НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«БЕЗПЕКА ТА КОНФІДЕНЦІЙНІСТЬ В АРХІТЕКТУРІ СИСТЕМИ ОРКЕСТРАЦІЇ КЛАСТЕРІВ»

Виконав: здобувач 4 курсу

галузь знань: 12 «»

спеціальність: 125 «Кібербезпека»

Астахов Даниїл Юрійович

Керівник: к.т.н., доцент

Бойко Віктор Дмитрович

Рецензент: доцент, к.ф.-м.н.

Чепурна Олена Євгеніївна

Одеса 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

Факультет **кібербезпеки та інформаційних технологій**

Кафедра **кібербезпеки**

Галузь знань: **12 Інформаційні технології**

Спеціальність: **125 Кібербезпека**

Назва освітньої програми: **Кібербезпека**

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки
професор С.А. Горбаченко
_____ «____ »_____ 20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Астахов Даниїл Юрійович

1. Тема роботи: «Безпека та конфіденційність в архітектурі системи оркестрації кластерів»
Керівник роботи: к.т.н, доцент Бойко Віктор Дмитрович
затверджені наказом НУ ОЮА від «__» _____ 20___ року № _____
2. Строк подання здобувачем роботи «__» _____ 20___ рік
3. Дата видачі завдання«__» _____ 20___ рік

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка

Здобувач _____
(підпис) (прізвище та ініціали)

Керівник роботи _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Безпека та конфіденційність в архітектурі системи оркестрації кластерів» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, освітньою програмою: Кібербезпека, містить 24 рисунків, 21 літературне джерело за переліком посилань. Робота виконана на 42 сторінках загального тексту і 33 сторінках основного тексту.

У роботі розглянуто основні архітектурні компоненти систем оркестрації кластерів та їх вразливості. Проаналізовано потенційні атаки на ці системи, такі як несанкціонований доступ, ескалація привілеїв, атаки на мережевий трафік та інші. Також досліджено механізми захисту, які можуть бути застосовані для зменшення ризиків, включаючи автентифікацію, авторизацію, шифрування даних, моніторинг та аудит.

Робота є актуальною для інженерів організацій, які використовують та підтримують контейнеризовані додатки, а також для дослідників у галузі кібербезпеки. Висновки та рекомендації, представлені в роботі, можуть бути використані для підвищення рівня безпеки в існуючих та майбутніх системах оркестрації кластерів.

ANNOTATION

The qualification work on the Topic "Security and Privacy in Cluster Orchestration System Architecture" for obtaining the first (bachelor's) level of higher education in the specialty 125 Cybersecurity, educational program: Cybersecurity, contains 24 figures, 21 literature sources listed in the references. The work is completed on 42 pages of total text and 33 pages of main text.

The work examines the main architectural components of cluster orchestration systems and their vulnerabilities. Potential attacks on these systems, such as unauthorized access, privilege escalation, network traffic attacks, and others, are analyzed. Additionally, protection mechanisms that can be applied to reduce risks, including authentication, authorization, data encryption, monitoring, and auditing, are explored.

The work is relevant for engineers in organizations that use and maintain containerized applications, as well as for researchers in the field of cybersecurity. The conclusions and recommendations presented in the work can be used to enhance the security level of existing and future cluster orchestration systems.

ЗМІСТ

ВСТУП

На сьогоднішній день все більше та більше великих корпорацій та маленьких підприємств переходять на систему Kubernetes. Це дуже зручна система оркестрації контейнерів в кластері. Вона дозволяє налаштувати повністю готовий сервер включаючи будь-який застосунок написаний будь-якою мовою. Але як і завжди – застосунок, який знаходиться в мережі є головною ціллю зловмисника, не важливо, чи хоче він порушити конфіденційність, продивившись секретні дані, цілісність, шляхом змінення або видалення даних тощо, через те, що стандартні налаштування Kubernetes не є безпечними. Підвищення рівня безпеки є вимогою для захисту від ризиків, пов'язаних з несанкціонованим доступом, а також від витоку конфіденційної інформації.

Метою дослідження є налаштування робочого сервера та імплементація його повного захисту для нього самого та застосунку з базою даних, які є на цьому сервері.

Для цього потрібно вирішити наступні завдання:

1. Виконати огляд літературних джерел по системах оркестрації.
2. Проаналізувати можливі проблеми в безпеці цієї системи.
3. Налаштувати кластер з повної конфігурацією інструментів захисту.

Предметом дослідження є вразливість кластера без налаштування його належним чином. Об’єктом дослідження є сам сервер, налаштований за допомогою оркестрації контейнерів.

Практичним значенням отриманих результатів є буде забезпечення захисту даних користувача та системи в цілому за допомогою налаштування вищезазначених інструментів, що може допомогти забезпечувати безпеку подібного сервера швидше через зазначені в роботі кращі практики захисту та моніторингу інформації. Через налаштування моніторингу за системою для

виявлення незвичних дій із даними застосунка чи з самим кластером; налаштування обмеженого доступу до ресурсів, також буде налаштовано систему секретів на віддаленому ресурсі для їх безпечного зберігання. Результатом роботи

Основні висновки роботи докладались та обговорювались на наукових конференціях та викладені в [1][2].

1 БЕЗПЕКА В СИСТЕМІ ОРКЕСТРАЦІЇ KUBERNETES

1.1. Система оркестрації контейнерів Kubernetes

Головним аспектом запуску та експлуатації системи в мережі інтернет є імплементація, розвиток та постійна підтримка кібербезпеки. Її складність полягає в тому, що зловмисники та ї методи розвиваються одночасно, а то й набагато швидше простих методів захисту, це являється справжнім викликом для спеціалістів в цій сфері. Основні цілі кібербезпеки є запобігання несанкціонованого доступу, крадіжки, пошкодження та переривання інформаційних систем, мереж, пристроїв і даних. Вона має забезпечити довіру, конфіденційність та загальну стабільність у цифровій сфері. Правильно налаштована безпека дає повну гарантію, що усі конфіденційні дані не будуть знищені, змінені та переглянуті зловмисником. Ще один важливим критерієм кібербезпеки є підготовка персоналу, щоб запобігти людським помилкам, які можуть призвести до дуже негативних обставин. Особам, які мають хоча б частковий доступ до секретних даних має бути проведено навчання, щоб могли зрозуміти підозрілу діяльність у вигляді соціальної інженерії, фішінгового сайту чи повідомлення. Цей етап має проводитись регулярно, як мінімум кожен рік, але чи частіше, тим набагато краще. Кібербезпека як явище існує з тих пір, як розробили та запустили мережу під назвою інтернет, який міг поєднувати декілька комп’ютерів в одну екосистему. Це відкрило цілий світ нових можливостей не тільки позитивних, як відкритий доступ до будь-якої інформації, так і негативних у вигляді того, що це дало нові «виклики» для зловмисників в інформаційних системах.

Усі системи, які знаходяться в мережі інтернет будуються за основними стандартами. Їх по світу існує безліч, наприклад, в Сполучених Штатах Америки використовуються стандарти типу NIST, для Європейського Союзу використовують постанови, директиви та рекомендації, які постійно

оновлюються разом із новими тенденціями в інформаційних системах. Основними в Україні є серія стандартів ISO/IEC[3]:

1. ISO/IEC27000:2019 - Інформаційні технології - Методи і засоби забезпечення безпеки - Системи управління інформаційною безпекою - Загальні відомості і словник.
2. ISO/IEC 27001:2013 - Інформаційні технології - Методи захисту - Системи управління інформаційною безпекою – Вимоги.
3. ISO/IEC 27002:2013/COR 2:2015 - Інформаційні технології - Методи захисту - Звід рекомендованих правил для управління інформаційною безпекою.
4. ISO/IEC 27003:2017 - Інформаційні технології - Методи безпеки - Системи управління інформаційною безпекою – Керівництво.
5. ISO/IEC 27004:2016 - Інформаційні технології - Методи безпеки - Управління інформаційною безпекою - Моніторинг, вимір, аналіз і оцінка тощо.

Також існують стандарти, які були створені не під окремі регіони, а для налаштування кібербезпеки по всьому світу, наприклад, ANSI/ISA 62443, COBIT5 (Control Objectives for Information and Related Technologies) / Цілі управління інформаційними та суміжними технологіями, COSOERM2017 тощо. Всі стандарти направлені на підтримку та розвиток кібербезпеки для покращення роботи організацій будь-якого типу та розміру[3].

Система оркестрації контейнерів - це програмне забезпечення або інфраструктура, яка дозволяє керувати і масштабувати кластери серверів або вузлів у великих обчислювальних системах. Ці системи роблять управління кластерами більш ефективним, забезпечуючи високу доступність, автоматичне відновлення, моніторинг і різноманітні інші функції. Сам контейнер – дуже легкий застосунок, навіть, окрема екосистема, в якій знаходиться застосунок з усіма потрібними йому змінними та залежностями.

Контейнеризація не користувалась великою популярністю серед інженерів до 2008 року, коли Linux увімкнув у себе цю функцію до свого ядра, яка, по суті, стала початком цілої епохи розробки нових інструментів такого типу, перше місце в цій гонці, мабуть, і до сьогодні став Docker, який був створений в 2013 і став класикою, коли річ іде за можливість контейнеризації застосунків чи систем[4]. Після того, як контейнеризація почала ставати все більш популярною, стало питання про її оптимізацію. Коли організації почали використовувати всі більше мікросервісів, пов’язаних з контейнерами, то не є проблемою, коли їх 10, але коли таких сервісів створюється більше 100, їх організація на сервері стає майже неможливою, ось тут на допомогу прийшла система оркестрації контейнерів. Це система, ка повністю автоматизує процеси пов’язані з запуском контейнеризованих служб, це включає в себе управління, запуск, підтримку та багато інших процесів пов’язаних з контейнерами[5]. З цим нововведенням, створилося нове направлення в світі IT – DevOps, робота якого дуже тісно пов’язана з автоматизацією процесі усієї інфраструктури в цілому, а від нього і пішов DevSecOps, яке вже є напрямком більш широкого класу більш пов’язаним саме з безпекою інфраструктури та застосунків окремо.

Найпоширенішими системами оркестрації контейнерів є:

1. Kubernetes існує для більш великих компаній з розширеним бюджетом та складнішою інфраструктурою.
2. Docker Swarm існує для організацій поменше, у яких обмежений бюджет та легша інфраструктура.

В цій роботі буде розглянута саме система оркестрації контейнерів в Kubernetes вона є більш надійною з точки зору працездатності та має набагато більше нюансів та інструментів підтримки роботи та навіть захисту. Kubernetes – це платформа з відкритим вихідним кодом для автоматичного управління контейнерними програмами. Часто можете зустріти іншу назву –

k8s. Технологія надає базові інструменти для розгортання, масштабування та підтримки контейнерів[6].

