КВАЛІФІКАЦІЙНА РОБОТА

Група МІПЗ-2022

Вацик Ю.Ю.

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Вацик Юрій Юрійович

УДК 004.453

Дослідження та аналіз актуальності застосування Kubernetes в умовах сучасного IT ринку

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації магістра

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Вацик Ю.Ю.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

_____ к.ф-м.н., доц. Бойчук А.М.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «магістр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

« 19 » лютого 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Вацик Юрій Юрійович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Дослідження та аналіз актуальності застосування Kubernetes в умовах сучасного IT ринку

керівник роботи:
Бойчук Андрій Михайлович, кандидат фізико-математичних наук, доцент

затверджена наказом вищого навчального закладу від «26» червня 2023 року № 32/1 с

2. Термін подання студентом роботи 16.02.2024

3. Вихідні дані роботи: технологія оркестрації контейнерів Kubernetes, інструмент інфраструктури як коду Terraform, менеджер пакетів Helm

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
   1. Дослідити теоретичні основи використання технології Kubernetes;
   2. Виокремити тренди застосування Kubernetes;
   3. Проаналізувати практичні приклади використання компаніями Kubernetes;
   4. Розгорнути Kubernetes на базі одного із хмарних провайдерів

5. Дата видачі завдання 29.06.2023

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Дослідження теоретичних основ	15.08.2023	Виконано
2.	Аналіз трендів застосування Kubernetes	16.09.2023	Виконано
3.	Дослідження прикладів використання Kubernetes компаніями у практичній діяльності	09.10.2023	Виконано
4.	Розгортання Kubernetes на базі одного із хмарних провайдерів	26.10.2023	Виконано
5.	Оформлення пояснювальної записки	14.11.2023	Виконано
6.	Оформлення графічного матеріалу та підготовка до захисту роботи	07.12.2023	Виконано

Студент _____ Вацик Ю.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бойчук А.М.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
11	Рисунок 1.1 – Відмінності елементів систем віртуальних машинах та Docker контейнерах, а також їх ізоляція	15	Рисунок 1.2 – Мікросервісна архітектура в Kubernetes
17	Рисунок 1.3 – Архітектура Kubernetes та її зв'язок із базовою інфраструктурою	21	Рисунок 1.4 – Найпопулярніші середовища для роботи кластерів Kubernetes у всьому світі
27	Рисунок 2.1 – Тенденції розміщення кластерів в хмарі та локально	28	Рисунок 2.2. Тенденції використання мультихмарних середових для розміщення кластерів Kubernetes
30	Рисунок 2.3 – Бізнес переваги Kubernetes	31	Рисунок 2.4 – Покращення від використання Kubernetes
32	Рисунок 2.5 – Операційні переваги Kubernetes	33	Рисунок 2.6 – Типи робочих навантажень, які виконуються в кластері Kubernetes

АНОТАЦІЯ

Кваліфікаційна робота присвячена вивченню технології Kubernetes та її впливу на розробку програмного забезпечення.

У першому розділі роботи розглядаються теоретичні основи використання Kubernetes, зокрема контейнеризацію та її вплив на розробку програмного забезпечення.

В другому розділі проведено дослідження трендів застосування Kubernetes, висвітлено поточний стан впровадження технології, фактори, які спонукають до використання, а також проблеми та перешкоди.

Третій розділ аналізує практику використання Kubernetes через приклади реалізації та історії успіху впровадження. Робота докладно розглядає використання Kubernetes в хмарних середовищах. Висвітлює отримані уроки та висновки з процесу впровадження Kubernetes.

Останній розділ присвячено розробленню та імплементації Kubernetes кластера на базі Amazon Elastic Kubernetes Service. Здійснено аналіз вимог до кластера, вивчено процес автоматизації розгортання кластера за допомогою Terraform та Helm, налаштовано системні компоненти кластера та розглянуто використання Argo CD для Continuous Delivery.

КЛЮЧОВІ СЛОВА: КОНТЕЙНЕРИЗАЦІЯ, ОРКЕСТРАЦІЯ КОНТЕЙНЕРІВ, KUBERNETES, ХМАРНІ СЕРЕДОВИЩА, АВТОМАТИЗАЦІЯ РОЗГОРТАННЯ, TERRAFORM, HELM.

SUMMARY

The research paper is devoted to the study of Kubernetes technology and its impact on software development.

The first section of the paper examines the theoretical foundations of using Kubernetes, including containerization and its impact on software development.

In the second chapter, a study of trends in the use of Kubernetes was carried out, the current state of technology implementation, factors that encourage its use, as well as problems and obstacles are highlighted.

The third section analyzes Kubernetes usage practices through implementation examples and success stories. The work examines the use of Kubernetes in cloud environments in detail. Highlights lessons learned and takeaways from the Kubernetes implementation process.

The last section is devoted to the development and implementation of a Kubernetes cluster based on Amazon Elastic Kubernetes Service. An analysis of the requirements for the cluster was performed, the process of automating the deployment of the cluster using Terraform and Helm was studied, the system components of the cluster were configured, and the use of Argo CD for Continuous Delivery was considered.

KEYWORDS: CONTAINERIZATION, CONTAINER ORCHESTRATION, KUBERNETES, CLOUD ENVIRONMENTS, DEPLOYMENT AUTOMATION, TERRAFORM, HELM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AKS	–	Azure Kubernetes Service
ALB	–	Application Load Balancers
AWS	–	Amazon Web Services
EKS	–	Amazon Elastic Kubernetes Service
GCP	–	Google Cloud Platform
GKE	–	Google Kubernetes Engine
IAM	–	AWS Identity Access Management
NLB	–	Network Load Balancers
OKE	–	Oracle Kubernetes Engine
ІТ	–	Інформаційні технології
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення

ВСТУП

Актуальність теми. Kubernetes, як платформа оркестрації контейнерів з відкритим кодом, отримала широке поширення завдяки своїй здатності спрощувати й автоматизувати розгортання, масштабування та керування контейнерними програмами та забезпечувати ефективне використання ресурсів.

Kubernetes спрощує розгортання та керування контейнерними програмами, надаючи стандартизований та ефективний спосіб організації та масштабування програм. Розуміння його актуальності допомагає організаціям оптимізувати використання ресурсів, скоротити час простою та підвищити загальну операційну ефективність.

У динамічному ландшафті сучасного IT-ринку масштабованість і гнучкість є вирішальними. Kubernetes забезпечує автоматичне масштабування програм на основі попиту, забезпечуючи оптимальне використання ресурсів. Дослідження в цій галузі допомагають організаціям адаптувати свою інфраструктуру відповідно до мінливих вимог бізнесу без зайвих витрат.

Отже, дослідження та аналіз актуальності використання Kubernetes на сучасному ринку IT є важливими для організацій, які прагнуть залишатися конкурентоспроможними, гнучкими та безпечними в технологічному середовищі, що швидко розвивається. Прийняття Kubernetes може дати можливість компаніям використовувати весь потенціал контейнерезованих додатків, одночасно вирішуючи проблеми масштабованості, гнучкості та ефективності в цифрову еру.

Окрім того, наукова важливість дослідження Kubernetes на сучасному IT-ринку полягає в його потенціалі для просування знань у різних сферах, включаючи розподілені системи, управління ресурсами, автономні обчислення, безпеку та розробку, керовану спільнотою. Це дослідження може сприяти розумінню проблем і рішень, пов’язаних із розгортанням і керуванням контейнерними програмами в складних динамічних IT-середовищах.

Мета і завдання дослідження. Метою цієї роботи є дослідження та аналіз актуальності використання Kubernetes на сучасному ринку IT.

Для досягнення мети сформульовані наступні завдання роботи:

- визначити основні характеристики системи оркестрації контейнерів Kubernetes;
- вивчити найбільш поширені приклади розгортання кластерів Kubernetes в хмарних середовищах;
- дослідити тренди застовування Kubernetes, зокрема фактори, що спонукають до його використання і проблеми та перешкоди для впровадження;
- проаналізувати практики та приклади використання Kubernetes із виокремлення історій успіху та отриманих уроків в його застосуванні;
- розробити та імплементувати Kubernetes кластер на базі Amazon Elastic Kubernetes Service, із врахування бізнес вимог та використанням автоматизації розгортання кластеру та програмних застосунків.

Об’єктом дослідження є процес впровадження та застосування систем оркестрації контейнерів.

Предметом дослідження виступає використання технології оркестрації контейнерів Kubernetes та її вплив на сучасний IT ринок.

Методи дослідження – кейс-метод, технічний та економічний аналіз, практична розробка та імплементація.

Наукова новизна одержаних результатів полягає у виокремлених тенденціях розвитку технології Kubernetes, аналізу їхнього впливу на діяльність та розвиток компанії в умовах сучасного IT ринку, а також імплементація кластера Kubernetes в реальних проектах. Основні результати, які були отримані в процесі вирішення поставлених завдань та становлять наукову новизну дослідження, полягають у наступному:

1) Дістало подальший розвиток:

- визначення основних характеристик технології оркестрації контейнерів Kubernetes із врахування специфіки та динаміки IT ринку останніх років, зокрема розвитку хмарних технологій;

- структурування та дослідження чинників, які стимулюють та сприяють еволюції Kubernetes, а також проблем та перешкод, які можуть виникнути під час впровадження цієї технології;
- аналіз історій успішного впровадження, конкретних сценаріїв використання та практичної реалізації технології Kubernetes, із виділенням отриманих уроків та висновків, що можуть бути використані іншими компаніями;

2) Удосконалено: практичні та методичні рекомендації щодо розгортання Kubernetes кластера на базі Amazon Elastic Kubernetes Service, із застосування автоматизації розгортання за допомогою Terraform, налаштування системних компонентів (Cluster Autoscaler, Metrics Server, AWS Load Balancer Controller, Cert Manager) та використанням GitOps методології для Continuous Delivery процесу.

Практичне значення одержаних результатів. Практичне значення одержаних результатів полягає у науковій обґрунтованості та прикладній спрямованості теоретичних положень, підходів і рекомендацій, викладених у роботі, використання яких може поліпшити процес розробки, впровадження та підтримки програмного забезпечення з використанням технології Kubernetes.

Апробація результатів дослідження. Результати цієї роботи впроваджено в роботу Компанії “Untapped Global”, що підтверджується листом від “02” січня 2024 року.

Структура. Ця робота складається із вступу, чотирьох розділів, висновків, додатків і списку використаних джерел. Загальний обсяг дослідження становить 94 сторінки друкованого тексту. Список використаних джерел містить 72 позицій і розміщений на 8 сторінках.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ВИКОРИСТАННЯ ТЕХНОЛОГІЇ KUBERNETES

1.1. Контейнеризація та її вплив на розробку програмного забезпечення

Контейнеризація – це технологічна інновація, що змінила підхід до розробки та розгортання програмного забезпечення. Вона полягає в упакуванні програми та всіх її залежностей в єдиний блок, відомий як контейнер. Ці контейнери створюють однорідне та ізольоване середовище, яке забезпечує надійну та послідовну роботу програми в різних контекстах, таких як розробка, тестування та виробництво. Наприклад, із бібліотеки образів DockerHub можна завантажити понад 9 млн одиниць різного ПЗ у вигляді контейнерів. Найпопулярніші контейнери, що мають понад мільярд завантажень: Ubuntu, redis, alpine, node та postgres [71].

