МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,
освітньо–наукової програми «Інформатика»
на тему: «Автоматизована інформаційна система з використанням хмарних обчислень»
здобувача групи ІН–06–2 Давидова Романа Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Роман ДАВИДОВ
(підпис)

Керівник,
старший викладач кафедри комп'ютерних наук, к.ф.–м.н. Дмитро ВЕЛИКОДНИЙ _____
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавра
зі спеціальності 122 – Комп'ютерних наук, освітньо–професійної програми «Інформатика»
здобувача групи ІН–06–2 Давидов Р. С.

1. Тема роботи: «Автоматизована інформаційна система з використанням хмарних обчислень» затверджую наказом по СумДУ від 22 квітня 2024 № 0414-VI
2. Термін здачі здобувачем кваліфікаційної роботи до 13 червня 2024 року
3. Вхідні дані до кваліфікаційної роботи
_____
4. Зміст розрахунково–пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження.____
2) Огляд технологій, що використовуються для розробки інформаційного та програмного забезпечення автоматизованої інформаційної системи з використанням хмарних обчислень.
3) Розробка інформаційного та програмного забезпечення автоматизованої інформаційної системи з використанням хмарних обчислень 4) Аналіз результатів._____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до Керівник
_____ _____ _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Аналіз проблеми предметної області, постановка й формування завдань дослідження		
2	Огляд технологій для автоматизованої інформаційної системи з використанням хмарних обчислень.		
3	Розробка інформаційного та програмного забезпечення автоматизованої інформаційної системи з використанням хмарних обчислень		
4	Аналіз отриманих результатів		
5	Оформлення пояснювальної записки до кваліфікаційної роботи		

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 69 стор., 29 рис., 4 додатки, 33 джерел.

Обґрунтування актуальності теми роботи – тема дослідження даної кваліфікаційної роботи належить до актуальних напрямків сучасної інформаційно–технологічної сфери. Розвиток хмарних обчислень стає необхідним у зв'язку з постійним зростанням обсягів даних та потреби у швидкому та ефективному доступі до інформації. Використання автоматизованих інформаційних систем з використанням хмарних обчислень є ключовим фактором у забезпеченні конкурентоспроможності сучасних підприємств та організацій.

Об’єкт дослідження – процес автоматизації розгортання інформаційних систем з використанням хмарних обчислень..

Мета роботи – розробка автоматизованої інформаційної системи з використанням хмарних обчислень, спрямованої на оптимізацію процесів розгортання та управління додатками у хмарних середовищах.

Методи дослідження – аналіз існуючих постачальників хмарних послуг, вивчення технологій автоматизованого розгортання додатків, огляд технологій шаблонізації додатків та автоматизації розгортання.

Результати – розроблено інформаційну систему, яка дозволяє ефективно використовувати хмарні обчислення для розгортання додатків. Система пройшла успішне тестування та демонструє здатність до швидкого та надійного розгортання додатків у хмарних середовищах.

ІНФОРМАЦІЙНА СИСТЕМА, СИСТЕМА ОБЛІКУ СТУДЕНТІВ, KUBERNETES, BASH, PIXIE, TERRAFORM, AZURE, IaC, IaaS, HELM DOCKER.

Зміст

ВСТУП

Актуальність. В умовах постійного розвитку сучасних технологій та зростаючих потреб користувачів, використання хмарних обчислень та автоматизованих інформаційних систем стає ключовим для підтримки конкурентоспроможності та успішного функціонування бізнесу. Постійне зростання обсягу даних та потреба у швидкому доступі до них вимагають ефективних інструментів управління інформаційними процесами. У цьому контексті, розробка та впровадження автоматизованої інформаційної системи з використанням хмарних обчислень має стратегічне значення для підприємств, що прагнуть забезпечити надійність, швидкість та масштабованість своїх інформаційних систем.

Об'єкт дослідження – це автоматизована інформаційна система з використанням хмарних обчислень

Предмет дослідження – це методи та засоби для ефективної автоматизації запуску кластерних додатків, моніторингу та оркестрації у середовищі Kubernetes.

Гіпотеза – ефективне управління додатками у Kubernetes–кластері можливе завдяки розробці та впровадженню системи, яка буде автоматизовано керувати життєвим циклом додатків з використанням засобів контейнеризації.

Наукова новизна полягає в тому, що запропонована система автоматизованого управління додатками у Kubernetes–кластері надає можливість забезпечити більш високу ступінь автоматизації та гнучкість управління додатками, що розгортаються, порівняно з існуючими підходами.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибір методу розв'язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Автоматизоване розгортання

Програмне забезпечення [1] – це складний та динамічний продукт, який постійно змінюється та вдосконалюється. Розробники програмного забезпечення повинні не тільки створювати новий код, але й перевіряти його на наявність помилок, сумісності та функціональності. Крім того, вони повинні доставляти свої рішення до клієнтів та кінцевих користувачів, які можуть мати різні вимоги, очікування та середовища.

Автоматизоване розгортання [2] – це процес, який дозволяє автоматично переносити програмне забезпечення з одного середовища в інше. Автоматизоване розгортання має багато переваг, таких як:

- Зменшення помилок, які можуть виникнути при ручному розгортанні, таких як неправильна конфігурація, пропущені кроки або несумісність версій.
- Підвищення швидкості та ефективності розгортання, оскільки не потрібно зважати на людський фактор або виконувати складні процедури.
- Поліпшення якості та надійності програмного забезпечення, оскільки можна використовувати стандартизовані та перевірені сценарії розгортання, а також проводити автоматичне тестування після кожного розгортання.
- Забезпечення послідовності та сумісності середовищ, оскільки можна використовувати однаковий код, дані та конфігурацію для всіх середовищ.

- Зменшення витрат на розгортання, оскільки можна зменшити час, ресурси та зусилля, необхідні для розгортання.

Для успішного автоматизованого розгортання слід дотримуватися деяких найкращих практик, таких як:

- Використовувати інструменти та платформи, які підтримують автоматизоване розгортання, такі як Jenkins [3], Ansible [4], Docker [5], Kubernetes [6] тощо.
- Використовувати версіонування коду та конфігурації, щоб мати повний контроль над змінами та можливість відновити попередню версію у разі необхідності.
- Використовувати неперервну інтеграцію та неперервну доставку (CI/CD [7]), щоб автоматизувати процес збирання, тестування та розгортання коду за допомогою пайплайнів [8].
- Використовувати моніторинг та логування, щоб стежити за станом розгортання та виявляти проблеми або помилки на ранньому етапі.
- Використовувати стратегії розгортання, які зменшують ризик та забезпечують безперебійну роботу програмного забезпечення, такі як блакитно–зелене розгортання (blue–green deployment [9]), канаркове розгортання (canary deployment [10]), A/B–тестування [11] тощо.

Автоматизоване розгортання – це сучасний та ефективний спосіб доставки програмного забезпечення, який має багато переваг для розробників, тестувальників, клієнтів та кінцевих користувачів. Автоматизоване розгортання зменшує помилки, підвищує швидкість, поліпшує якість, забезпечує сумісність та зменшує витрати на розгортання. Для досягнення оптимальних результатів

слід використовувати відповідні інструменти, платформи, методології та стратегії розгортання.

1.2 Компоненти автоматизованого розгортання інформаційної системи

Компоненти автоматизованого розгортання інформаційної системи є ключовими для ефективного та надійного управління інфраструктурою в хмарних обчисленнях. Важливість цих компонентів полягає у їхній здатності спростити та оптимізувати процеси розгортання і підтримки інформаційних систем.

Контейнеризація [12] відіграє ключову роль у сучасному автоматизованому розгортанні інформаційних систем. Цей підхід дозволяє ізолювати додатки та їх залежності в індивідуальні контейнери [13]. Важливість контейнеризації полягає в тому, що вона сприяє уникненню конфліктів між додатками та спрощує процес переносу між різними середовищами, що є критичним для забезпечення стабільності та надійності системи.

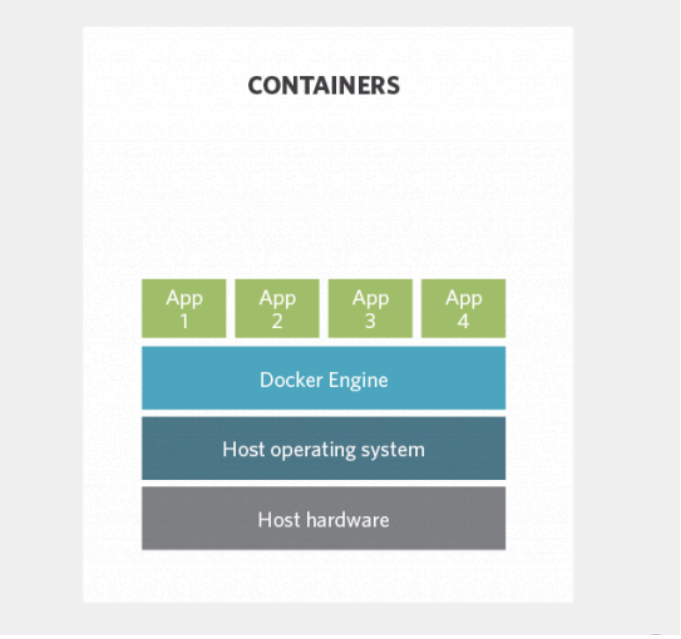

Рисунок 1.1 – Контейнери

Контейнеризація також відкриває можливості для швидкого розгортання та масштабування додатків. Оскільки контейнери містять усі необхідні компоненти, їх можна легко створювати та запускати на будь–якому хості або в хмарному середовищі, які підтримують контейнеризацію. Це зменшує час, необхідний для розгортання додатків, та дозволяє швидко відгукуватися на змінні потреби бізнесу.

Контейнеризація стала важливим інструментом для автоматизованого розгортання інформаційних систем, спрощуючи процеси розгортання, підтримки та масштабування, а також забезпечуючи стабільність та надійність системи в умовах різних середовищ розгортання.

Оркестрація [14] контейнерів є ключовою складовою сучасного автоматизованого розгортання інформаційних систем, яка спрямована на управління та координацію контейнерами у розподіленому середовищі. Оркестрація дозволяє автоматизувати розгортання, масштабування та керування контейнерами великих масштабів, що може бути важливим у складних та великих інфраструктурах.

Завдяки оркестрації, можна легко створювати та розгортати контейнери на різних хостах, а також забезпечити автоматичну заміну або відновлення контейнерів у випадку неполадок. Це робить систему стійкою до відмов та гарантує високу доступність.

Оркестратори надають інструменти для автоматизованого керування контейнерами, масштабування за потребою, розподіл навантаження та балансування навантаження. Ці можливості роблять оркестрацію незамінною для забезпечення стабільності та надійності системи в умовах високих навантажень і складних конфігурацій. Таким чином, оркестрація стає важливим інструментом для забезпечення успішного розгортання та управління контейнеризованими додатками в хмарних та розподілених обчислювальних середовищах.

Інструменти для управління конфігурацією стають незамінними компонентами у процесі автоматизованого розгортання та управління інфраструктурою. Вони дозволяють визначити структуру та параметри інфраструктури як код, що може бути легко розгорнуто та керовано автоматично.

Однією з ключових переваг цих інструментів є можливість створення інфраструктури як коду (Infrastructure as Code [15], IaC). За допомогою спеціальних мов програмування або декларативних конфігураційних файлів, інженери можуть описати всі аспекти своєї інфраструктури, включаючи сервери, мережі, сховища та інші ресурси. Це дозволяє створювати, змінювати та видаляти інфраструктурні ресурси швидко та безпечно.

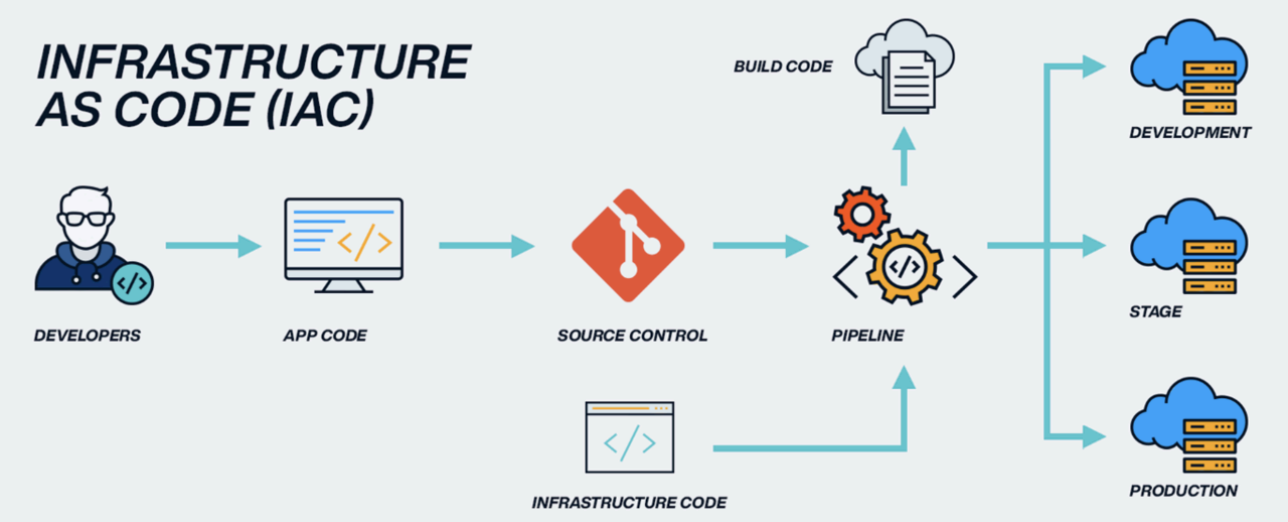

Рисунок 1.2 – Логічна структура IaC [16]

Крім того, інструменти для управління конфігурацією забезпечують консистентність середовищ розгортання. Один і той же конфігураційний файл може бути використаний для розгортання інфраструктури на різних стадіях розробки та в різних середовищах (розробка, тестування, виробництво), що дозволяє уникнути непередбачуваних помилок і забезпечити стабільність та надійність системи в умовах постійних змін і масштабування.

Усе це робить інструменти для управління конфігурацією важливими складовими для автоматизованого розгортання інформаційних систем,

сприяючи швидкому та надійному створенню і підтримці інфраструктури на всіх етапах розробки та експлуатації системи.

1.3 Хмарні обчислення

Хмарні обчислення – це технологія, яка дозволяє користувачам отримувати доступ до обчислювальних ресурсів, таких як сервери, сховища, бази даних, мережі, програмне забезпечення та аналітика через Інтернет.

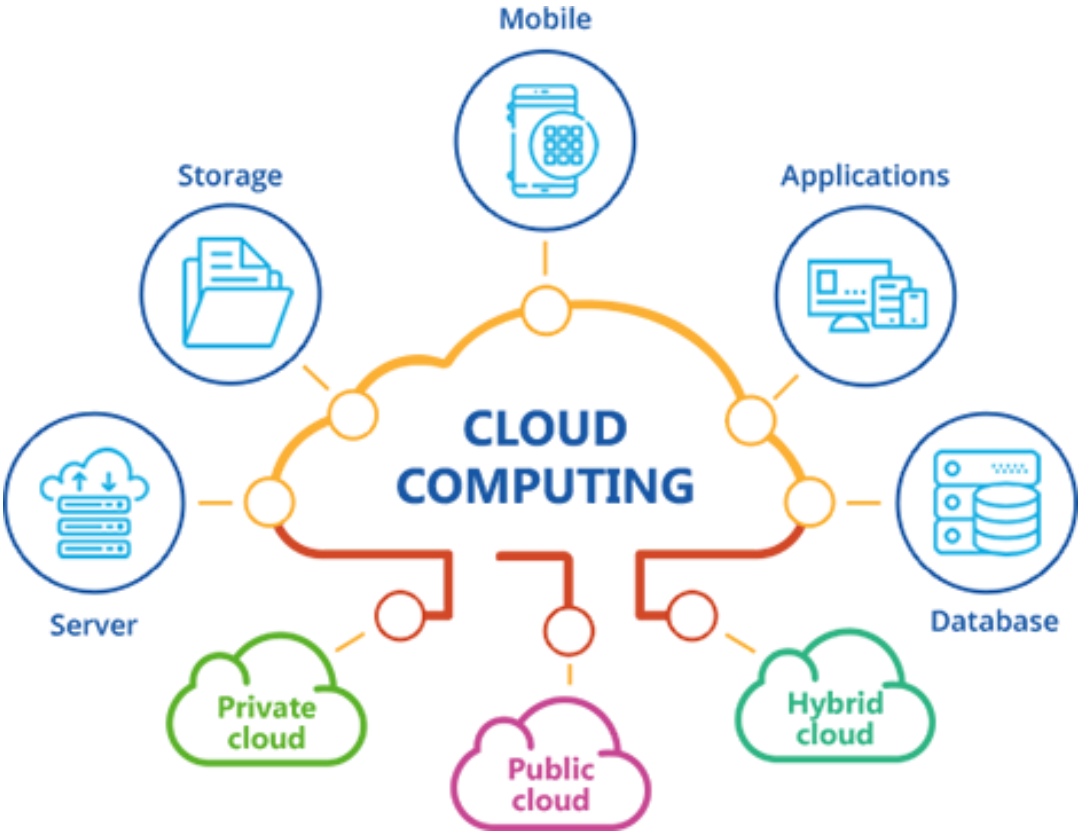

Рисунок 1.3 – Типи хмарних обчисленнь

Хмарні обчислення надаються третіми сторонами, які забезпечують управління, оновлення та безпеку інфраструктури в хмарі. Користувачам не потрібно купувати, встановлювати або підтримувати власне обладнання або програмне забезпечення на своїх пристроях. Замість цього вони можуть просто платити за використання ресурсів за певний період часу або за обсягом споживання.