Система оркестрації Kubernetes останні декілька років стає все більш та більш популярною, починаючи з 2022 років майже 30% компаній використовували Kubernetes як основне рішення для розгортання інфраструктури, це складає близько 3 мільйонів кластерів, але після 2024 року це число має всі перспективи вирости до 75%, що буде складати набагато більше 4 мільйонів активних користувачів[7][8]. На даний момент часу 70% великий IT-компаній світу використовують саме його. Попри його зручність використання, правильно не налаштований кластер є дуже вразливим для будь-якого виду атак, тому в 2023 році близько 50% компаній призупинили процес переносу інфраструктури на Kubernetes через його вразливість, саме тому треба правильно навчитись впроваджувати безпеку в цю систему.

1.2. Системи моніторингу для кластера на основі Kubernetes

Одним і, мабуть, головним способом виявлення потенційних проблем в інформаційній системі є впровадження сервісів моніторингу роботи цієї системи. Ефективний моніторинг для кластерів Kubernetes спрощує управління вашими контейнеризованими завданнями, відстежуючи час роботи, використання ресурсів кластера та взаємодію між компонентами кластера це допомагає відстежувати майже всі аспекти, які можуть бути вразливими до атак зловмисника, чи то атака DDOS, який буде впливати на внутрішні ресурси кластера, чи втручання, логи якого будуть записані тим самим моніторингом тощо. Майже усі ці перевірки націлені на відстеження подій, які можна поділити на наступні категорії[2]:

1. Моніторинг доступу – він допомагає дивитися, чи насправді авторизовані користувачі, які мають доступ до одного чи іншого ресурсу.

2. Моніторинг мережі – постійний перегляд трафіку та виділення будь-яких нетипових змін, що може значити втручання зловмисника чи іншу атаку на кластер.
3. Моніторинг системних ресурсів – постійний перегляд на кластері процесів на наявність нетипових поведінок системних ресурсів таких як процесор чи оперативна пам’ять.
4. Виявлення вразливостей – автоматичне виявлення недоліків сервера чи застосунку з точки зору кібербезпеки, це можуть бути атаки типу SQL-ін’єкцій, DDOS, отримання доступу зловмисником тощо.
5. Журналювання подій чи логування – запис усіх подій застосунка та кластера в цілому, починаючи від звичайних перевірок на доступність в мережу інтернет застосунку, закінчуючи записом подій по небажаному втручанню до сервера зловмисників.

Це дозволяє відстежувати майже усі критичні показники сервера, неважливо, якого розміру компанія його підтримує. Це не просто зручність, це життєво важливий сервіс для виявлення проблем з безпекою, стабільності та ефективності, це може бути критичним критерієм через те, що в час падіння сервера логи можуть не зберігатися, а це може призвести до того, що зловмисник пройде на кластер непоміченим. Існують безліч ситуацій, коли навіть невеликі відхилення можуть вказувати на потенційні проблеми, які, якщо їх не виявити вчасно, можуть призвести до серйозних негараздів.

Окрім самостійного налаштування моніторингу по критеріям специфікації сервера та застосунку варто користуватися кращими практиками, основними є[9]:

1. Встановлення та перегляд базових метрик
2. Налаштування комплексних сповіщень
3. Організація та пріоритезація сповіщень

Справедливо буде сказати, що 100% компаній використовують хоча б один інструмент моніторингу. Важливо зазначити, що моніторинг не лише стежить за встановленими політиками безпеки, але й допомагає активно їх оновлювати відповідно до змін у вимогах законодавства та потреб організації. Це важливо, оскільки кіберзагрози постійно еволюціонують, і важливо залишатися на кроці з останніми тенденціями в цій сфері. Постійний моніторинг дозволяє оперативно реагувати на можливі атаки або інші загрози, забезпечуючи швидке виявлення, аналіз та відповідь на події. Це допомагає зменшити час відновлення та мінімізувати вплив можливих атак. Крім того, моніторинг допомагає підтримувати стійкість інфраструктури. Шляхом аналізу показників роботи системи можна виявити потенційні проблеми та уникнути втрати продуктивності або навіть відмови системи.

Постійний моніторинг є практикою кібербезпеки, яка включає неперервний нагляд та аналіз IT-інфраструктури, систем та додатків організації для виявлення потенційних кіберзагроз та вразливостей. В ньому є безліч переваги, але основними з них є[10]:

1. Раннє виявлення загрози – постійний моніторинг дозволяє виявляти загрози до того, як вона станеться, щоб запобігти загострення можливого інциденту.
2. Проактивна реакція – цей тип моніторингу дозволяє бачити можливі атаки у реальному часі, що дозволяє набагато швидше реагувати на інциденти та атаки, які можуть призвести до негативних наслідків.
3. Ефективніше управління ризиками – допомагає точніше виявити тип загрози для пріоритезації дій.
4. Безперервна відповідність – допомагає організаціям притримувати свої політики безпеки відповідно до стандартів безпеки в законодавстві, наприклад стандарти ISO/IEC 27001:2022, NIST 800-53 тощо

5. Покращене реагування на інциденти – дозволяє більш детально описувати факт інциденту, його причини та першоджерело, що результаті допомагає не допускати це знову та отримати масштаб завданої шкоди.
6. Покращена видимість – покращується прозорість трафіку в інфраструктурі сервера компанії, для виявлення потенційний загроз або підозрілу поведінку мережі організації.
7. Усвідомлене прийняття рішень – в кінцевому підсумку може допомогти організаціям перейти від управління ризиками, що ґрунтується на виконанні вимог, до управління ризиками, що ґрунтується на даних.

Було розглянуто літературні джерела та теоретичні відомості стосовно систем оркестрації. З декількох варіантів було обрано саме Kubernetes, так як зараз кластери на основі саме цієї системи є більш стабільними та гнучкими, порівняно з іншими рішеннями. Також була визначена важливість його доналаштування разом з імплементацією моніторингу.

2 АНАЛІЗ РОБОТИ ТА БЕЗПЕКИ СЕРВЕРУ НА ОСНОВІ KUBERNETES

2.1. Аналіз системи оркестрації контейнерів Kubernetes на вразливість

Kubernetes представляє собою дуже зручну та корисну систему оркестрацією контейнерів, її основними перевагами є[11]:

1. Автоматизовані випуски та відкати – це дозволяє запустити нову версію чи відкотитися на стару без перерви у роботі застосунка.
2. Виявлення служб та балансування навантаження – це означає, що кожний найменший компонент системи(под) має свою IP-адресу для підключення до інших внутрішніх застосунків чи «виходу» в інтернет та сам Kubernetes вміє управляти трафіком, який заходить на сервер та розподіляти його по потрібних сервісах.
3. Оркестрація зберігання – автоматичне підключення сховища даних, це може бути чи локальні ресурси, чи віддалені, наприклад EBS або EFS для Amazon Web Services.
4. Автовідновлення – автоматично перезапускає контейнер, який не зміг запуститися та перекидає поди на нову ноду, якщо стара була видалена.
5. Керування секретами та конфігурацією – в системі Kubernetes є спеціальний сервіс для зберігання секретів для внутрішнього середовища застосунка без зайвого їх показу при запуску застосунка в мережу.
6. Автоматична упаковка контейнерів – дозволяє розташувати контейнери відповідно до їх вимог до ресурсів та інших обмежень, при цьому не жертвуючи доступністю.
7. Виконання пакетів – окрім сервісів самого Kubernetes є можливість підтримки сторонніх, наприклад Gitlab CI.
8. Горизонтальне масштабування – масштабування сервера вручну чи автоматично по навантаженню процесора.

9. Підтримка двох стеків IPv4/IPv6.
10. Розроблений для розширення – можна покращувати роботу сервера не змінюючи програмний код.

Система оркестрації Kubernetes може бути розкрита як на фізичному сервері так і на хмарних рішення таких як AWS чи GCP. В останніх є спеціальні сервіси для розкриття такого типу сервера EKS чи AKS для Amazon Web Services та GKE для Google Cloud Platform. Вони дозволяють додати додаткові налаштування для забезпечення безперервної та безпечної експлуатації. Наприклад, в них встановлена IAM (Identity and Access Management (IAM) — це веб-служба, яка допомагає безпечно контролювати доступ до ресурсів. За допомогою IAM можна централізовано керувати дозволами, які контролюють, до яких ресурсів мають доступ користувачі, хто пройшов автентифікацію і має дозволи на використання ресурсів[12]. Security Group, яка пропускає тільки дозволений трафік до застосунка та сервера в цілому тощо. В той час, при розкриванні такої системи на фізичному сервері - треба додавати такі налаштуванні вручну, але привілеями саме такого рішення є набагато дешевша експлуатація, так як в хмарних рішеннях такі сервери можуть коштувати до 1000 доларів на місяць активного використання для середньо-великого типу компаній.

По факту розкриття проекту чи застосунку є декілька етапів, якій застосунок має пройде до і в процесі попадання на кластер у вигляді контейнера з повністю робочою системою. Таких існує три фази, а саме: build phase security(фаза будування), deployment phase security(фаза розгортання), runtime phase security(фаза запуску)[13]:

1. Фаза будування – фаза застосунку, на якій він лише починає завантажувати потрібні для нього пакети даних та починає запускати код.

Головним аспектом захисту на даному етапі є захист чутливих даних по тип паролів, токенів тощо. Тому головними рішеннями для захисту застосунку на цьому етапі – впровадження безпеки в момент будування чи підготувати інформацію потрібну для його роботи. Для першого варіанта потрібно, щоб застосунок вимагав якісь секретні дані, наприклад, токен для отримання даних з віддаленого сервісу. Для другого треб підготувати фундамент, який забезпечить передачу вразливих даних в застосунок вже після його будування, це допоможе повністю вимкнути можливість перехвату секретних даних при передачі в застосунок до його імплементації на сервер.

2. Фаза розгортання – фаза, при якій застосунок вже дійшов до кластера, але тільки починає налаштування інфраструктури під себе на основі конфігурації сервера

Саме на цьому етапі використовується забезпечення безпеки через впровадження систем автентифікації, як було казано раніше – IAM для хмарних рішень та контроль доступу за допомогою системи RBAC. Також, система оркестарції контейнерів Kubernetes дозволяє налаштовувати мережеві політики, що дозволяє фільтрувати вхідний та вихідний трафік, бажано майже повністю обмежувати зв’язок застосунку окрім сервісів, з якими у нього має бути комунікація для правильної роботи.

3. Фаза запуску – фаза, при якій застосунок вже працює, але все ще потребує забезпечення підтримки безпеки.

