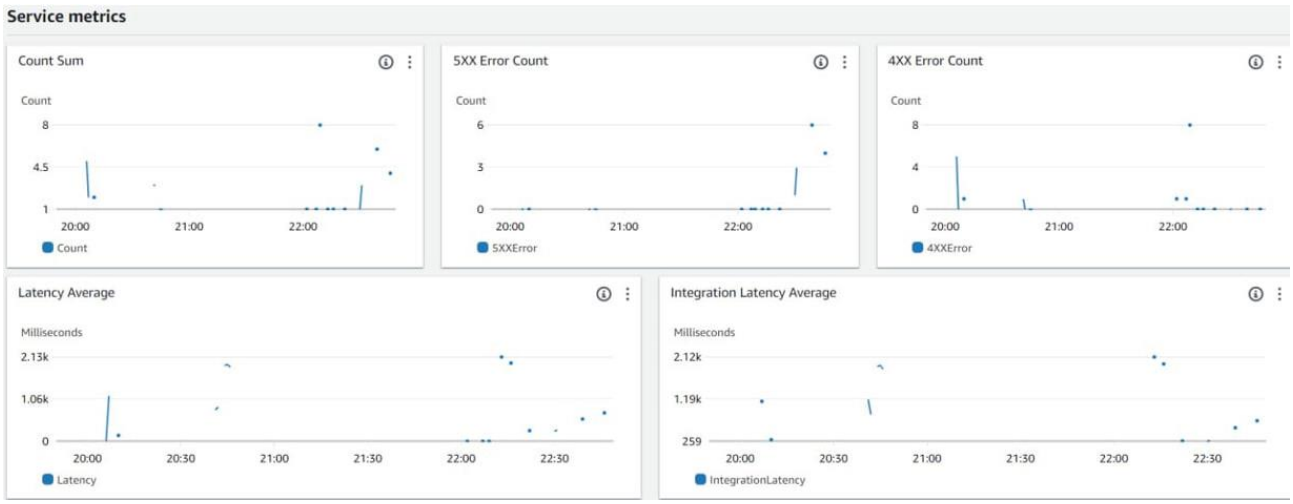


Рисунок 3.16 – Метрики сервісу.

Налаштування AWS Grafana дозволяє ефективно використовувати цей інструмент для візуалізації та моніторингу даних у середовищі Amazon Web

Services (AWS). Перш за все, слід встановити та налаштувати Grafana на сервері або скористатися хмарним сервісом. Після входу в інтерфейс Grafana обираємо "Settings" та переходимо до "Data Sources" (рис. 3.17). Тут додаємо нове джерело даних, вказуючи X-Ray як джерело, оскільки це часто використовується для моніторингу AWS-ресурсів.

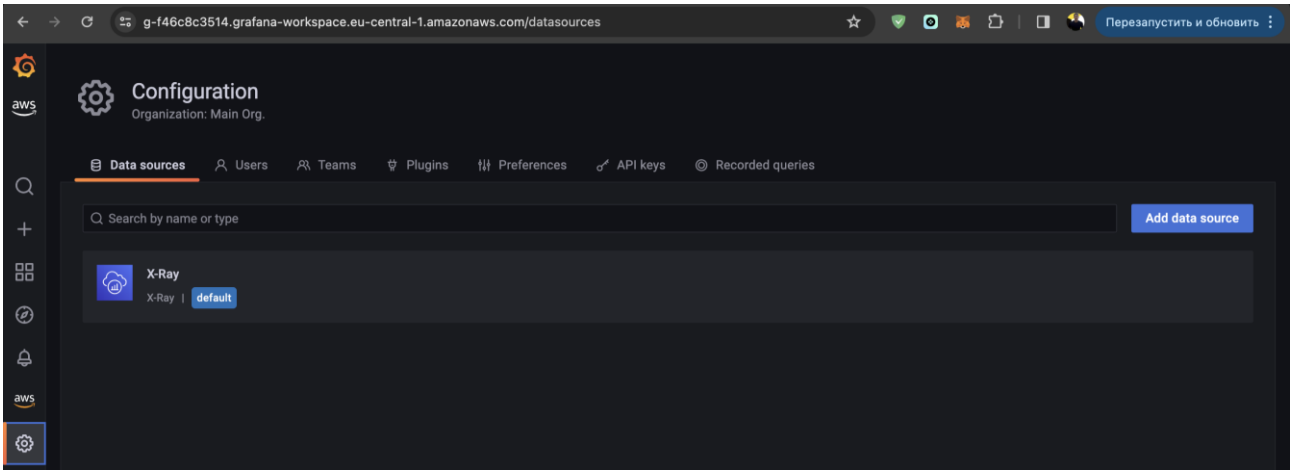


Рисунок 3.17 – Додавання Data Sources для Grafana.

Після цього налаштовуємо параметри графіка, визначаємо інтервал часу та функції агрегації. Зберігаємо панель та можемо візуалізувати дані. Крім того, можливо зберегти та обмінюватися графіками, а також вбудовувати їх у веб-сторінки.

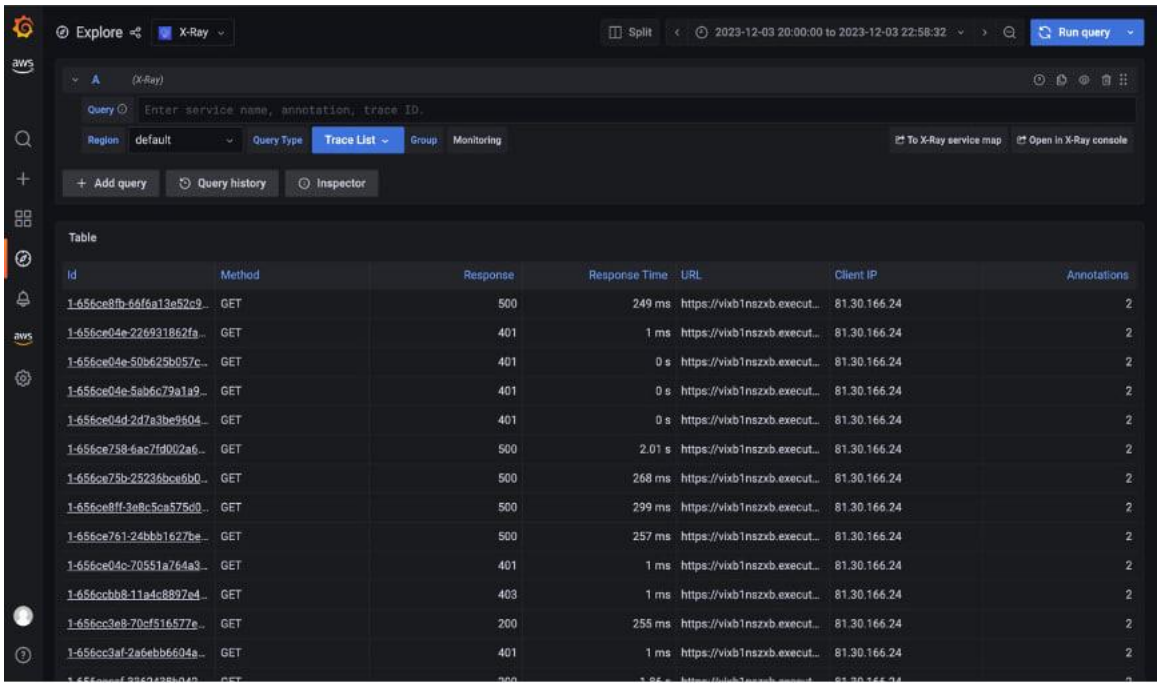


Рисунок 3.18 – Відстеження викликів API в Grafana

Розглядаючи розширені налаштування та плагіни, ви можете отримати доступ до додаткового функціоналу. Такий підхід дозволяє ефективно використовувати Grafana для моніторингу та аналізу різних метрик та ресурсів у вашому AWS-середовищі. Побудова графіка із результатів викликів вашого API за допомогою Grafana. Створивши новий Dashboard в розділі "Dashboard", де відображаються графіки. Наступним кроком є додавання панелі для графіка, обираючи тип "Time Series", який відобразить дані в часовому розрізі. Налаштувавши джерело даних для панелі, вказуючи відповідне джерело, яке вказує API.