Зараз багато компаній переходять на хмарні обчислення, оскільки це має багато переваг, таких як:

- Гнучкість та масштабованість. Хмарні обчислення дозволяють швидко збільшувати або зменшувати потребу в ресурсах в залежності від потреб бізнесу. Користувачам не потрібно хвилюватися про недостатню або надлишкову ємність.
- Економія витрат. Хмарні обчислення зменшують капітальні витрати на придбання та утримання обладнання та програмного забезпечення. Користувачам не потрібно платити за ресурси, які вони не використовують.
- Доступність та надійність. Хмарні обчислення забезпечують високий рівень доступності та надійності ресурсів, оскільки вони розподілені по різних географічних регіонах і можуть автоматично відновлюватися в разі збоїв. Користувачам не потрібно хвилюватися про простої або втрату даних.
- Безпека та конфіденційність. Хмарні обчислення захищають дані та програми в хмарі за допомогою шифрування, автентифікації, авторизації, фаєрволів, аудиту та інших механізмів безпеки. Користувачам не потрібно піклуватися про крадіжку або пошкодження даних на своїх пристроях.

Існують три основні типи хмарних обчислень, які називаються моделями обслуговування: IaaS [17] (infrastructure as a service), PaaS [18] (platform as a service) і SaaS [19] (software as a service).

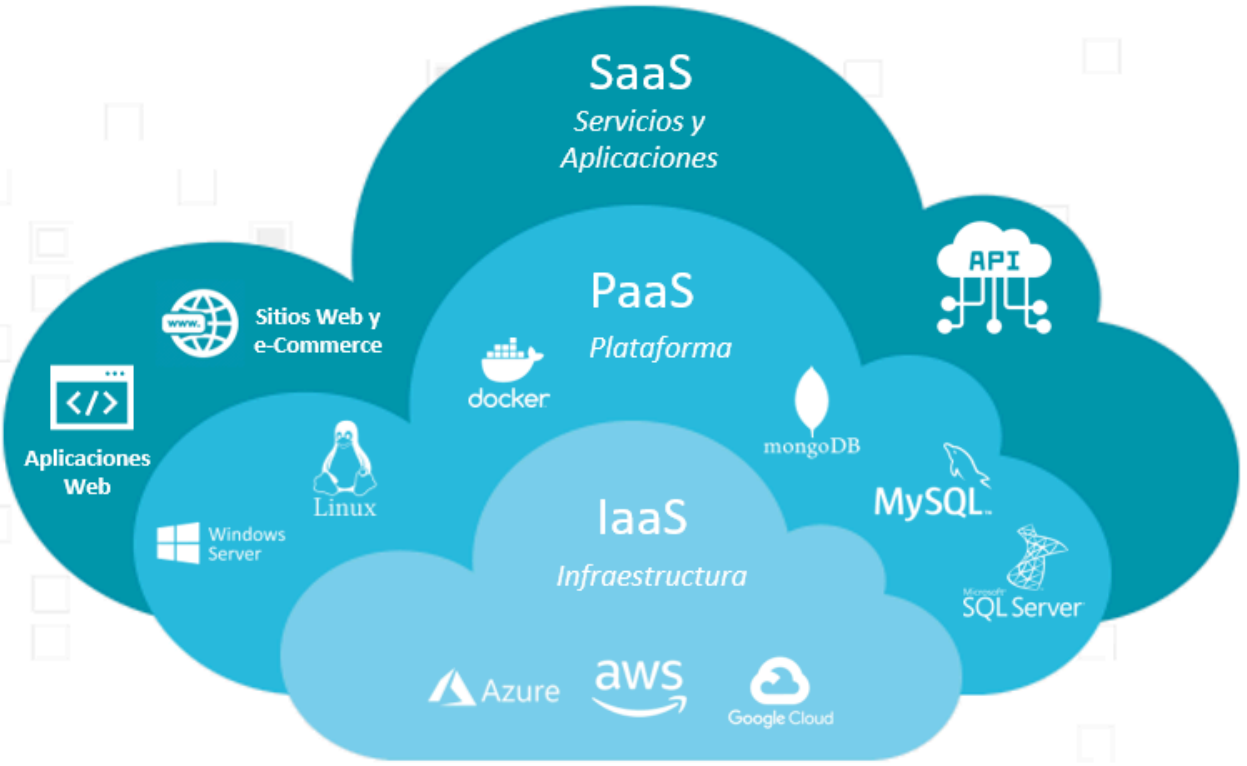

Рисунок 1.4 – Хмарні сервісні моделі

Кожна модель покриває певний рівень управління для користувачів:

– IaaS (інфраструктура як послуга) надає користувачам доступ до хмарної інфраструктури, такої як обчислювальні, сховищні, мережеві та віртуалізаційні ресурси. Користувачі несуть відповідальність за операційну систему, проміжне програмне забезпечення, віртуальні машини та будь–які додатки або дані. Постачальник хмари забезпечує доступ до та управління мережею, серверами, віртуалізацією та сховищем. Користувачі не повинні підтримувати або оновлювати свій власний дата–центр, оскільки постачальник робить це за них. Замість цього вони отримують і контролюють інфраструктуру за допомогою API або панелі керування. IaaS дає користувачам гнучкість купувати тільки ті компоненти, які їм потрібні, і масштабувати їх за потребою.

Низькі загальні витрати і відсутність витрат на обслуговування роблять IaaS дуже доступною опцією.

- PaaS (платформа як послуга) надає користувачам доступ до хмарної платформи, яка дозволяє розробляти, запускати і керувати додатками через хмару. Користувачі несуть відповідальність за свої дані і додатки, але не за операційну систему, проміжне програмне забезпечення, сервери, сховище, мережі або інші аспекти інфраструктури. Постачальник хмари забезпечує доступ до та управління всіма цими ресурсами за допомогою хмарної платформи. Користувачам не потрібно встановлювати або оновлювати своє програмне забезпечення для розробки додатків, оскільки постачальник робить це за них. Замість цього вони можуть просто використовувати хмарну платформу для створення, розгортання та управління своїми додатками. PaaS дає користувачам продуктивність та швидкість розробки, оскільки вони можуть сконцентруватися на своїй бізнес–логіці, а не на технічних деталях. Висока сумісність та інтеграція з іншими хмарними сервісами роблять PaaS дуже зручною опцією.

- SaaS (програмне забезпечення як послуга) надає користувачам доступ до хмарних додатків, які працюють в хмарі і доступні через Інтернет. Користувачам не потрібно ні за що відповідати, крім своїх даних і налаштувань.

- Постачальник хмари забезпечує доступ до управління всіма аспектами програмного забезпечення, включаючи інфраструктуру, платформу, операційну систему, проміжне програмне забезпечення, бази даних і сам додаток. Користувачам не потрібно встановлювати або оновлювати своє програмне забезпечення на своїх пристроях, оскільки постачальник робить це за них. Замість цього вони можуть просто використовувати хмарний додаток через веб–браузер або мобільний клієнт. SaaS дає користувачам зручність та доступність, оскільки вони можуть отримати доступ до своїх додатків з

будь–якого пристрою і місця. Низька вартість і легкість використання роблять SaaS дуже популярною опцією.

Кожна модель покриває певний рівень управління для користувачів. IaaS надає користувачам доступ до хмарної інфраструктури, PaaS надає користувачам доступ до хмарної платформи, а SaaS надає користувачам доступ до хмарних додатків, вибір моделі залежить від типу та вимог до додатку.

1.4 Постановка задачі

Метою даного проекту є створення автоматизованої інформаційної системи, що базується на використанні хмарних обчислень для ефективного обліку студентів та їхніх даних.

Розроблена система повинна відповідати наступним вимогам:

- Збір та збереження інформації про студентів, включаючи особисті дані, академічні досягнення та інші важливі дані.
- Візуалізація та аналіз даних про студентів, зокрема, їх академічні прогреси та участь у додаткових заходах.
- Можливість перегляду даних про кількість та структуру студентів за певний період часу.
- Надсилання сповіщень на електронну пошту адміністраторів про важливі події, такі як реєстрація нових студентів чи оновлення даних.
- Для досягнення поставлених цілей необхідно вирішити наступні завдання:
- Провести аналіз сучасних підходів та визначити актуальність використання хмарних обчислень у сфері освіти та обліку студентів.

- Порівняти різні системи обліку студентів та їх можливості.
- Вибрати технології для розробки програмного забезпечення, що базується на хмарних обчисленнях та відповідає потребам освітнього середовища.
- Реалізувати модель інформаційної системи для обліку студентів та їхніх даних з використанням хмарних обчислень.

2 ВИБІР МЕТОДІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Огляд постачальників хмарних послуг

Хмарні послуги – це надання обчислювальних ресурсів, таких як сервери, сховища, бази даних, мережі, програмне забезпечення, аналітика та інтелект, через Інтернет. Хмарні послуги дозволяють користувачам отримувати доступ до потужної інфраструктури без необхідності придбання, встановлення або підтримки власного обладнання або програмного забезпечення. Хмарні послуги також забезпечують гнучкість, масштабованість, доступність, надійність, безпеку та інновації.

Існує багато постачальників хмарних послуг, які пропонують різноманітні хмарні рішення для різних потреб та сценаріїв такі як: Google Cloud Platform [20], Amazon Web Services [21], Microsoft Azure [22], IBM Cloud [23] та інші.

Google Cloud Platform (GCP) – це хмарна платформа від компанії Google. GCP використовує ту саму інфраструктуру, яку використовує Google для своїх власних продуктів, таких як пошук Google, YouTube, Gmail та інші.

Переваги GCP:

- Висока продуктивність та швидкодія. GCP використовує передові технології Google для забезпечення високої продуктивності та швидкодії своїх хмарних сервісів. GCP також пропонує інноваційні продукти, такі як BigQuery [24] для аналізу великих даних і TensorFlow [25] для машинного навчання.
- Гнучке ціноутворення. GCP пропонує конкурентоспроможне і гнучке ціноутворення для своїх хмарних сервісів. GCP пропонує можливість оплати за секунду за використання обчислювальних ресурсів, що дозволяє заощадити кошти за невикористану ємність.
- Відкритість і сумісність. GCP підтримує відкриті стандарти і технології, що дозволяє легко інтегрувати і мігрувати хмарні рішення. GCP

підтримує Kubernetes для управління контейнерами, Cloud Foundry [26] для розгортання додатків і Apache Beam [27] для обробки потокових даних. GCP також співпрацює з іншими хмарними постачальниками, такими як AWS і Azure.

GCP спеціалізується на наданні хмарних сервісів для аналітики, машинного навчання та великих даних. GCP пропонує широкий спектр продуктів для збору, обробки, аналізу та візуалізації даних будь–якого обсягу та формату.

Amazon Web Services (AWS) – це хмарна платформа від компанії Amazon, яка є першим і найбільшим постачальником хмарних послуг у світі.

Переваги AWS:

- Широкий спектр продуктів і функцій. AWS пропонує найбільшу кількість хмарних сервісів для різних потреб і сценаріїв, також має унікальні продукти, такі як Snowball [28] для перенесення даних з хмари фізичним шляхом і Outposts [29] для запуску хмарних сервісів на локальному обладнанні.
- Висока надійність і доступність. AWS забезпечує високий рівень надійності і доступності своїх хмарних сервісів, оскільки вони розподілені по різних географічних регіонах і зонах.
- Глибока інтеграція і сумісність. AWS пропонує глибоку інтеграцію і сумісність між своїми хмарними сервісами, що дозволяє легко і ефективно створювати і управляти складними хмарними розв’язками.

Microsoft Azure – це хмарна платформа від компанії Microsoft, яка є другим за величиною постачальником хмарних послуг у світі.

Переваги Azure:

- Сильна інтеграція з продуктами Microsoft. Azure підтримує інтеграцію з іншими продуктами Microsoft, такими як Windows , Office 365 ,

Dynamics 365 , Power BI і інші. Це дозволяє користувачам легко і ефективно переносити і використовувати свої дані і додатки в хмарі. Azure також пропонує сумісність з іншими платформами і технологіями, такими як Linux , Java, Python , Node.js і інші.

- Висока гібридність і багатохмарність. Azure пропонує гібридні і багатохмарні розв'язки для користувачів, які хочуть поєднувати свою локальну і хмарну інфраструктуру або використовувати декілька хмарних постачальників.
- Сильна позиція в сфері AI та ML . Azure пропонує широкий спектр продуктів для AI та ML для різних потреб і рівнів складності. Azure має такі продукти, як Azure Cognitive Services для надання готових API для розпізнавання зображень, мови, тексту та іншого

Azure спеціалізується на наданні хмарних сервісів для великих корпорацій, урядових організацій та інших секторів, які потребують високої безпеки, дотримання нормативів та інтеграції з продуктами Microsoft.

З усіх постачальників хмарних послуг, було обрано Microsoft Azure як найкращий варіант для розробки.

2.2 Огляд технологій хмарних обчислень

Технології хмарних обчислень можуть бути поділені на три основні моделі обслуговування:

- PaaS (Platform as a Service) – це модель, в якій постачальник хмарних послуг надає платформу для розробки, запуску та управління прикладними програмами. Користувач має доступ до різних інструментів, мов програмування, баз даних, фреймворків тощо, але не контролює і не підтримує основну інфраструктуру. Прикладами PaaS є Azure App Service , Google App Engine , Heroku тощо. Архітектура PaaS зображена на рисунку 2.1

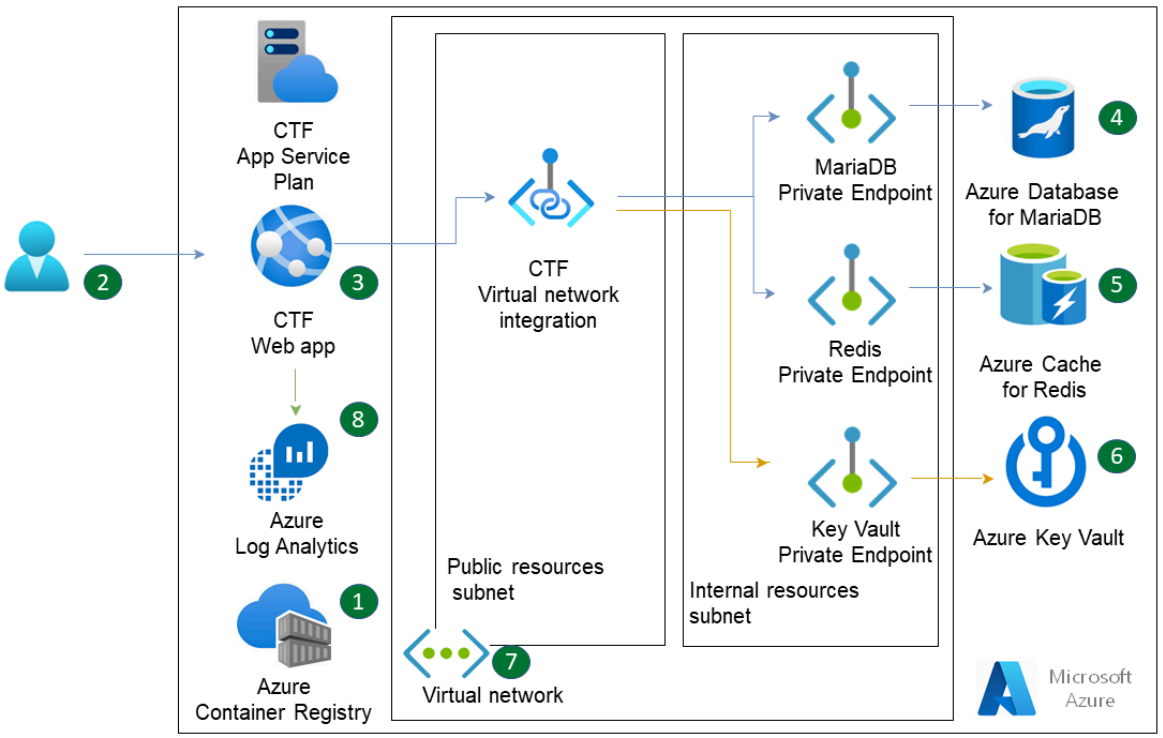

Рисунок 2.1 – Архітектура PaaS [30]

– SaaS (Software as a Service) – це модель, в якій постачальник хмарних послуг надає готове програмне забезпечення як онлайн–сервіс. Користувач має доступ до програми через веб–браузер або мобільний додаток, але не контролює і не підтримує ні інфраструктуру, ні платформу, ні програмне забезпечення. Прикладами SaaS є Microsoft 365, Google Workspace , Salesforce тощо. Архітектура SaaS зображена на рисунку 2.2