Безпека застосунку на цьому етапі впроваджується за допомогою сервісів моніторингу та постійним оновленням політик безпеки для кожного контейнера окремо, в залежності від його конфігурації, також потрібно обов’язково завантажувати лише перевірені офіціальні конфігурації для сторонніх ресурсів застосунку, що значно зменшує шанси на отримання шкідливого програмного забезпечення.

2.2. Аналіз та вибір систем моніторингу для кластера на основі Kubernetes

Існує дуже багато різних інструментів для відстеження навантаження, журналювання та візуалізації даних, деякі сервіси роблять це окремо, деякі включають в собі одразу все. Такі сервіси поділяються на два типи: локально налаштовані та сервіси, які працюють віддалено, але на кластері знаходиться агент, який ці сервіси пов’язують з сервером[14].

Найрозповсюдженішими сервісами, які налаштовується локально є Prometheus+Loki+Grafana Stack, який є класичний інструментом моніторингу, та ELK Stack(Elasticsearch, Logstash, Kibana), їх основними мінусами є складній налаштування, але це можна виділити і як плюс, тому що їх можна повністю конфігурувати під будь-яку специфіку кластера та застосунка, їх основна різниця – Prometheus+Loki+Grafana Stack використовується для більш маленьких застосунків, в той час, як ELK Stack використовується для великого об’єму даних. Кожний з цих інструментів розділяється ще на декілька сервісів, наприклад, для збирання метрик використовуються Prometheus або Elasticsearch, для логів – Loki або Logstash, візуалізація реалізовується за допомогою Grafana або Kibana[15].

Також є сервіси, які існують окремо від кластера, але для підключення треба встановити агент, прикладом такого сервісу є Datadog – це швидкоростучий інструмент, який вже встиг собі заробити ім’я через активне використання великими компаніями, такими як Panasonic Corp, eBay, Samsung, Ubisoft тощо. Його головна перевага є в тому, що він в собі включає одразу всі сервіси, окремі перевірки на втручання, постійні «синтетичні» запроси, які можуть перевіряти роботу застосунку і все це в одному місці із зручним інтерфейсом. Також треба враховувати те, що цей сервіс є віддаленим відносно кластера, тому, якщо якимось чином постраждає сервер, то Datadog залишається в повному порядку, це є однозначний плюс, але може і статися таке, що сам інструмент може втратити працездатність, це ніяк не вплине на

роботу кластера, але може порушити роботу відправки логів та метрик, що може призвести до неможливості розуміння виниклої проблеми в застосунку.

Дуже важливим критерієм налаштування моніторингу є встановлення системи повідомлень про інциденти працездатності системи та кібербезпеки. Тому що за законом Всесвіту, такі помилки трапляються у максимально незручний час, наприклад о 3-тій ночі, коли всі сплять, через те і легко можна пропустити таке важливе повідомлення. Тому дуже критично налаштувати таку систему. Є два типи сповіщень[16]:

1. Критичні для пробудження інженера. Такі повідомлення роблять для ситуацій, коли якісь сервіс, пов’язаний з основним сервером, до якого мають доступ користувачі. Для таких сповіщень підключається система дзвінків.
2. Некритичні для загальної інформації. Робляться для ситуацій на тестовому сервері, до якого доступ мають тільки розробники. Для таких сповіщень підключається система повідомлень.

Моніторинг в системі оркестрації включає в собі налаштування дуже великої кількості конфігурацій, з якими можна експериментувати, але і є кращі практики, які вважаються найбільш стабільними та надійними. Ось декілька прикладів таких практик[17]:

1. Моніторинг метрик Kubernetes за допомогою єдиного «вікна» - означає, що не треба налаштовувати декілька інтерфейсів, для слідкування за даними, оскільки з декількома дуже просто пропустити важливе повідомлення про помилку чи втручання неавторизованого користувача.
2. Забезпечення масштабованості систем моніторингу та достатню збереженість даних – це є дуже важливо, тому що перенавантаження системи без можливості розширення її ресурсів може привести до того, що важливе повідомлення просто не відправиться та є дуже

велика ймовірність не зреагувати вчасно на виниклу проблему. Таким самим чином, якщо у системи немає місця, то важливим логам немає куди зберігатися, що також може бути критичним при аналізі виниклої проблеми.

3. Забезпечення генерацію сповіщень та їх доставка відповідному персоналу – кожному розробнику не мають приходити усі можливі повідомлення про проблеми на сервері, тому що реально важливі помилки та проблеми можуть залишитись непоміченими через занадто сильний спам. Тому важливо налаштувати специфічні сповіщення та навіть доступи для конкретного персоналу окремо.
4. Інтегрування системи моніторингу з CI/CD конвеєром – теж є дуже корисною практикою тому що буде можливість за допомогою системи моніторингу слідкувати не тільки за роботою застосунку, а й за етапом його побудови та запуску.

Використання кращих практик не виключає використання своїх налаштувань, які будуть більше підходити саме для конфігурації окремого кластера зі специфічним налаштуванням інфраструктури та застосунку. Це є лише рекомендаціями, які є стандартизованими під кожний існуючий сервер, налаштований не тільки на основі системи оркестрації контейнерів Kubernetes, а і будь-якої іншої конфігурації.

2.3. Вибір додаткових інструментів безпеки для кластера

І все ж таки головною задачею моніторингу сервера залишається виявлення проблем «по факту» їх появи, тому для запобігання атаки на сервер, треба налаштовувати додаткові сервіси захисту. Є безліч сторонніх чи внутрішніх сервісів, які можуть допомогти налаштування безпечне середовище для всієї інфраструктури, але в цій дипломній роботі будуть розглянуті саме надважливі, без яких саме існування сервера на основі системи

оркестрації Kubernetes неможливо, якщо говорити про його експлуатацію для реального проекту:

1. Система RBAC(Role-Based Access Control) – це система контролю за доступами до окремих ресурсів та кластеру в цілому.

Система RBAC для кластерів, які розкриті в хмарних рішеннях вже налаштована при запуску, але під кожне рішення є свої нюанси. Наприклад, для GCP достатньо мати дозвіл через управління користувачами, в той час, як для AWS окрім того, що треба мати доступ до певного ресурсу через управління користувачами, адміністратор кластера має надати дозвіл через внутрішню систему контроля доступу в самому Kubernetes, що є більш незручним, але безпечнішим ніж в хмарному рішенні від Google. Для фізично налаштованих кластерів в стандартних налаштуваннях ця система має бути увімкнена вручну з окремими налаштуваннями доступу під окремого користувача або групу користувачів.

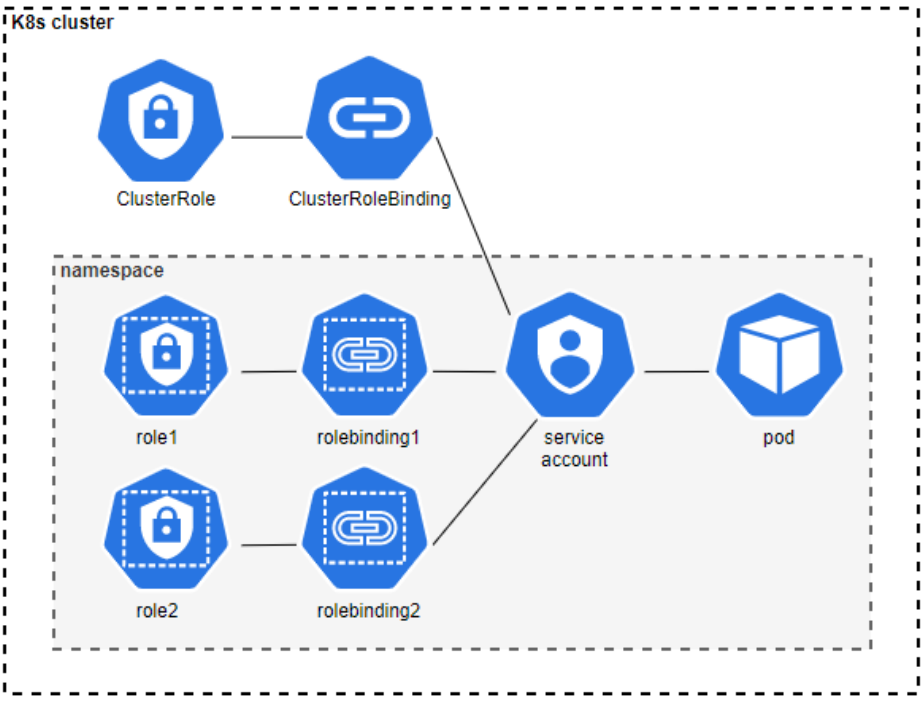


Рисунок 2.1 – схема роботи RBAC в Kubernetes[18]

2. Сховище секретів – віддалене чи локальне сховище, яке включає в собі безпечне зберігання токенів, паролів, назв застосунків тощо, яке не є доступним для звичайного користувача.

Секрети в хмарному середовищі - це пари ключ-значення, які надають конкретному користувачу або групі користувачів авторизацію або автентифікацію для API та сервісів. Секрети використовуються в вихідному коді та конфігураційних файлах, які контролюються за допомогою системи контролю версій[19]. Вони є основним сховищем для будь-яких чутливих даних, єдиним мінусом є те, що вони не налаштовуються автоматично без додаткових інструментів. А налаштовувати такі дані вручну можу бути небезпечним через можливий виток даних через необережність адміністратора тих самих даних.

Сховищем секретів можна вважати застосунок чи веб-сайт в якому зберігаються вразливі дані, потрібні для правильної роботи застосунку, до яких є доступ тільки, якщо це дозволяють права. Як і з системами моніторинг, є сервіси, які встановлюються на самому кластері чи проекті, так і може бути встановлений агент, який пов’язує кластер із віддаленим сервісом. Прикладом останнього є система Doppler, який може зберігати в собі усі потрібні для застосунку чи бази даних секрети. Окрім того, що це є безпечною системою зберігання паролів, вона є і дуже зручною тому що не треба вручну кожний раз впроваджувати новий пароль чи нове посилання напряму в кластер, достатньо лише записати ці дані в потрібне оточення і вони автоматично підтягнуться до кластера. Також існує сервіс, який встановлюється прямо в проект, який на основі секретного ключа шифрує файл, який мітить дані, які звичайному користувачу, а тим більше зловмиснику бачити категорично не можна, цей сервіс називається sops.

3. Лімітований доступ самого застосунку – найбанальніший спосіб захисту застосунку та сервера в цілому є звичайний закритий доступ

до кожного із сервісів, окрім тих, що повинні мати відкритий доступ в інтернет мережу.

Застосунки, які працюють з чутливими даними, які ніхто не має бачити, повинні залишатися існувати в свої локальній мережі кластера, щоб мати змогу доступу лише до потрібних для нього ресурсів, наприклад, для бази даних не треба робити вихід в відкриту мережу, тому що її дані опрацьовуються лише всередині кластера за допомогою іншого застосунка чи сервісу.