У вересні 2021 року компанія “Dell” доручила Aberdeen Strategy and Research провести опитування сотень осіб, які приймають рішення в IT, які мають досвід вибору або розгортання контейнерів. Мета була краще зрозуміти, як і чому контейнери та Kubernetes розгортаються на підприємствах середнього розміру, а також у великих підприємствах, оцінити переваги продуктивності, пов’язані з контейнерами, і виявити проблеми, пов’язані з Kubernetes і контейнерними середовищами. Опитування показало, що в середньому понад 50% застосунків є контейнерними [16]. Також, у 2022 році компанія “Docker” встановила, що 44% розробників використовують деякі форми безперервної інтеграції та розробки з контейнерами [64]. Згідно із звітом “State of Cloud Native Development” станом на 2020 рік 60 відсотків бек-енд-розробників використовували контейнери [59, ст. 7].

Контейнеризація суттєво впливає на процес розробки програмного забезпечення. Зокрема, контейнери гарантують, що програмне забезпечення працює однаково в різних середовищах, зменшуючи горезвісну проблему “воно

працює на моїй машині”. Розробники можуть узгоджено створювати, тестувати та розгортати програми, що зменшує кількість помилок і полегшує співпрацю між членами команди. У контейнерах усі залежності, необхідні для програми, об’єднуються в контейнер. На основі проведеного дослідження “Відмінності в продуктивності між контейнеризацією та віртуалізацією з акцентом на HTTP-запити”, Йоханнес Берггрен та Єнс Карлссон вказують, що налаштування віртуальної машини займає більше часу, ніж налаштування контейнера Docker [36]. Це означає, що розробникам не потрібно турбуватися про керування складними налаштуваннями залежностей, що забезпечує швидші цикли розробки та менше часу, витраченого на налаштування середовищ.

Також, контейнери дозволяють легко створювати легкі та ізольовані середовища розробки. Розробники можуть швидко запускати контейнери для тестування нових функцій або виправлення помилок, не впливаючи на інші елементи системи. Детальніше дивіться на рисунку 1.

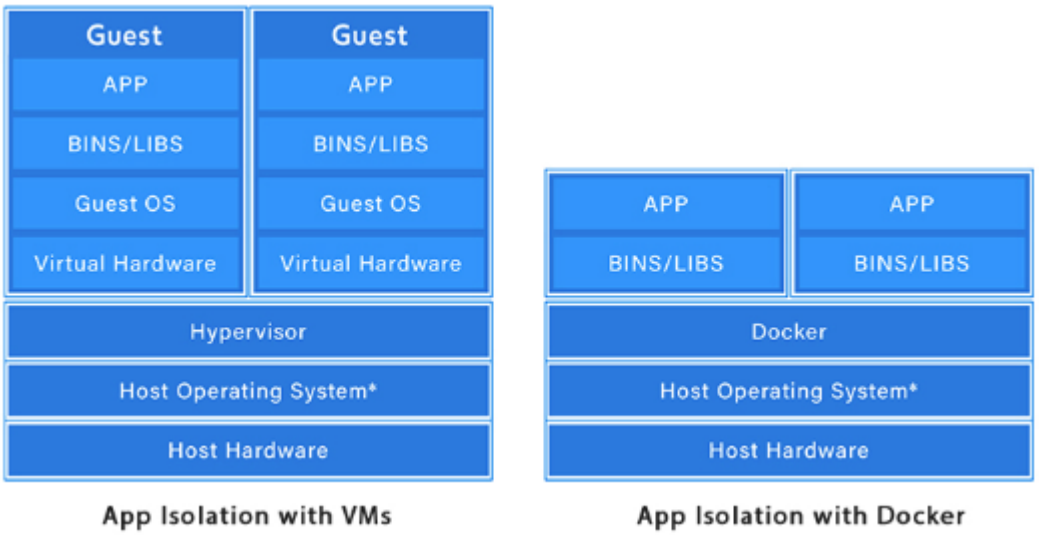


Рисунок 1.1 – Відмінності елементів систем віртуальних машинах та Docker контейнерах, а також їх ізоляція

Слід виділити, що контейнери розроблені таким чином, щоб бути легкими та використовувати ядро основної ОС. Вони споживають менше ресурсів порівняно з традиційними віртуальними машинами, що робить їх

ефективнішими з точки зору використання пам’яті та ЦП. Це дозволяє програмам легко масштабуватися як вертикально (в межах одного хосту), так і горизонтально (на кількох хостах).

Клаус Паль, Антоніо Брогі, Якопо Солдані та Пуян Джамшіді у статті “Cloud Container Technologies: a State-of-the-Art Review” вказують, що результати останніх досліджень показують зростаючий інтерес і використання технологій на основі контейнерів (таких як LXC або Docker) як легких рішень віртуалізації на рівні інфраструктури як послуги (IaaS) і як рішень для керування програмами на рівні PaaS. Ми можемо помітити, що контейнери позитивно впливають як на аспекти розробки, так і на розгортання. Наприклад, архітектура в хмарі переходить до підходів на основі DevOps, підтримуючи конвеєр безперервної розробки та розгортання з урахуванням рішень власної хмарної архітектури на основі контейнерів та їх оркестровки [21, ст. 25].

Окрім того, контейнеризація добре узгоджується з архітектурою мікросервісів. Мікросервіси включають розбиття програми на менші, слабко пов’язані служби. Кожну службу можна контейнеризувати незалежно, що забезпечує швидку розробку, розгортання та масштабування окремих компонентів. Зокрема, останнім часом популярні архітектури на основі мікросервісів можна реалізувати в цій хмарній структурі через контейнери [44].

Контейнери відіграють важливу роль у CI/CD. Розробники можуть упаковувати свої програми в контейнери на етапі створення, а потім розгортати їх на різних етапах, від тестування до виробництва, не турбуючись про невідповідності середовища. Використання контейнерів, також, сприяє більш плавній співпраці між командами розробки та операцій. Через те, що контейнерна програма демонструє консистентну роботу у різних середовищах, це знижує конфлікти між командами та підвищує загальну продуктивність, забезпечує однакове середовище для тестування та розробки, і зменшує можливі проблеми, пов’язані з різними конфігураціями середовищ.

Незважаючи на численні переваги, контейнеризація також приносить певні проблеми, такі як керування оркестрацією контейнерів, питання безпеки

та забезпечення належного моніторингу та використання ресурсів. Однак, загалом, контейнеризація справила глибокий вплив на ландшафт розробки програмного забезпечення, забезпечивши більш ефективні та гнучкі методи розробки, одночасно значно покращивши розгортання та масштабованість сучасних програм.

1.2. Основні характеристики системи оркестрації контейнерів Kubernetes

Оркестрація контейнерів автоматизує розгортання, керування, масштабування та роботу в мережі. Підприємства, яким необхідно розгортати та керувати сотнями або тисячами контейнерів і хостів Linux, можуть отримати вигоду від оркестрації контейнерів. Оркестрація контейнерів можна використовувати в будь-якому середовищі, де ви використовуєте контейнери. Це може допомогти вам розгорнути ту саму програму в різних середовищах без необхідності її перепроектування. А мікросервіси в контейнерах спрощують організацію служб, зокрема зберігання, мережі та безпеки [68].

Системи оркестрації контейнерів спрощують розгортання та обслуговування додатків на основі контейнерів, але розробка ефективних і чітко визначених систем оркестровки є проблемою. Сьогодні Kubernetes є провідною платформою оркестровки контейнерів з відкритим вихідним кодом, яка стала стандартом де-факто.

Так, у 2014 році Google відкрила проект Kubernetes. Kubernetes поєднує в собі понад 15-річний досвід Google у створенні великих робочих навантажень із найкращими ідеями та практиками спільноти [17].

Kubernetes – це портативна, розширювана платформа з відкритим вихідним кодом для керування контейнерними робочими навантаженнями та службами, яка полегшує як декларативне налаштування, так і автоматизацію. Він має велику екосистему, що швидко розвивається. Сервіси, підтримка та інструменти Kubernetes широко доступні [50].

1.2.1. Застосування Kubernetes на ринку IT

Kubernetes знайшов значне впровадження та використання в різних галузях і організаціях. Kubernetes вважається одним із найпопулярніших інструментів оркестровки контейнерів з відкритим кодом і використовується в комерційних та некомергійних організаціях, зокрема Adidas, Nokia, Spotify [43] і Міністерство оборони США [70].

Переваги використання Kubernetes були задокументовані: наприклад, використання Kubernetes у Міністерстві оборони США призвело до скорочення восьмимісячного розгортання програмного забезпечення до одного тижня [70]. Для Adidas час завантаження веб-сайту електронної комерції скоротився вдвічі, а частота випусків зросла з одного разу на 4-6 тижнів до 3-4 разів на день [43].

Серед ключових аспектів використання Kubernetes слід виділити його використання у хмарних середовищах. Kubernetes надає необхідні функції оркестровки та масштабованості для підтримки практик розробки в хмарі. Зокрема, багатство інструментів у cloud native ecosystem стало вагомою причиною для багатьох людей прийняти Kubernetes. Якщо ви використовуєте власну екосистему хмари, ви можете використовувати створені спільнотою та підтримувані проекти майже для кожної частини вашої системи, дозволяючи вам зосередитися на розробці основної бізнес-логіки та послуг, які є виключно вашими [39, ст. 17].

Масштабованість і висока доступність є одними із ключових переваг Kubernetes. Він може автоматично збільшувати або зменшувати кількість реплік залежно від використання ресурсів або вхідного трафіку, забезпечуючи ефективну роботу програми з різними навантаженнями. У Kubernetes HorizontalPodAutoscaler автоматично оновлює ресурс робочого навантаження (наприклад, Deployment або StatefulSet) з метою автоматичного масштабування робочого навантаження відповідно до потреб. Горизонтальне масштабування означає, що реакцією на збільшення навантаження є розгортання більшої кількості модулів. Це відрізняється від вертикального масштабування, яке для

Kubernetes означатиме призначення додаткових ресурсів (наприклад, пам'яті чи ЦП) для модулів, які вже запущені для робочого навантаження [32].

Kubernetes також підтримує конфігурації високої доступності, що дозволяє програмам залишатися доступними навіть у разі збою вузла.

Окрім того, Kubernetes добре узгоджується з архітектурним стилем мікросервісів. За допомогою Kubernetes кожен мікросервіс можна розгорнути як окремий контейнер у модулі, що забезпечує незалежне масштабування, керування та швидку розробку додатків на основі мікросервісів (рис. 1.2).