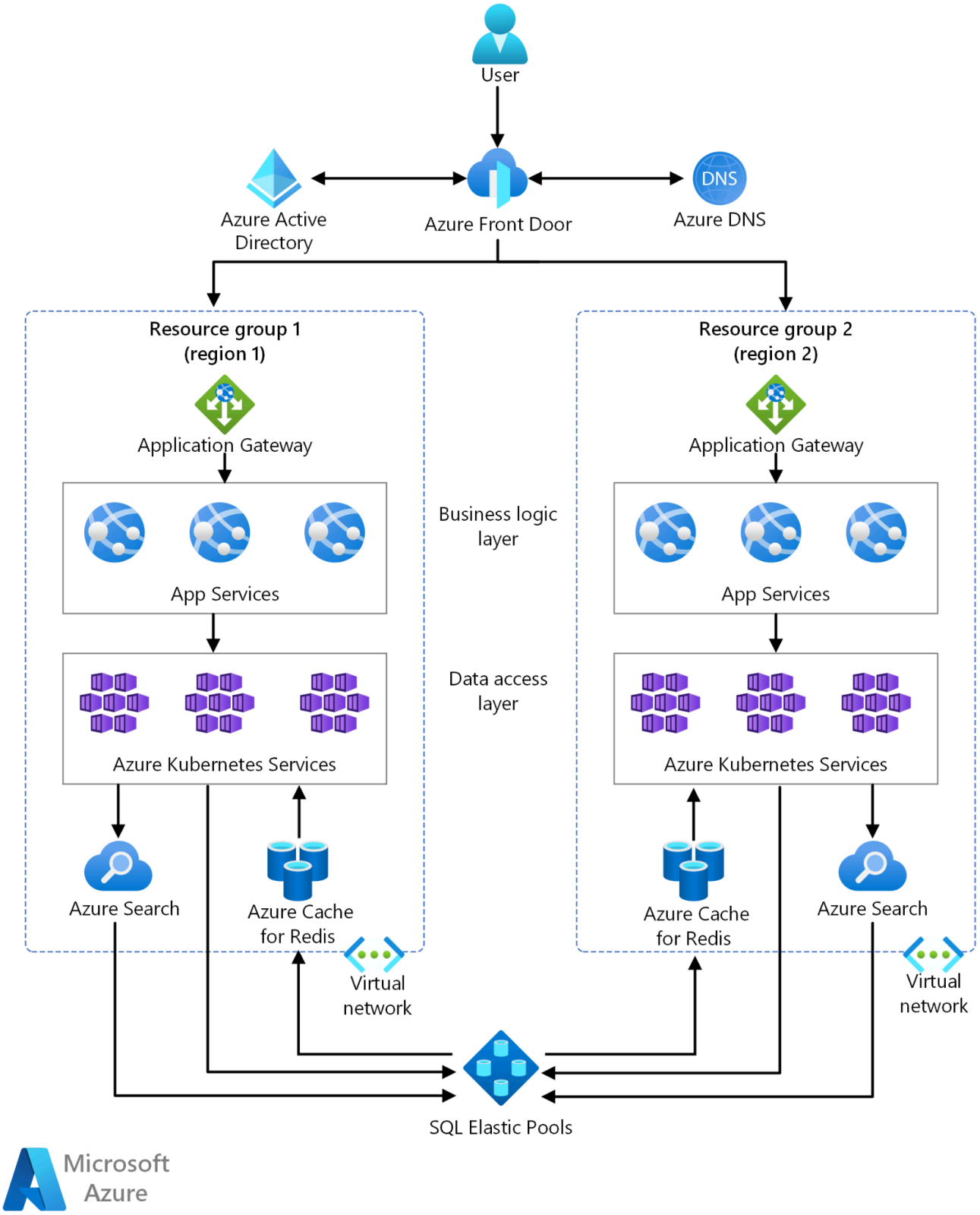

Рисунок 2.2 – Архітектура SaaS [31]

– IaaS (Infrastructure as a Service) – це модель, в якій постачальник хмарних послуг надає основні ресурси для обчислень, сховища і сетевого підключення за запитом з оплатою за використання. Користувач має повний контроль над вибором і налаштуванням операційної системи, програмного забезпечення та прикладних програм, але не контролює і не підтримує фізичного обладнання. Прикладами IaaS є Azure Virtual Machines, Amazon EC2 , Google Compute Engine тощо.Архітектура додатку з реляційною БД IaaS зображена на рисунку 2.3

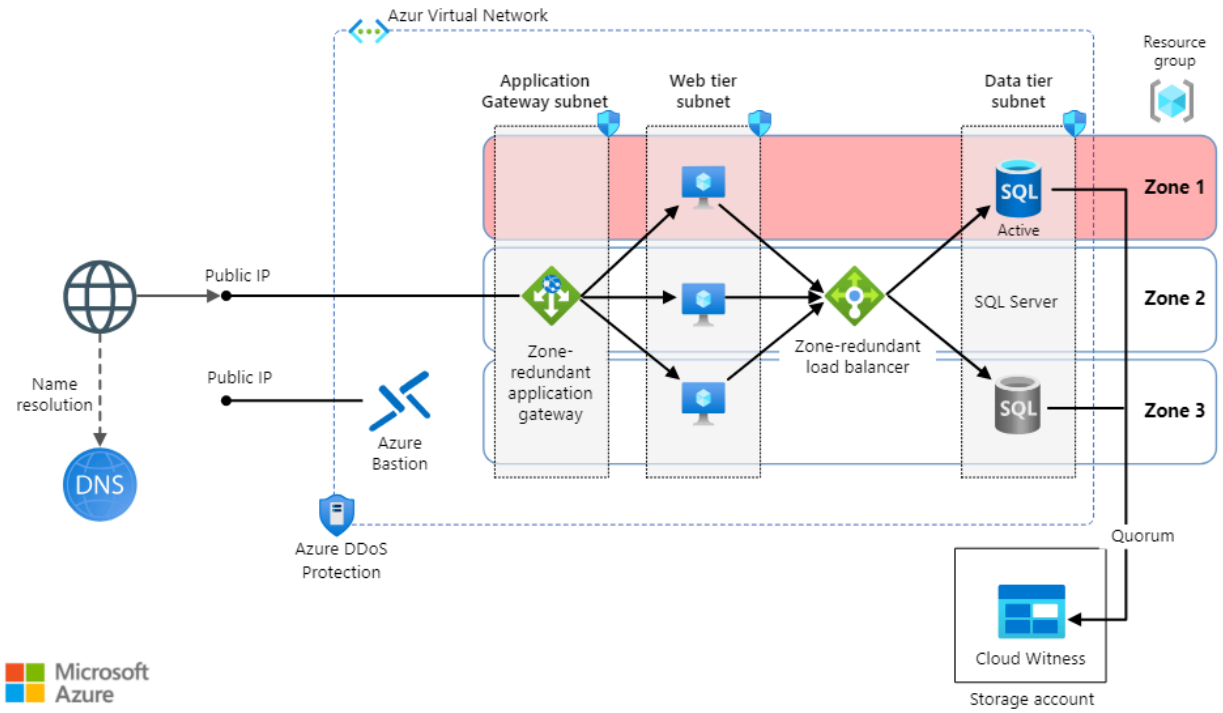

Рисунок 2.3 – Архітектура SaaS [32]

Переваги інструментів IaaS в Azure:

– Інструменти IaaS в Azure дозволяють користувачам мати повний контроль над своїми обчислювальними ресурсами в хмарі, а також налаштовувати їх за своїми потребами та бюджетом. Користувач може вибрати з

різних типів і розмірів віртуальних машин, дисків, мереж та інших компонентів, а також змінювати їх за необхідності.

- Інструменти IaaS в Azure забезпечують високу продуктивність, надійність та доступність обчислювальних ресурсів в хмарі, оскільки вони використовують передову інфраструктуру Microsoft, яка розподілена по різних географічних регіонах і зонах. Користувач може також скористатися гарантованим рівнем якості обслуговування (SLA) для своїх ресурсів, а також можливостями для реплікації, шифрування, резервного копіювання та відновлення даних.
- Інструменти IaaS в Azure надають гнучке та конкурентоспроможне ціноутворення для своїх обчислювальних ресурсів в хмарі, оскільки вони пропонують оплату за використання за годину або за секунду, залежно від типу та розміру ресурсу. Користувач може також скористатися знижками за тривале використання або за попереднє резервування ресурсів.

Було обрано саме IaaS, адже вона відповідає поставленим вимогам

2.3 Огляд технологій автоматизованого розгортання додатків

Контейнеризація є популярною технологією для упаковки та розгортання додатків та їх залежностей в ізольованих середовищах, відомих як контейнери.

Docker та rkt (Rocket) є двома популярними технологіями контейнеризації, які використовуються для упаковки та розгортання додатків у контейнерах.

Docker – це відома технологія контейнеризації, яка дозволяє упаковувати додатки та їх залежності в ізольовані контейнери, що незалежні від операційної системи хоста. Docker був вперше випущений у 2013 році і швидко здобув велику популярність завдяки своїм перевагам та зручності використання.

Переваги Docker:

- Контейнеризація: Docker надає легкий і стандартизований спосіб упакувати додатки та їх залежності у контейнерах. Кожен контейнер має своє власне ізольоване середовище, що дозволяє запускати додатки незалежно від конфліктів з іншими програмами та залежностями. Це спрощує розробку, тестування та розгортання додатків.

- Мультиплатформенність: Docker робить контейнери переносимими між різними операційними системами та архітектурами, такими як Linux, Windows і macOS. Це дозволяє розробникам та адміністраторам легко розгортати однакові контейнери на різних платформах.

- Швидкість і легкість: Контейнери Docker запускаються швидше, оскільки вони використовують спільне ядро операційної системи хоста і не потребують емуляції апаратного забезпечення. Вони також вимагають менше обсягу системних ресурсів, що зменшує навантаження на хост–систему та дозволяє більше контейнерів на одному сервері.

- Управління версіями: Docker дозволяє створювати контейнерні образи за допомогою Dockerfile , які містять всі необхідні компоненти та конфігурацію додатка. Образи можна зберігати в Docker Registry , що дозволяє відстежувати та керувати версіями додатків. Діаграма функціонування кросплатформеності Docker зображена на рисунку 2.4

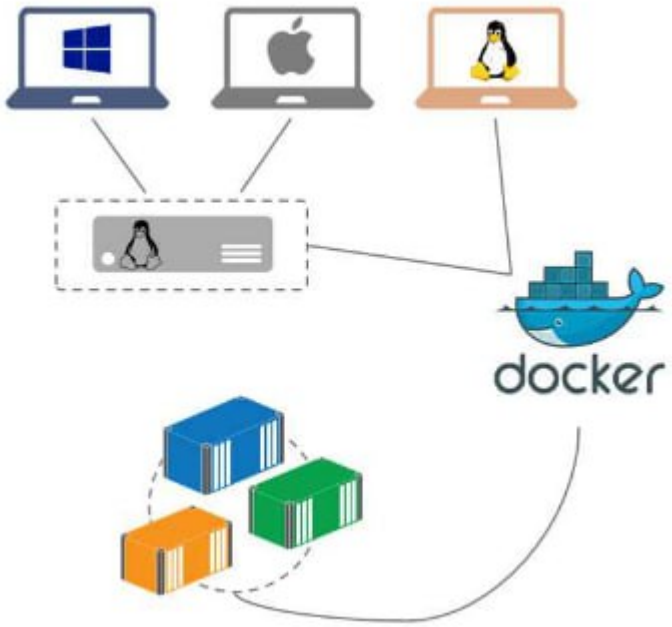

Рисунок 2.4 – Діаграма функціонування кросплатформеності Docker

Rocket – це альтернативна технологія контейнеризації, розроблена компанією CoreOS. Rocket був створений з метою представити альтернативу Docker і відрізняється від нього в деяких ключових аспектах.

Основні переваги Rocket:

- Архітектура: Rocket має спрощену архітектуру порівняно з Docker. Він ставить на перший план простоту і безпеку, уникнувши складних функцій, таких як демон контейнера. Замість цього, Rocket працює за допомогою інструментів командного рядка.
- Спрощений підхід до безпеки: Rocket ретельно дотримується принципу "принципу найменших привілеїв". Він робить контейнеризацію більш безпечною, не використовуючи складних скриптів або додаткових служб для управління контейнерами. Це може бути важливо для сценаріїв, де безпека є пріоритетом.

- Гнучкість і відкритість: Rocket ставить на відкритість і має гнучкість у виборі різних компонентів для роботи з контейнерами. Він може використовувати різні інструменти для сховища образів та мережі, що дає більше варіантів для інтеграції з іншими системами.
- ACI (App Container Image): Rocket використовує формат образів App Container Image (ACI), який вважається менш складним та більш прозорим у порівнянні з форматом образів Docker. ACI спрощує створення, зберігання та поширення контейнерних образів.

Через ряд переваг було обрано Docker.

Оркестрація додатків – це процес управління та автоматизації розгортання, масштабування та керування контейнерами або іншими компонентами додатків у розподілених середовищах. Існує багато технологій для оркестрації додатків, які допомагають спростити та забезпечити надійне управління додатками.

Kubernetes – це відкрита платформа для автоматизації розгортання, керування та масштабування контейнерами. Він був розроблений Google і зараз підтримується Cloud Native Computing Foundation (CNCF). Kubernetes дозволяє легко керувати розподіленими додатками, використовуючи контейнери, такі як Docker, у виробничих середовищах.

Kubernetes має багато переваг, які роблять його популярним і важливим інструментом для управління контейнерами та оркестрації додатків. Ось декілька з найважливіших переваг Kubernetes:

- Автоматизація розгортання і масштабування: Kubernetes дозволяє автоматизовано розгортати, масштабувати і керувати додатками і контейнерами. Ви можете описати стан свого додатку в конфігураційних файлах і дозволити Kubernetes забезпечити, щоб стан додатку відповідав цим описам.

- Гнучкість та розширюваність: Kubernetes надзвичайно гнучкий і легко розширюється. Ви можете встановити його на будь–якому хмарному сервісі, власному центрі обробки даних або гібридному середовищі. Він може керувати великими кластерами серверів і багатьма додатками одночасно.

- Самовідновлення: Kubernetes автоматично відновлює контейнери та додатки у випадку неполадок. Якщо якийсь контейнер чи вузол виходить з ладу, Kubernetes може перезапустити його або розмістити на іншому вузлі, забезпечуючи стійкість додатків.

- Збалансування навантаження: Kubernetes має вбудовані засоби для розподілу навантаження між різними частинами додатку. Він забезпечує рівномірне розподілення трафіку і автоматично визначає, де розміщувати нові контейнери.

- Декларативна конфігурація: Ви описуєте бажаний стан свого додатку в конфігураційних файлах і Kubernetes робить все необхідне, щоб досягти цього стану. Це дозволяє уникнути складних скриптів та команд для налаштування системи.

Docker Swarm – це вбудований оркестратор контейнерів, який поставляється разом з Docker, однією з найпопулярніших технологій контейнеризації. Docker Swarm призначений для оркестрації контейнерів Docker у великих мережах і кластерах серверів.

Переваги Docker Swarm:

- Простота використання: Однією з основних переваг Docker Swarm є його простота в налаштуванні та використанні. Це ідеальний вибір для тих, хто вже працює з Docker, оскільки багато функціональності і інтерфейси схожі.

- Масштабованість: Docker Swarm може легко масштабуватися. Ви можете додавати нові вузли до кластера для збільшення обчислювальної потужності або видаляти їх для зменшення. Docker Swarm автоматично розподіляє контейнери між вузлами.
- Самовідновлення: Docker Swarm має вбудовані засоби для виявлення неполадок і автоматичного відновлення контейнерів. Якщо контейнер виходить з ладу, Swarm може автоматично перезапустити його на іншому вузлі.
- Інтеграція з Docker Compose: Ви можете використовувати існуючі конфігураційні файли Docker Compose для запуску контейнерів на Docker Swarm без змін.
- Хоча Docker Swarm має свої переваги, важливо враховувати, що Kubernetes є більш потужним і функціонально розширеним інструментом для оркестрації контейнерів.

Ряд переваг Kubernetes над Docker Swarm:

- Більша функціональність: Kubernetes має більше функціональних можливостей порівняно з Docker Swarm. Він підтримує багато додаткових функцій, таких як створення розподілених мереж, налаштування політик безпеки, автоматизоване розгортання і масштабування додатків, горизонтальне автоматичне масштабування і багато інших.
- Більша спільнота і підтримка: Kubernetes має надзвичайно велику та активну спільноту розробників і користувачів. Це означає більше ресурсів, інструментів та рішень для розв'язання різних завдань.

- Здатність до роботи в більших масштабах: Kubernetes призначений для роботи в розподілених середовищах і може легко масштабуватися для обслуговування більшої кількості контейнерів і вузлів.
- Сумісність і міграція: Kubernetes зазвичай більш сумісний з різними хмарними платформами і іншими середовищами. Це робить його кращим вибором для організацій, які планують мігрувати додатки або використовувати різні хмарні рішення.

Через ряд переваг було обрано Kubernetes.

Інструменти автоматизації розгортання допомагають розробникам і адміністраторам автоматизувати процеси розгортання додатків та інфраструктури. Два з найпопулярніших інструментів цього типу – це Terraform і Ansible .

Terraform – це інструмент для інфраструктурного коду, розроблений компанією HashiCorp, який дозволяє автоматизувати процес створення, налаштування і керування інфраструктурою та ресурсами в хмарних платформах та центрах обробки даних. Його переваги:

- Інфраструктурний код: Terraform використовує декларативний підхід до налаштування, дозволяючи вам описувати бажаний стан вашої інфраструктури за допомогою файлів конфігурації HCL (HashiCorp Configuration Language). Цей код можна зберігати в системах контролю версій.
- Мультиплатформенність: Terraform підтримує багато провайдерів для різних хмарних платформ і інфраструктурних сервісів, таких як AWS, Azure, Google Cloud, Kubernetes, Docker, і багато інших. Це дозволяє створювати і керувати ресурсами в різних середовищах з одного інструмента.