Далі конфігурується запит для отримання необхідних даних, враховуючи параметри, які важливі для відслідковування результатів викликів. При збережені налаштованого графіку та розглянуто можливість використання додаткових налаштувань та плагінів для покращення функціональності та аналізу результатів викликів вашого API. Такий графік дозволить ефективно візуалізувати та аналізувати продуктивність та ефективність API на основі зібраних даних.

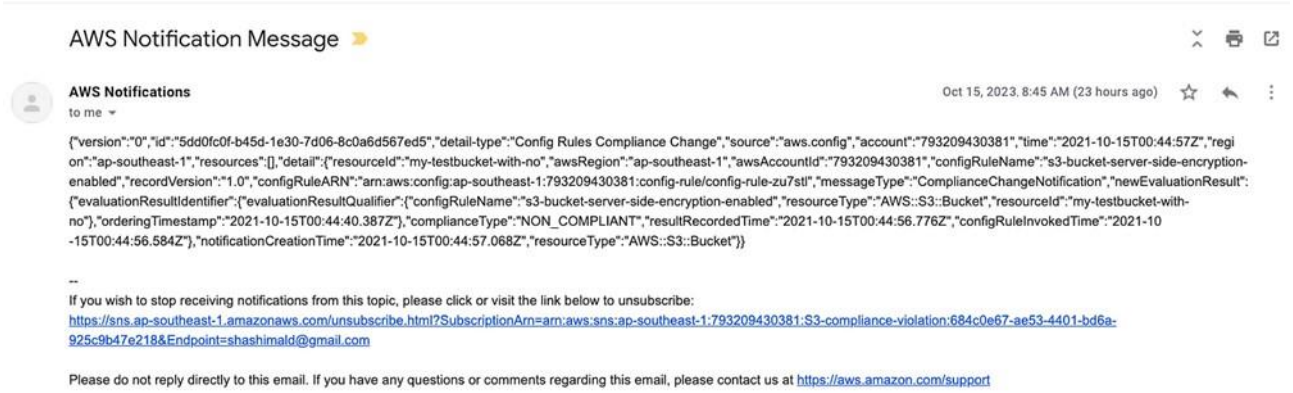
Рисунок 3.19 – Побудований графік із результатів виклику API.

Перегляд результатів відпрацювання Lambda в Grafana надає вам можливість визначити ефективність функцій, швидкість виконання та, при

необхідності, виявлення можливих проблем. Налаштування параметрів запиту таких як час виконання функцій чи можливі помилки, дозволяє стежити за ключовими аспектами роботи AWS Lambda.

Рисунок 3.20 – Результати роботи AWS Lambda.

Додавання системи сповіщень при виявленні аномалій, наприклад, брутфорс-атаки, є важливим етапом в розробці підсистеми моніторингу. Використання Amazon Simple Email Service (SES) для надсилання сповіщень, цей сервіс може легко інтегруватися із підсистемою моніторингу для відправлення повідомлень про аномалії на електронну пошту адміністраторів або відповідальних осіб. Цей підхід дозволяє ефективно використовувати Amazon SES для відправлення повідомлень про аномалії, забезпечуючи надійну інтеграцію між вашою підсистемою моніторингу та сервісом доставки електронної пошти.


AWS Notification Message

AWS Notifications Oct 15, 2023, 8:45 AM (23 hours ago)
to me

{"version":"0","id":"5dd0fc0f-b45d-1e30-7d06-8c0a6d567ed5","detail-type":"Config Rules Compliance Change","source":"aws.config","account":"793209430381","time":"2021-10-15T00:44:57Z","region":"ap-southeast-1","resources":[],"detail":{"resourceId":"my-testbucket-with-no","awsRegion":"ap-southeast-1","awsAccountId":"793209430381","configRuleName":"s3-bucket-server-side-encryption-enabled","recordVersion":"1.0","configRuleARN":"arn:aws:config:ap-southeast-1:793209430381:config-rule/config-rule-zu7stl","messageType":"ComplianceChangeNotification","newEvaluationResult":{"evaluationResultIdentifier":{"evaluationResultQualifier":{"configRuleName":"s3-bucket-server-side-encryption-enabled","resourceType":"AWS::S3::Bucket","resourceId":"my-testbucket-with-no"},"orderingTimestamp":"2021-10-15T00:44:40.387Z"},"complianceType":"NON_COMPLIANT","resultRecordedTime":"2021-10-15T00:44:56.776Z","configRuleInvokedTime":"2021-10-15T00:44:56.584Z"},"notificationCreationTime":"2021-10-15T00:44:57.068Z","resourceType":"AWS::S3::Bucket"}}

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.ap-southeast-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:ap-southeast-1:793209430381:S3-compliance-violation:684c0e67-ae53-4401-bd6a-925c9b47e218&Endpoint=shashimald@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

Рисунок 3.21 – Приклад відправки сповіщення на пошту.

Впровадження системи SMS-сповіщень дозволить оперативно і ефективно повідомляти відповідальних осіб про важливі події чи аномалії. Підсистема

моніторингу використовує для цього Amazon Simple Notification Service (SNS) для надсилання SMS-повідомлень на зазначені номери телефонів.

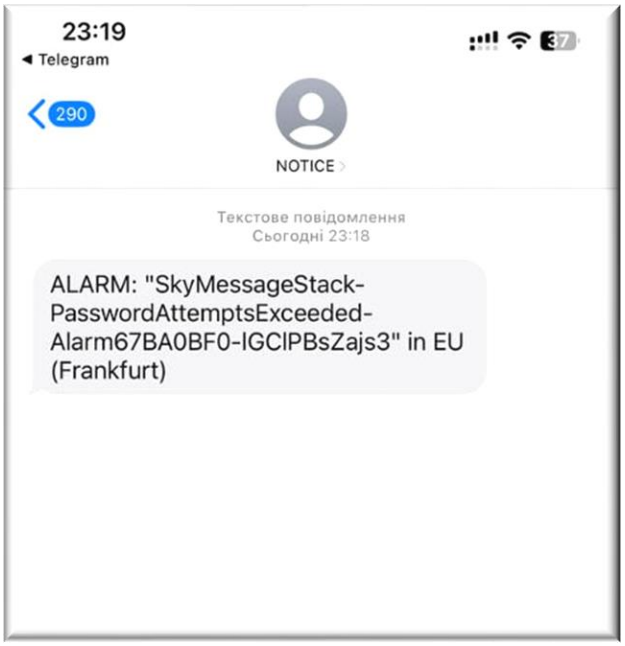


Рисунок 3.22 – Приклад відправки SMS - сповіщення.

В результаті тестування підсистеми моніторингу в AWS можна визначити, що вона успішно виконує свої ключові функції. Висока надійність та ефективність підсистеми були виявлені під час виявлення аномалій та моніторингу важливих метрик. Інтеграція з різними сервісами AWS, такими як CloudWatch, Lambda і SNS, забезпечує ефективний обмін даними та спрощує процес моніторингу. Особливо важливим є успішне використання SMS-сповіщень за допомогою Amazon SNS, що дозволяє швидко та оперативно реагувати на важливі ситуації. Аналіз результатів через інструменти, такі як AWS CloudWatch та Grafana, забезпечує зручність та прозорість в роботі системи. Здатність системи ефективно реагувати на аномалії та відповідати вимогам підтверджує готовність підсистеми до впровадження в реальному.