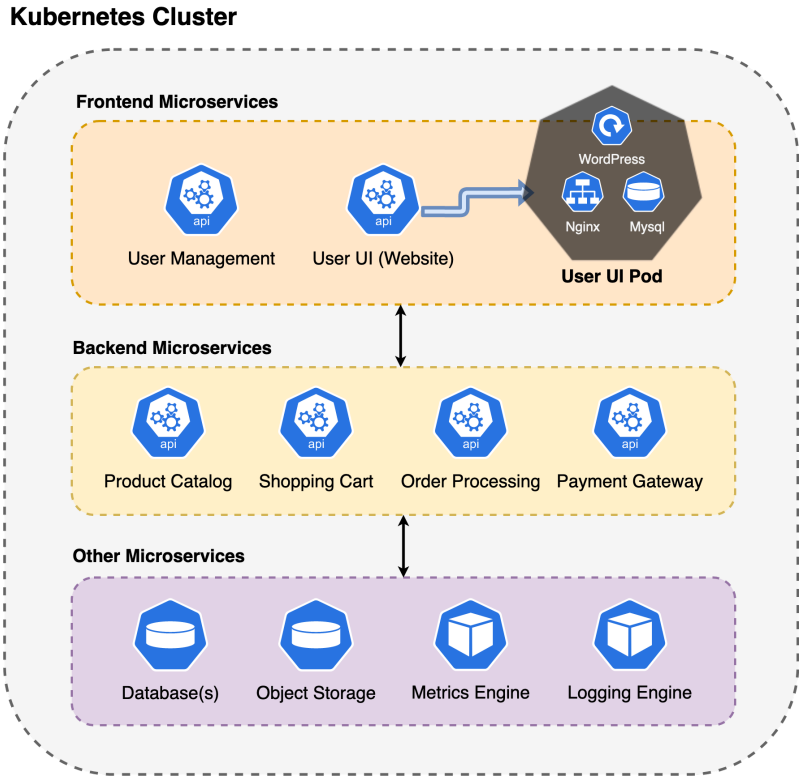


Рисунок 1.2 – Мікросервісна архітектура в Kubernetes

Функції перевірки працездатності та розгортання Kubernetes гарантують послідовний підхід до розгортання додатків і надійність, що гарантує, що збільшення кількості команд мікросервісів не призведе до розповсюдження різних підходи до життєвого циклу та операцій виробництва послуг [39, ст. 14].

Kubernetes відіграє вирішальну роль у CI/CD процесах. Розробники можуть використовувати Kubernetes для розгортання та тестування програм на різних етапах, від розробки до виробництва, використовуючи автоматизовані

процеси. Це забезпечує швидшу та надійнішу доставку додатків. Конвеєр CI/CD, який розгортається на Kubernetes, полегшує контрольований випуск програмного забезпечення, оскільки інженери DevOps можуть налаштовувати поетапні випуски, як-от “blue-green” чи “canary” розгортання [2]. Це допомагає досягти нульового простою під час випуску та зменшує ризик випуску програми для всіх користувачів одночасно. Автоматизація всього SDLC (життєвого циклу розробки програмного забезпечення) за допомогою конвеєра CI/CD допомагає знизити витрати за рахунок скорочення багатьох постійних витрат, пов’язаних із процесом випуску [38].

Незважаючи на те, що Kubernetes відомий ефективним керуванням програмами без збереження стану, вона досягла значного прогресу в підтримці програм із збереженням стану за допомогою таких функцій, як StatefulSets і Persistent Volumes (PV). Це зробило його більш універсальним для роботи з різними видами навантажень. Окрім того, Kubernetes можна використовувати для обробки великих даних. Його можна використовувати для розподілу обробки великих даних між кількома машинами, що може допомогти покращити продуктивність обробки великих даних.

Незважаючи на відомі переваги, користувачі Kubernetes повідомили про свої занепокоєння щодо безпеки Kubernetes. Cloud Native Computing Foundation провів опитування 1337 практиків і повідомив, що 40% учасників опитування стурбовані безпекою Kubernetes [25]. Докази підтверджують занепокоєння практикуючих спеціалістів щодо безпеки Kubernetes. Наприклад, у 2018 році зловмисники отримали доступ до ресурсів Tesla Amazon Web Services (AWS) за допомогою незахищеної консолі Kubernetes [63].

Підсумовуючи можна відзначити, що Kubernetes отримав широке поширення в корпоративному світі. Організації будь-якого розміру, від стартапів до великих підприємств, прийняли Kubernetes за його здатність спрощувати керування контейнерами, покращувати використання ресурсів і сприяти співпраці між командами розробки та операцій.

1.2.2. Архітектура Kubernetes

Архітектура Kubernetes проста та інтуїтивно зрозуміла. Концепція “loose coupling” між площиною керування та вузлом забезпечує майже нескінченну гнучкість і здатність програми практично миттєво масштабуватись відповідно до мінливих потреб, переміщувати користувачів у нові збірки та підтримувати міграцію з локальних на хмарні вузли або між кількома хмарами, щоб скористатися бажаними функціями кожного постачальника хмар [69].

Інсталяцію Kubernetes називають кластером Kubernetes [47]. Кожен кластер Kubernetes містить набір робочих машин, визначених як вузли. Як показано на рисунку 1.3, для Kubernetes існують два типи вузлів: головні вузли та робочі вузли.

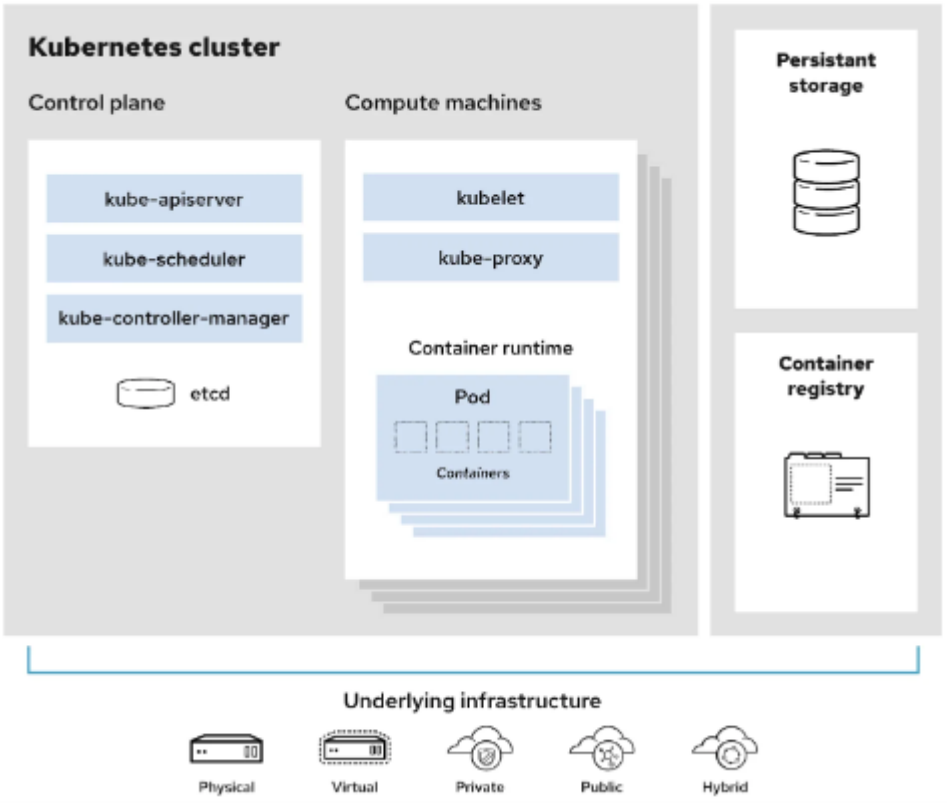


Рисунок 1.3 – Архітектура Kubernetes та її зв’язок із базовою інфраструктурою

Кожен головний вузол включає в себе наступні компоненти: “сервер API”, “планувальник”, “контролер” і “etcd” [47]. “Сервер API” відповідає за організацію всіх операцій у кластері. Kubernetes обслуговує свої функції через інтерфейс прикладної програми з “сервера API”. “Контролер” – це компонент головного кластера, який спостерігає за станом кластера через “сервер API” та змінює поточний стан на бажаний. “Планувальник” – це компонент у площині керування, відповідальний за планування модулів між кількома вузлами. “etcd” – це база даних на основі ключів і значень, яка зберігає всю інформацію про конфігурацію для кластера Kubernetes. Менеджер хмарного контролера (додатково): інтерфейси з API основного хмарного постачальника для керування спеціальними хмарними ресурсами, як-от балансувальники навантаження та обсяги зберігання.

Користувачі використовують інструмент командного рядка “Kubectl” для зв’язку з “сервером API” у головному вузлі.

Робочі вузли розміщують програми, які працюють на Kubernetes [47]. До робочого вузла входять такі компоненти: “kube-proxy”, “kubelet” і “pod”. “kube-proxy” підтримує мережеві правила на вузлах, щоб забезпечити зв’язок між модулями та зовнішніми джерелами. “kubelet” – це агент, який працює на кожному робочому вузлі та взаємодіє з сервером API. Він забезпечує роботу контейнерів у модулі (найменшому розгортаючому модулі в Kubernetes) відповідно до бажаного стану. Він відповідає за запуск контейнерів у Pods.

Pod – це найменша сутність Kubernetes, яка містить принаймні один активний контейнер. Kubernetes підтримує різні середовища виконання контейнерів, як-от Docker, container тощо. Контейнер – це стандартний програмний блок, який пакує код і пов’язані з ним залежності для виконання в будь-якому обчислювальному середовищі.

Kubernetes надає широкий вибір вбудованих API для декларативного керування вашими робочими навантаженнями та компонентами цих робочих навантажень. Взагальному, програми працюють як контейнери всередині Pods; однак керування окремими модулями складе багато зусиль. Наприклад, якщо

Pod виходить з ладу, можливо, ви захочете запустити новий Pod, щоб замінити його. Kubernetes може робити автоматизовано.

API Kubernetes може бути використаний для створення об’єкта робочого навантаження, який представляє вищий рівень абстракції, ніж Pod, а потім площина керування Kubernetes автоматично керує об’єктами Pod від вашого імені на основі специфікації для об’єкта робочого навантаження, який ви визначили [47].

Серед вбудованих API для керування навантаженнями найбільш часто використовуються наступні:

- Deployment (і, опосередковано, ReplicaSet) – найпоширеніший спосіб запуску програми на вашому кластері. Розгортання добре підходить для керування робочим навантаженням додатків без стану на вашому кластері, де будь-які модулі в розгортанні є взаємозамінними та можуть бути замінені за потреби.
- StatefulSet дозволяє керувати одним або декількома модулями. Усі вони працюють з однаковим кодом програми, де модулі покладаються на наявність окремої ідентифікації. Це відрізняється від Deployment, де очікується, що модулі будуть взаємозамінними. Найпоширенішим використанням StatefulSet є можливість встановлення зв’язку між його модулями та їх постійним сховищем. Наприклад, ви можете запустити StatefulSet, який пов’язує кожен Pod з PersistentVolume. Якщо один із модулів у StatefulSet виходить з ладу, Kubernetes створює замінний модуль, підключений до того самого PersistentVolume.
- DaemonSet визначає модулі, які надають можливості, локальні для певного вузла; наприклад, драйвер, який дозволяє контейнерам на цьому вузлі отримувати доступ до системи зберігання. DaemonSet використовується, коли драйвер або інша служба рівня вузла має працювати на вузлі, де це корисно. Кожен Pod у DaemonSet виконує роль, подібну до системного демона на класичному сервері Unix / POSIX. DaemonSet може бути основоположним для роботи вашого кластера, наприклад, плагін, який дозволяє цьому вузлу

отримати доступ до мережі кластера, він може допомогти вам керувати вузлом або він може надавати менш важливі засоби, які покращують платформу контейнера, яку ви використовуєте.