- Планування і виконання: Terraform надає можливість спочатку створити план змін, які будуть внесені до інфраструктури, і відповідно перевірити цей план. Після цього ви можете виконати ці зміни, і Terraform забезпечить їх виконання в правильному порядку.
- Модульність і повторне використання коду: Terraform дозволяє вам створювати модулі, які можна використовувати для повторного використання коду та створення більш складних інфраструктурних конфігурацій.
- Спільнота та розширення: Terraform має велику та активну спільноту розробників і користувачів. Інструмент постійно розвивається, і у нього є багато розширень та сторонніх модулів, що полегшує роботу.
- Інфраструктура як код: Terraform допомагає реалізувати концепцію "інфраструктура як код", що полегшує автоматизацію і створення резервних копій інфраструктури. На рисунку зображено основні провайдери Terraform 2.5

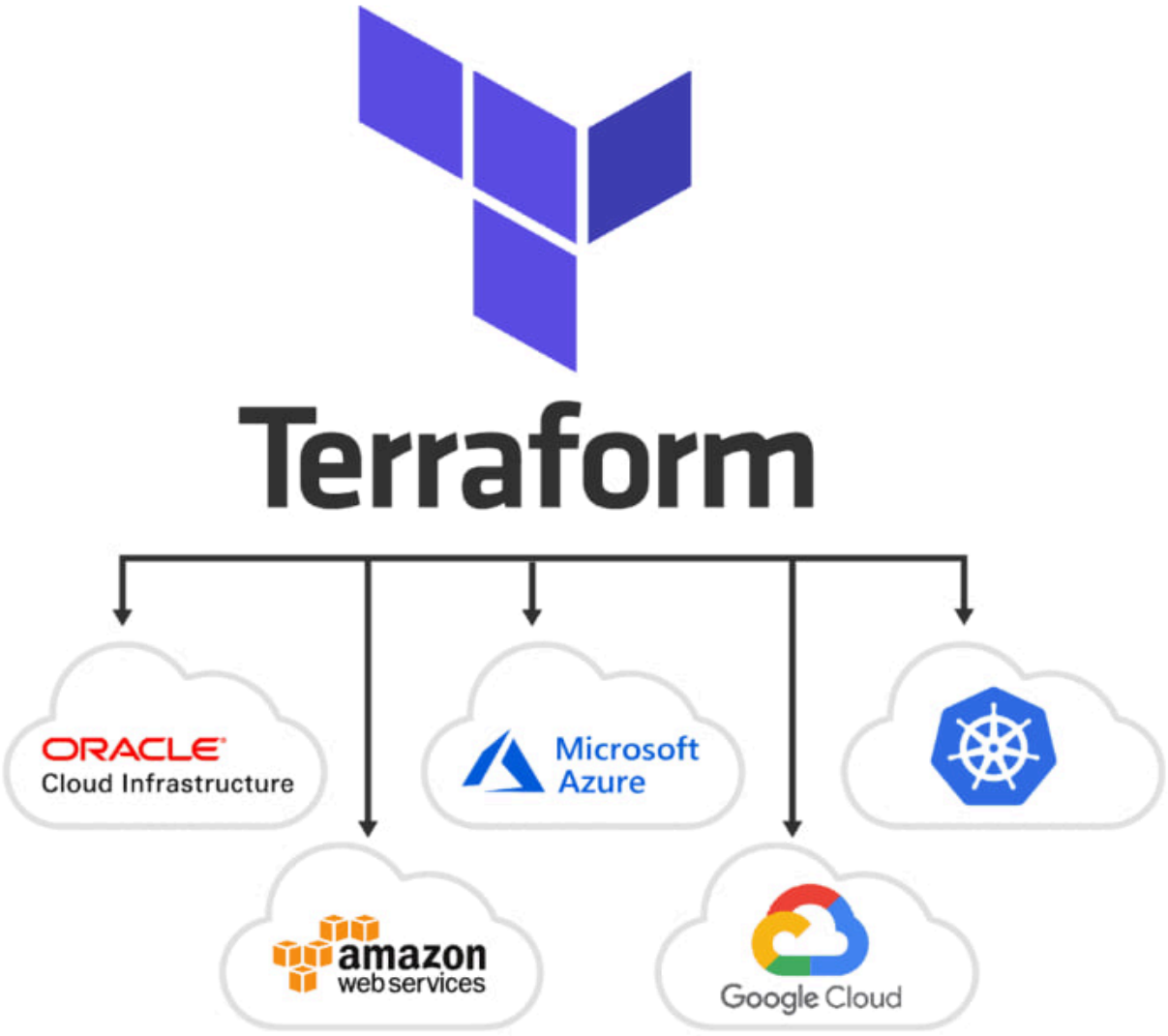

Рисунок 2.5 – Основні провайдери Terraform

Ansible – це інструмент автоматизації та оркестрації, розроблений компанією Red Hat, який дозволяє автоматизувати задачі конфігурації, управлінням серверами, деплойменту додатків та інші операції на серверах та інших комп'ютерах. Його переваги:

- Агентless: Ansible працює в агентлес режимі, що означає, що вам не потрібно встановлювати додаткові агенти або клієнти на сервери, як це робиться в багатьох інших інструментах. Він використовує протоколи, такі як

SSH, для взаємодії з серверами, що робить його легким для встановлення та налаштування.

- Простий синтаксис YAML: Ansible використовує простий і читабельний синтаксис YAML для опису задач та конфігурації. Це полегшує створення, редагування та розуміння коду Ansible.
- Декларативний підхід: Ansible використовує декларативний підхід до налаштування, що означає, що ви описуєте бажаний стан системи, і Ansible самостійно визначає необхідні дії для досягнення цього стану. Це полегшує роботу з конфігурацією та управлінням серверами.
- Кросплатформенність: Ansible підтримує різні операційні системи і хмарні платформи, що дозволяє використовувати його для різних типів інфраструктури.
- Інфраструктура як код: Ansible допомагає реалізувати концепцію "інфраструктура як код", що полегшує автоматизацію та створення резервних копій інфраструктури.

Ansible і Terraform – це два різних інструмента, які використовуються для різних завдань у сфері автоматизації та інфраструктурного управління. Вони мають різні сценарії застосування та переваги. Для роботи було обрано Terraform.

2.4 Огляд технологій шаблонізації додатків

Шаблонування розгортання маніфестів Kubernetes – це процес створення та застосування параметризованих шаблонів для опису ресурсів Kubernetes, які потрібно розгорнути в кластері. Це дозволяє уникнути дублювання коду, спростити конфігурацію та забезпечити узгодженість між різними

середовищами. Шаблонування розгортання маніфестів Kubernetes використовується для таких цілей, як:

- Генерація динамічних значень, таких як імена, мітки, адреси тощо.
- Внесення змін до базових маніфестів без їх зміни або перезапису.
- Перевикористання та комбінування існуючих маніфестів для створення нових.
- Управління залежностями та порядком розгортання між ресурсами.

Для шаблонування розгортання маніфестів Kubernetes існують різні інструменти, такі як Helm, Kustomize, Jsonnet, Pulumi тощо. У цьому випадку ми розглянемо два з них: Helm та Kustomize.

Helm – це пакетний менеджер для Kubernetes, який дозволяє створювати, встановлювати та оновлювати навіть найскладніші застосунки Kubernetes. Helm використовує концепцію чартів – пакетів з параметризованими шаблонами маніфестів, які можна легко створювати, версіонувати, ділитися та публікувати.

Helm також надає можливості для генерації ресурсів, таких як ConfigMaps та Secrets, для налаштування застосунків, а також для виконання гаків для керування життєвим циклом застосунків. На рисунку 2.6 зображено діаграму роботи Helm.

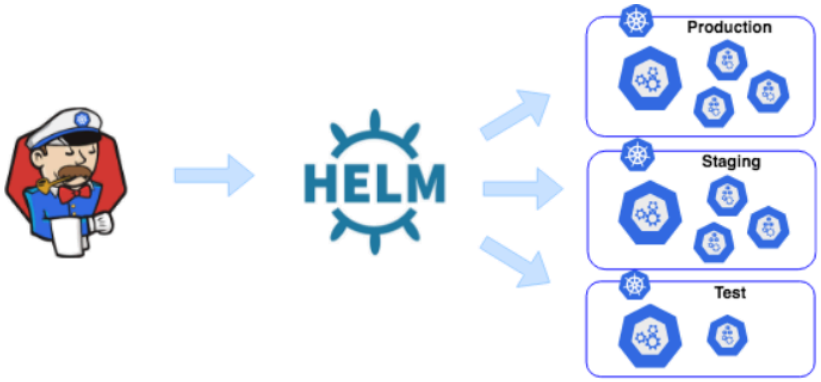

Рисунок 2.6 – Діаграма роботи Helm [33]

Kustomize – це інструмент для налаштування конфігурацій Kubernetes, який використовує принцип нашарування для збереження базових налаштувань застосунків та компонентів шляхом накладання декларативних yaml–артефактів (називаються латками), які вибірково перевизначають налаштування за замовчуванням без фактичної зміни початкових файлів. Kustomize також може генерувати ресурси, такі як ConfigMaps та Secrets, з інших представлень.

Переваги шаблонування розгортання маніфестів Kubernetes:

- Зменшення помилок та несумісностей при розгортанні застосунків в різних середовищах, таких як розробка, тестування, продакшн тощо.
- Полегшення співпраці, перевикористання та управління кодом конфігурації за допомогою стандартизованих форматів та інструментів.
- Покращення продуктивності, гнучкості та автоматизації процесу розгортання застосунків в Kubernetes.

Було обрано для роботи Helm, через ряд його переваг.

2.5 Автоматизація розгортання додатків за допомогою Helm Charts

В сучасному світі, де швидкість впровадження програмного забезпечення є ключовим фактором конкурентоспроможності, автоматизація процесу розгортання додатків стає невід'ємною частиною розвитку та управління інфраструктурою. В цьому контексті Helm та Helm Charts виявляються важливими інструментами, спрямованими на забезпечення швидкого та ефективного впровадження додатків у контейнеризованих середовищах.

Helm є інструментом управління пакетами для Kubernetes, що дозволяє спростити та автоматизувати процеси розгортання, оновлення та керування додатками. Використання Helm дозволяє стандартизувати розгортання додатків, що полегшує їхнє масштабування та управління.

Helm Charts є пакунками, які містять всю необхідну інформацію для розгортання додатків в Kubernetes. Вони включають в себе конфігураційні файли, шаблони ресурсів Kubernetes, метадані та залежності. Завдяки Helm Charts розробники можуть ефективно організувати та розподілити свої додатки, а також легко керувати конфігурацією середовища.

Використання Helm та Helm Charts принесе численні переваги для команд розробників та DevOps інженерів. По–перше, вони дозволяють стандартизувати процес розгортання, що сприяє зменшенню ризику помилок та забезпеченню консистентності середовищ. По–друге, автоматизація розгортання за допомогою Helm Charts прискорює процес впровадження змін та оновлень, що робить його більш ефективним та масштабованим.

Helm та Helm Charts знаходять широке застосування в індустрії розробки програмного забезпечення, зокрема в управлінні мікросервісною архітектурою та в контейнеризованих середовищах. Вони використовуються для розгортання та керування додатками у великих корпоративних системах, хмарних інфраструктурах та на місцевих серверах.

Структура Helm Charts складається з наступних основних елементів:

- Chart.yaml: Цей файл містить метадані про Helm Chart, такі як назва, версія, опис тощо.
- values.yaml: Цей файл містить значення параметрів, які можуть бути використані для налаштування додатку під час розгортання.
- templates/: У цьому каталозі розміщуються шаблони конфігураційних файлів Kubernetes, які будуть використовуватися для створення ресурсів (наприклад, ресурси Deployment, Service, ConfigMap тощо).
- charts/: Якщо ваш Helm Chart має залежності від інших Charts, вони можуть бути включені в цей каталог.

Для налаштування Helm Charts можна змінювати значення параметрів у файлі values.yaml відповідно до індивідуальних вимог. Ці параметри можуть включати такі дані, як розмір ресурсів, порти, URL–адреси служб, налаштування бази даних та інші конфігураційні параметри, які можуть бути унікальними для вашого додатку.

Helm Charts – це потужний інструмент, який дозволяє розробникам та DevOps інженерам стандартизувати та автоматизувати процес розгортання додатків в Kubernetes. Налаштування, розробка та використання Helm Charts робить процес розгортання більш простим та ефективним, забезпечуючи консистентність та швидкість впровадження.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Розробка IaC з використанням хмарних обчислень

Infrastructure as Code (IaC) – це підхід до управління інфраструктурою IT–систем, який полягає в автоматизації процесу створення, налаштування та управління інфраструктурними ресурсами через використання коду. Замість традиційних ручних процесів, де адміністратори вручну налаштовують сервери, мережі і інші компоненти інфраструктури, в IaC інфраструктура описується у вигляді коду, що дозволяє автоматизувати її розгортання та керування.

Для розгортання IaC було обрано Terraform. На рисунку 3.1 зображено структуру розробленого Terraform проєкту.

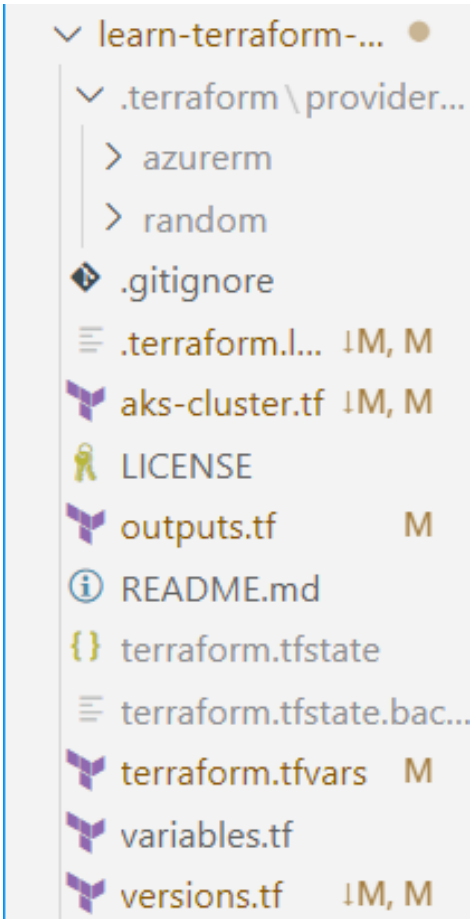

Рисунок 3.1 – Структура Terraform

Terraform складається з таких основних компонентів як:

Providers – це плагіни, які дозволяють Terraform взаємодіяти з різними хмарними сервісами. Кожен провайдер має набір доступних ресурсів, які він може створити та керувати. Провайдер azurerm дозволяє Terraform взаємодіяти з Azure Resource Manager, а провайдер random дозволяє генерувати випадкові числа для використання в конфігураціях.

Resources – це об'єкти інфраструктури, які Terraform створює, модифікує або видаляє. Кожен ресурс пов'язаний з певним провайдером і має набір властивостей, які ви можете налаштувати. Файл aks–cluster.tf може містити опис ресурсу для створення кластера Kubernetes в Azure.

Variables – використовуються для зберігання значень, які ви можете використовувати у всьому проекті. Вони допомагають зробити ваші конфігурації більш гнучкими та повторно використовуваними. Файли змінних, такі як terraform.tfvars та variables.tf, містять значення цих змінних.

Outputs – це значення, які генеруються після застосування конфігурації Terraform. Ви можете використовувати виводи для отримання інформації про вашу інфраструктуру після її створення. Файл outputs.tf може містити виводи, які показують IP–адресу створеного сервера.

State Files – це файли, які зберігають інформацію про поточний стан ресурсів у вашому проекті. Вони важливі для збереження історії інфраструктури та для відслідковування змін. Файли стану, такі як terraform.tfstate, зберігають цю інформацію.

На зображені 3.2 зображено вивід командного рядка після ініціалізації Terraform. Вивід деталізує процес ініціалізації, включаючи ініціалізацію бекенду та плагінів провайдера. Повідомлення вказують, що попередні версії hashicorp/azurerm та hashicorp/random повторно використовуються, а також відзначаються нові встановлення. Є вказівка, що Terraform вніс деякі зміни в вибір залежностей провайдера, записаний у файлі .terraform.lock.hcl. Виділене повідомлення стверджує, що "Terraform було успішно ініціалізовано!". Після

цього йдуть інструкції про те, як почати роботу з Terraform, включаючи запуск “terraform plan”, щоб побачити будь–які необхідні зміни для інфраструктури.