4 ЕКОНОМІЧНА ЧАСТИНА

Для успішного впровадження науково-технічної розробки важливо, щоб вона відповідала актуальним вимогам науково-технічного прогресу та ураховувала економічні аспекти. Надання оцінки економічної ефективності результатів науково-дослідної роботи є важливою складовою цього процесу. Дослідження, що представлене у магістерській роботі та присвячене розробці та вивченню "Системи автоматизованого керування безпекою застосунків у хмарному середовищі ", віднесено до науково-технічних робіт, спрямованих на виведення на ринок. Рішення щодо комерціалізації розробки може бути прийняте під час самого виконання роботи, розкриваючи можливості для подальшого введення на ринок. Цей напрямок розглядається як пріоритетний, оскільки розроблені результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди. Однак для успішної реалізації цього процесу важливо залучити зацікавленого інвестора, який виявить інтерес до втілення даного проекту, і переконати його у доцільності інвестування у цю розробку. З цією метою були визначені наступні етапи виконання робіт:

1. Проведення комерційного аудиту науково-технічної розробки, включаючи визначення науково-технічного рівня та комерційного потенціалу.

2. Розрахунок витрат на реалізацію науково-технічної розробки.

3. Проведення розрахунку економічної ефективності впровадження та комерціалізації науково-технічної розробки для потенційного інвестора, а також обґрунтування економічної доцільності комерціалізації з точки зору інвестора.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Система автоматизованого керування безпекою застосунків у хмарному середовищі» є розробка та впровадження підсистеми моніторингу застосунків у хмарному середовищі для покращення безпеки та надійності роботи застосунків.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 **[Козловський, Лесько, Кавецький]**.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту	Технічні та споживчі влас-тивості продукту	Технічні та споживчі властивості продукту на	Технічні та споживчі властивості продукту	Технічні та споживчі властивості продукту
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційни х витрат	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела	Потрібні незначні фінансові ресурси. Джерела фінансування	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуют ь ся у військово	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуют
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів Вінницького національного технічного університету кафедри «Захисту інформації»: Дудатьєв Андрій Веніамінович, к. т. н., доцент, Куперштейн Леонід Михайлович, к. т. н., доцент, Баришев Юрій Володимирович, к. т. н., доцент.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Дудатьєв А. В.	Куперштейн Л. М.	Баришев Ю. В.
	Бали, виставлені експертами:		
1. Технічна здійсненність концепції	4	4	5
2. Ринкові переваги (наявність аналогів)	2	3	3
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	2	2	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	3	4	5
12. Практична здійсненність (розробка документів)	4	5	4
Сума балів	$СБ_1$=39	$СБ_2$=44	$СБ_3$=45
Середньоарифметична сума балів $СБ_с$	42,7		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 **[Козловський, Лесько, Кавецький]**.

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Система автоматизованого керування безпекою застосунків у хмарному середовищі» становить 43 бали, що, відповідно до таблиці 4.3 рівень комерційного потенціалу розробки високий, що свідчить про комерційну важливість проведення даних досліджень.

Магістерська кваліфікаційна робота «Система автоматизованого керування безпекою застосунків у хмарному середовищі» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок, тобто при цьому відбувається комерціалізація науково-технічної розробки. Цей напрямок є для нас пріоритетним, оскільки результатами розробки можуть користуватися не тільки самі розробники, а й інші споживачі, отримуючи при цьому суттєвий економічний ефект.

4.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою **[Козловський, Лесько, Кавецький]**:

$$q_i = \frac{P_i}{P_{базі}}. \qquad (4.1)$$

де q_i – одиничний параметричний індекс, розрахований за i-м параметром;

P_i – значення i-го параметра виробу;

$P_{базі}$ – аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 4.4.

Таблиця 4.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
мультиплатформність, шт	1	3	3	40%
складність вивчення, год	32	16	2	10%
складність застосування, стр. коду	50	20	2,5	10%
інтегрованість, бал 1-5	3	5	1,7	20%
розширюваність, бал 1-5	2	5	2,5	20%

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою **[Козловський, Лесько, Кавецький]**:

$$I_{НП} = \prod_{i=1}^{n} q_i \,, \tag{4.2}$$

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами;

q_i – одиничний (частинний) показник за i-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра **[Козловський, Лесько, Кавецький]**:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \ , \qquad (4.3)$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{mn} = 3 \cdot 0{,}4 + 2 \cdot 0{,}1 + 2{,}5 \cdot 0{,}1 + 1{,}7 \cdot 0{,}2 + 2{,}5 \cdot 0{,}2 = 2{,}49.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою **[Козловський, Лесько, Кавецький]**:

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою **[Козловський, Лесько, Кавецький]:**

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \qquad (4.5)$$

$$K_{ІНТ} = 1 \cdot 2{,}49 / 0{,}80 = 4{,}98.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Система автоматизованого керування безпекою застосунків у хмарному

середовищі», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою **[Козловський, Лесько, Кавецький]**:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 15000 \cdot 40 / 21 = 3409 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту розробки та	15000	681,8	5	3409

дослідження інформаційної технології управління проектами				
Інженер-програміст 1-ї категорії	18000	818,2	35	28636
Всього				32045

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Система автоматизованого керування безпекою застосунків у хмарному середовищі» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (4.7)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{М_М \cdot К_i \cdot К_с}{Т_р \cdot t_{зм}}, \qquad (4.8)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6500 грн;

$К_i$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) **[Козловський, Лесько, Кавецький]**;

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 21 дн;

$t_{зм}$ – тривалість зміни, год.

C_1 = 6700,00 · · 1,65 / (21 · 8) = 65,8 грн.
$З_{р1}$ = 65,8 · 2 = 131,6 грн.

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1. Підготовка робочого місця інженера-розробника ПЗ	2	1	65,8	131,6
2. Інсталяція програмного забезпечення середовищ моделювання та розробки	2	3	88,8	177,7
3.Розробка програмної архітектури та алгоритмів	4	5	111,9	447,5
4. Написання програмного коду модулів	6	3	88,8	533,0
5. Програмне тестування дослідного зразка	4	2	59,8	239,3
Всього				1529,0

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (32045+ 1529,0) · 11 / 100% = 3693,19 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Система автоматизованого керування безпекою застосунків у хмарному середовищі» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 1500,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою **[Козловський, Лесько, Кавецький]**:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.21)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (1 \cdot 500 + 1500 \cdot 2000) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) = 683391{,}41$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (1 \cdot 500 + 1500 \cdot (2000 + 1500)) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) = 1196285{,}5$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (1 \cdot 500 + 1500 \cdot (2000 + 1500 + 1000)) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) = 1537938{,}5$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.22)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 18\%$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 683391{,}41 / (1+0{,}18)^1 + 1196285{,}5 / (1+0{,}18)^2 + 1537938{,}5 / (1+0{,}18)^3 =$$
$$= 2294396{,}96 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \qquad (4.23)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 354480,97 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 354480{,}97 = 708961{,}94$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.24}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 2294396,96 грн;

PV – теперішня вартість початкових інвестицій, 708961,94 грн.

$E_{абс} = ПП - PV$ = 2294396,96 - 708961,94 = 1585435,01 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.25}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+1585435{,}01 / 708961{,}94)^{1/3} - 1 = 0{,}76.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.26)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,76 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (4.27)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,76 = 1,3 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Система автоматизованого керування безпекою застосунків у хмарному середовищі» становить 43 бали, що, свідчить про комерційну

важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 4,98 рази.

Також термін окупності становить 1,3 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія моніторингу безпеки даних програмного забезпечення».

ВИСНОВКИ

Магістерська робота представляє собою значний внесок у розвиток моніторингу застосунків у хмарному середовищі, зокрема в контексті їхньої безпеки та ефективності. Основний акцент на аспектах безпеки та ефективності моніторингу застосунків в хмарному середовищі є відзначеною особливістю цього дослідження.