4. Налаштування паролів для застосунків, у яких є доступ в мережу – це пункт є продовженням попереднього, таким чином, що є сервіси(наприклад, фронтенд) які не можуть існувати без виходу в мережу інтернет.

Деяким застосункам для їх повноцінної роботи повинен бути відкритий доступ до мережі інтернет, але якщо застосунок, який має відкритий доступ в інтернет повинен бути «невидимий» чи просто захищений від несанкціонованого доступу – на нього обов’язково потрібно поставити звичайну систему автентифікації, це може бути чи звичайний логін та пароль, чи підключення зовнішніх сервісів на прикладі акаунта Google чи Гітлаб.

Було проаналізовано можливі проблеми в безпеці цієї системи. Було обрано систему моніторингу, яка більш підходить саме до конфігурації даного кластера, було обрано додаткові інструменти безпеки, які виконують функції додаткового захисту застосунку та всієї інфраструктури.

3 ПРАКТИЧНЕ НАЛАШТУВАННЯ ТА ЗАХИСТ ІНФРАСРУКТУРИ

3.1. Налаштування кластера на основі системи оркестрації контейнерів Kubernetes

В цій роботі був налаштований кластер на основі системи оркестрації контейнерів Kubernetes на локально інстальованій системі Minikube. Усе адміністрування сервером ведеться за допомогою спеціального інструменту K9S, який є дуже зручним на відміну від користування звичайним терміналом за допомогою команд типу kubectl[20]. Для коректної та повноцінної роботи кластера я мав налаштувати 2 ноди, які використовують 4 ядра процесора та 8гб оперативної пам’яті кожна. Нода – один з багатьох Kubernetes, яка налаштовує реальну машину з потрібними для запуска сервера ресурсами. Для запуску сервера та налаштування всієї інфраструктури потрібно запустити завчасно написаний скрипт, який повністю робить кластер працездатним.

1. Першим етапом цього скрипту є запуск кластера, який буде мати в собі усі потрібні налаштування для проекту (Рисунок 3.1-3.2).

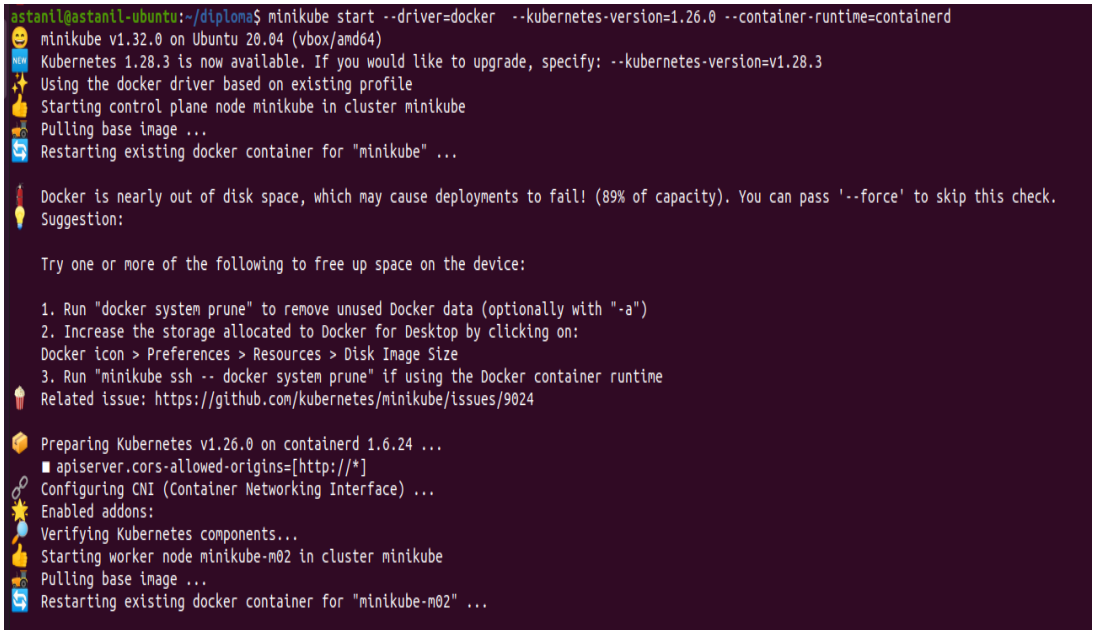


Рисунок 3.1 – запуск кластера на основі системи оркестрації Kubernetes

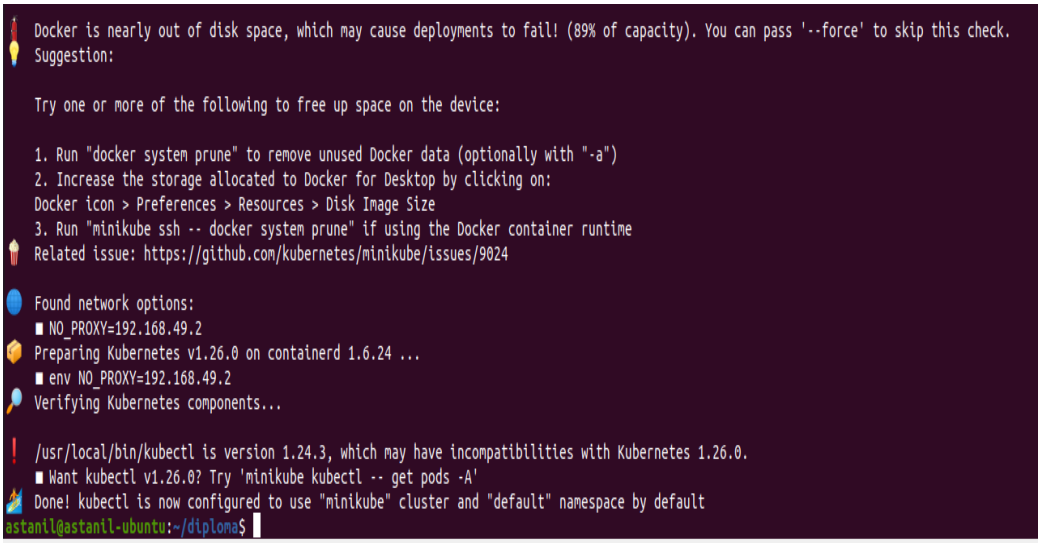


Рисунок 3.2 – продовження запуску кластера на основі системи оркестрації Kubernetes

2. Другим етапом налаштовуються окремі іменні простори для більш легкого управління інфраструктурою під окрему частину кластера (Рисунок 3.3), наприклад, застосунок та база даних мають різні секретні дані, та конфігурації під своїми просторами імен.

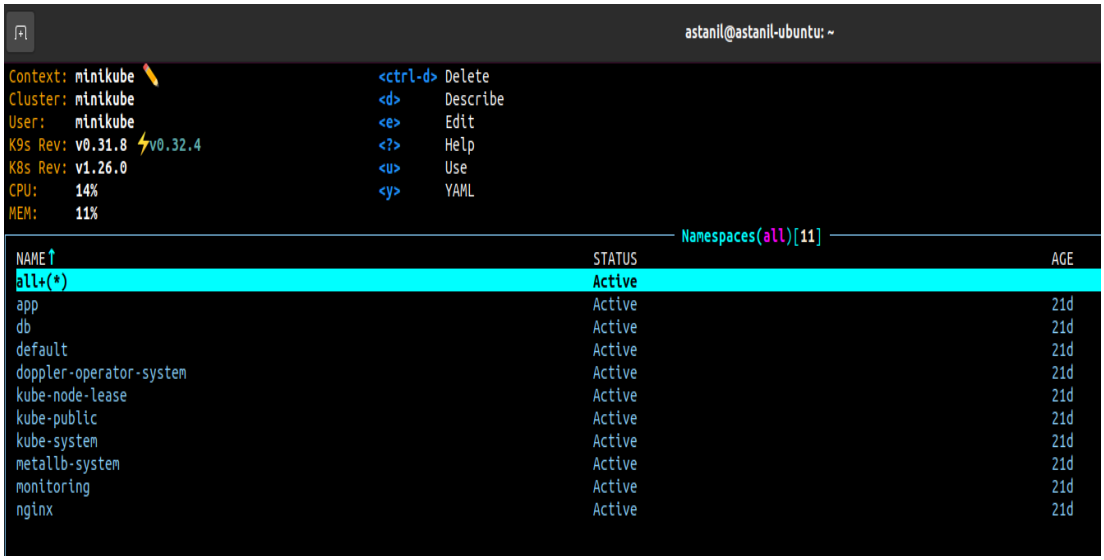


Рисунок 3.3 – Показ простору імен в кластері.

3. Третій етап – будування та переніс даних застосунка до віддаленого сервера за допомогою Docker (Рисунок 3.4), переніс даних можливий тільки авторизації користувача цього інструменту.