- Job та/або CronJob застосовується, щоб визначити завдання, які виконуються до завершення, а потім зупиняються. Завдання представляє одноразове завдання, тоді як кожне CronJob повторюється за розкладом.

У екосистемі Kubernetes наявні ресурси робочого навантаження, які забезпечують додаткову поведінку. Використовуючи Custom resource definition, можна додати сторонній ресурс робочого навантаження, якщо небхідно додати певну поведінку, яка не є частиною ядра Kubernetes. Наприклад, якщо потрібно запустити групу Pods для програми, але припинити роботу, доки всі Pods не будуть доступні (наприклад, для розподіленого завдання з високою пропускною здатністю), можна застосувати або встановити розширення, яке надає цю функцію.

Отже, підхід Kubernetes до керування робочими навантаженнями через різноманітні ресурси, такі як Deployment, StatefulSet, DaemonSet, Job та CronJob, демонструє широкий спектр можливостей для керування різнорідними застосунками та їх поведінкою. Забезпечуючи вбудовані API для декларативного керування, Kubernetes дозволяє користувачам ефективно керувати своїми робочими навантаженнями та компонентами, забезпечуючи високий рівень абстракції та автоматизації.

1.3. Розгортання кластерів Kubernetes в хмарних середовищах

Kubernetes продовжує отримувати широке впровадження в різних галузях і організаціях. Компанії будь-якого розміру, від стартапів до підприємств, використовують Kubernetes для розгортання та керування контейнерними програмами в хмарі. Хмарні постачальники розширювали свої керовані сервіси Kubernetes. Ці служби, як-от Amazon Elastic Kubernetes Service [6], Google Kubernetes Engine [29] і Azure Kubernetes Service [15], полегшили користувачам

розгортання та керування кластерами Kubernetes без складнощів адміністрування кластерів.

Не дивно, що у 2022 році AWS, Microsoft Azure, Google Cloud і VMware займають значну частку середовищ, у яких працюють кластери Kubernetes, залишаючи позаду інших конкурентів на ринку. Зокрема, згідно з дослідження “Найпопулярніші середовища для роботи кластерів Kubernetes у всьому світі 2022”, опублікованого Лайонел Суджай Вейлшері, 21 вересня 2022 р. [46], більше половини респондентів зазначили, що веб-сервіси AWS були найпопулярнішим середовищем для роботи кластерів Kubernetes у всьому світі у 2022 році. Деякі інші популярні середовища для респондентів включають Microsoft Azure, Google Cloud Platform і VMware. Детальніше дивіться будь ласка рисунок 1.4.

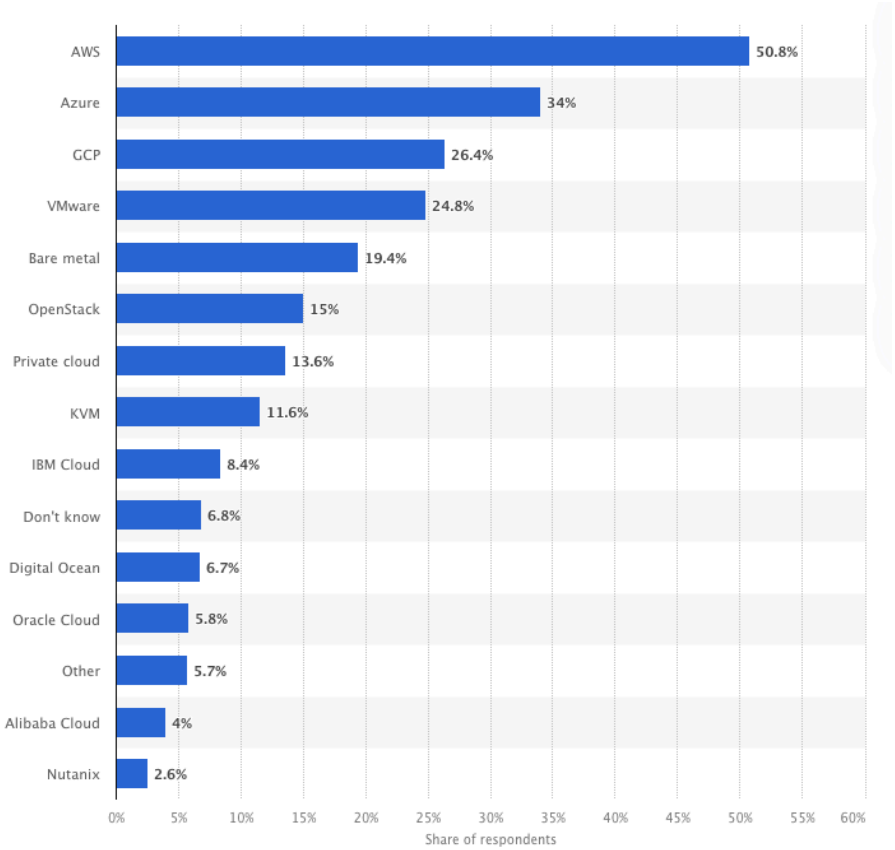


Рисунок 1.4 – Найпопулярніші середовища для роботи кластерів Kubernetes у всьому світі

Наприклад, Amazon EKS – це керований сервіс Kubernetes від AWS. Це спрощує розгортання та керування кластерами Kubernetes в інфраструктурі AWS. EKS інтегрується з іншими службами AWS, що полегшує створення та масштабування контейнерних програм [6]. GKE – це служба Kubernetes, якою керує Google [29]. Він відомий своєю глибокою інтеграцією з Kubernetes і сервісами Google Cloud. GKE надає такі функції, як автоматичне масштабування, пули вузлів і моніторинг, що робить його популярним вибором для виконання робочих навантажень Kubernetes на GCP. AKS – це керована пропозиція Kubernetes від Microsoft Azure. Його розроблено, щоб спростити розгортання та керування Kubernetes в Azure. AKS тісно інтегровано зі службами Azure, що дозволяє безперешкодно використовувати ресурси Azure.

Також, Oracle Kubernetes Engine [49] – це керована служба Kubernetes Oracle Cloud. Він призначений для запуску контейнерних програм в інфраструктурі Oracle Cloud. DigitalOcean Kubernetes [26] пропонує простий і економічно ефективний спосіб розгортання кластерів Kubernetes у хмарній інфраструктурі DigitalOcean. Його віддають перевагу меншим командам і стартапам. Слід згадати, також, Red Hat OpenShift [54], який є дистрибутивом Kubernetes із додатковими функціями для корпоративних середовищ. Хоча його можна запускати на різних хмарних постачальниках, Red Hat пропонує OpenShift як службу на деяких платформах. Окрім того, VMware Tanzu [67] надає рішення Kubernetes корпоративного рівня та пропонує Tanzu Kubernetes Grid для багатохмарного керування Kubernetes.

Слід відзначити, що безсерверні пропозиції для Kubernetes, такі як AWS Fargate для EKS [12] і GKE Autopilot [28], набули популярності. Ці служби спрямовані на абстрактне управління інфраструктурою, дозволяючи користувачам зосередитися на коді програми, а не на адмініструванні кластера.

Використання Kubernetes у хмарних середовищах також допомагає покращити безпеку середовищ. Користувачі Kubernetes інвестували в методи безпеки, такі як сканування контейнерів, мережеві політики та керування ідентифікацією та доступом (IAM), щоб захистити свої середовища Kubernetes.

Зокрема, в межах Amazon EKS можна використати функцію сканування контейнерів Amazon ECR [5]. Також, як керована служба, Amazon Elastic Kubernetes Service захищена системою безпеки глобальної мережі AWS. Щоб отримати інформацію про служби безпеки AWS і те, як AWS захищає інфраструктуру [35].

В AWS, адміністратори IAM контролюють, хто може пройти аутентифікацію (увійти) і авторизуватися (мати дозволи) на використання ресурсів Amazon EKS [34]. Також, AWS App Mesh забезпечує роботу в мережі на рівні програми, щоб ваші служби могли обмінюватися даними між кількома типами обчислювальної інфраструктури [10].

Підсумовуючи, Kubernetes у хмарі еволюціонував, став більш зручним для користувачів, адаптованим до різноманітних випадків використання та інтегрованим із різними хмарними службами та інструментами, що робить його надійним вибором для оркестровки контейнерів у хмарі.

Висновки до розділу 1

Написання цієї роботи є доцільним заважаючи на стан дослідження Kubernetes та активність його розвитку. Kubernetes, як провідна платформа оркестровки контейнерів, пропонує безліч дослідницьких можливостей у таких сферах, як контейнеризація, хмарні обчислення, DevOps, масштабованість тощо. Доцільність цього дослідження підтверджується зростаючою популярністю Kubernetes і зростаючим попитом на професіоналів із досвідом Kubernetes. Дослідження Kubernetes можуть сприяти розвитку найкращих практик, методів оптимізації та інноваційних рішень для керування контейнерними програмами. Крім того, вони узгоджується з галузевими тенденціями та вирішує реальні проблеми, що робить його актуальним і ефективним вибором для досліджень у сфері хмарних обчислень і контейнеризації.

Використання Kubernetes у хмарних середовищах слід виділити як одну із найпоширеніших практик. Kubernetes дозволяє автоматично масштабувати додатки в залежності від навантаження, що особливо важливо в хмарних середовищах. Хмарні постачальники, такі як AWS, Microsoft Azure, та Google Cloud, розширюють свої керовані сервіси Kubernetes. Сервіси, такі як Amazon EKS, Google Kubernetes Engine, і Azure Kubernetes Service, спрощують розгортання та керування кластерами Kubernetes. Kubernetes продовжує еволюціонувати, стаючи більш зручним та інтегрованим з різноманітними хмарними службами та інструментами. Його використання сприяє поліпшенню ефективності, безпеки та гнучкості оркестрації контейнерів у хмарі.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ТРЕНДІВ ЗАСТОСУВАННЯ KUBERNETES

2.1. Поточний стан впровадження Kubernetes

Сучасні хмарні обчислення неможливо відокремити від контейнерів і прийняття Kubernetes. Незважаючи на те, що Kubernetes є відносно молодою технологією, більшість глобальних підприємств використовують її для запуску критично важливих для бізнесу програм у виробництві. Швидке впровадження зумовлене і викликане постійно зростаючою екосистемою технологій Kubernetes, які додають розширені функції платформи, такі як безпека, комунікації мікросервісів, можливість спостереження, масштабування, використання ресурсів тощо.

Kubernetes використовується багатьма організаціями, від невеликих стартапів до великих підприємств. За даними CNCF, Kubernetes продовжує розвиватися та досягла найвищого рівня за всю історію: 96% організацій використовують або оцінюють Kubernetes [1].