Проведений глибокий аналіз сучасного стану та перспектив розвитку автоматизованих систем управління моніторингом в хмарних обчисленнях виявив необхідність комплексного підходу, який враховує технічні, організаційні та правові аспекти цієї проблематики. Це стає ключовою передумовою для подальшого успішного розвитку та вдосконалення систем моніторингу у хмарних обчисленнях.

При розробці автоматизованої підсистеми управління моніторингом, велика увага приділялася детальному опису архітектури системи, компонентів, алгоритмів взаємодії та методів автоматизації процесів управління безпекою. Важливість цього підходу визначається гнучкістю та масштабованістю системи, що стає критично важливими характеристиками у змінному хмарному середовищі.

Отримані результати аналізу ефективності підсистеми моніторингу підтверджують її здатність ефективно виявляти та реагувати на різноманітні загрози безпеки, що забезпечує високий рівень захисту додатків у хмарному середовищі. З економічного погляду робота підкреслює економічну доцільність розробленої системи, особливо враховуючи ростучу потребу в ефективних інструментах моніторингу в хмарних середовищах.

Ця робота не лише відкриває нові можливості для подальшого розвитку технологій управління моніторингом хмарних додатків, але і стає основою для розробки більш надійних та ефективних систем. Такий підхід забезпечить кращий контроль та безпеку даних у хмарному середовищі, що стає додатковим стимулом для подальшого розвитку цього напрямку. У кінцевому результаті, робота має не тільки технічне, але й важливе практичне значення, сприяючи

підвищенню рівня моніторингу та безпеки в цифровому світі. Все це робить магістерську роботу важливим кроком у розвитку сфери хмарних обчислень.

.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Державна служба спеціального зв’язку та захисту інформації України. URL: https://cip.gov.ua/ua/news (дата звернення: 07.06.2022).

2. Essential Cyber Security Handbook In Ukrainian URL: https://www.google.com.ua/books/edition/Essential_Cyber_Security_Handbook_In_Ukr/WMBTDwAAQBAJ?hl=ru&gbpv=0 (дата звернення: 11.06.2023)

3. Cloud Technologies: An Overview of Cloud Computing Technologies URL: https://www.google.com.ua/books/edition/Cloud_Technologies/1w0jEAAAQBAJ?hl=ru&gbpv=0 (дата звернення 11.06.2023)

4. Jesus Montesa, Alberto Sánchez, Bunjamin Memishic, Maria S. Perez, Gabriel Antoniu, “GMonE: A complete approach to cloud monitoring”, Vol 29, Issue 8, Future Generation Computer Systems, Elsevier, 2013, 2016.

5. Sander, William V., and Christopher C. J. Fong. *Intrusion Detection Systems*. CRC Press, 2014.

6. Gupta, Brij B., and Srivathsan Srinivasagopalan. *Handbook of Research on Intrusion Detection Systems*. Information Science Reference, 2020..

7. Dobrescu, Radu, and Florin Ionescu. *Large Scale Networks Modeling and Simulation*. CRC Press, 2010..

8. Majumdar, Suryadipta, Taous Madi, and Yushun Wang. *Cloud Security Auditing*. CRC Press, 2011.

9. Halpert, Ben. *Auditing Cloud Computing: A Security and Privacy Guide*. Wiley, 2011..

10. Hugos, Michael H., and Derek Hulitzky. *Business in the Cloud: What Every Business Needs to Know About Cloud Computing*. Wiley & Sons, 2010.

11. Крис Скиннер / Людина цифрова. Четверта революція в історії людства, яка торкнеться кожного.

12. Dhivakar, Ashwin. *A Multi-level Security in Cloud Computing: Image Sequencing and RSA Algorithm*. Anchor Publishing, 2014.

13. Chang, Victor, Robert John Walters, and Gary Wills. *Delivery and Adoption of Cloud Computing Services in Contemporary Organizations*. IGI Global, 2015.

14. Albert, Anthony. *Mastering AWS Security: Create and Maintain a Secure Cloud Ecosystem*. Packt Publishing, 2018.

15. Mukherjee, Adrin. *AWS All-in-one Security Guide: Design, Build, Monitor, and Manage a Fortified Application Ecosystem on AWS*. Packt Publishing, 2016.

16. Baron, Joe, Hisham Baz, Tim Bixler, Biff Gaut, Kevin E. Kelly, Sean Senior, and John Stamper. *AWS Certified Solutions Architect Official Study Guide: Associate Exam*. Sybex, 2016.

17. Ryan, M., & Lucifredi, F. (2018). AWS System Administration: Best Practices for Sysadmins in the Amazon Cloud. Manning Publications.

18. Lakhera, Prashant. AWS for System Administrators: Build, Automate, and Manage Your Infrastructure on the Most Popular Cloud Platform – AWS. Packt Publishing, 2021.

19. Milenkovic, M. (2020). Internet of Things: Concepts and System Design. Springer.

20. Armstrong, J. (2020). Migrating to AWS: A Manager's Guide: How to Foster Agility, Reduce Costs, and Bring a Competitive Edge to Your Business. O'Reilly Media.

21. Tulloch, M., et al. (2012). Introducing Microsoft System Center 2012 R2. Microsoft Press.

22. Martin, T. (2022). Designing Secure IoT Devices with the Arm Platform Security Architecture and Cortex-M33. Elsevier.

23. The NIST Definition of Cloud Computing, NIST Special Publication 800-145, 2011.

24. Guidelines on Security and Privacy in Public Cloud Computing , NIST SP800-144, 2011.

25. Cloud Computing Synopsis and Recommendations DRAFT, NIST, 2011

26. Security Guidance for Critical Areas of Focus in Cloud Computing, Version 3.0. Technical report, Cloud Security Alliance, 2011. URL: http://www.cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf

27. D. Catteddu and G. Hogben. Cloud Computing Security Risk Assessment. Technical report, European

28. D. Catteddu and G. Hogben. Cloud Computing Information Assurance Framework. Technical report, European Network and Information Security Agency, November 2009.

29. Cloud-computing-information-assurance-framework Wayne Jansen, Timothy Grance Guidelines on Security

30. AWS Certified Solutions Architect Official Study Guide: Associate Exam (Aws Certified Solutions Architect Official: Associate Exam) – Joe Baron.

31. AWS Certified Solutions Architect Associate Training Notes 2019 – Neal Davis.

32. Amazon Web Services in Action - Andreas Wittig

33. Learn Amazon Web Services in a Month of Lunches - David Clinton

34. Automating Security in the Cloud: Modernizing Governance through Security Design 1st Edition - Tim Sandage

35. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

91

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Система автоматизованого керування безпекою застосунків у хмарному середовищі. Частина 1. Підсистема моніторингу застосунків.