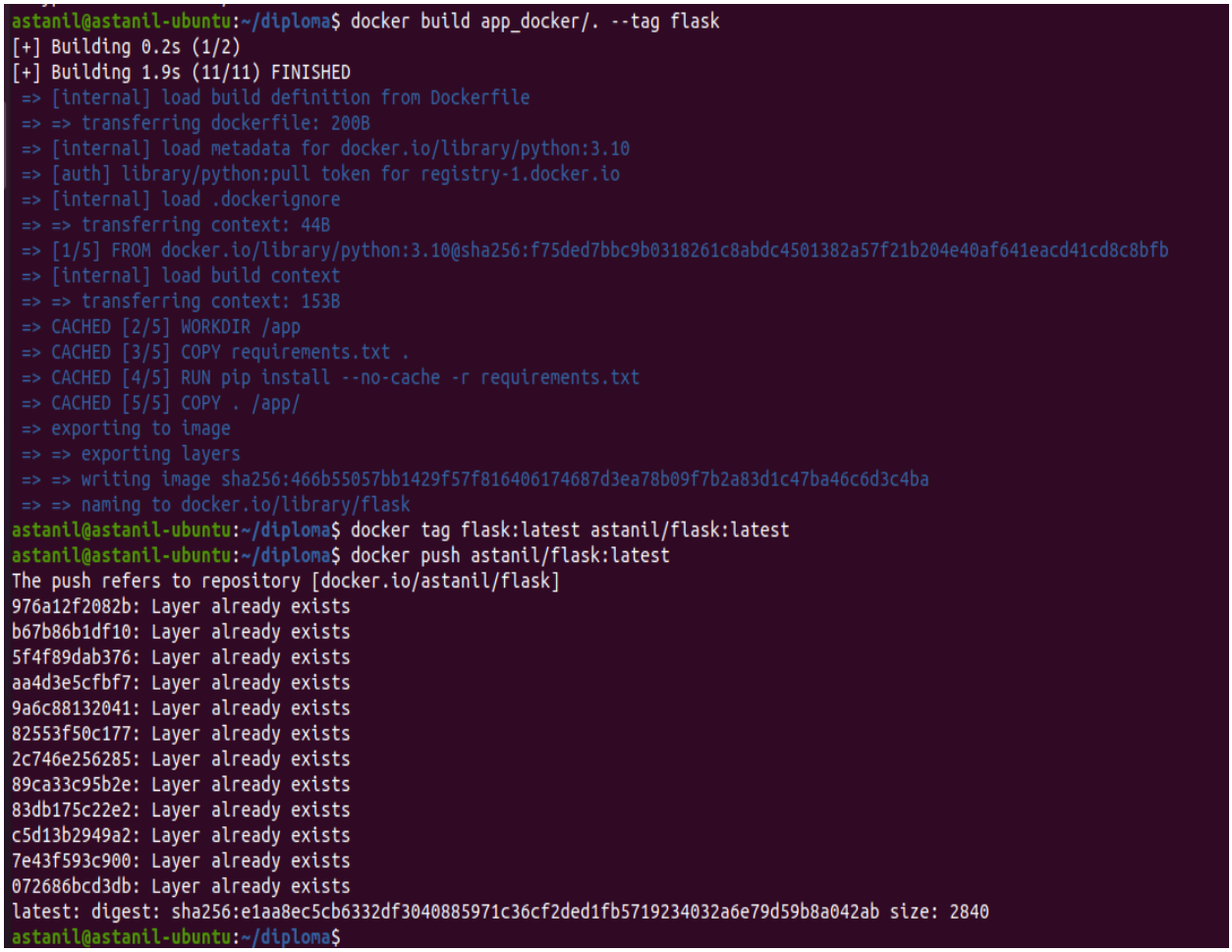
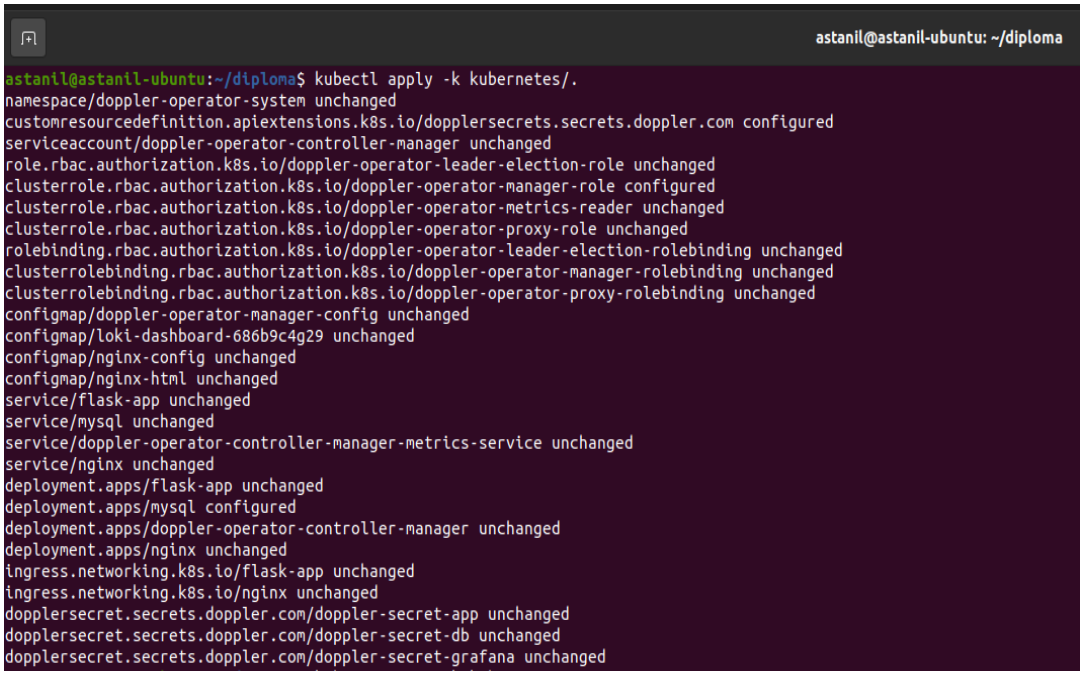


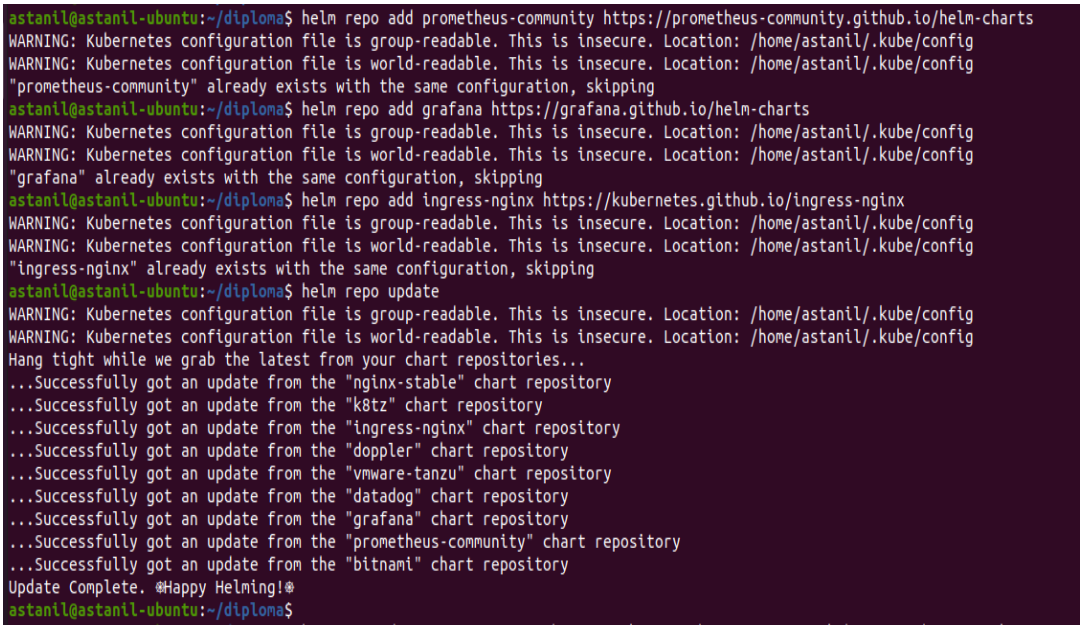
Рисунок 3.4 – будування та переніс даних застосунку на віддалений сервер Docker

Перед четвертим етапом потрібно вручну додати секрети, які містять секретні токени для автоматичного створювання оточення під окремий сервіс на кластері за допомогою Doppler.

4. Четвертий етап скрипту включає в себе запуск усіх застосунків та інструментів, щоб приготувати проект до повноцінної роботи (Рисунок 3.5-3.8). Включає в себе запуск: застосунку(backend та frontend частини), бази даних, сховище секретів Doppler, конфігурації для моніторингу, сам моніторинг та nginx.



Рисунки 3.5 – запуск застосунку та частини інструментів



Рисунки 3.6 – додавання віддалених директорій з інструментами

```
astanil@astanil-ubuntu:~/diploma$ helm upgrade --install promtail grafana/promtail -n monitoring --values kubernetes/monitoring/loki/values-promtail.yaml
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/astanil/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/astanil/.kube/config
Release "promtail" has been upgraded. Happy Helming!
NAME: promtail
LAST DEPLOYED: Sun May 12 18:52:40 2024
NAMESPACE: monitoring
STATUS: deployed
REVISION: 2
TEST SUITE: None
NOTES:
**********************************************************************
 Welcome to Grafana Promtail
 Chart version: 6.15.5
 Promtail version: 2.9.3
**********************************************************************

Verify the application is working by running these commands:
* kubectl --namespace monitoring port-forward daemonset/promtail 3101
* curl http://127.0.0.1:3101/metrics
```

Рисунки 3.7 – приклад запуску інструменту за допомогою helm

```
astanil@astanil-ubuntu:~/diploma$ helm upgrade --install nginx ingress-nginx/ingress-nginx -n nginx --values kubernetes/nginx/controller/values.yaml
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/astanil/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/astanil/.kube/config
Release "nginx" has been upgraded. Happy Helming!
NAME: nginx
LAST DEPLOYED: Sun May 12 18:56:24 2024
NAMESPACE: nginx
STATUS: deployed
REVISION: 5
TEST SUITE: None
NOTES:
The ingress-nginx controller has been installed.
It may take a few minutes for the load balancer IP to be available.
You can watch the status by running 'kubectl get service --namespace nginx nginx-ingress-nginx-controller --output wide --watch'

An example Ingress that makes use of the controller:
  apiVersion: networking.k8s.io/v1
  kind: Ingress
  metadata:
    name: example
    namespace: foo
  spec:
    ingressClassName: nginx
    rules:
      - host: www.example.com
        http:
          paths:
            - pathType: Prefix
              backend:
                service:
                  name: exampleService
                  port:
                    number: 80
              path: /
    # This section is only required if TLS is to be enabled for the Ingress
    tls:
      - hosts:
        - www.example.com
        secretName: example-tls
```

Рисунки 3.8 – запуск Nginx за допомогою helm

5. Останній етап включає в собі команду `minikube tunnel` що замінює функцію LoadBalancer в AWS, що робить можливість «тунелю» до зовнішньої мережі.

```
astanil@astanil-ubuntu:~/diploma$ minikube tunnel
[sudo] password for astanil:
Status:
        machine: minikube
        pid: 249058
        route: 10.96.0.0/12 -> 192.168.49.2
        minikube: Running
        services: [flask-app, nginx, nginx-ingress-nginx-controller]
    errors:
                minikube: no errors
                router: no errors
                loadbalancer emulator: no errors
```

Рисунок 3.9 – запуск доступу кластера до зовнішньої мережі інтернет.

3.2. Приклад роботи застосунки на мові програмуванні Python

Backend застосунку написаний за допомогою мови програмування Python, його робота полягає в записі, видаленні, змінені та виводу даних з бази даних MySQL. В ньому є декілька кінцевих точок:

1. /api для отримання документації з приводу застосунку.
2. /api/students для отримання усіх даних про студентів з бази даних.
3. /api/students/get/<int:id> для отримання даних про конкретного студента за його id.
4. /api/students/add для додавання нового запису в базу даних.
5. /api/students/modify/<int:id> для зміни запису даних по id студента.
6. /api/students/change/<int:id> для нового запису чи його зміни по id.
7. /api/students/delete/<int:id> для видалення запису по його id.
8. /api/heath-check/ok та /api/heath-check/bad для перевірки на працездатність застосунку.