Слід відзначити, що Kubernetes є безумовним лідером серед CNCF проєктів станом на 01.01.2022-01.01.2023 (3774 авторів) [24]. Також, широко використовуються CNCF проєкти, які пов’язані із використанням у Kubernetes середовищах. Серед них, слід виділити Argo (858 авторів), Prometheus (426 авторів), Envoy (411 авторів), Istio (360 авторів) та інші.

Компанія “DZone” розробила звіт “Kubernetes in the Enterprise” за жовтень 2022 року [40], який містить дані опитування про те, як команди розгортають і використовують Kubernetes, а також проблеми, які виникають у масштабному впровадженні реальних кластерів. У звіті міститься інформація про розробку надійної стратегії Kubernetes, забезпеченої безпекою та можливістю спостереження для спрощення обслуговування та моніторингу.

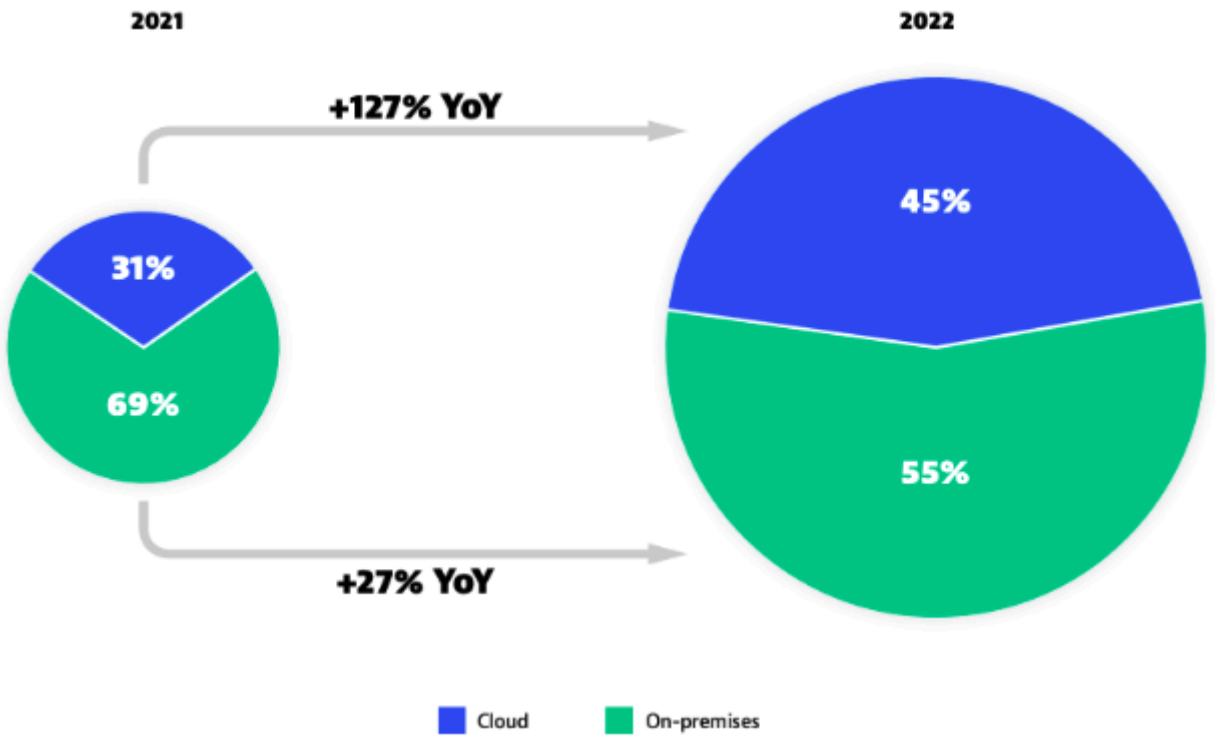
Так, DZone опитав розробників програмного забезпечення, архітекторів та інших IT-фахівців і отримав майже 470 відповідей. Ось кілька ключових висновків зроблених на основі їх дослідження [40]:

- впровадження контейнерів залишається на позначці 90% або близько неї;
- впровадження Kubernetes зупинилося приблизно на 73%;
- 14,8% розгортань Kubernetes знаходяться в розробці, а 17% – у виробництві;
- 85,5% учасників з великих компаній сказали, що вони керують кластерами Kubernetes, тоді як 63,4% респондентів з менших компаній заявили, що вони їх використовують;
- більше половини респондентів (52%) використовують робочі навантаження з підтримкою стану в Kubernetes.

Окрім того, опитування щодо Kubernetes проведене компанією Dynatrace у 2023 [41] вказує на наступні тенденції:

- у 2022 році Kubernetes перейшов у хмару;
- моделі інфраструктури Kubernetes відрізняються між хмарними та локальними моделями;
- Kubernetes стає “операційною системою” хмари;
- найсильніші сфери розвитку Kubernetes – це безпека, бази даних і технології CI/CD;
- програмне забезпечення з відкритим вихідним кодом забезпечує живу екосистему Kubernetes;
- Java, Go і Node.js є трьома найкращими мовами програмування для робочих навантажень програм у Kubernetes.

Наприклад, У 2022 році Kubernetes став ключовою платформою для переміщення навантаження на публічну хмару. При річному прирості +127%, кількість кластерів Kubernetes, розміщених у хмарі зростав приблизно в п’ять разів швидше, ніж кластери, розміщені локально. Так само частка хмарних кластерів зросла з 31% у 2021 році до 45% у 2022 році. [41]. Детальніше дивіться рисунок 2.1.



с

Зазначене свідчить про стійкий і надалі зростаючий інтерес до використання Kubernetes у хмарному середовищі. Ця тенденція, ймовірно, залишиться актуальною на протязі найближчих кількох років. Рові інструменти та сервіси для спрощення управління Kubernetes у хмарному середовищі допомагають знизити операційні витрати та спрощує розгортання й управління кластерами Kubernetes для підприємств. За останніми тенденціями, Kubernetes також стає важливою основою для розвитку інших технологій, таких як машинне навчання та інші розумні додатки, що можуть бути розгорнуті в хмарному середовищі. Це підсилює перспективи щодо подальшого зростання популярності Kubernetes у хмарних середовищах.

Більше того, слід звернути увагу на мультихмарні тенденції використання Kubernetes. Так, у звіті “The State of Kubernetes 2023” [65, ст. 7] підготовленого компанією VMware вказується, що оскільки організації розширюють свою мультихмарну зону дії Kubernetes, багато виклики розгортання та управління зростають, зокрема неадекватний внутрішній досвід, складність інтеграції з поточним інфраструктура та відсутність мобільності додатків. Вибравши

Kubernetes дистрибутивів та інструментів ретельно, підприємства можуть зменшити складність і отримати свободу запускати будь-яку програму в будь-якій хмарі. На рисунку 2.2 наведені тенденція використання мультихмарних середових для розміщення кластерів Kubernetes.

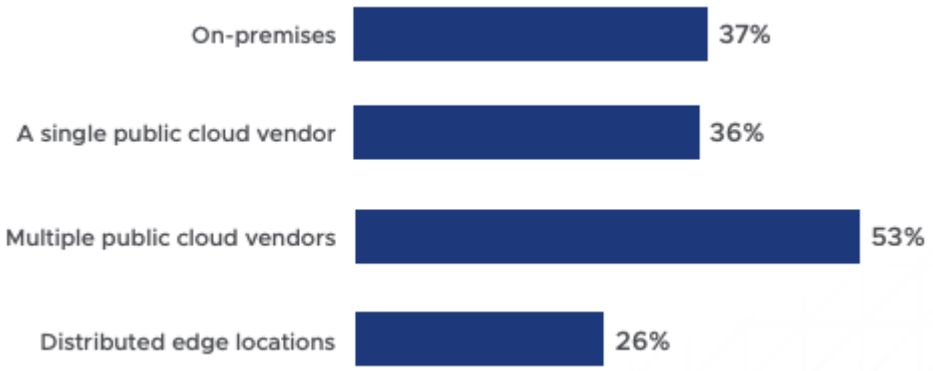


Рисунок 2.2. Тенденції використання мультихмарних середових для розміщення кластерів Kubernetes

Звіт “The State of Kubernetes 2023” підверджує, що мультихмарні середовища стала домінуючим типом розгортання. Понад три чверті опитаних (76%) використовують кілька хмар. Відсоток багатохмарних операцій навіть вищий у таких галузях, як телекомунікації (89%), фінансові послуги та страхування (87%) та програмне забезпечення (84%). На запитання, де організації зараз розгорнули Kubernetes, 44% є гібридними, а 44% – лише хмарними. Лише 10% респондентів кажуть, що працюють лише на місці, порівняно з 15% два роки тому [65, ст. 7].

Багато компаній будують плани на майбутнє із стратегією використання багатохмарних середовищ для розміщення Kubernetes. Більше половини (53%) планують додавати або збільшити потужність у багатьох постачальників публічних хмар порівняно з 37% локальне розширення та 36% розширення в одній загальнодоступній хмарі [65, ст. 7].

2.2. Фактори, що спонукають до впровадження Kubernetes

Kubernetes доклав багато зусиль, щоб допомогти користувачам керувати кінцевим станом того, що розробнику потрібно від системи, водночас автоматично підтримувати роботу реплік і процес відновлення. Тут єдина конфігурація циркулює між усіма доступними кластерами. Таким чином, виникає єдина стандартизована платформа для постійного використання в усій галузі. Це явище допомагає нормалізувати та стандартизувати використання, щоб зберігати всі залежні програми в одному контейнері, задля того, щоби Kubernetes керував ними ефективно. Для більшості клієнтів Kubernetes став стандартним способом організації та керування контейнерами. Kubernetes успішно організовує конкретне розгортання на певних сайтах на кількох сайтах із життєвими циклами програми [56].

У звіті “The State of Kubernetes 2023” [65, ст. 5] підготовленого компанією VMware проаналізовані бізнес переваги, які надає Kubernetes. Зокрема, дві основні переваги для бізнесу пов’язані з ефективністю операцій і груп розробників: IT-оператори ефективніші (64%), а розробники продуктивніші (60%). Ці переваги, особливо продуктивність розробників, надзвичайно важливі, враховуючи сучасну залежність від цифрових технологій. Підвищення продуктивності розробника, усунення складності та зменшення трудомісткості забезпечують швидший шлях до виробництва та скорочують час виходу на ринок, що скорочує час для отримання вартості. Більше третини (37%) повідомили, що Kubernetes допомагає IT-керівництву показати IT як джерело доходу, а не просто центр витрат. Три додаткові переваги свідчать про те, що багато зацікавлених сторін у сфері технологій зараз бачать прямий вплив на прибутки в результаті використання Kubernetes: бізнес бачить зростання частки ринку (25%), створено нові умови для клієнтів, які приносять прибуток (21%), а норма прибутку зростає (20%) (рис. 2.3). Ці переваги вказують на те, що Kubernetes забезпечує значну віддачу від інвестицій (ROI) для багатьох організацій.