Автор роботи: Черновол Борис Віталійович
Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 98,37 %. Схожість – 1,63 %.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Валентина КАПЛУН

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Борис ЧЕРНОВОЛ

Керівник роботи _____ (підпис) Галина ШЕЛЕПАЛО

Додаток Б Лілстинг програми.

```
from __future__ import print_function
import os, boto3, json, base64, argparse, sys, socket, uuid
import urllib.request, urllib.parse
import sys
import logging
from datetime import datetime
from datetime import timedelta
from datetime import date

https_prefix = "https://"
cloudwatch_url = "https://console.aws.amazon.com/cloudwatch/home?region="
date_format = "%Y-%m-%dT%H:%M:%S.%f%z"
date_format_display = "%Y-%m-%dT%H:%M:%S"
log_level = os.environ["LOG_LEVEL"] if "LOG_LEVEL" in os.environ else "INFO"
correlation_id = str(uuid.uuid4())
information_source_icon = ":information_source:"
unset_text = "NOT_SET"


# Initialise logging
def setup_logging(logger_level):
    """Set the default logger with json output."""
    the_logger = logging.getLogger()
    for old_handler in the_logger.handlers:
        the_logger.removeHandler(old_handler)

    new_handler = logging.StreamHandler(sys.stdout)

    hostname = socket.gethostname()

    json_format = (
        '{   "timestamp":   "%(asctime)s",   "log_level":   "%(levelname)s",
"message": "%(message)s", '
        f'"environment":                "{args.environment}","application":
"{args.application}", '
        f'"module": "%(module)s", "process":"%(process)s", '
        f'"thread": "[%(thread)s]", "hostname": "{hostname}" }}'
    )

    new_handler.setFormatter(logging.Formatter(json_format))
    the_logger.addHandler(new_handler)
```

```
        new_level = logging.getLevelName(logger_level.upper())
        the_logger.setLevel(new_level)

        if the_logger.isEnabledFor(logging.DEBUG):
            # Log everything from boto3
            boto3.set_stream_logger()
            the_logger.debug(f'Using boto3", "version": "{boto3.__version__}')

        return the_logger


    def get_parameters():
        """Define and parse command line args."""
        parser = argparse.ArgumentParser(
            description="Send HTME SQS messages for defined or default topics."
        )

        # Parse command line inputs and set defaults
        parser.add_argument("--aws-profile", default="default")
        parser.add_argument("--aws-region", default="eu-west-2")
        parser.add_argument("--environment",               default=unset_text,
help="Environment value")
        parser.add_argument("--application",               default=unset_text,
help="Application")

        _args = parser.parse_args()

        # Override arguments with environment variables where set
        if "AWS_PROFILE" in os.environ:
            _args.aws_profile = os.environ["AWS_PROFILE"]

        if "AWS_REGION" in os.environ:
            _args.aws_region = os.environ["AWS_REGION"]

        if "ENVIRONMENT" in os.environ:
            _args.environment = os.environ["ENVIRONMENT"]

        if "APPLICATION" in os.environ:
            _args.application = os.environ["APPLICATION"]

        return _args
```

```
    args = get_parameters()
    logger = setup_logging(log_level)


    # Decrypt encrypted URL with KMS
    def decrypt(encrypted_url):
        region = os.environ["AWS_REGION"]
        logger.info(
            f'Decrypting URL", "encrypted_url": "{encrypted_url}", "region":
"{region}", "correlation_id": "{correlation_id}'
        )

        try:
            kms = boto3.client("kms", region_name=region)
            plaintext                                                    =
kms.decrypt(CiphertextBlob=base64.b64decode(encrypted_url))[
                "Plaintext"
            ]
            logger.info(
                f'Decrypted    URL",    "encrypted_url":    "{encrypted_url}",
"plaintext":    "{plaintext}",    "region":    "{region}",    "correlation_id":
"{correlation_id}'
            )
            return plaintext.decode()
        except Exception:
            logging.exception("Failed to decrypt URL with KMS")
            logger.error(
                f'Failed   to   decrypt   URL   with   KMS",   "encrypted_url":
"{encrypted_url}", "region": "{region}", "correlation_id": "{correlation_id}'
            )


    def config_notification(message, region, payload):
        dumped_message = get_escaped_json_string(message)
        logger.info(
            f'Processing     config     notification",     "dumped_message":
{dumped_message}, "region": "{region}", "correlation_id": "{correlation_id}'
        )

        no_emojis = {
            "COMPLIANT": "Compliant",
            "NOT_APPLICABLE": "N/A",
            "NON_COMPLIANT": "Non-Compliant",
```

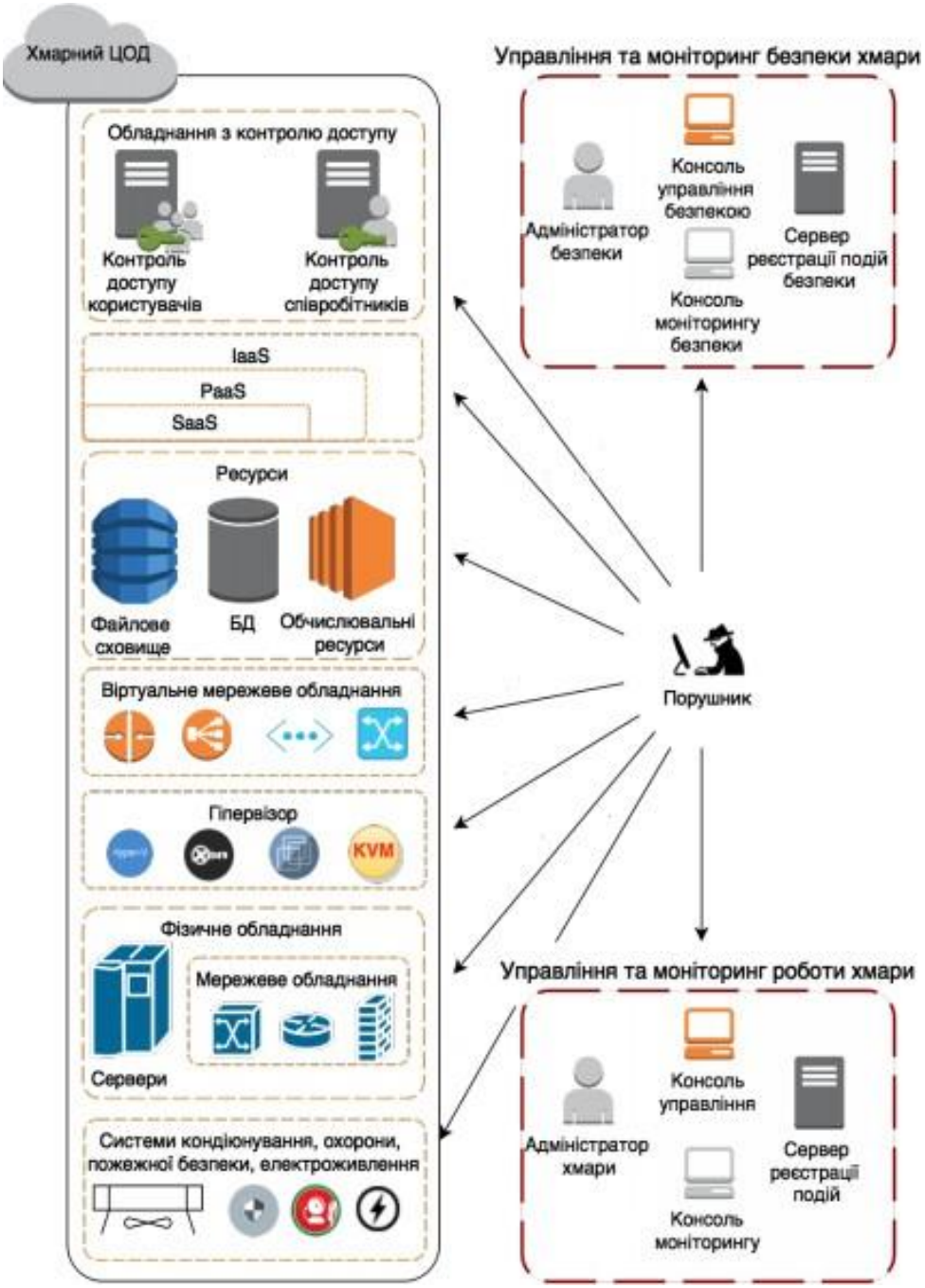
```
        }
        emojis = {
            "COMPLIANT": ":white_check_mark:*Compliant*:white_check_mark:",
            "NOT_APPLICABLE": ":white_check_mark:*N/A*:white_check_mark:",
            "NON_COMPLIANT": ":x:*Non-Compliant*:x:",
        }
        config_rule_name                                                    =
message["newEvaluationResult"]["evaluationResultIdentifier"][
            "evaluationResultQualifier"
        ]["configRuleName"]
        # If an object has just been created, it does not have a
'complianceType'.
        config_old_state = "NOT_APPLICABLE"
        if message["oldEvaluationResult"] != None:
            config_old_state                                                =
message["oldEvaluationResult"]["complianceType"]
        config_new_state = message["newEvaluationResult"]["complianceType"]
        # We constantly get spammed with restricted-ssh periodically going from
NON_COMPLIANT to NOT_APPLICABLE
        if config_new_state == "NOT_APPLICABLE":
            print("DEBUG: new state NOT_APPLICABLE message - exiting.")
            quit()
        elif config_old_state == "NOT_APPLICABLE" and config_new_state ==
"COMPLIANT":
            # TT: inserted this because we are not really interested in the
creation of a new resource whose state is COMPLIANT
            print(
                "DEBUG: old state NOT_APPLICABLE & new state COMPLIANT message
- exiting."
            )
            quit()
        config_url = (
            https_prefix
            + region
            + ".console.aws.amazon.com/config/home#/rules/rule-details/"
            + urllib.parse.quote_plus(
                message["newEvaluationResult"]["evaluationResultIdentifier"][
                    "evaluationResultQualifier"
                ]["configRuleName"]
            )
        )

        payload["blocks"] = [
```

```
            {
                "type": "section",
                "text": {
                    "type": "mrkdwn",
                    "text":  "AWS  Config  Compliance  Change  detected  "  +
config_rule_name,
                },
            },
            {
                "type": "context",
                "elements": [
                    {
                        "type": "mrkdwn",
                        "text": "AWS Config Compliance Rule ["
                        + config_rule_name
                        + "] has changed from ["
                        + no_emojis[config_old_state]
                        + "] to ["
                        + emojis[config_new_state]
                        + "]\nThis script can't tell if everything is compliant
or not. For full details check the AWS Console: "
                        + config_url,
                    }
                ],
            },
        ]
        return payload
```