Frontend застосунку зроблений на основі Html з використанням чистого Javascript для отримання запитів з Backend частини застосунку (Рисунки 3.10-3.14).

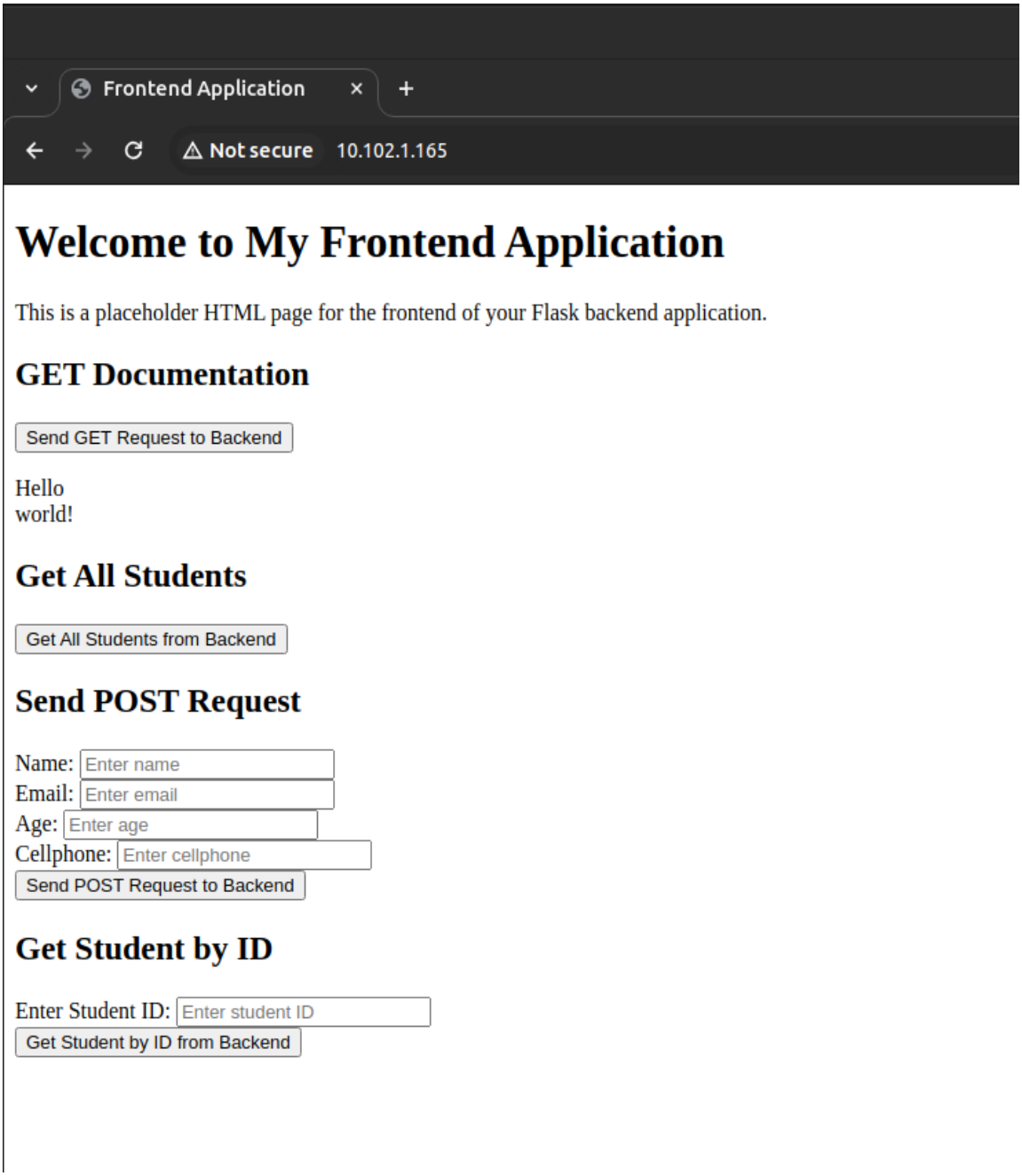


Рисунок 3.10 – вигляд frontend частини застосунку

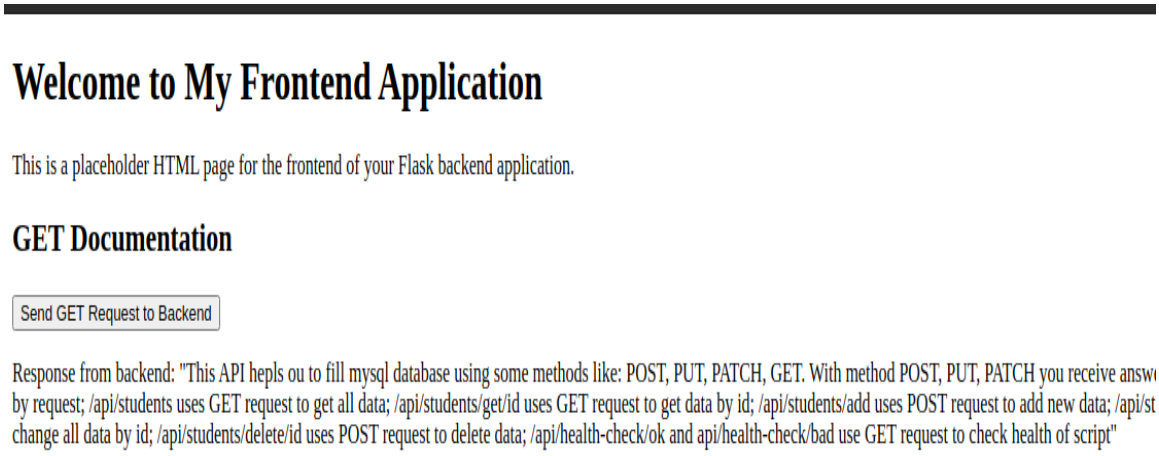


Рисунок 3.11 – робота застосунку при виклику документації

Повний вивід на екран виглядає так: «`Response from backend: "This API hepls ou to fill mysql database using some methods like: POST, PUT, PATCH, GET. With method POST, PUT, PATCH you receive answer code 201 which show you the data you want to add. With GET you receive code 200 which shows you data by request; /api/students uses GET request to get all data; /api/students/get/id uses GET request to get data by id; /api/students/add uses POST request to add new data; /api/students/modify/id uses PATCH request to modify part of data by id; /api/students/change/id uses PUT request to change all data by id; /api/students/delete/id uses POST request to delete data; /api/health-check/ok and api/health-check/bad use GET request to check health of script`»

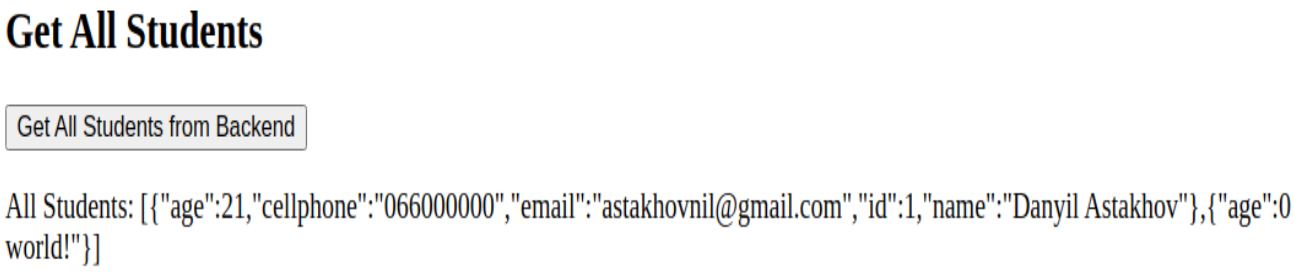


Рисунок 3.12 – робота застосунку при виклику всіх студентів

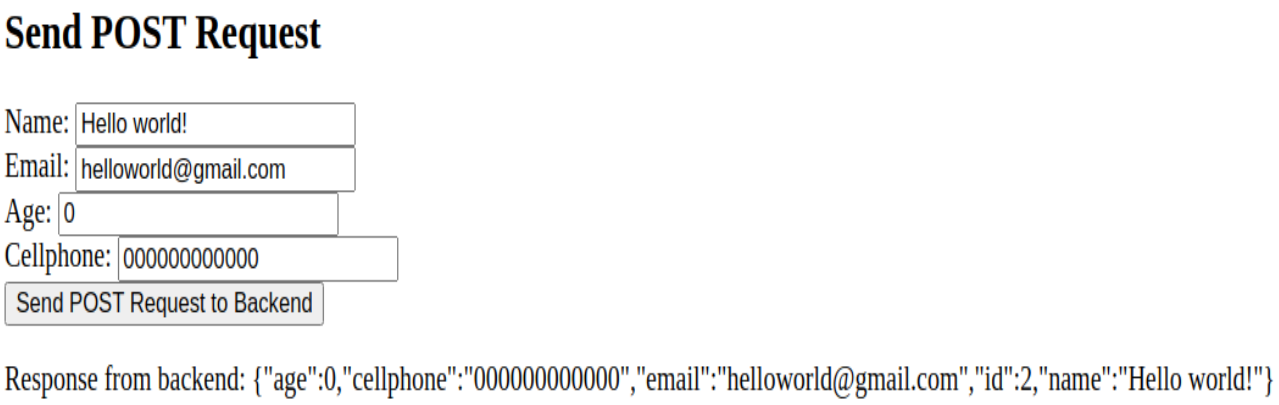


Рисунок 3.13 – робота застосунку при виклику всіх студентів та запису нового

Повний вивід на екран виглядає так: «`All Students: [{"age":21,"cellphone":"066000000","email":"astakhovnil@gmail.com","id":1,"name":"Danyil Astakhov"},{"age":0,"cellphone":"000000000000","email":"helloworld@gmail.com","id":2,"name":"Hello world!"}]`»