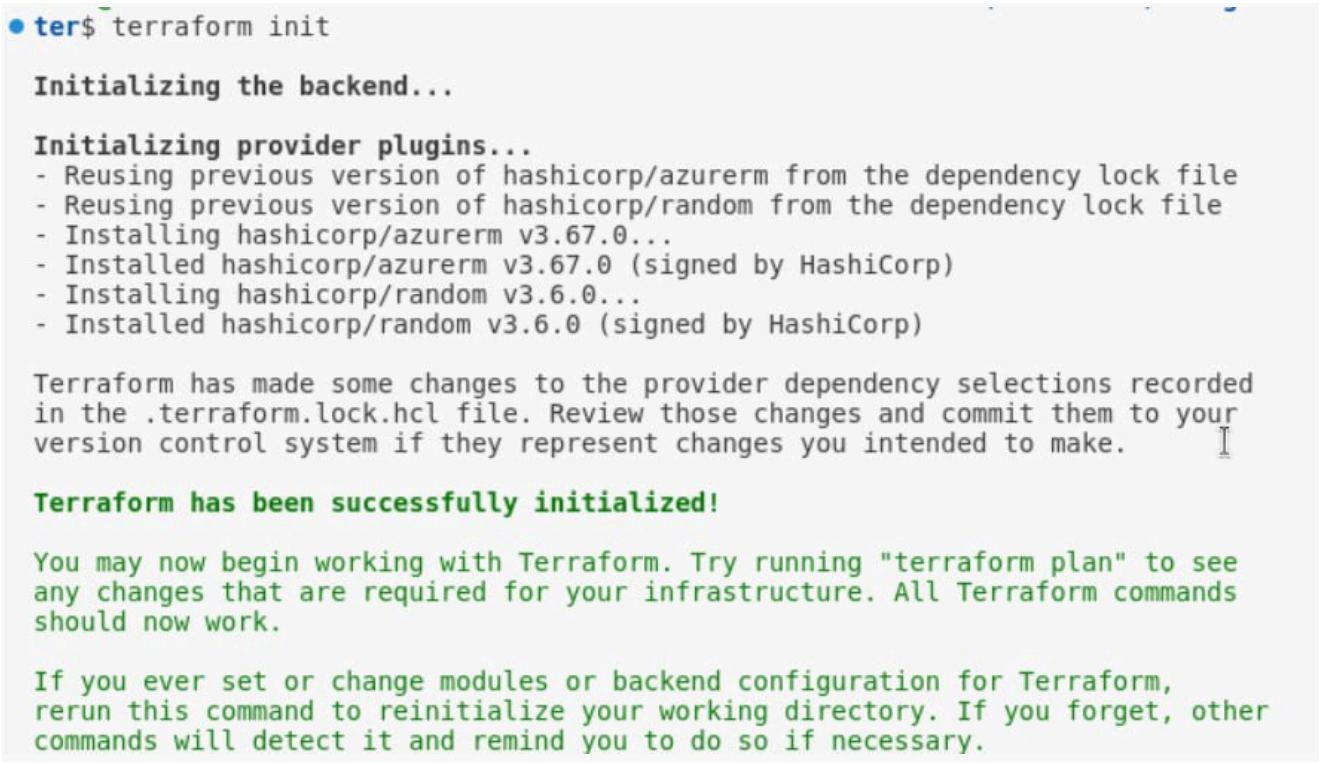

Рисунок 3.2 – Успішна ініціалізація Terraform

На зображені 3.3 зображено вивід інформації про ресурси, які будуть створені.

```
ter$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_kubernetes_cluster.default will be created
  + resource "azurerm_kubernetes_cluster" "default" {
      + api_server_authorized_ip_ranges     = (known after apply)
      + dns_prefix                          = (known after apply)
      + fqdn                                = (known after apply)
      + http_application_routing_zone_name  = (known after apply)
      + id                                  = (known after apply)
      + image_cleaner_enabled               = false
      + image_cleaner_interval_hours        = 48
      + kube_admin_config                   = (sensitive value)
      + kube_admin_config_raw               = (sensitive value)
      + kube_config                         = (sensitive value)
      + kube_config_raw                     = (sensitive value)
      + kubernetes_version                  = "1.28"
      + location                            = "australiaeast"
      + name                                = (known after apply)
      + node_resource_group                 = (known after apply)
      + node_resource_group_id              = (known after apply)
      + oidc_issuer_url                     = (known after apply)
      + portal_fqdn                         = (known after apply)
      + private_cluster_enabled             = false
      + private_cluster_public_fqdn_enabled = false
      + private_dns_zone_id                 = (known after apply)
      + private_fqdn                        = (known after apply)
      + public_network_access_enabled       = true
      + resource_group_name                 = (known after apply)
      + role_based_access_control_enabled   = true
      + run_command_enabled                 = true
      + sku_tier                            = "Free"
      + tags                                = {
          + "environment" = "Demo"
        }
```

Рисунок 3.3 – План розгортання Terraform

Різні параметри та їх значення або статуси (“known after apply”) перераховані, такі як назви груп ресурсів, місцезнаходження, версія Kubernetes тощо. Символи, такі як “+”, що вказують на створення ресурсів.

За допомогою наведеного IaC коду Terraform, буде розгорнуто кластер Kubernetes зображений на рисунку 3.4 та Azure Database for MySQL зображений на рисунку 3.5.

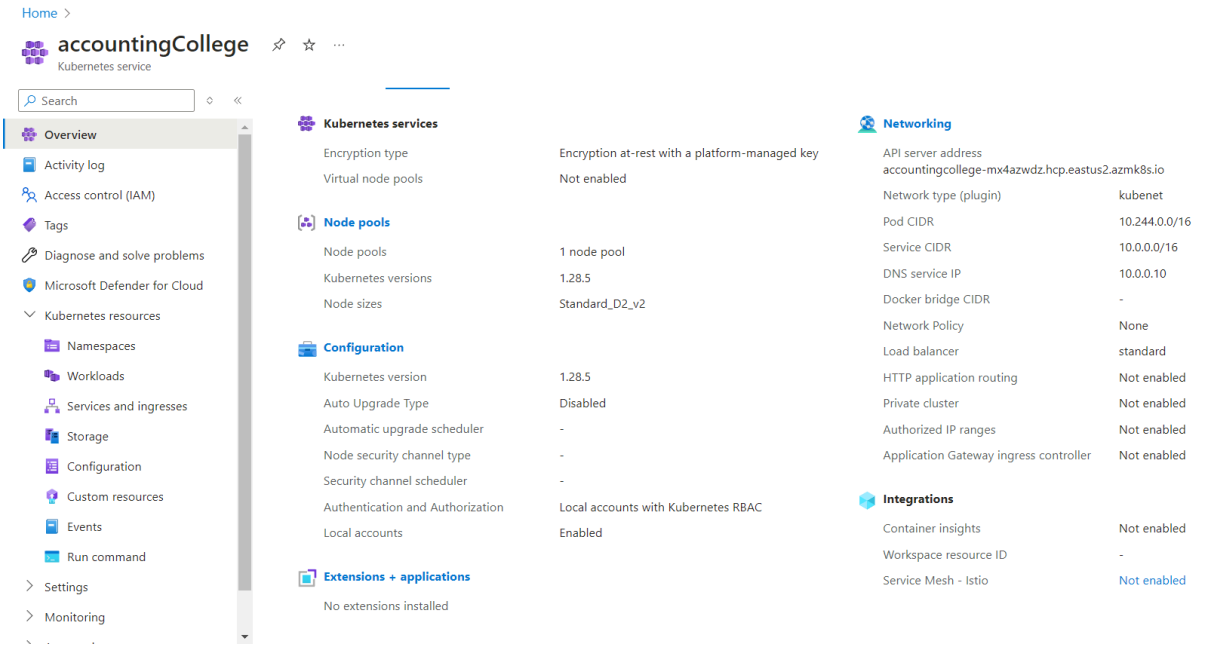

Рисунок 3.4 – Розгорнутий кластер Kubernetes

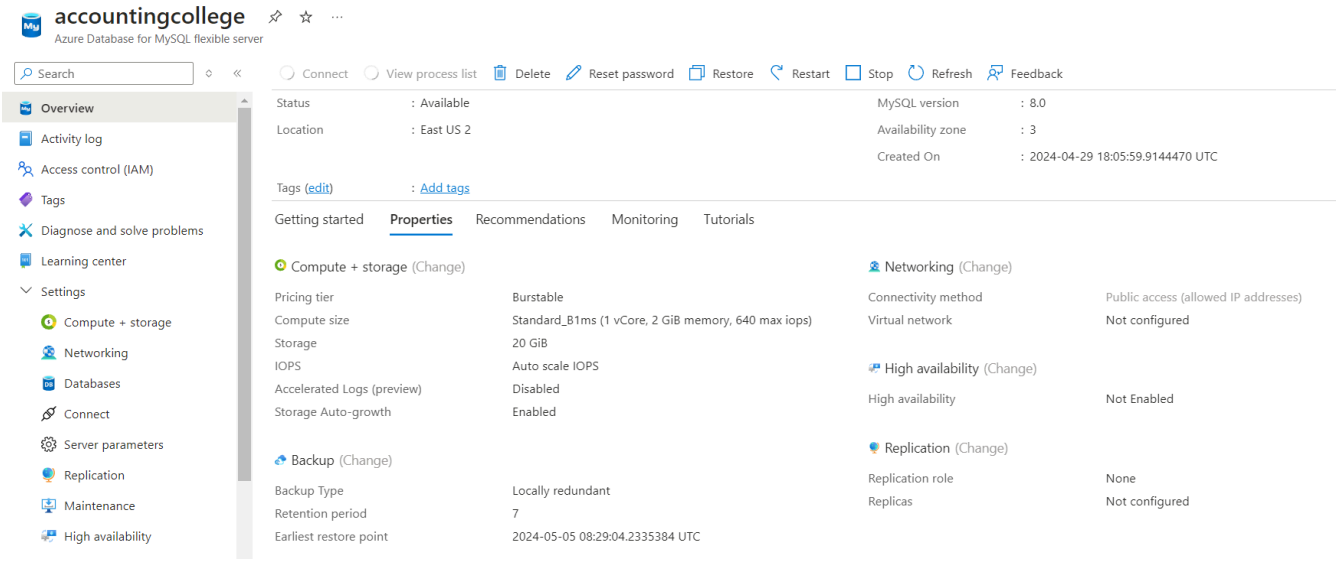

Рисунок 3.5 – Розгорнута база даних Azure Database for MySQL

3.2 Інтеграція контейнерів для оптимізації інформаційної системи

Використання контейнерів є важливим етапом у вдосконаленні інформаційних систем. Ця технологія дозволяє збільшити ефективність та гнучкість системи шляхом ізоляції додатків та їх залежностей. Інтеграція

контейнерів в інформаційну систему забезпечує стандартизацію середовища виконання, спрощує розгортання додатків і полегшує їх масштабування.

Для інтеграції контейнерів в інформаційну систему використовується Docker, одна з найпопулярніших платформ для контейнеризації додатків. Завдання полягає в створенні Docker–образу додатку, що включає всі необхідні компоненти та залежності, його побудови за допомогою команди docker build. В додатку А наведено код Dockerfile.

Процес побудови контейнера включає такі кроки:

- Оновлення списку пакетів для системи внутрішнього контейнера.
- Встановлення та активація розширень PHP mysqli та pdo, необхідних для роботи з базами даних.
- Копіювання усіх файлів та папок з поточного каталогу у контейнер у каталог /var/www/html/.
- Встановлення додаткових пакетів, таких як git, zip, unzip, необхідних для роботи з додатком.
- Завантаження та встановлення Composer для керування залежностями PHP–додатку.
- Виконання команди composer install для встановлення залежностей PHP–додатку з файлу composer.json.
- Активація модуля Apache rewrite, який дозволяє використовувати перезапис URL.
- Налаштування власника та групи для каталогу /var/www/html/.

- Відкриття порту 80 для забезпечення доступу до веб–сервера Apache.
- Запуск веб–сервера Apache в режимі переднього плану.

На зображенні 3.6 відображено логи побудови образу

```
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 536B
=> [internal] load metadata for docker.io/library/php:7.2-apache
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [ 1/10] FROM docker.io/library/php:7.2-apache@sha256:4dc0f0115acf8c2f0df69295ae822e49f5ad5fe849725847f15aa0e5802b55f8
=> [internal] load build context
=> => transferring context: 8.64kB
=> CACHED [ 2/10] RUN apt-get update
=> CACHED [ 3/10] RUN docker-php-ext-install mysqli pdo
=> CACHED [ 4/10] RUN docker-php-ext-enable mysqli pdo
=> CACHED [ 5/10] COPY . /var/www/html/
=> CACHED [ 6/10] RUN apt-get install -y     git     zip     unzip     && rm -rf /var/lib/apt/lists/*
=> CACHED [ 7/10] RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer
```

Рисунок 3.6 – Логи побудови образу

Після успішної побудови зображення необхідно завантажити його до Docker Hub. Це централізований репозиторій Docker. Після завантаження образу в Docker Hub він стає доступним для використання іншими користувачами та для розгортання в інших середовищах. Такий процес дозволяє забезпечити консистентність та легкість управління додатком у контейнеризованому середовищі. На зображені 3.7 можна побачити готовий Docker Image інформаційної системи управління обліком студентів розміщений в Docker Hub

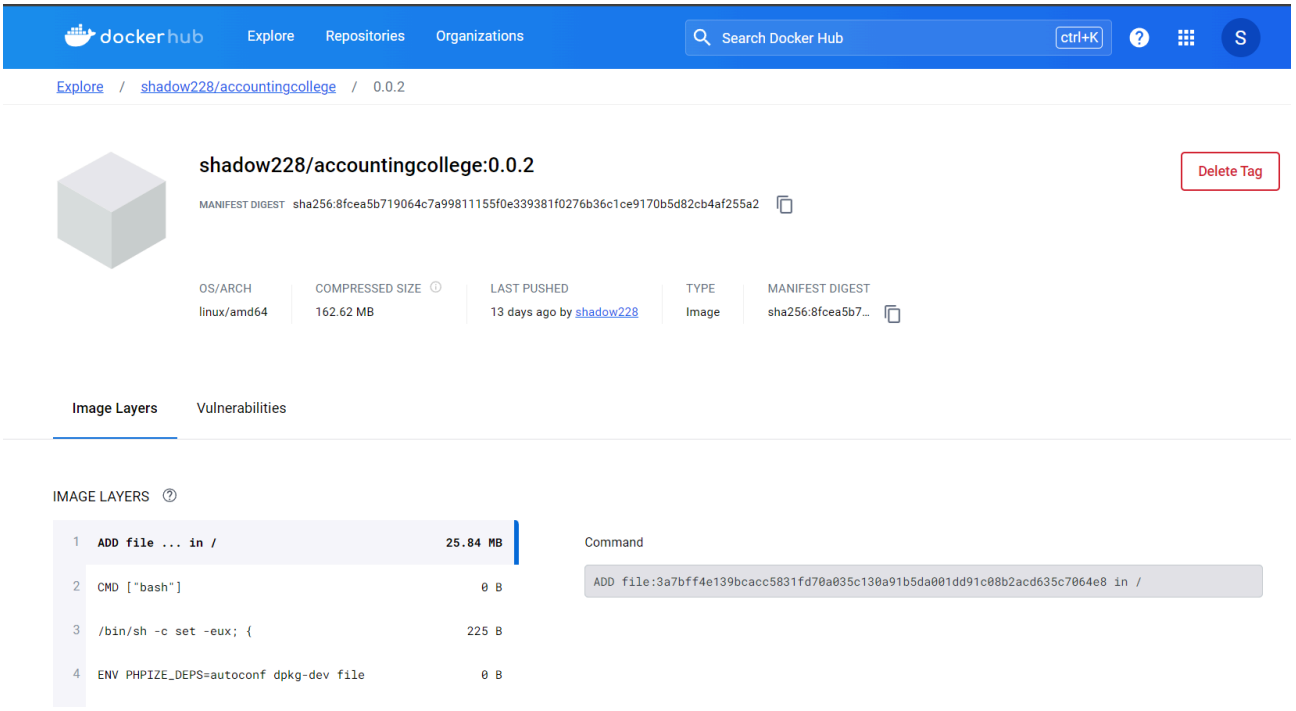

Рисунок 3.7 – Docker Image інформаційної системи управління обліком студентів

Інтеграція контейнерів для оптимізації інформаційної системи полягає в використанні їхніх переваг а саме:

– Ізоляція середовища: Контейнери дозволяють ізолювати додатки та їх залежності від інших системних компонентів. Це дозволяє забезпечити стабільну роботу додатків навіть у разі змін в середовищі виконання або оновленнях інших компонентів.

– Масштабованість: Контейнери можуть бути легко масштабовані вгору або вниз в залежності від потреб системи. Це дозволяє ефективно використовувати ресурси серверів та забезпечує гнучкість управління навантаженням.

– Швидкість розгортання: Контейнери мають мінімальний час розгортання порівняно з віртуальними машинами, що дозволяє швидко

впроваджувати нові версії додатків та вносити зміни без великих перерв у роботі системи.

- Портативність: Контейнери забезпечують стандартизоване середовище для додатків, що дозволяє легко переміщати їх між різними хостами та хмарними сервісами без необхідності виконання додаткових налаштувань.
- Інтеграція та розширення: Контейнери дозволяють легко інтегрувати різноманітні компоненти системи та розширювати їх функціональність, наприклад, за допомогою мікросервісної архітектури.

Інтеграція контейнерів у інформаційну систему допомагає оптимізувати роботу системи завдяки ізоляції середовища, масштабованості, швидкості розгортання, портативності та можливості інтеграції та розширення функціональності. Використання контейнерів сприяє підвищенню ефективності, надійності та гнучкості інформаційної системи, роблячи її більш готовою до змін та вимог користувачів та ринку.

3.3 Шаблонізація інформаційної системи пакетним менеджером Helm

Helm – це пакетний менеджер для Kubernetes, забезпечує ефективну шаблонізацію та управління додатками. Helm дозволяє створювати конфігураційні шаблони (часто називані "charts"), які можна легко встановлювати, оновлювати та видаляти, спрощуючи процес розгортання та управління мікросервісами та додатками в Kubernetes–середовищі.

Структура Helm Chart зображена на рисунку 3.8.

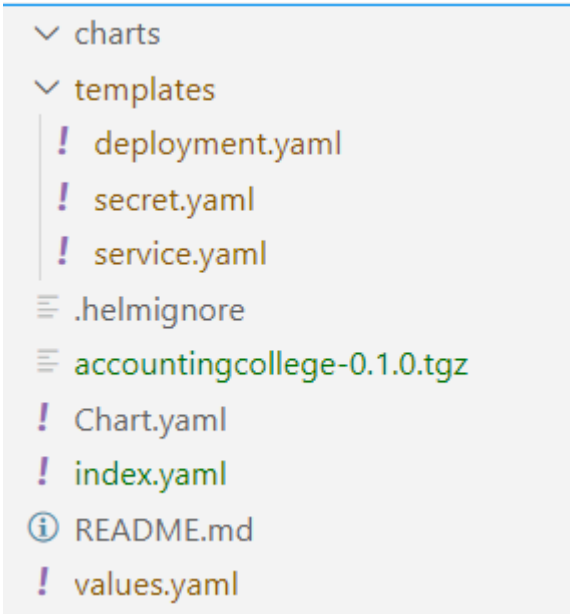

Рисунок 3.8 – Структура Helm Chart

Структура Helm Chart включає директорії та файли, які визначають ресурси Kubernetes та конфігурації, призначені для розгортання.

Зображення 3.8 демонструє інтерфейс файлового менеджера, який відображає структуру Helm chart:

- Головний каталог називається “charts”.
- Всередині каталогу “charts” знаходиться підкаталог “templates”, який містить наступні файли:
  - deployment.yaml
  - secret.yaml
  - service.yaml
- В головному каталозі “charts” також розташований файл .helmignore.
- Нарешті, в каталозі “charts” є чотири окремі файли: Chart.yaml, index.yaml, README.md та values.yaml.