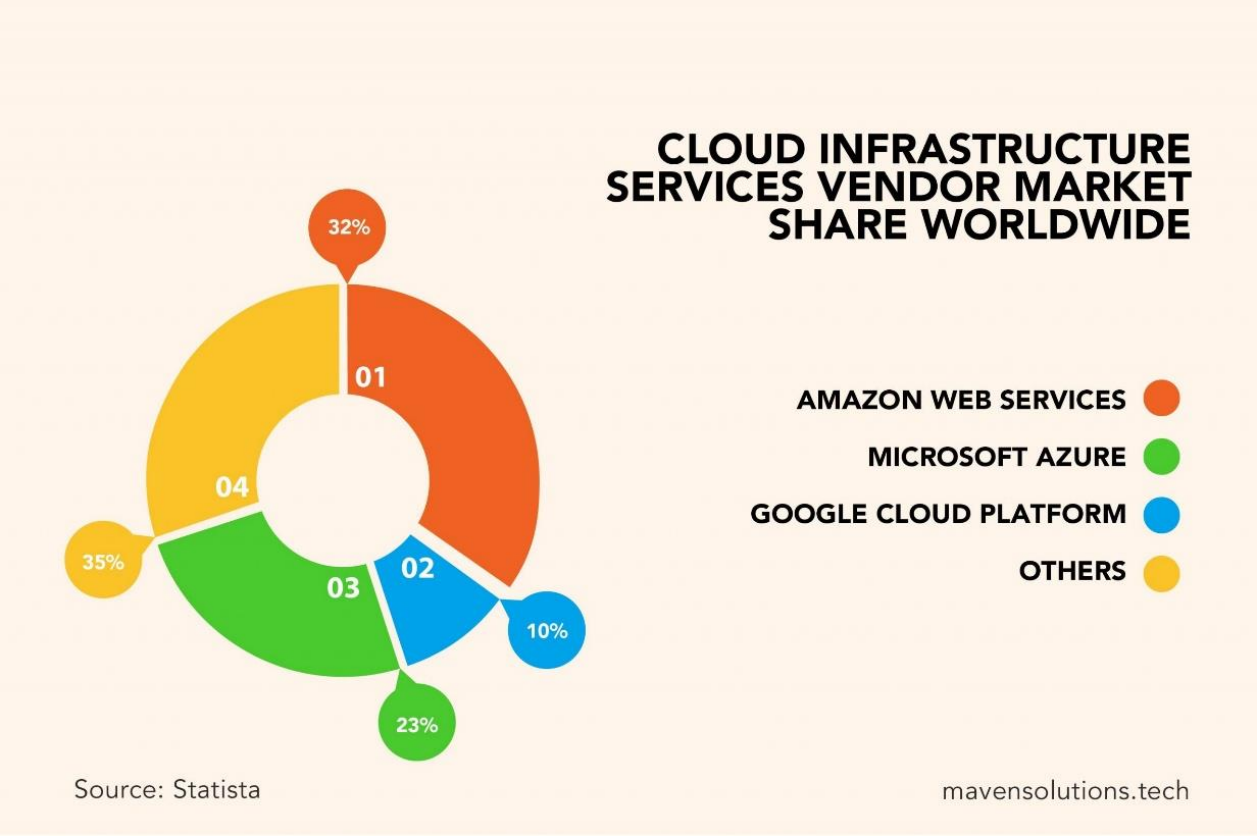
Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА
СИСТЕМА АВТОМАТИЗОВАНОГО КЕРУВАННЯ БЕЗПЕКОЮ ЗАСТОСУНКІВ У ХМАРНОМУ СЕРЕДОВИЩІ. ЧАСТИНА 1. ПІДСИСТЕМА МОНІТОРИНГУ ЗАСТОСУНКІВ