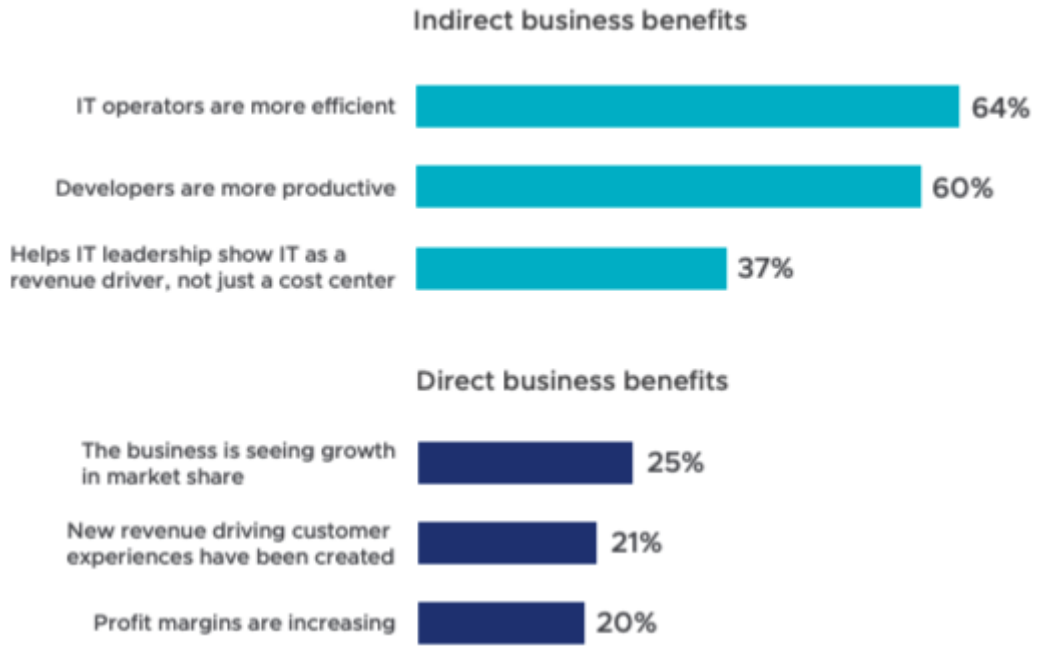


Рисунок 2.3 – Бізнес переваги Kubernetes

Також, у звіті “Kubernetes in the Enterprise” [40], визначені наступні спостереження:

- Kubernetes розглядався як корисний для CI/CD минулого року (2021), ця тенденція посилилася значно цього року (2022). Це, мабуть, знак зростаючої компетентності та впевненості Kubernetes для однієї з його основних функцій. Ми також побачили великий стрибок у думках людей про вплив Kubernetes на модульність після нього раніше був стабільним. Оскільки модульність стає все більш широкою реалізовано, це можливо, навіть ймовірно, що є інструменти та досвід Kubernetes змінено для його підтримки.
- Цього року (2022), як вважає майже половина респондентів Kubernetes відіграв роль у забезпеченні надійності, що не відповідає минулому року. Це могло б бути ознакою того, що демографічні показники аудиторії відрізняються щороку, або просто ознакою того, що Kubernetes був на неправильному шляху та курс виправлено в досить ефективний спосіб.
- Минулорічний (2021) стрибок у безпеці зберігся. Це все ще непопулярна думка, але вплив Kubernetes на безпеку, здається, стабілізується з року в рік.

- Цікаво, що розрив у витратах скорочується. Хоча ми не опублікували результати тут у формі таблиці, ми також поставили запитання: “Як Kubernetes покращив вашу організацію?”. Безперечно, найбільшим вибором була вартість (37,0%). Буде цікаво побачити, чи цьогорічне зростання залишиться стабільним чи навіть перевершить протилежну точку зору – чи навпаки знизиться до рівня 2020 року. Для цікавих, наступними найближчими виборами щодо того, що Kubernetes зробив краще, були безпека (26,1%) та архітектурний рефакторинг (24,4%). В іншому випадку історія, яку розповідають, здається про стабілізацію та помірний спад. Наприклад, менше половини респондентів зараз вважають, що Kubernetes покращує створення мікросервісів порівняно з минулим роком (52,3%) і 2020 роком (53,6%), архітектурний рефакторинг дещо знизився, а автомасштабування трохи підвищилося після незначного минулорічного падіння. Детальніше дивіться будь ласка рисунок 2.4.

Area	% of Total Respondents		
	2020	2021	2022
Deployment in general	66.0%	64.3%	61.7%
Autoscaling	65.0%	62.8%	63.8%
CI/CD	63.7%	61.1%	73.8%
Building microservices	53.6%	52.3%	47.5%
Reliability	46.0%	39.6%	48.2%
Application modularity	44.3%	43.3%	51.1%
Architectural refactoring	36.1%	32.2%	31.2%
Overall system design	33.5%	32.2%	32.6%
Cost	28.9%	28.2%	34.8%
Security	24.9%	31.3%	31.2%
Other	3.2%	2.6%	4.3%

Рисунок 2.4 – Покращення від використання Kubernetes

Під час проведення дослідження “The State of Kubernetes 2023” [65, ст. 6] респондентів запитали про використання Kubernetes, 98% зацікавлених сторін повідомили, що бачать переваги. Головною перевагою було покращене використання ресурсів (50%), а потім скорочені цикли розробки ПЗ (41%). Слід також відзначити кілька операційних переваг, пов’язаних із хмарою: можливість переходу в хмару (36%) і зниження витрат на публічну хмару (33%). Kubernetes збільшує успіх хмари та полегшує гібридні та багатохмарні операції, які стали важливими для цифрового бізнесу.

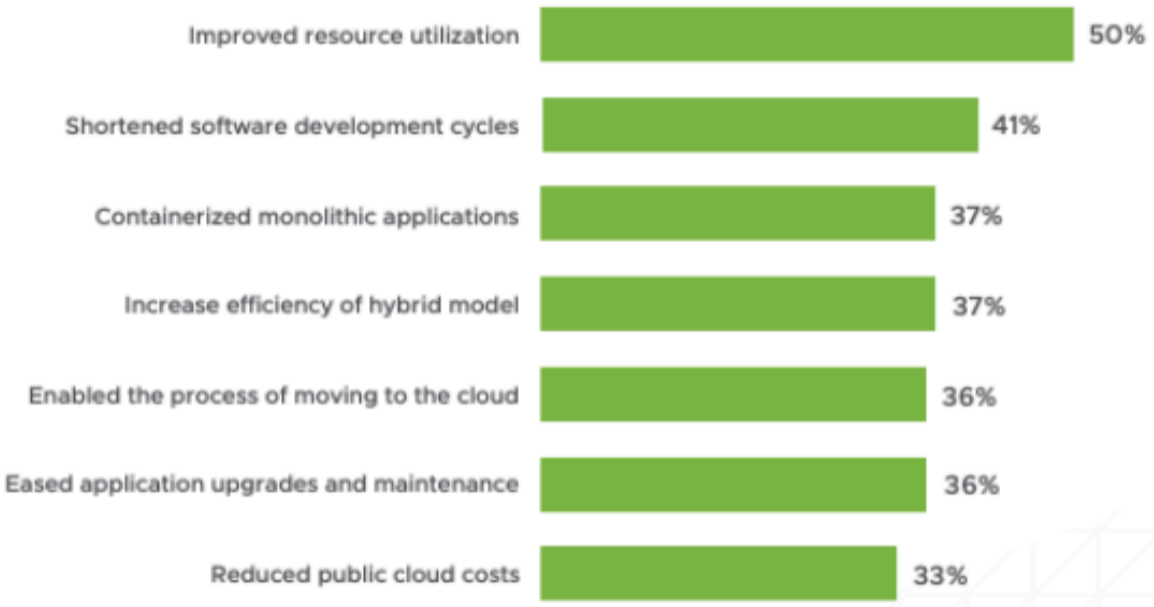


Рисунок 2.5 – Операційні переваги Kubernetes

Kubernetes використовується у різних сферах. Зокрема, Kubernetes ідеально підходить для всього: від веб-додатків до програм, що інтенсивно використовують процесор, GPI та пам’ять, а також для зберігання даних і загальних обчислювальних функцій. І масштабованість є ключовою – Kubernetes може обробляти дуже змінний трафік без надлишкових ресурсів [40]. Рисунок 2.6 описує типи робочих навантажень, які виконуються в кластері Kubernetes.

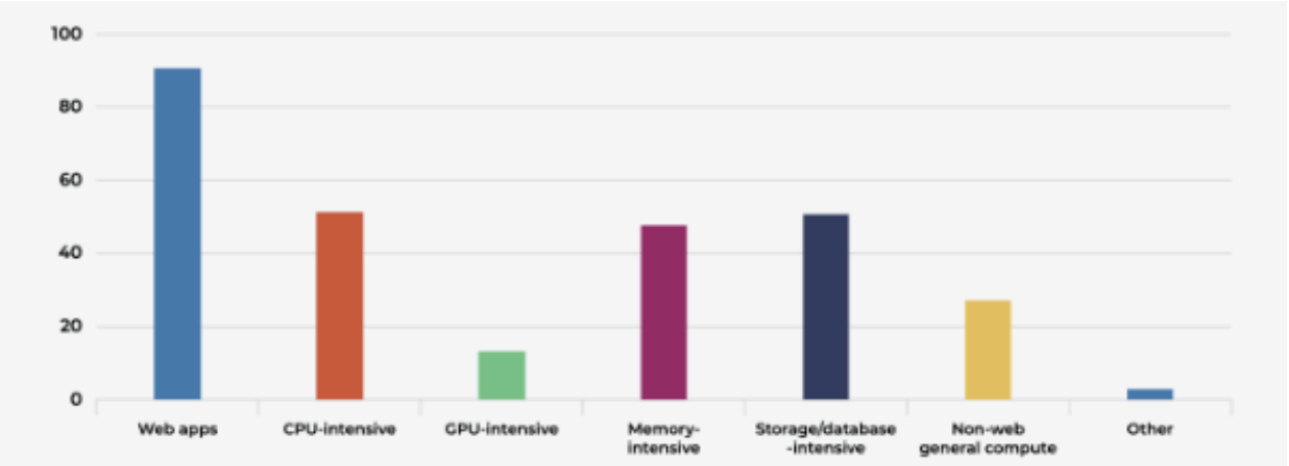


Рисунок 2.6 – Типи робочих навантажень, які виконуються в кластері Kubernetes

Проаналізувавши вказані вище дослідження, варто відзначити, що впровадження Kubernetes може бути стимульоване рядом факторів, які визначають його популярність та важливість у сфері розгортання та управління контейнерами. Серед них слід виділити наступні: 1) Kubernetes надає потужні інструменти для автоматизації процесів розгортання, масштабування, керування та відновлення додатків у контейнерах, що дозволяє забезпечити надійність та доступність додатків; 2) Kubernetes дозволяє працювати в різних хмарних середовищах та власних центрах обробки даних, що робить його гнучким рішенням для компаній, які використовують різні інфраструктурні платформи; 3) здатність Kubernetes ефективно масштабувати додатки вгору або вниз дозволяє адаптувати їх до змінних обсягів трафіку та завдань, що особливо важливо в умовах зростаючого навантаження; 4) Kubernetes забезпечує інструменти для безпеки контейнеризованих додатків; 5) завдяки автоматизації та стандартизації, Kubernetes полегшує розробку, тестування та впровадження додатків, що дозволяє швидше реагувати на зміни вимог та прискорює час виходу на ринок; 6) Kubernetes допомагає ефективно використовувати ресурси, розміщуючи додатки на вузлах із оптимальними потужностями задля зниження витрат та максимізації використання обчислювальних потужностей; 7) за допомогою широкої екосистеми інструментів та додатків, які підтримують

Kubernetes, користувачі можуть розширювати його функціональність та використовувати різноманітні рішення для своїх потреб; 8) впровадження Kubernetes спрощує рутинні завдання адміністрування, такі як масштабування, відновлення та моніторинг, що дозволяє командам економити час та ресурси.