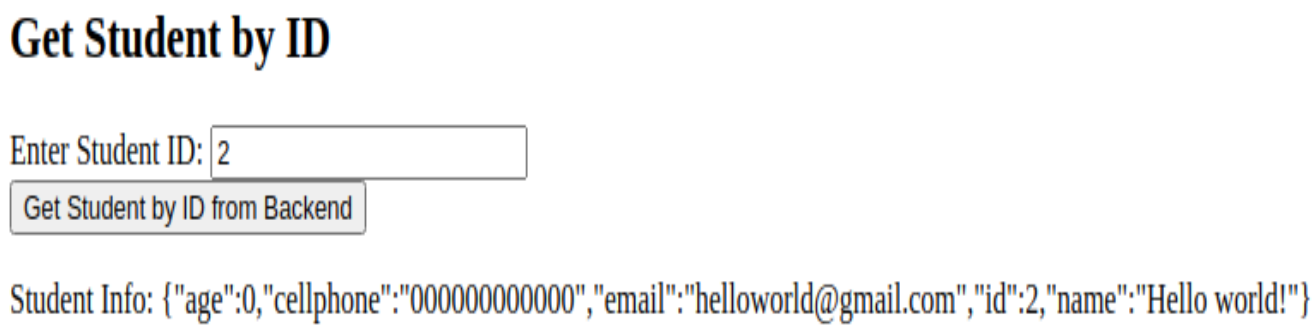


Рисунок 3.14 – робота застосунку при виклику студента по його id

Інші функції по кінцевим точкам не розкриті через frontend, тому що з точки зору безпеки неправильно використовувати такі методи, як зміна, видалення даних через застосунок з повністю відкритим доступом. Щоб повністю розкрити весь потенціал застосунку, треба використовувати функцію port-forward, який допомагає перенести сервіс на локальну систему, що є дуже безпечним варіантом, так як робота з базою даних йде через закриту

систему. Для цього можна використовувати застосунок Postman (Рисунки 3.15-3.16):

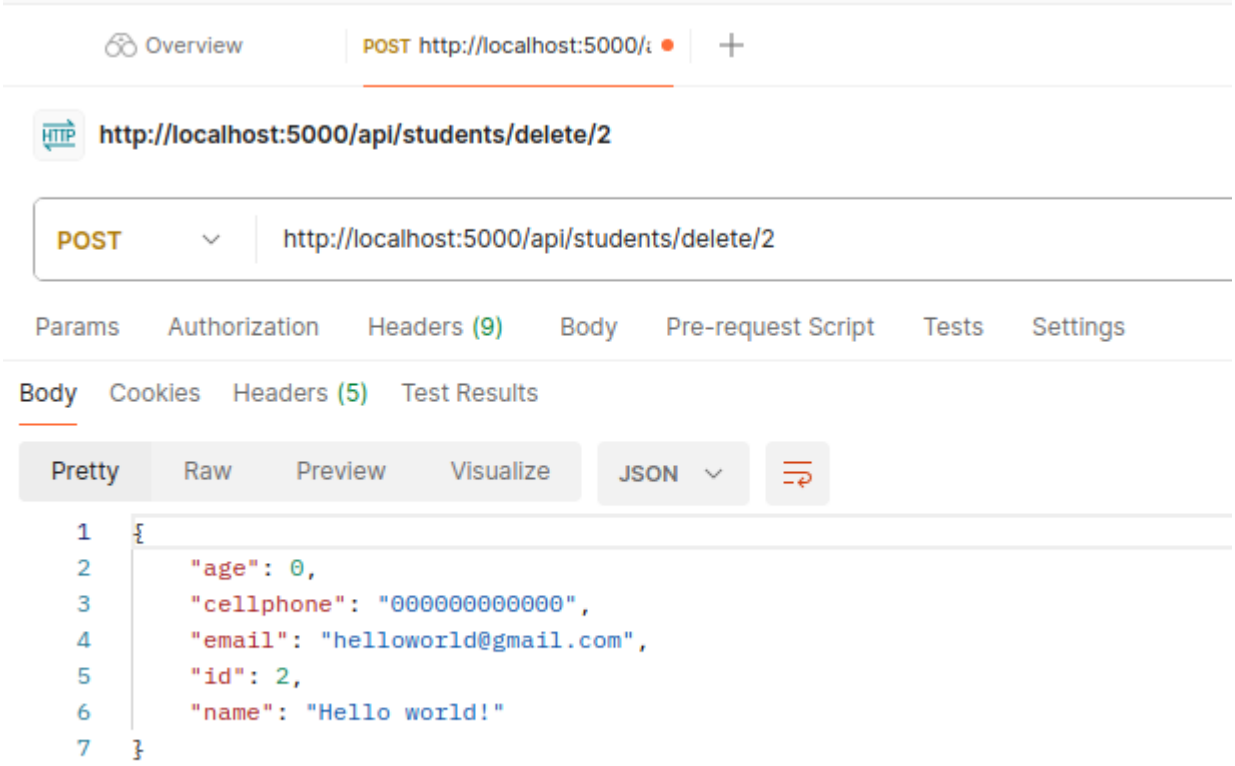


Рисунок 3.15 – приклад роботи функціоналу видалення даних

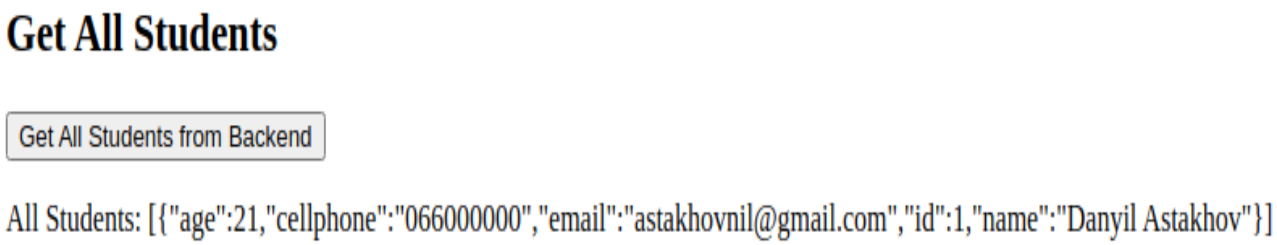


Рисунок 3.16 – результат видалення поля з бази даних

3.3. Налаштування системи моніторингу Prometheus для кластера на основі Kubernetes

В цій дипломній роботі була налаштована система моніторингу Prometheus+Loki+ Grafana Stack.

Першим інструментом було налаштовано графічний інтерфейс Grafana (Рисунок 3.17)

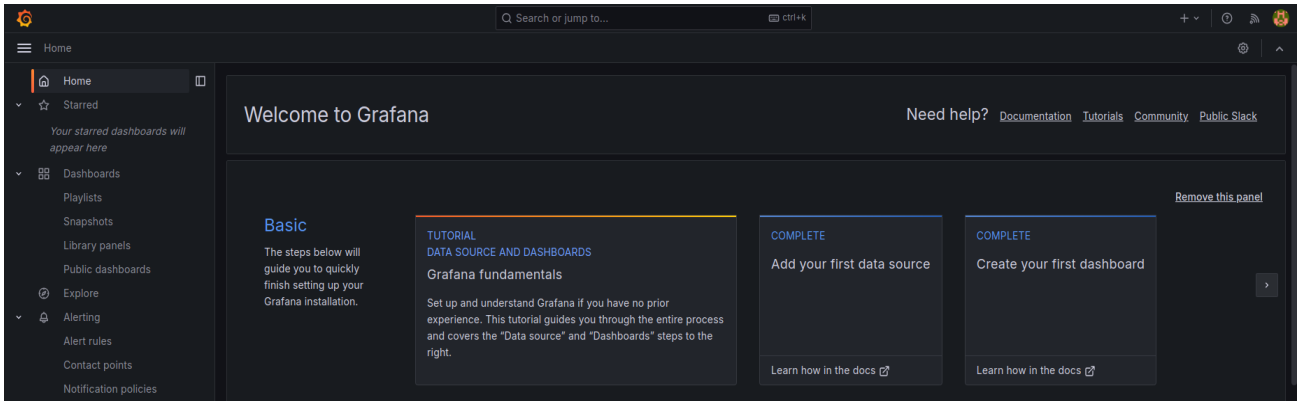


Рисунок 3.17 – приклад графічного інтерфейсу Grafana

Далі було налаштовано Prometheus – систему, яка збирає та відправляє метрики до графічного інтерфейсу Grafana (Рисунок 3.18)

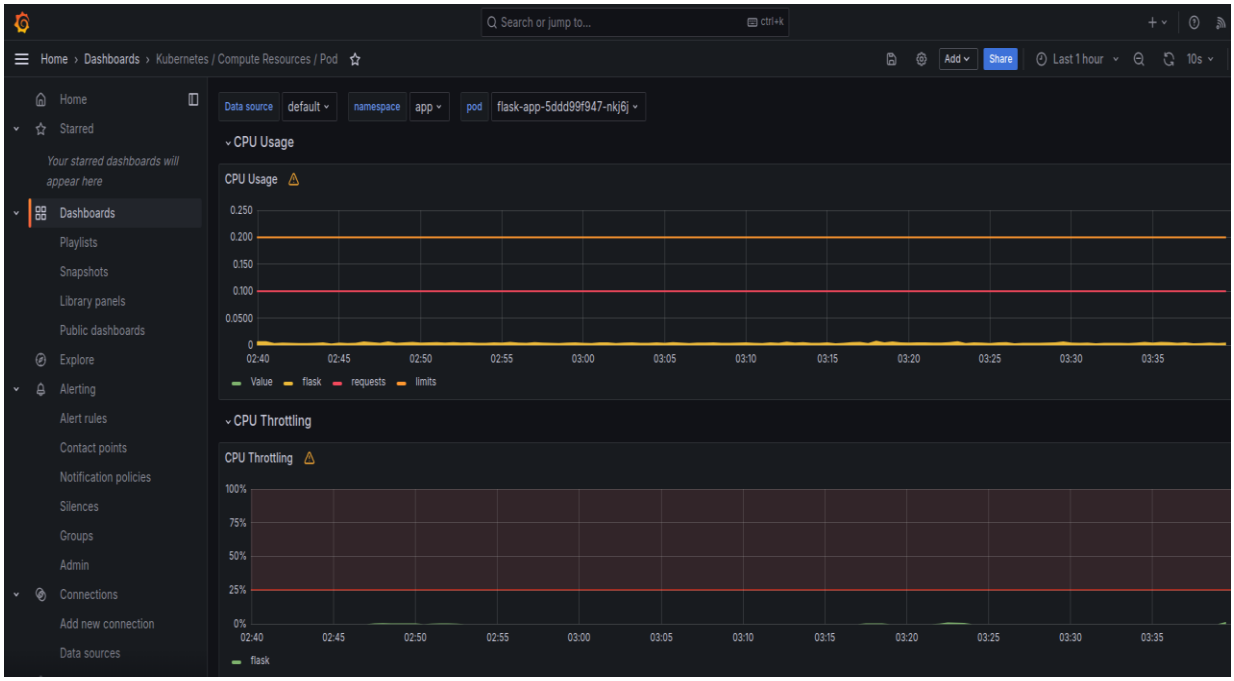


Рисунок 3.18 – показ метрик застосунку

Третій і останнім інструментом було налаштовано систему логів та журналювання Loki, який так само відправляє дані до Grafana (Рисунок 3.19)

Рисунок 3.19 – показ логів застосунку за останню годину

3.4. Налаштування додаткових інструментів безпеки для кластера

Для забезпечення більшої безпеки, були налаштовані:

1. Сховище секретів, які знаходяться на віддаленому сервері (Рисунок 3.20). Після того, як ці секрети стягуються на кластер, застосунок та інструменти записують їх значення до себе, як змінні (Рисунок 3.21).