Кожен файл виконує свою специфічну роль. Файл values.yaml використовується для задання значень, які можуть бути перевизначені при встановленні чи оновленні чарта. Файл Chart.yaml містить основну інформацію про чарт, а файли в каталозі “templates” використовуються для генерації Kubernetes маніфестів на основі значень, визначених в файлі values.yaml.

У Helm Charts, особливо у розгортанні додатків в Kubernetes, кілька ключових файлів відіграють важливу роль у налаштуванні та управлінні конфігурацією. Основну роль відіграють файли deployment.yaml, secret.yaml та service.yaml.

- deployment.yaml: Цей файл містить опис розгортання (Deployment) додатку в Kubernetes. Він визначає, яким чином контейнери додатку повинні бути розгорнуті, управляти, оновлювати та масштабувати. У цьому файлі зазвичай вказується ім'я контейнера, образ додатку, параметри розгортання, налаштування ресурсів та інші параметри, які визначають спосіб роботи додатку в Kubernetes–кластері.
- secret.yaml: Файл secret.yaml містить конфіденційні дані, такі як паролі, токени або ключі доступу, які використовуються додатком. Ці дані зазвичай шифруються та зберігаються в безпечному репозиторії, щоб запобігти їхньому викриттю. Secret використовується для передачі чутливої інформації до контейнерів додатку без збереження її у відкритому вигляді.
- service.yaml: Цей файл містить опис сервісу (Service) для доступу до розгорнутого додатку. Service визначає, як зовнішні клієнти можуть звертатися до додатку в Kubernetes–кластері, встановлюючи правила маршрутизації трафіку та забезпечуючи надійність доступу до додатку. У цьому файлі вказуються порти, протоколи, типи сервісу (наприклад, ClusterIP, NodePort, LoadBalancer), а також інші параметри, які регулюють роботу сервісу.

Ці три файли разом визначають архітектурну основу для розгортання додатку в Kubernetes–кластері за допомогою Helm–чарта. Вони дозволяють налаштувати, захистити та забезпечити доступ до додатку, забезпечуючи при цьому високу доступність, безпеку та ефективність роботи додатку в умовах розподіленого середовища.

Для підготовки до публікації Helm Chart необхідно згенерувати файл index.yaml, командою “helm repo index {pathtofolder}”. Вміст файлу відображений в додатку Б. Цей файл містить основну інформацію для формування та публікації Helm Chart. Необхідно упакувати усі файли в архів tgz, можливостями інструменту helm командою “helm package {pathtofolder}”.

Підготовлені файли необхідно опублікувати на GitHub, структура зображена на рисунку 3.9.

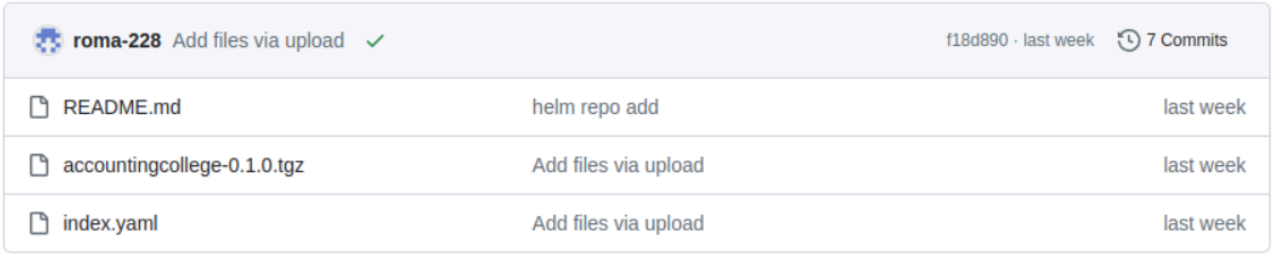

Рисунок 3.9 – Структура Helm Chart на GitHub

На фінальному етапі публікації в Helm необхідно заповнити відповідні поля у Artifact hub (рис. 3.10).

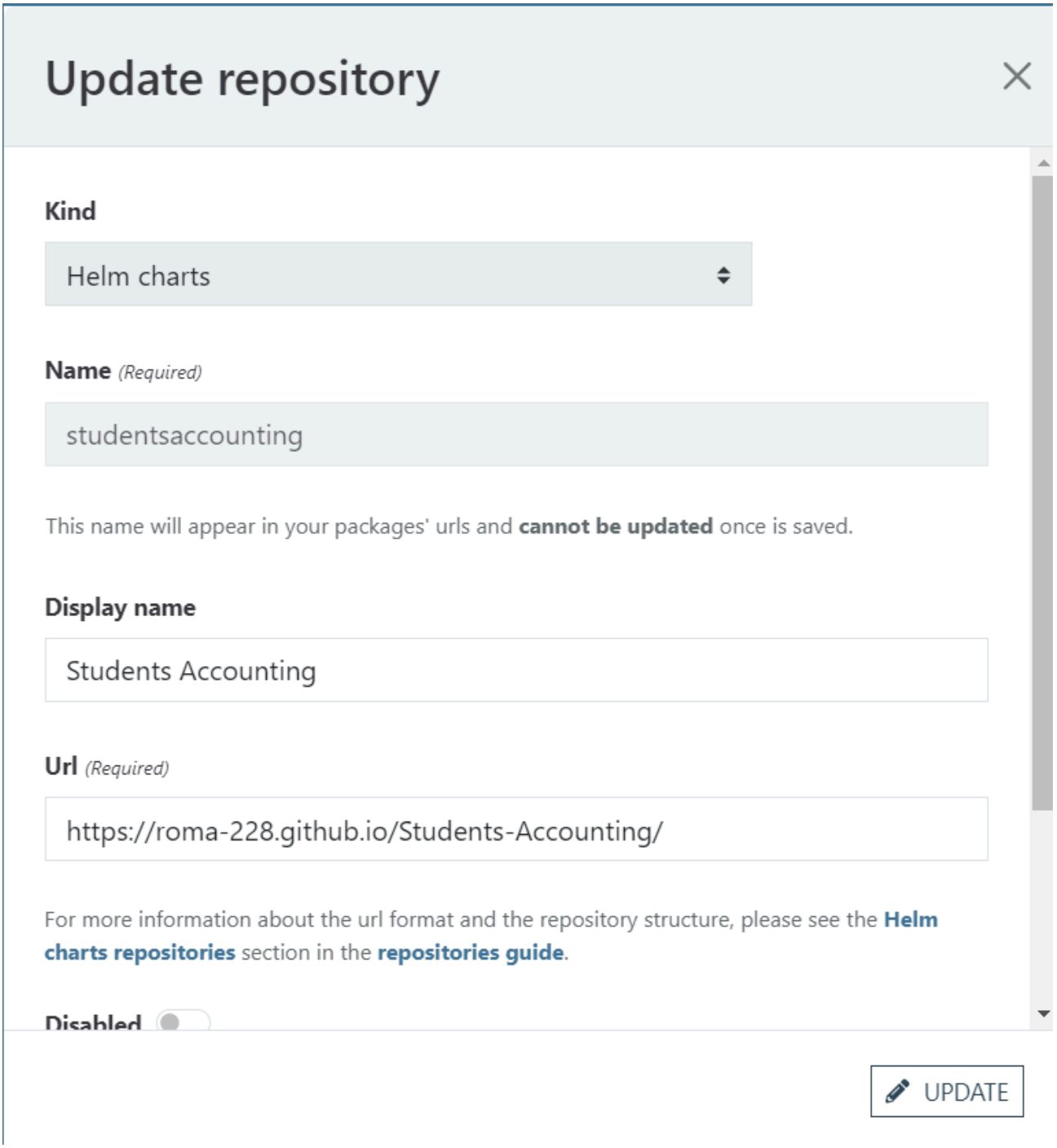

Рисунок 3.10 – Публікація Helm Chart у Artifact hub

Після успішної публікації Helm Chart, його можна буде знайти в Artifact hub (рис 3.11)

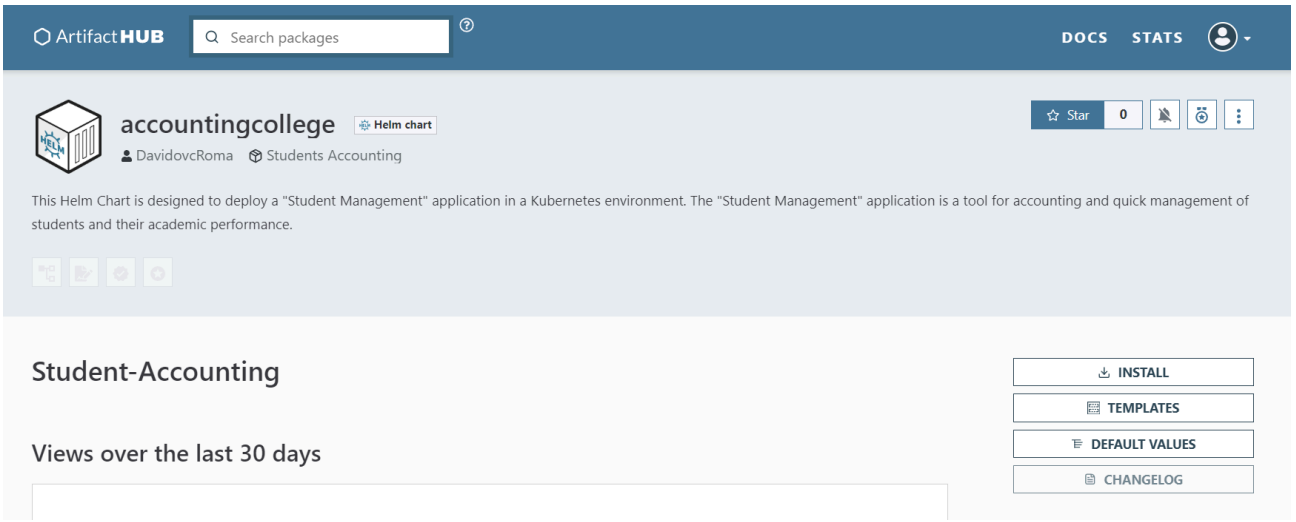

Рисунок 3.11 – Опублікований Helm Chart у Artifact hub

3.4 Автоматизоване управління кластером

Автоматизоване розгортання інфраструктури на кластері передбачає використання інструментів та методів, які дозволяють знизити трудомісткість та помилковість процесу розгортання, забезпечуючи швидке створення та налагодження середовища. Такий підхід дозволяє підвищити швидкість реакції на зміни, забезпечити високу доступність та надійність системи, а також зменшити витрати на обслуговування та управління інфраструктурою.

Автоматизація розгортання була виконана за допомогою Bash–скрипту – це файл зі скриптовою мовою програмування Bash, який містить набір команд для автоматизації різноманітних завдань у середовищі командного рядка Linux або UNIX. Використання Bash–скрипту дозволяє розробникам автоматизувати послідовні дії, такі як створення, конфігурування та запуск інфраструктури на кластері, безпосередньо з командного рядка або у скриптах для автоматизації процесів. Це забезпечує простоту та гнучкість у реалізації складних операцій, а також забезпечує можливість швидко реагувати на зміни та автоматично

виконувати рутинні завдання без необхідності ручного втручання.

Файл автоматизації наведений у додатку В. Скрипт Bash, призначений для автоматизації процесу розгортання та конфігурування інфраструктури на кластері. Він використовує команди Azure CLI та Helm для виконання наступних дій:

- Встановлення облікового запису Azure: Встановлюється обліковий запис Azure з певною підпискою для подальшої роботи.
- Отримання облікових даних для кластера AKS: Отримуються облікові дані для доступу до кластера Azure Kubernetes Service (AKS).
- Додавання репозиторію Helm–чартів Pixie Operator: Додається репозиторій Helm–чартів Pixie Operator для доступу до необхідного ПЗ.
- Оновлення репозитаріїв Helm: Виконує оновлення всіх репозитаріїв Helm, щоб мати актуальну інформацію про доступні чарти.
- Встановлення Helm–чарту Pixie Operator: Встановлюється Helm–чарт Pixie Operator на кластер AKS з певними параметрами конфігурації.
- Додавання репозиторію Helm–чартів Students Accounting: Додається репозиторій Helm–чартів Students Accounting для доступу до іншого ПЗ.
- Встановлення Helm–чарту Students Accounting: Встановлюється Helm–чарт Students Accounting з певними параметрами конфігурації, такими як налаштування підключення до бази даних.

Bash–скрипту демонструє ефективну стратегію автоматизації розгортання та конфігурування інфраструктури на кластері за допомогою команд Azure CLI та Helm. Використання Bash–скрипту спрощує процеси установки, оновлення та керування різноманітними компонентами інфраструктури, забезпечуючи

швидкість та точність виконання завдань. Автоматизація таких операцій допомагає зменшити ризики помилок під час виконання, підвищує продуктивність розробників та адміністраторів, а також сприяє створенню стабільної та масштабованої інфраструктури.

3.5 Комплексний аналіз та візуалізація результатів роботи

У рамках бакалаврської роботи було реалізовано автоматизовану систему розгортання системи обліку студентів та моніторингу веб–додатку.

Після успішного розгортання коду автоматизованого розгортання користувач отримає інформацію про розгорнутий додаток, який можна побачити в Azure на сторінці Kubernetes service у вкладці Services and ingresses (рис. 3.12). Навпроти сервісу accountingcollege–deployment буде знаходитися певна IP–адреса за якою буде знаходитися інформаційна система (рис. 3.13). Результат розгорнутого додатку зображено на рисунку 3.14.