Ці фактори роблять Kubernetes популярним та важливим інструментом для оркестрації та управління контейнеризованими додатками в різних сценаріях використання.

2.3. Проблеми та перешкоди для впровадження

Багато компаній вважають Kubernetes незамінним інструментом для розробки та управління сучасними хмарними додатками. З правильним підходом та належною підтримкою, вони можуть успішно використовувати Kubernetes для підвищення ефективності, масштабування та інновацій у своїх проектах. Однак, при реалізації цієї мети компанії стикаються зі значними викликами.

Зокрема, згідно опитування “Провідні виклики Kubernetes для компаній у всьому світі 2022” опублікованого Лайонел Суджай Вейлшері, 14 червня 2023 року [45] більше половини респондентів зазначили, що під час переходу або використання Kubernetes і контейнерів брак внутрішніх навичок і, отже, обмежена робоча сила є найбільшою проблемою, з якою вони стикаються у 2022 році. IT-структура компанії (37,7%) та несумісність із застарілими системами (31,9%) також були популярними проблемами, з якими стикаються співробітники під час використання Kubernetes. Слід відзначити також проблеми щодо складності навчання користувачів (29,3%), питання безпеки та відповідності стандартам безпеки (24,8%) та інтеграції хмарна програм разом (21,6%). Детальніше дивіться будь ласка рисунок 2.7.

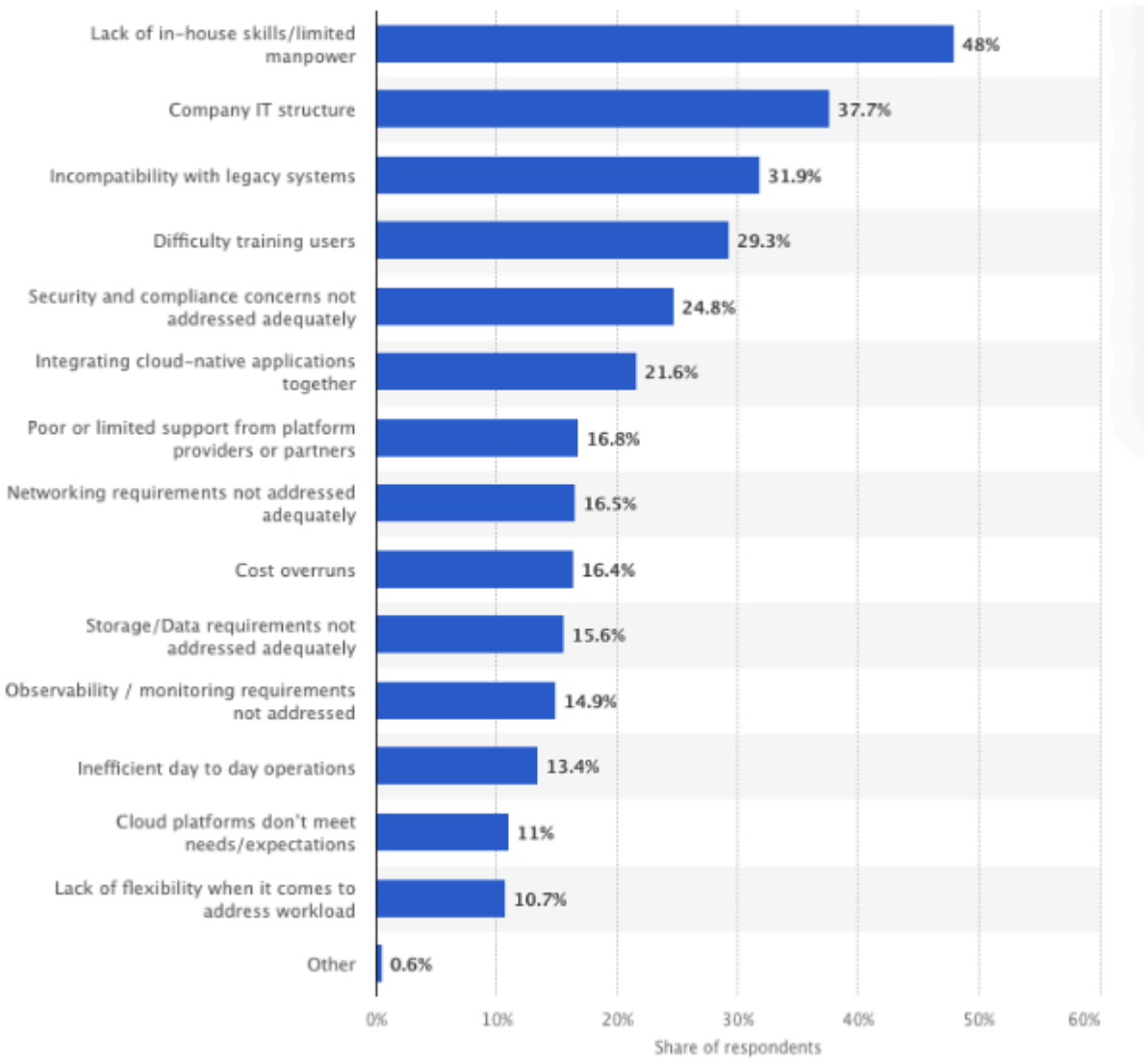


Рисунок 2.7 – Провідні виклики Kubernetes для компаній у всьому світі 2022

Також, у звіті “Sysdig 2022 Cloud-Native Security and Usage Report” [61, ст.29] виділені наступні ключові тенденції:

- Більш широке впровадження хмари та контейнерів призводить до багатьох небезпечних дій, таких як контейнери, що працюють від імені root і хмарні облікові записи, які мають надмірні привілеї. Однак постійне зростання Falco свідчить про те, що організації активно використовуються хмарні інструменти безпеки, щоб спробувати зменшити ризик.
- У середовищі Kubernetes складно планувати потужність. Тільки з належними обмеженнями ресурсу контейнера та безперервно моніторинг може

переконатися, що організації не перевитрачають або не ризикують проблемами продуктивності своїх програм.

- Kubernetes і контейнерні середовища продовжують розвиватися, оскільки організації переносять свої робочі навантаження в хмару. Використання відкриті стандарти, такі як Prometheus, і дотримання найкращих практик безпеки є критично важливою поведінкою, яка збільшить видимість і зниження ризику для хмарних додатків.

Окрім того, у звіті “State of Kubernetes Security Report 2023” [60, ст. 3], що підготовлений компанією “RedHat” вказується, що учасники опитування повідомили про затримку або сповільнення розгортання через проблеми безпеки Kubernetes (67%), дохід або втрата клієнтів через інцидент безпеки контейнера/Kubernetes (37%), називають безпеку головною проблемою для контейнерів і стратегій Kubernetes (38%), повідомили, що не мають ініціатив DevSecOps, а DevOps і Security залишаються розділеними (17%), сказали, що їхні існуючі контейнери та рішення безпеки Kubernetes сповільнюють розвиток (35%), та визначили вразливі місця як найбільшу проблему для свого контейнера та середовища Kubernetes (30%).

Будучи де-факто організатором контейнерів, впровадження Kubernetes продовжує зростати разом із попитом для хмарної рідної архітектури та збільшення контейнеризації. Це зростання було не завжди а потім таке саме зростання інвестицій у безпеку. Інвестування в безпеку контейнерів і Kubernetes означає розуміння складності Kubernetes і потенційні ризики безпеки, пов’язані з контейнерними програмами, а також впровадження необхідних елементів керування, які охоплюють усі шари контейнера та стек Kubernetes. Це включає базову інфраструктуру, площину керування Kubernetes, мережі, образи контейнерів і реєстри, а також багато інших компонентів.

Так, у відповіднь на питання “Що вас найбільше турбує щодо контейнерної стратегії вашої компанії?” учасники опитувань “State of Kubernetes Security Report 2023” [60, ст. 3] відповіли наступне:

- 38% – несерйозне ставлення до безпеки та неадекватне інвестування в безпеку;
- 25% – прогрес надто повільний;
- 14% – не враховує потреби відповідності стандартам безпеки;
- 13% – не усуває прогалини в навичках в нашій команді;
- 10% – не враховує культуру або обробляти зміни (рис. 2.8).

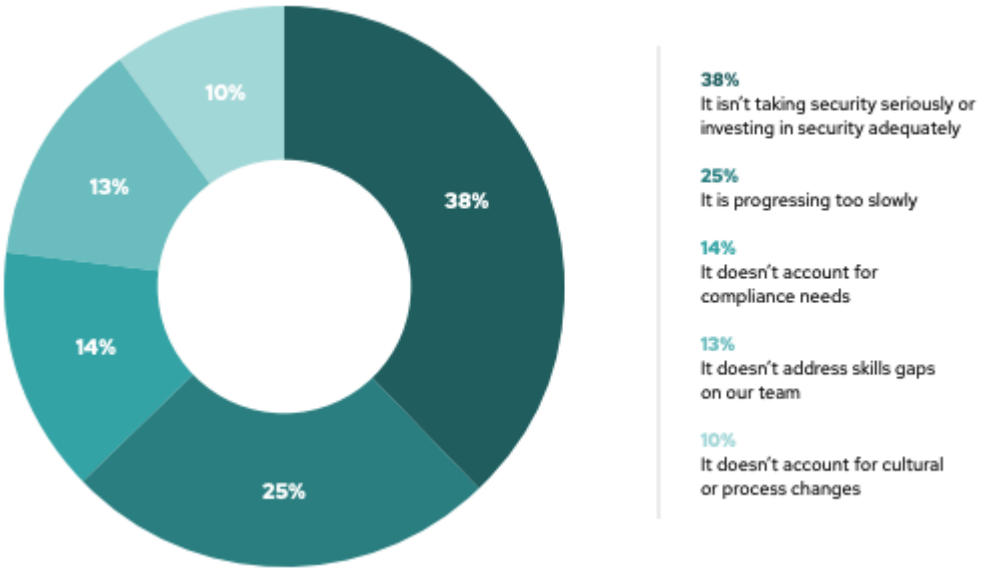


Рисунок 2.8 – Найчасті занепокоєння щодо контейнерної стратегії компанії

У звіті “The State of Kubernetes 2023” [65, ст. 14] відзначається важливість питань безпеки Kubernetes. Так, найбільше занепокоєння викликають неправильні конфігурації/експозиції (55%). У міру того як організації розширюються в хмари, ризик зростає кількість неправильних конфігурацій зростає. Інші важливі проблеми включають послідовне застосування політики для всіх кластерів і команди (42%), невиправлені CVE у дистрибутиві Kubernetes (42%), невідповідність (38%) і контроль доступу до кластерів (33%). Детальніше дивіться рисунок 2.9.