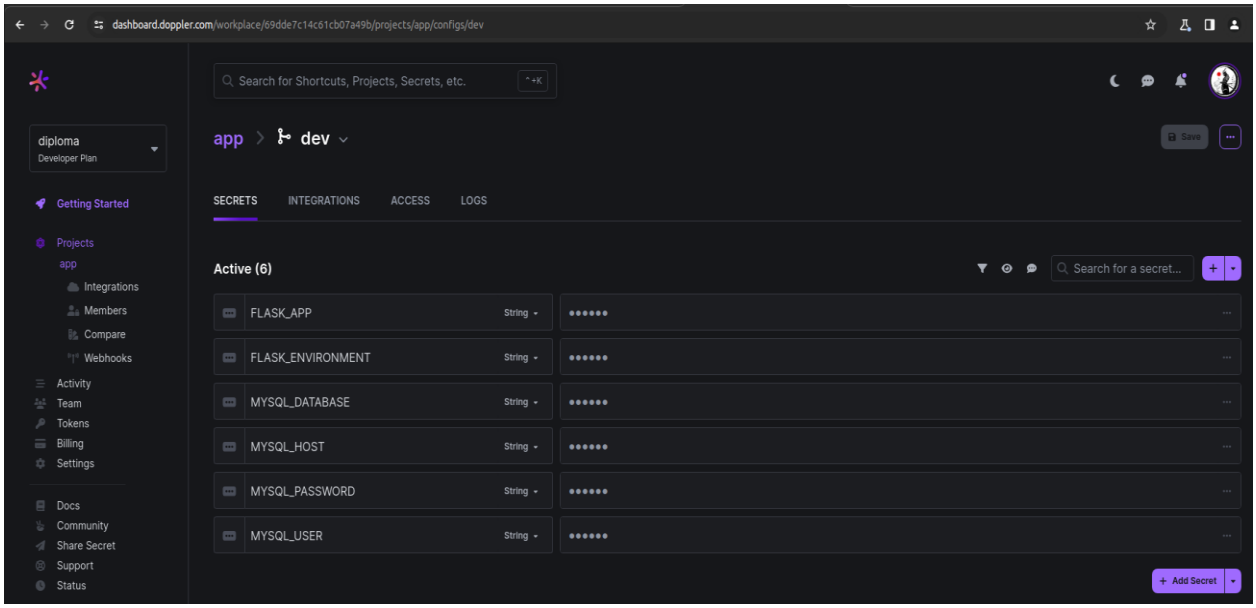


Рисунок 3.20 – приклад інтерфейсу Doppler

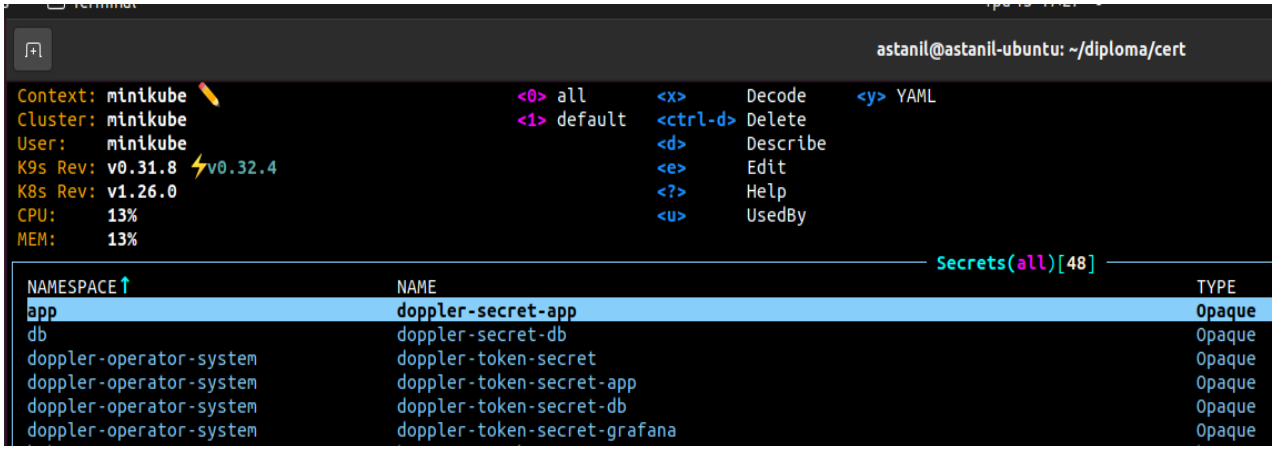


Рисунок 3.21 – дані у вигляді секретів на кластері

2. Також був налаштований обмежений доступ до інфраструктури для користувачів, які не повинні бачити всі дані кластера, наприклад, розробники (Рисунки 3.22-3.23)[21].

Рисунок 3.22 – створення нового окремого середовища для розробників

Рисунок 3.23 – перевірка на доступність обмежених ресурсів

Було розкрито кластер на основі системі оркестрації контейнерів Kubernetes. Інстальовано застосунок на мові програмування Python з базою даних MySQL. Було впроваджено систему моніторингу Prometheus, включаючи додаткові параметри під специфічну конфігурацію застосунка. Також, налаштовані додаткові інструменти безпеки такі як Doppler та RBAC. Це привело до того, що кластер став захищеним та повністю готовим до експлуатації.

ВИСНОВОК

В цій роботі було проаналізовано найпопулярніші на сьогоднішній день методології налаштування кластерів, зокрема:

1. Системи оркестрації контейнерів Kubernetes в цілому.
2. Системи моніторингу для кластера на основі Kubernetes.
3. Додаткові інструменти безпеки для кластера

Були досліджені та аналізовані різні аспекти безпеки та конфіденційності в системах оркестрації кластерів, був налаштований робочий кластер на основі системи оркестрації Kubernetes та повністю імплементований захист застосунку, бази даних та інших окремих та кластеру в цілому. Було виявлено, що використання кластеризації та оркестрації дозволяє покращити ефективність та масштабованість систем, але також може збільшити ризики безпеки та конфіденційності.

Для покращення безпеки та конфіденційності ми використали різні інструменти та методи, такі як обмежений доступ до ресурсів кластера, окреме сховище секретів, моніторинг ресурсів та доступів тощо. Результати практичного виконання дипломної роботи можуть бути використані для покращення безпеки та конфіденційності в системах оркестрації контейнерів, що є критично важливим для захисту інформації та захисту від несанкціонованого доступу, їх зміни та видалення.

Налаштування безпеки сервера підготовило його до повноцінної експлуатації, а саме допомогло додати можливість захистити сервер від втручання, з легкістю можна побачити втручання зловмисника, та полегшити втрати при його втручанні.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Астахов, Д. Безпека та конфіденційність в архітектурі системи оркестрації кластерів // Кіберпростір в умовах війни та глобальних викликів XXI століття: теорія та практика : матер. Міжнародної наук.-практ. конференції (м. Одеса, 24 листопада 2023 р.) / Нац. ун-т «Одеська юридична академія» ; ф-т кібербезпеки та інформ. технологій ; каф. кібербезпеки ; за ред. О. В. Дикого. – Одеса, 2023. – С. 57-59. – Режим доступу : https://doi.org/10.32837/11300.27179
2. Астахов, Д. Безпека та конфіденційність в архітектурі системи оркестрації кластерів // Інформаційне суспільство: проблеми та перспективи : матеріали IX Всеукр. наук.-практич. конф. (м. Одеса, 24 травня 2024 р.) / відп. ред. Н.І.Логінова. Одеса,2024, ДЕПОНОВАНО
3. ПЕРЕЛІК ДОКУМЕНТІВ, ЩО РЕГУЛЮЮТЬ ПИТАННЯ КІБЕРБЕЗПЕКИ УКРАЇНИ – URL: https://ligabezinfo.org/legislation/
4. What is container orchestration? – URL: https://www.ibm.com/topics/container-orchestration
5. What is container orchestration? – URL: https://www.vmware.com/topics/glossary/content/container-orchestration.html
6. What is Kubernetes? – URL: https://www.redhat.com/en/topics/containers/what-is-kubernetes
7. Kubernetes in the wild report 2023 – URL: https://www.dynatrace.com/news/blog/kubernetes-in-the-wild-2023/
8. How Many Kubernetes Clusters Exist Today? – URL: https://cloudnativenow.com/topics/how-many-kubernetes-clusters-exist-today/
9. Kubernetes Monitoring | Tools + 9 Best Practices – URL: https://www.cherryservers.com/blog/kubernetes-monitoring-tools

10.7 Benefits of Continuous Monitoring & How Automation Can Maximize Impact – URL: https://secureframe.com/blog/continuous-monitoring-cybersecurity
11. Production-Grade Container Orchestration – URL: https://kubernetes.io/
12. What is IAM? – URL: https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html
13. Kubernetes security best practices. – URL: https://www.redhat.com/en/topics/containers/kubernetes-security-best-practices
14. What is infrastructure monitoring? – URL: https://www.ibm.com/topics/infrastructure-monitoring
15. Infrastructure monitoring: An introduction – URL: https://mrintegrity.medium.com/monitoring-from-scratch-ea2b83a8f8a5
16. Mike Julian. Practical Monitoring: Effective Strategies for the Real World. C. 31-33 – URL: https://books.google.com.ua/books?id=Mak7DwAAQBAJ&printsec=copyright&redir_esc=y#v=onepage&q&f=false
17. Kubernetes Monitoring: 5 Tools and 4 Best Practices You Must Know About – URL: https://www.tigera.io/learn/guides/kubernetes-monitoring/
18. Kubernetes Security Best Practices -Part 1: Role Based Access Control (RBAC) – URL: https://engineering.dynatrace.com/blog/kubernetes-security-part-1-role-based-access-control-rbac/
19. Lauri Koivunen, Tuomas Mäkilä. Secrets Management in a Multi-Cloud Kubernetes Environment. C. 5-6 - URL: https://www.utupub.fi/bitstream/handle/10024/151776/Secrets_Management_in_a_Multi_Cloud_Kubernetes_Environment_pdf-a.pdf?sequence=1
20. k9s. Kubernetes CLI To Manage Your Clusters In Style! – URL: https://k9scli.io/

21. RBAC with Kubernetes in Minikube – URL: https://medium.com/@HoussemDellai/rbac-with-kubernetes-in-minikube-4deed658ea7b