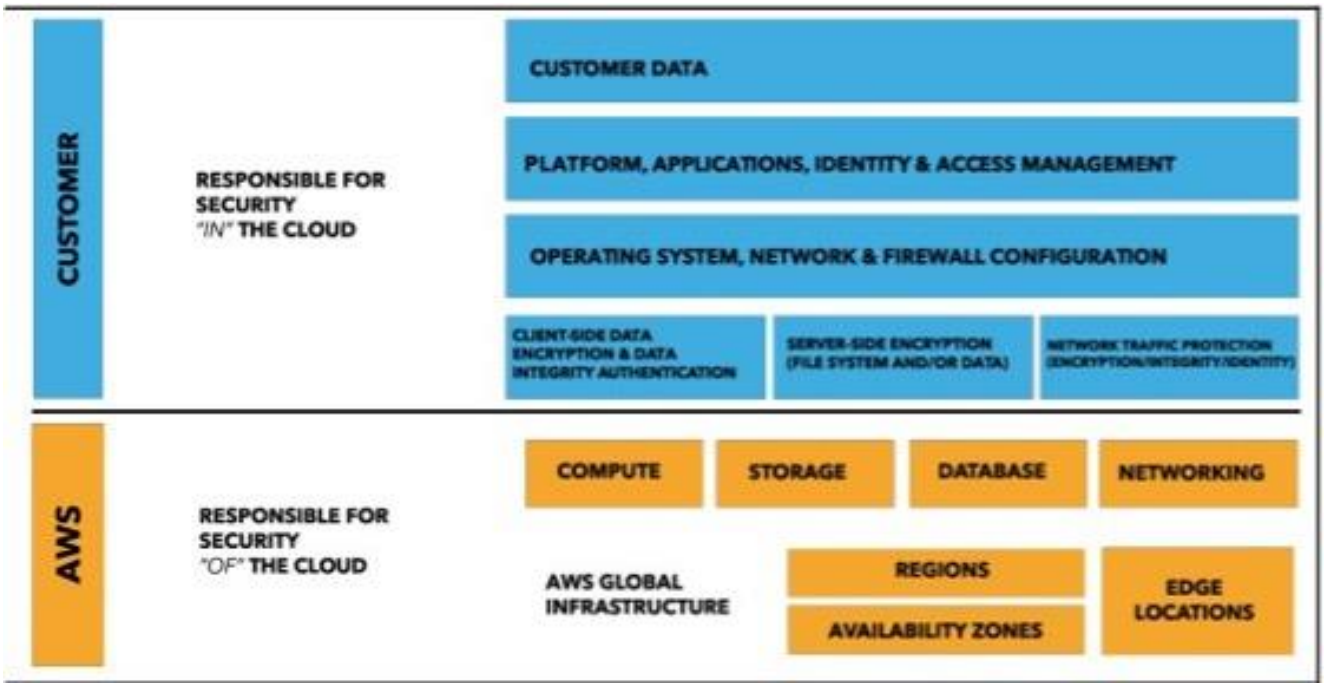
МОДЕЛЬ ЗАГРОЗ ДЛЯ ХМАРНИХ ОБЧИСЛЕНЬ



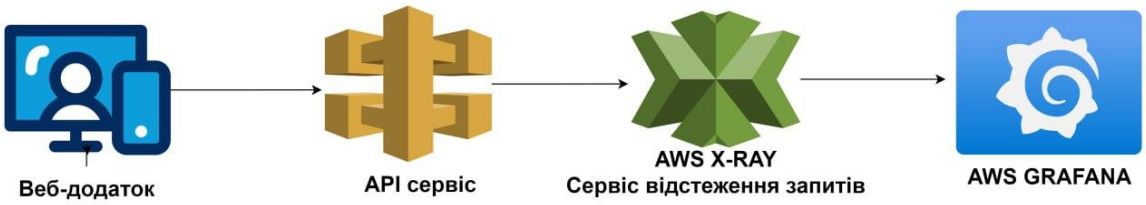
ЛІДЕРИ НА РИНКУ AWS, AZURE ТА GOOGLE CLOUD