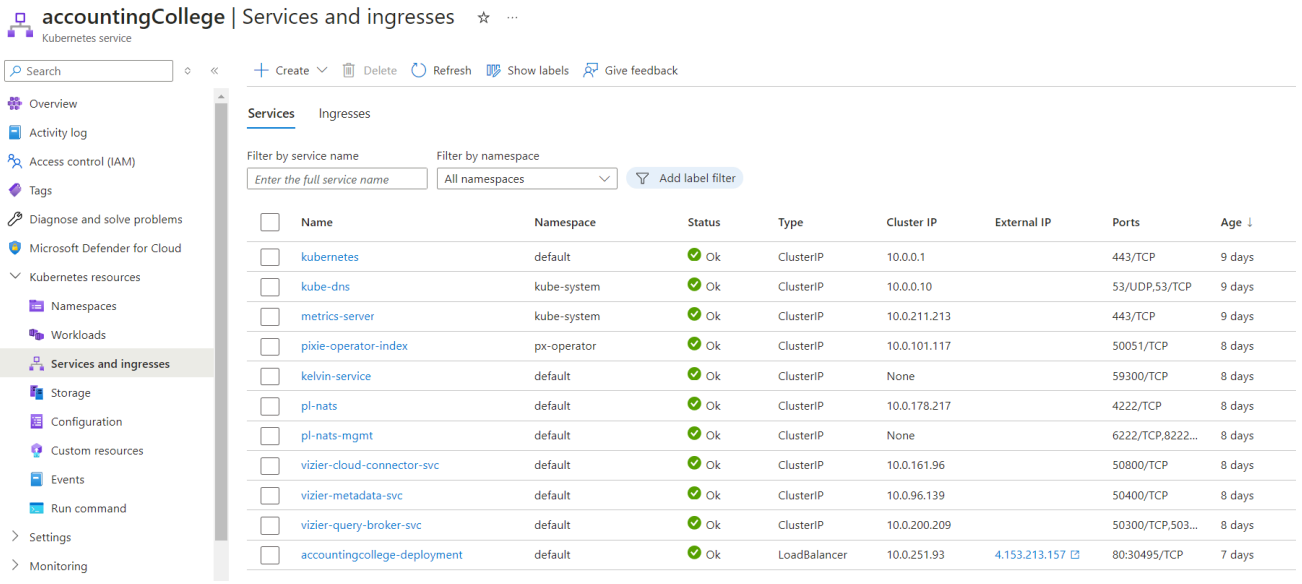

Рисунок 3.12 – Services and ingresses

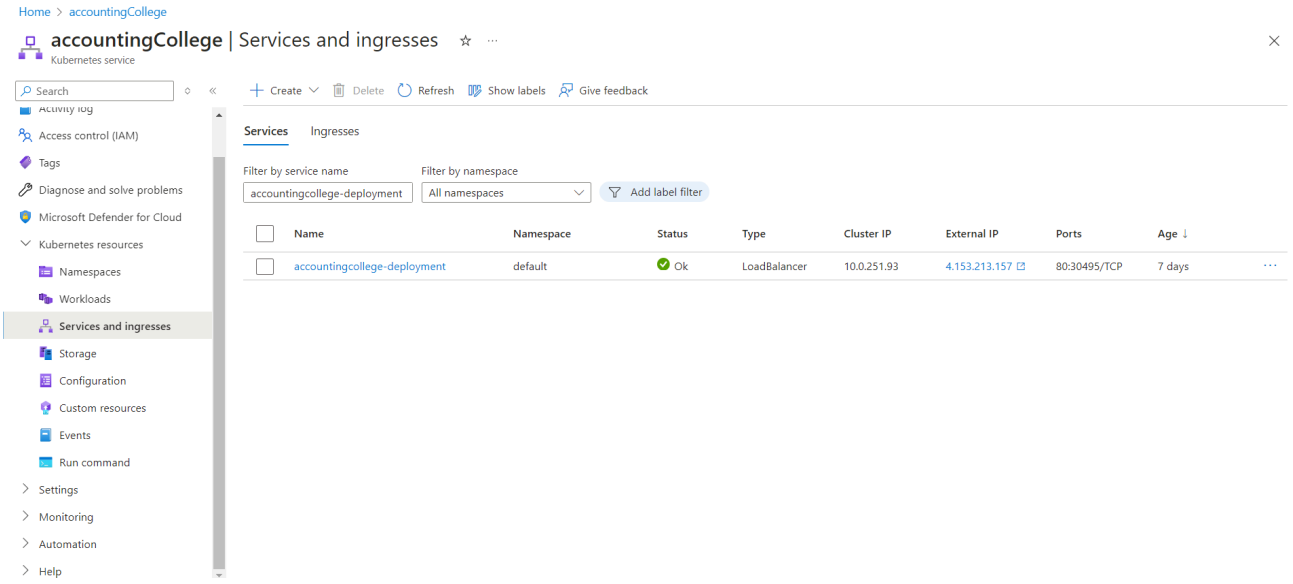

Рисунок 3.13 – IP–адреса інформаційної системи

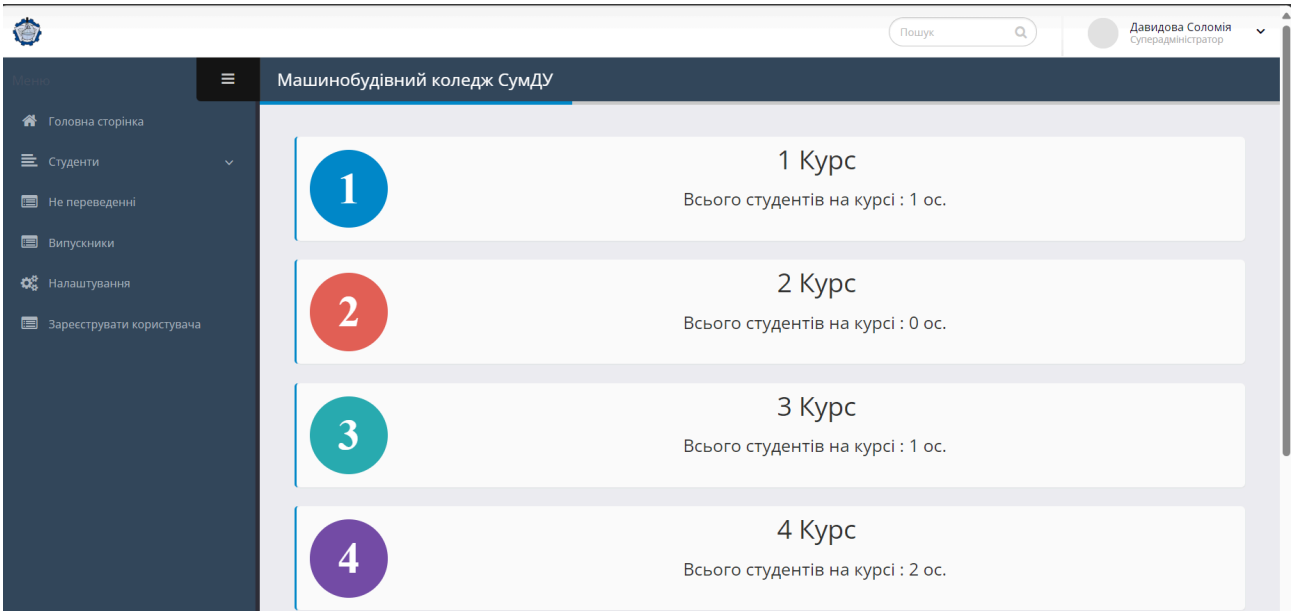

Рисунок 3.14 – Розгорнута інформаційна система

Azure має потужні інструменти для управління контейнеризованими додатками. У вкладці "Workloads" у платформі Azure AKS відображається інформація про робочі навантаження, що розгорнуті в кластері Kubernetes. Ця вкладка надає зручний інтерфейс для перегляду та управління додатками та мікрослужбами, що працюють у кластері (рис 3.15).

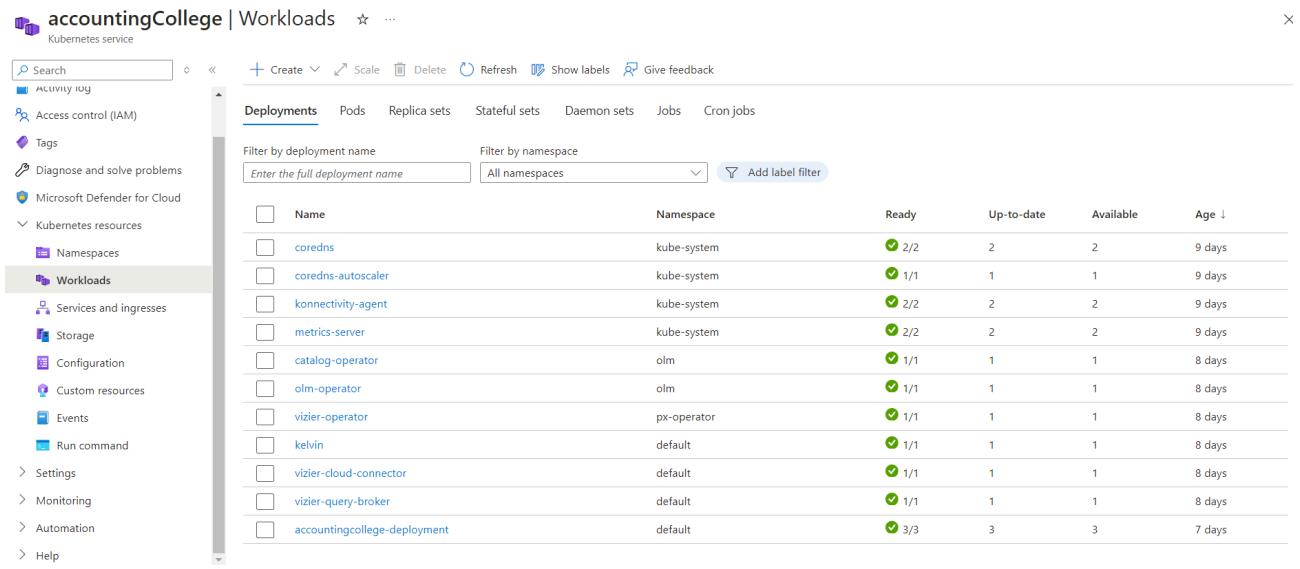

Рисунок 3.15 – Workloads

Тут можна знайти наступне:

- Інформацію про розгорнуті додатки (Deployments), зокрема їх імена, версії, кількість реплік та можливість керувати ними.
- Журнал подій (Events), який містить інформацію про події, пов'язані з робочими навантаженнями.
- Статуси розгортання (Deployment Statuses), які вказують на поточний стан додатків та мікрослужб.
- Загальний статус навантажень (Workload Status), який дає уявлення про стан системи в цілому.

Ця вкладка дозволяє зручно відстежувати та керувати робочими навантаженнями у Kubernetes–кластері на платформі Azure AKS.

Розгорнутий сайт вже моніториться інструментом Pixie зображений на зображені 3.16. Pixie має багато скриптів за допомогою яких можна більш детально відслідковувати навантаження на додаток.

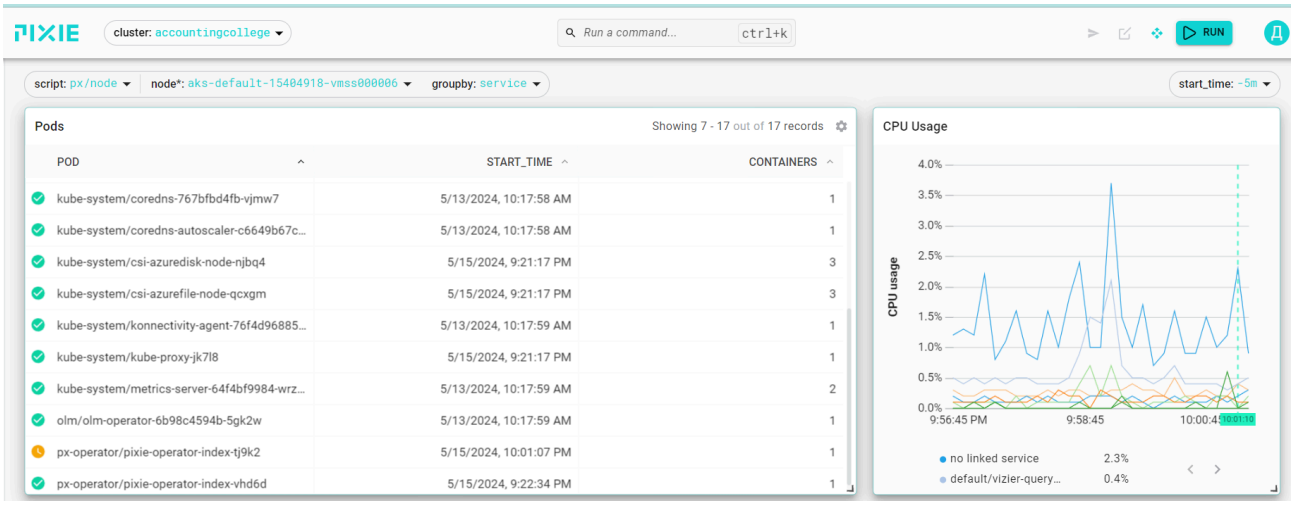

Рисунок 3.16 – Моніторинг Pixie

Було розроблено та модифіковано набір функцій на мові програмування Python для отримання та аналізу статистики роботи контейнерів у середовищі Kubernetes за допомогою бібліотеки px (рис. 3.17). Ці функції дозволяють збирати інформацію про використання ресурсів, мережевий трафік та стан контейнерів на конкретних вузлах Kubernetes. Відомості, отримані за допомогою цих функцій, можуть бути корисними для моніторингу та аналізу роботи системи, а також для ефективного управління ресурсами і виявлення можливих проблем. Код знаходиться у додатку Г.

Основна мета цього коду – отримання статистики щодо ресурсів (таких як CPU, пам'ять, диск) та мережевого трафіку, який генерують контейнери на певному вузлі Kubernetes. Крім того, він також надає можливість перегляду інформації про поточний стан контейнерів, включаючи їх кількість, час запуску та статус.

Функції у цьому коді призначені для виконання наступних завдань:

- pods_for_node: Отримання списку контейнерів, які працюють на вказаному вузлі Kubernetes разом з їхніми основними параметрами.
- resource_timeseries: Збір та агрегація часових рядів ресурсів контейнерів, таких як використання CPU та пам'яті, а також швидкість читання/запису на диск.

- network_stats: Отримання статистики щодо мережевого трафіку контейнерів, зокрема переданого та отриманого об'єму даних, а також помилок та втрат.
- stacktraces: Отримання стек–трейсів (звітів про виклики функцій) для процесів, що виконуються на вказаному вузлі Kubernetes, разом з відсотковим відношенням їх кількості до загальної кількості процесів на вузлі.

Ці функції допомагають адміністраторам моніторити та аналізувати роботу контейнерів у Kubernetes–кластері, що дозволяє ефективно керувати ресурсами та виявляти потенційні проблеми в роботі системи

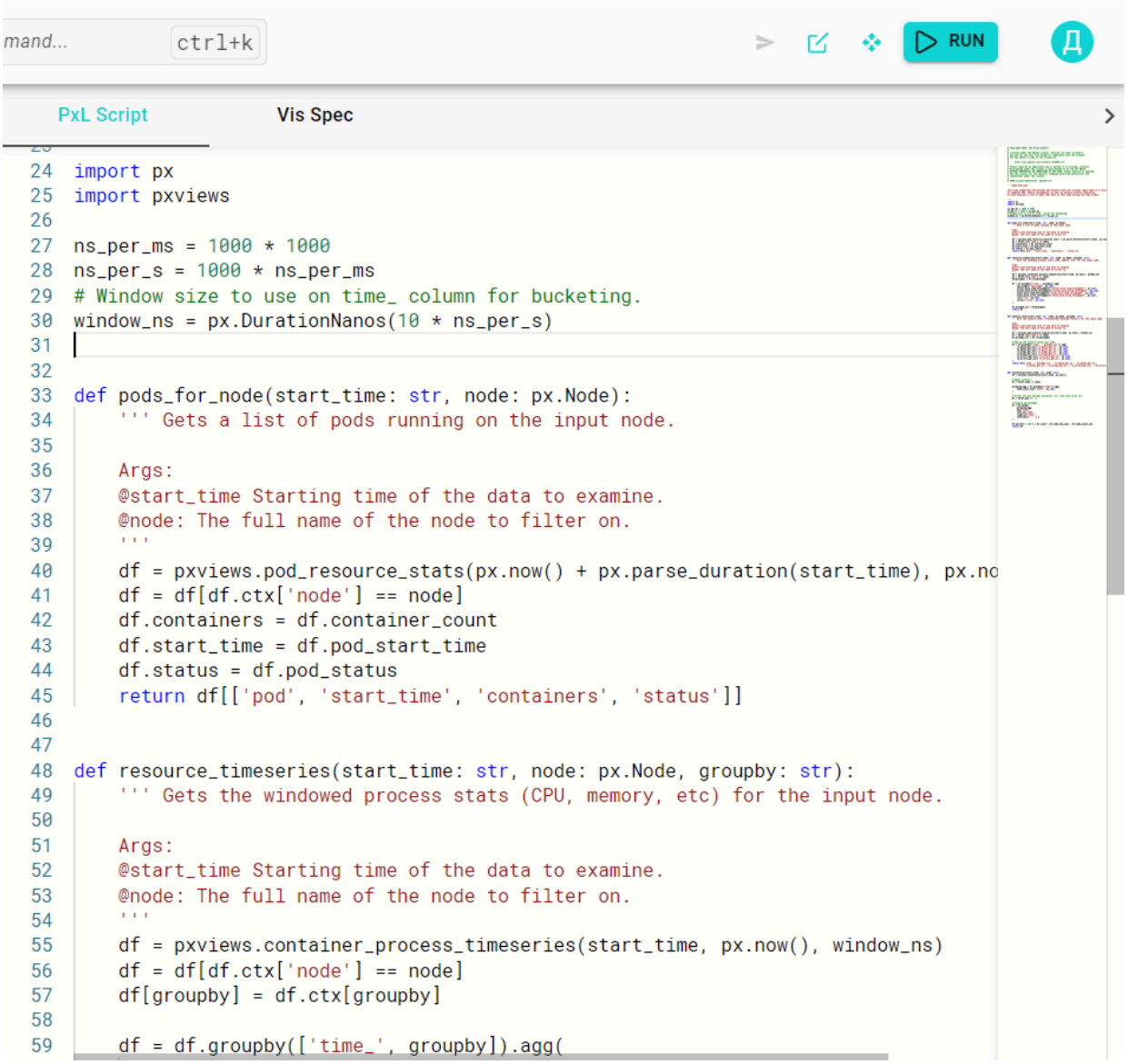

Рисунок 3.17 – Код Pixie

На зображені 3.18 зображено HTTP Service Map

Service Map в Pixie – це інструмент для візуалізації та аналізу HTTP–трафіку, що пройшов через додатки. Він допомагає зрозуміти, як ваші служби взаємодіють між собою та зовнішніми системами.

HTTP Service Map в Pixie надає наступні можливості:

- Візуалізація мережевого трафіку: HTTP Service Map дозволяє побачити всі HTTP–запити та відповіді між службами.
- Відстеження шляху запитів: Є можливість переглядати шлях, яким проходять HTTP–запити через додатки, що допомагає виявляти проблеми з продуктивністю та відстежувати підозрілу діяльність.
- Аналіз продуктивності: HTTP Service Map може надати інформацію про продуктивність додатку, включаючи час відповіді, коди відповіді та обсяг переданих даних.
- Виявлення проблем: Візуалізація HTTP–трафіку дозволяє виявляти аномальну поведінку, таку як помилки сервера, надмірні затримки тощо.

HTTP Service Map в Pixie допомагає зрозуміти структуру та здоров'я мережевого трафіку, що дозволяє ефективніше управляти та відлагоджувати додатки.

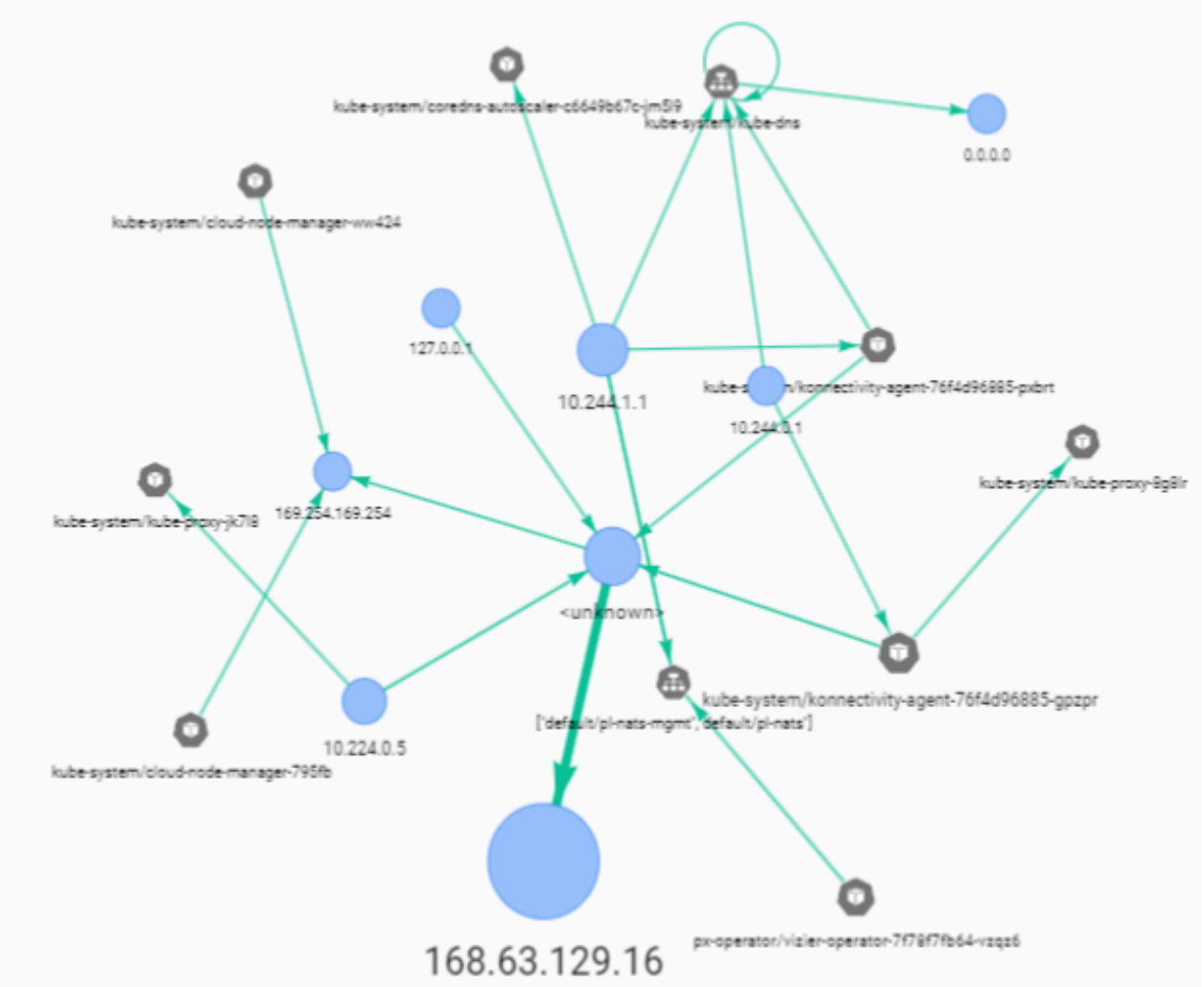

Рисунок 3.18 – HTTP Service Map

На зображені 3.19 зображені графіки які відображають різноманітне навантаження

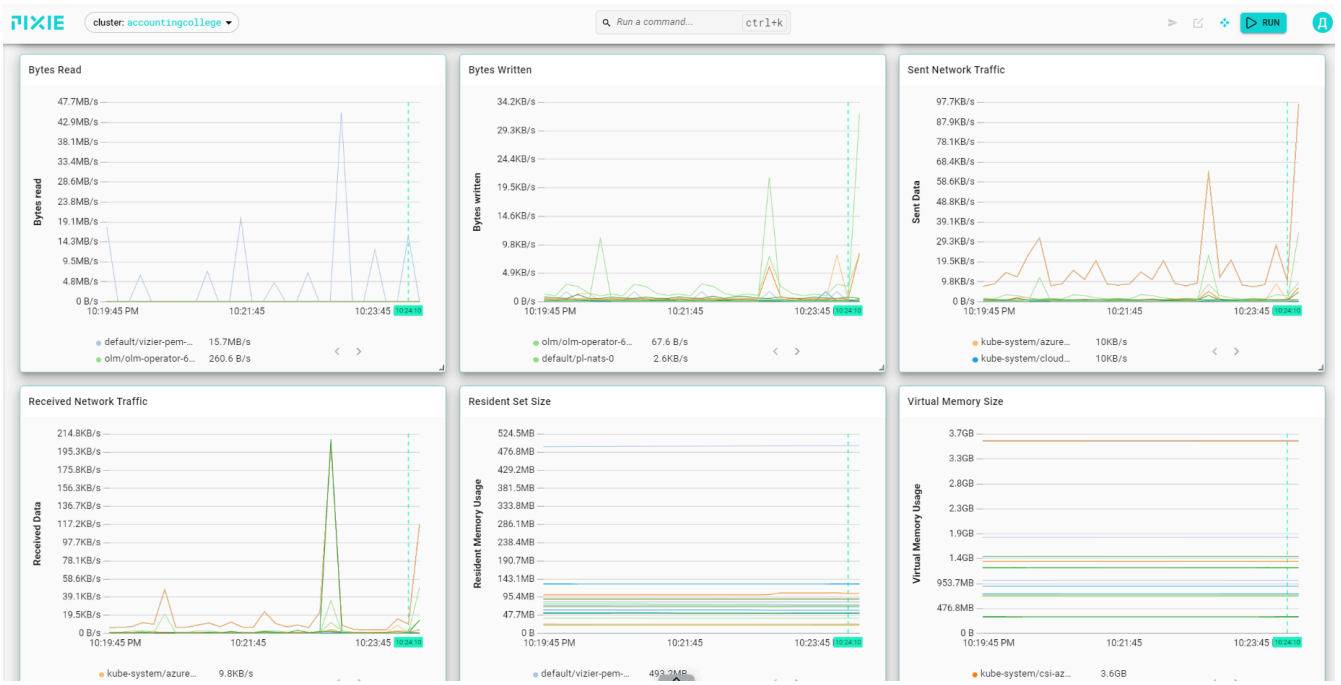

Рисунок 3.19 – Графіки навантаження

Автоматизація розгортання та моніторингу веб–додатку дозволяє швидко реагувати на зміни в середовищі, забезпечуючи при цьому стабільну та надійну роботу системи. Використання Helm–чартів та скриптів автоматизації дозволяє ефективно керувати конфігурацією системи та забезпечує швидке реагування на вимоги бізнесу та користувачів.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено аналіз та розробка автоматизованої інформаційної системи з використанням хмарних обчислень для обліку студентів. Визначено актуальність використання хмарних обчислень у сфері освіти та створення інформаційних систем для ефективного управління навчальним процесом.

У ході виконання кваліфікаційної роботи бакалавра було виконано наступні завдання

1) Комплексний аналіз потреб та вимог користувачів щодо системи обліку студентів.

2) Дослідження сучасних підходів та технологій у галузі хмарних обчислень та автоматизованого розгортання додатків.

3) Розроблено архітектуру системи, включаючи в себе компоненти, базу даних, інтерфейс користувача та логіку бізнес–процесів.

4) Виконано імплементацію інформаційної системи згідно з визначеною архітектурою та вимогами.

5) Створено Helm Charts для автоматизації розгортання додатків, що дозволить значно спростити процес розгортання та керування інформаційною системою обліку студентів

6) Протестовано розроблену систему з метою виявлення та виправлення можливих проблем та недоліків.

Отже, можна зробити висновок, що розроблена інформаційна система на основі хмарних обчислень є ефективним інструментом для автоматизації обліку студентів у навчальних закладах, що дозволить оптимізувати процеси управління та підвищити їхню продуктивність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is software? [Електроний ресурс] - URL:https://www.techtarget.com/searchapparchitecture/definition/software (дата звернення: 09.04.2024).

2. What is deployment automation? [Електроний ресурс] - URL: https://www.redhat.com/en/topics/automation/what-is-deployment-automation (дата звернення: 12.04.2024).

3. Jenkins official documentation [Електроний ресурс] - URL: https://www.jenkins.io/ (дата звернення: 15.04.2024).

4. Ansible official documentation [Електроний ресурс] - URL: https://www.ansible.com/ (дата звернення: 18.04.2024).

5. Docker official documentation [Електроний ресурс] - URL: https://www.docker.com/ (дата звернення: 21.04.2024).

6. Kubernetes official documentation [Електроний ресурс] - URL: https://kubernetes.io/ (дата звернення: 24.04.2024).

7. What is CI/CD? [Електроний ресурс] - URL: https://www.redhat.com/en/topics/devops/what-is-ci-cd (дата звернення: 27.04.2024).

8. What is a data pipeline? [Електроний ресурс] - URL: https://www.ibm.com/topics/data-pipeline (дата звернення: 30.04.2024).

9. What is blue green deployment? [Електроний ресурс] - URL: https://www.redhat.com/en/topics/devops/what-is-blue-green-deployment (дата звернення: 03.05.2024).

10. What Is Canary Deployment? [Електроний ресурс] - URL: https://semaphoreci.com/blog/what-is-canary-deployment (дата звернення: 05.05.2024).

11. What is A/B testing? [Електроний ресурс] - URL: https://vwo.com/ab-testing/ (дата звернення: 10.04.2024).

12. What is containerization? [Електроний ресурс] - URL: https://www.ibm.com/topics/containerization (дата звернення: 13.04.2024).

13. Use containers to Build, Share and Run your applications [Електроний ресурс] - URL: https://www.docker.com/resources/what-container/ (дата звернення: 16.04.2024).

14. What is container orchestration? [Електроний ресурс] - URL: https://www.redhat.com/en/topics/containers/what-is-container-orchestration (дата звернення: 19.04.2024).

15. What is infrastructure as code (IaC)? [Електроний ресурс] - URL: https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code (дата звернення: 22.04.2024).

16. Visualizing the Cloud: Mastering AWS Diagrams for Effective Architecture Design and Communication [Електронний ресурс] - URL: https://medium.com/@maheshwar.ramkrushna/visualizing-the-cloud-mastering-aws-diagrams-for-effective-architecture-design-and-communication-f4bec263bde2 (дата звернення: 20.05.2024).

17. What is IaaS (Infrastructure-as-a-Service)? [Електроний ресурс] - URL: https://www.ibm.com/topics/iaas (дата звернення: 28.04.2024).

18. What is Platform-as-a-Service (PaaS)? [Електроний ресурс] - URL: https://www.ibm.com/topics/paas (дата звернення: 01.05.2024).

19. What is SaaS (Software as a Service)? [Електроний ресурс] - URL: https://www.salesforce.com/saas/ (дата звернення: 04.05.2024).

20. Google Cloud Platform [Електроний ресурс] - URL: https://cloud.google.com/ (дата звернення: 09.04.2024).

21. Amazon Web Services [Електроний ресурс] - URL: https://aws.amazon.com/what-is-aws/ (дата звернення: 12.04.2024).

22. Microsoft Azure [Електроний ресурс] - URL: https://azure.microsoft.com/en-us/ (дата звернення: 15.04.2024).

23. IBM Cloud [Електроний ресурс] - URL:https://www.ibm.com/ (дата звернення: 18.04.2024).

24. Cloud data warehouse to power your data-driven innovation [Електроний ресурс] - URL: https://cloud.google.com/bigquery/?hl=uk (дата звернення: 21.04.2024).

25. TensorFlow [Електроний ресурс] - URL: https://www.tensorflow.org/ (дата звернення: 24.04.2024).

26. Cloud Foundry, The Proven Development Platform For Cloud-Native Applications [Електроний ресурс] - URL: https://www.cloudfoundry.org/ (дата звернення: 27.04.2024).

27. Apache Beam [Електроний ресурс] - URL: https://beam.apache.org/ (дата звернення: 30.04.2024).

28. Snowball [Електроний ресурс] - URL: https://snowballstem.org/ (дата звернення: 03.05.2024).

29. Outposts [Електроний ресурс] - URL: https://docs.aws.amazon.com/outposts/latest/userguide/work-with-outposts.html (дата звернення: 05.05.2024).

30. Capture the Flag Platform on Azure PaaS [Електронний ресурс] - URL: https://learn.microsoft.com/en-us/azure/architecture/example-scenario/apps/capture-the-flag-platform-on-azure-paas (дата звернення: 20.05.2024).

31. Azure Multitenant SaaS on Azure [Електронний ресурс] - URL: https://slides365.com/azure-multitenant-saas-on-azure/ (дата звернення: 20.05.2024).

32. Software Development [Електронний ресурс] - URL: https://www.wozavez.com/software-development (дата звернення: 20.05.2024).

33. Helm Best Practices [Електронний ресурс] - URL: https://blog.stackademic.com/helm-best-practices-93326ff8cbed (дата звернення: 20.05.2024).

Додаток А Dockerfile

```
FROM php:7.2-apache

RUN apt-get update

RUN docker-php-ext-install mysqli pdo

RUN docker-php-ext-enable mysqli pdo

COPY . /var/www/html/

RUN apt-get install -y \
    git \
    zip \
    unzip \
    && rm -rf /var/lib/apt/lists/*

RUN curl -sS https://getcomposer.org/installer | php --
--install-dir=/usr/local/bin --filename=composer

RUN composer install

RUN a2enmod rewrite

RUN chown -R www-data:www-data /var/www/html/

EXPOSE 80

CMD ["apache2-foreground"]
```

Додаток Б index.yaml Helm Chart

```yaml
apiVersion: v1
entries:
  accountingcollege:
  - apiVersion: v2
    appVersion: 1.16.0
    created: "2024-05-09T22:11:37.3328814+03:00"
    description: This Helm Chart is designed to deploy a "Student Management" application
      in a Kubernetes environment. The "Student Management" application is a tool
      for accounting and quick management of students and their academic performance.
    digest: 5f33cff08d7e32ab8d99f59404bfcb9a677f342167154b49110448bc166d7a1c
    name: accountingcollege
    type: application
    urls:
    - accountingcollege-0.1.0.tgz
    version: 0.1.0
generated: "2024-05-09T22:11:37.3318367+03:00"
```

Додаток В Автоматизація розгортання інфраструктури на кластері

```
#!/bin/bash

# Set the Azure account
az account set --subscription 8b0e9c10-1b38-42ff-b84d-f20e71474eaa

# Get the credentials for the AKS cluster
az aks get-credentials --resource-group accountingCollege --name accountingCollege --overwrite-existing

# Add the Pixie Operator helm chart repository
helm repo add pixie-operator https://artifacts.px.dev/helm_charts/operator

# Update the helm repositories
helm repo update

# Install the Pixie Operator helm chart
helm install pixie pixie-operator/pixie-operator-chart \
    --set deployKey='px-dep-9700c608-4d13-43b5-8764-ee63bfa0fd82' \
    --set clusterName='accountingCollege'

# Add the Students Accounting helm chart repository
helm repo add studentsaccounting https://roma-228.github.io/Students-Accounting/

# Install the Students Accounting helm chart
helm install my-accountingcollege studentsaccounting/accountingcollege \
    --version 0.1.0 \
    --set accounting.DB_HOST='YWNjb3VudGluZ2NvbGxlZ2UubXlzcWwuZGF0YWJhc2UuYXp1cmUuY29tCg==' \
    --set accounting.DB_USER='YWNjb3VudGluZ2NvbGxlZ2UK' \
    --set accounting.DB_PASSWORD='aEN4WHI0ZDhqYjl2NGUK' \
    --set accounting.DB_DATABASE='YWNjb3VudGluZ2NvbGxlZ2UK'
```

Додаток Г Код Pixie

```
import px
import pxviews

ns_per_ms = 1000 * 1000
ns_per_s = 1000 * ns_per_ms
# Window size to use on time_ column for bucketing.
window_ns = px.DurationNanos(10 * ns_per_s)


def pods_for_node(start_time: str, node: px.Node):
    ''' Gets a list of pods running on the input node.

    Args:
    @start_time Starting time of the data to examine.
    @node: The full name of the node to filter on.
    '''
    df = pxviews.pod_resource_stats(px.now() + px.parse_duration(start_time), px.now())
    df = df[df.ctx['node'] == node]
    df.containers = df.container_count
    df.start_time = df.pod_start_time
    df.status = df.pod_status
    return df[['pod', 'start_time', 'containers', 'status']]


def resource_timeseries(start_time: str, node: px.Node, groupby: str):
    ''' Gets the windowed process stats (CPU, memory, etc) for the input node.

    Args:
    @start_time Starting time of the data to examine.
    @node: The full name of the node to filter on.
    '''
    df = pxviews.container_process_timeseries(start_time, px.now(), window_ns)
    df = df[df.ctx['node'] == node]
    df[groupby] = df.ctx[groupby]

    df = df.groupby(['time_', groupby]).agg(
        cpu_usage=('cpu_usage', px.sum),
        actual_disk_read_throughput=('actual_disk_read_throughput', px.sum),
        actual_disk_write_throughput=('actual_disk_write_throughput', px.sum),
        total_disk_read_throughput=('total_disk_read_throughput', px.sum),
        total_disk_write_throughput=('total_disk_write_throughput', px.sum),
        rss=('rss', px.sum),
        vsize=('vsize', px.sum),
    )

    df.groupby_col = df[groupby]
    return df
```

```
def network_stats(start_time: str, node: px.Node, groupby: str):
    ''' Gets the network stats (transmitted/received traffic) for the input node.

    Args:
    @start_time Starting time of the data to examine.
    @node: The full name of the node to filter on.
    '''
    df = pxviews.pod_network_timeseries(start_time, px.now(), window_ns)
    df = df[df.ctx['node'] == node]
    df.groupby_col = df.ctx[groupby]

    # Add up the network values per node.
    df = df.groupby(['time_', 'groupby_col']).agg(
        rx_bytes_per_ns=('rx_bytes_per_ns', px.sum),
        tx_bytes_per_ns=('tx_bytes_per_ns', px.sum),
        rx_drops_per_ns=('rx_drops_per_ns', px.sum),
        tx_drops_per_ns=('tx_drops_per_ns', px.sum),
        rx_errors_per_ns=('rx_errors_per_ns', px.sum),
        tx_errors_per_ns=('tx_errors_per_ns', px.sum),
    )
    return df[['time_', 'groupby_col', 'rx_bytes_per_ns', 'tx_bytes_per_ns',
               'rx_drops_per_ns', 'tx_drops_per_ns', 'rx_errors_per_ns', 'tx_errors_per_ns']]
```

```
def stacktraces(start_time: str, node: str):
    df = pxviews.stacktraces(start_time, px.now())

    # Apply filters.
    df = df[df.node == node]

    grouping_agg = df.groupby(['node']).agg(
        node_count_sum=('count', px.sum),
    )

    # Filter out any non-k8s processes (it's now safe to do so).
    df = df[df.pod != '']

    # Compute percentages.
    df = df.merge(
        grouping_agg,
        how='inner',
        left_on='node',
        right_on='node',
        suffixes=['', '_x']
    )

    df.percent = 100.0 * df.count * df.node_num_cpus / df.node_count_sum
    return df
```