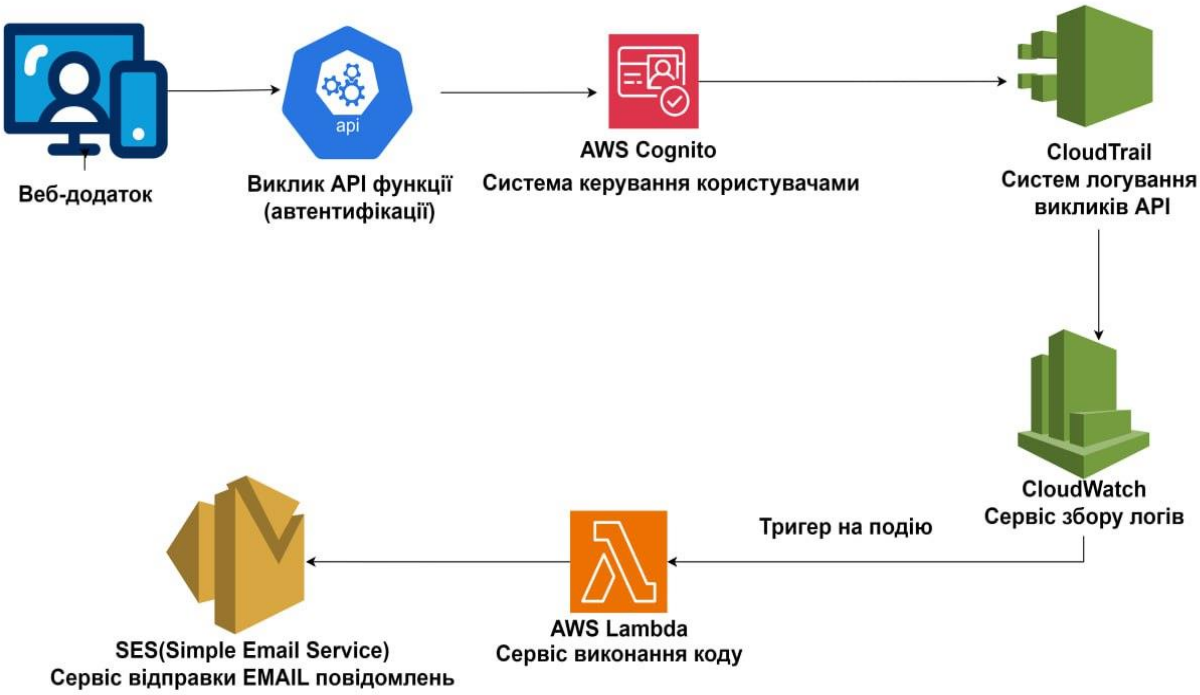
МОДЕЛЬ СПІЛЬНОЇ ВІДПОВІДАЛЬНОСТІ ЗА БЕЗПЕКУ AWS



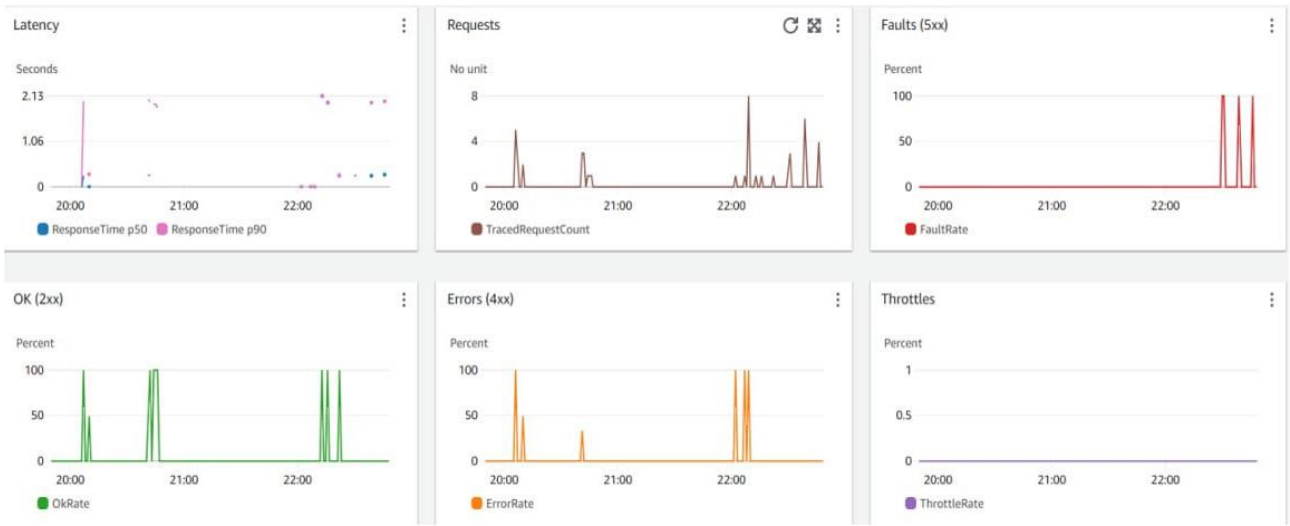
МОДУЛЬ МОНІТОРИНГУ API



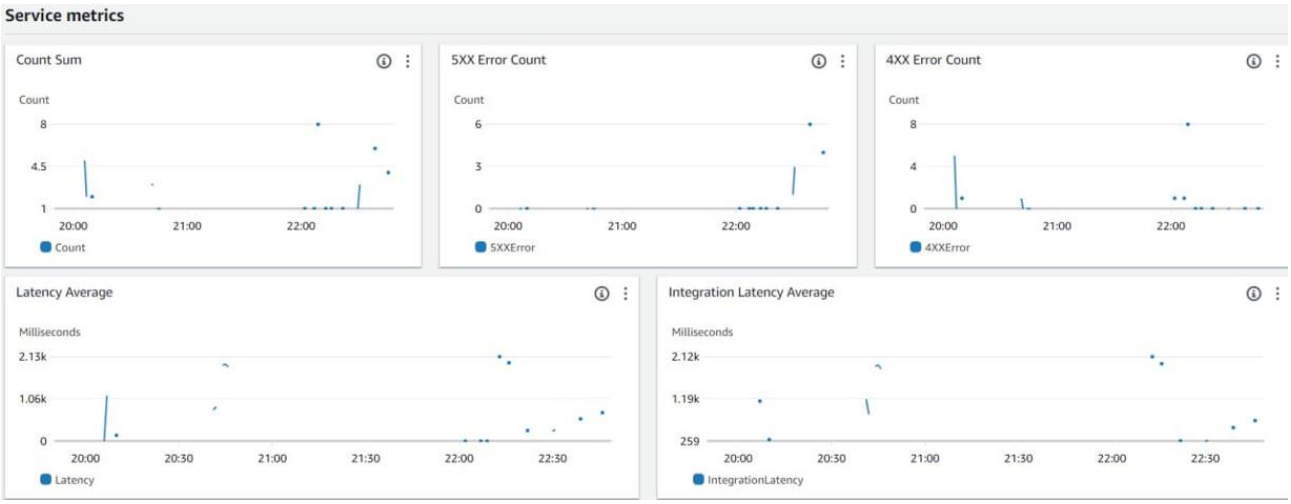
МОДУЛЬ МОНІТОРИНГУ СИСТЕМИ АВТЕНТИФІКАЦІЇ



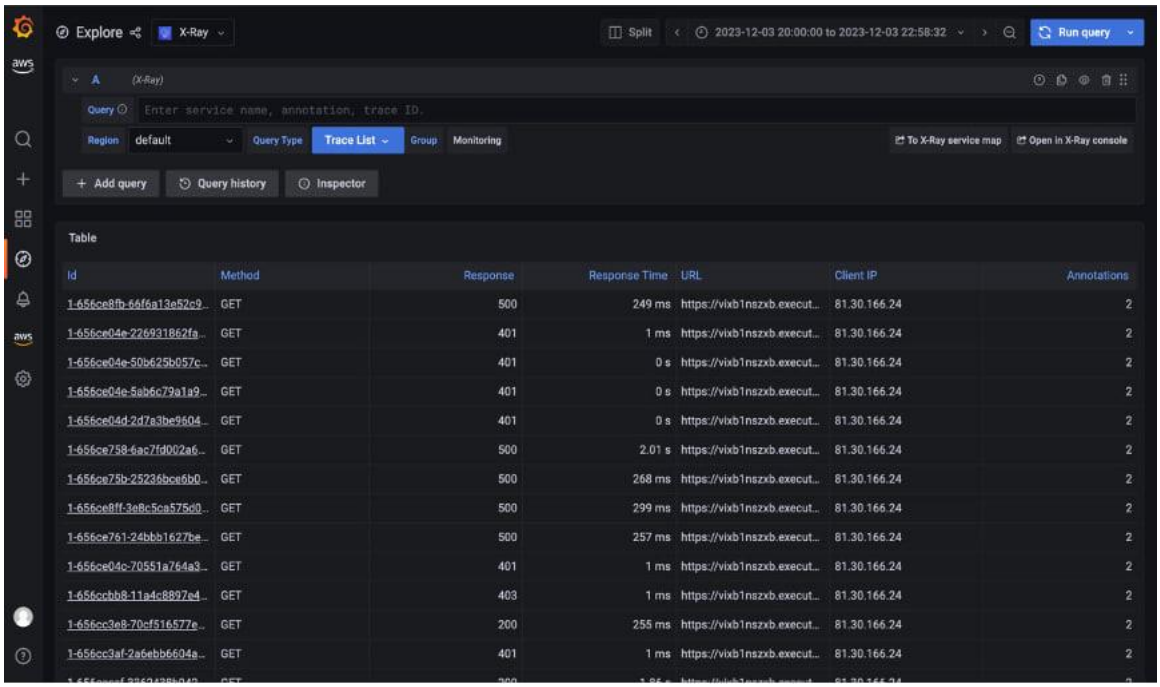
СТАТИСТИКА ВИКЛИКІВ API



МЕТРИКИ СЕРВІСУ



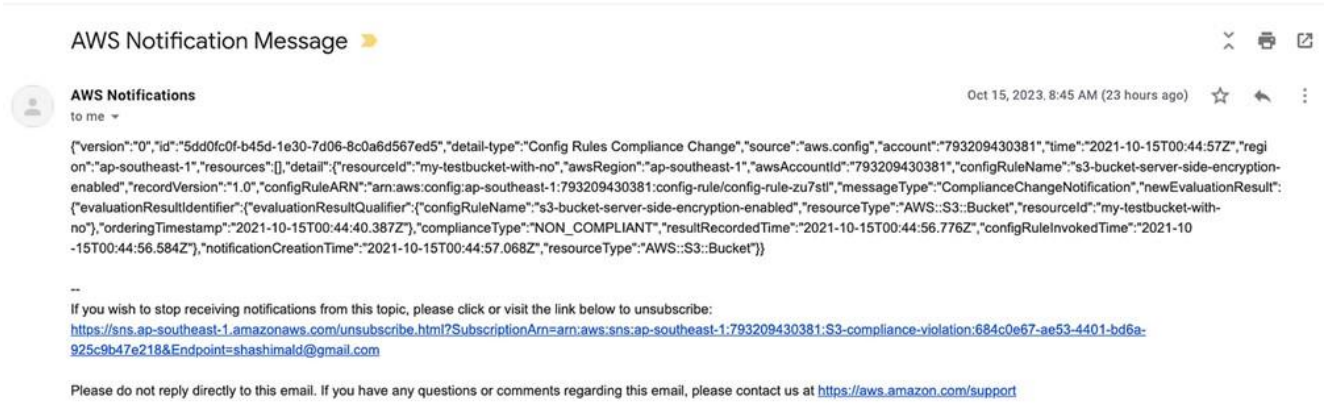
ВІДСТЕЖЕННЯ ВИКЛИКІВ API В GRAFANA



ПОБУДОВАНИЙ ГРАФІК ІЗ РЕЗУЛЬТАТІВ ВИКЛИКУ API

РЕЗУЛЬТАТИ РОБОТИ AWS LAMBDA

РЕЗУЛЬТАТИ РОБОТИ AWS LAMBDA



AWS Notification Message

AWS Notifications
to me

Oct 15, 2023, 8:45 AM (23 hours ago)

{"version":"0","id":"5dd0fc0f-b45d-1e30-7d06-8c0a6d567ed5","detail-type":"Config Rules Compliance Change","source":"aws.config","account":"793209430381","time":"2021-10-15T00:44:57Z","region":"ap-southeast-1","resources":[],"detail":{"resourceId":"my-testbucket-with-no","awsRegion":"ap-southeast-1","awsAccountId":"793209430381","configRuleName":"s3-bucket-server-side-encryption-enabled","recordVersion":"1.0","configRuleARN":"arn:aws:config:ap-southeast-1:793209430381:config-rule/config-rule-zu7stl","messageType":"ComplianceChangeNotification","newEvaluationResult":{"evaluationResultIdentifier":{"evaluationResultQualifier":{"configRuleName":"s3-bucket-server-side-encryption-enabled","resourceType":"AWS::S3::Bucket","resourceId":"my-testbucket-with-no"},"orderingTimestamp":"2021-10-15T00:44:40.387Z"},"complianceType":"NON_COMPLIANT","resultRecordedTime":"2021-10-15T00:44:56.776Z","configRuleInvokedTime":"2021-10-15T00:44:56.584Z"},"notificationCreationTime":"2021-10-15T00:44:57.068Z","resourceType":"AWS::S3::Bucket"}}

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.ap-southeast-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:ap-southeast-1:793209430381:S3-compliance-violation:684c0e67-ae53-4401-bd6a-925c9b47e218&Endpoint=shashimald@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

ПРИКЛАД ВІДПРАВКИ SMS - СПОВІЩЕННЯ