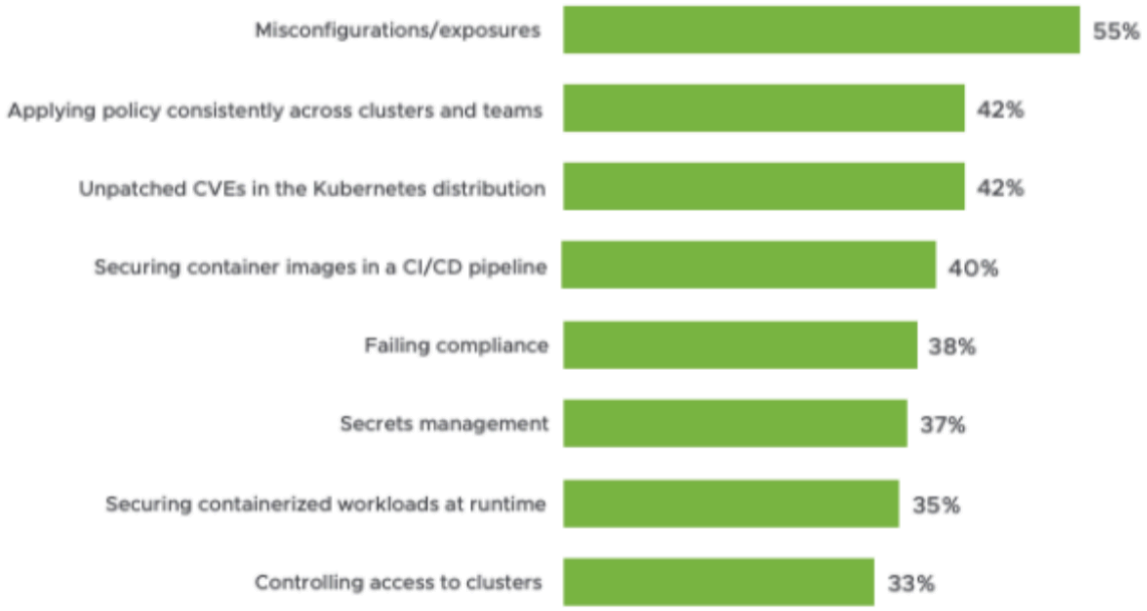


Рисунок 2.9 – Питання безпеки Kubernetes

Слід відзначити, що прагнучи стандартизувати та керувати використанням Kubernetes, 76% високоефективних організацій створили команди платформ [20]. Однак зростаючі проблеми, такі як складність, поширення інструментів, прогалини в навичках, вартість, безпека та відповідність вимогам, все ще залишаються, створюючи запит команд платформи на кращий підхід до управління та операцій Kubernetes.

Підсумовуючи аналіз проведених проблем та перешкод, із якими організації можуть зіткнутися під час впровадження Kubernetes, найбільш важливими є: 1) Kubernetes може виявитися складним для новачків через велику кількість параметрів конфігурації та необхідність ретельної настройки. Вказане вимагає глибокого розуміння архітектури та принципів його роботи; 2) Kubernetes надає велику кількість можливостей, але це також може стати викликом. Так, визначення оптимальних налаштувань та використання функцій може вимагати значних зусиль; 3) проблеми безпеки Kubernetes та складнощі із їх усуненням створюють додаткві обмежене на шляху його впровадження, де, в першу чергу, увага приділяється неправильним конфігураціям, вразливостям та контролю доступу; 4) різноманітність хмарних і локальних інфраструктур може призвести до труднощів у розгортанні і управлінні Kubernetes на різних

платформах. Вирішення цієї проблеми може вимагати додаткових зусиль для стандартизації середовищ; 5) впровадження Kubernetes може вимагати значних витрат на навчання персоналу, що включає в себе як навчання з основ контейнеризації, так і оволодіння роботою зі специфічними інструментами та поняттями Kubernetes; 6) ефективний моніторинг та забезпечення безпеки Kubernetes можуть виявитися викликом; 7) впровадження Kubernetes може вимагати змін в організаційній культурі та способі управління інфраструктурою, так як необхідно впроваджувати нові практики та процеси, що може спричинити опір з боку персоналу.

Не зважаючи на вищенаведене, Kubernetes став широко використовуваним стандартом управління контейнерами, і це призвело до значних користей для бізнесу у вигляді підвищеної ефективності, продуктивності та зменшення витрат.

Висновки до розділу 2

Таким чином, Kubernetes є невід'ємною частиною сучасних хмарних обчислень і використовується багатьма організаціями для розгортання критично важливих програм виробництва. Kubernetes стає ключовою платформою для переміщення навантаження в публічну хмару, з річним приростом і збільшенням кількості кластерів у хмарному середовищі. Ця тенденція до використання Kubernetes в хмарному середовищі ймовірно залишиться актуальною на протязі наступних років, зокрема завдяки розширенню інструментів для спрощення управління Kubernetes у хмарі.

Серед тенденцій Kubernetes слід також виділити те, що він забезпечує потужні інструменти для автоматизації процесів розгортання, масштабування, керування та відновлення додатків у контейнерах. Його стандартизована платформа дозволяє нормалізувати та стандартизувати використання в різних галузях. Окрім того, Kubernetes впливає на ефективність операцій та груп розробників, що є ключовими факторами для бізнесу. IT-оператори стають

ефективнішими, а розробники продуктивніші, що допомагає скорочувати час виходу на ринок та отримання вартості.

Узагальнюючи отримані результати та аналіз проблем, пов’язаних із впровадженням Kubernetes, можна виділити кілька ключових висновків. Попри те, що компанії активно прагнуть володіти технологією Kubernetes і використовувати її переваги, існують виклики, які стають на шляху успішного впровадження. Багато компаній вказують на відсутність внутрішніх навичок та обмежену робочу силу як основну проблему впровадження Kubernetes. Проблеми із IT-структурою та несумісністю із застарілими системами також виявилися важливими факторами. Зміни в організаційній культурі та способах управління, необхідність у великих витратах на навчання персоналу та необхідність стандартизації середовища також вносять свій внесок у складність впровадження Kubernetes. Питання безпеки Kubernetes стають джерелом занепокоєння для організацій, і учасники опитування вказують на проблеми зі затримкою розгортання та втратою клієнтів через інциденти безпеки. Особливу увагу приділяють неправильним конфігураціям, вразливостям та контролю доступу.

Незважаючи на ці виклики, важливо враховувати, що Kubernetes відзначається широким застосуванням і приносить значні переваги у вигляді підвищеної ефективності, продуктивності та зниження витрат для компаній, які успішно впроваджують цю технологію.

РОЗДІЛ 3. АНАЛІЗ ПРАКТИКИ ВИКОРИСТАННЯ KUBERNETES

3.1. Приклади реалізації Kubernetes

Для кращого розуміння сутності використання Kubernetes, варто детальніше розглянути його роль і можливості. Kubernetes, завдяки своїй потужності та гнучкості, став надзвичайно популярним у багатьох областях, і він може бути успішно використаний для керування контейнеризованими додатками в різних сферах.

Опитування щодо Kubernetes проведене компанією Dynatrace у 2023 [41] вказує вказує на наступні найпопулярніші категорії робочого навантаження Kubernetes (рис. 3.1).

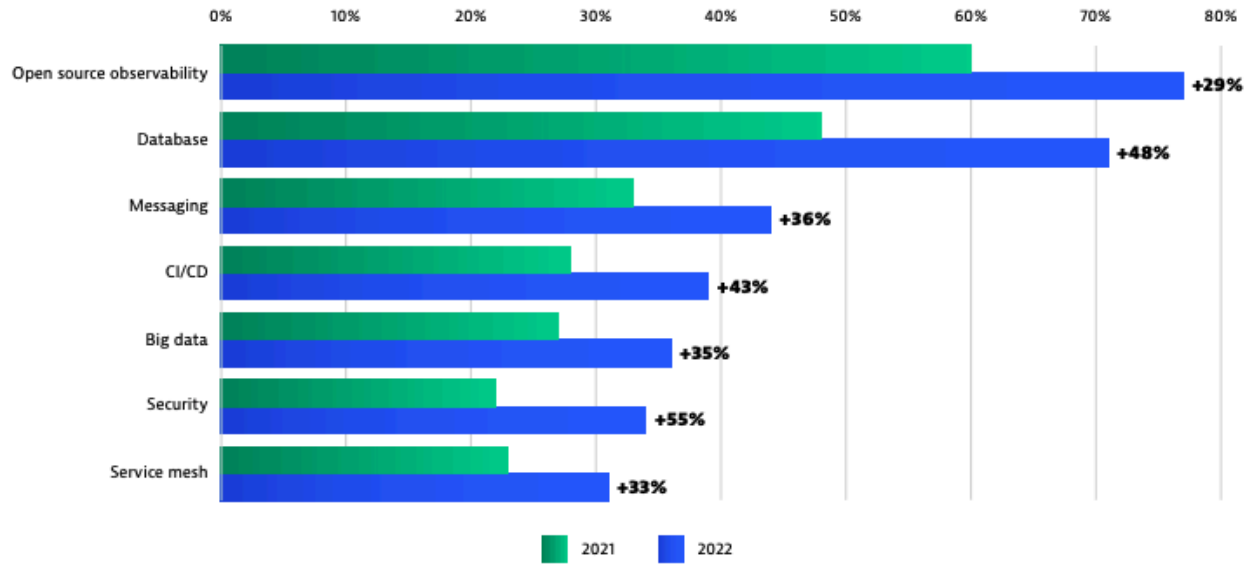


Рисунок 3.1 – Найпопулярніші категорії робочого навантаження Kubernetes

Так, у 2022 році організації визначили безпеку Kubernetes як головний пріоритет. Починаючи з низького базового рівня, відсоток збільшилася кількість організацій, які використовують інструменти безпеки Kubernetes з 22% у 2021 році до 34% у 2022 році. Це відповідає річний приріст +55%. Ця тенденція,

ймовірно, триватиме. Обізнаність про безпеку Kubernetes ще більше підвищується та з’являється новий клас рішень безпеки стає доступним.

71% організацій, які взяли участь в опитуванні Kubernetes, використовують бази даних і кеш-пам’яті в Kubernetes, що становить +48% зростання порівняно з минулим роком. Разом із системами обміну повідомленнями (+36% зростання), організації все частіше використовують бази даних і кеші для збереження станів робочого навантаження програми.

Також, зростали технології безперервної інтеграції та доставки (CI/CD). на +43% у порівнянні з минулим роком. Ця тенденція свідчить про те, що організації виділяють значно більше кластерів Kubernetes для роботи конвеєри створення, тестування та розгортання програмного забезпечення.

Найчастіше, Kubernetes використовується для розгортання веб-програм. Це допомагає в управлінні контейнерами програми, балансуванні навантаження, масштабуванні та поточних оновленнях. Такі інструменти, як NGINX Ingress Controller [48], можна використовувати для маршрутизації трафіку до програми. В той же час, розгортання веб-програм у Kubernetes накладає певні вимоги до архітектури такої програми. Зокрема, філософія дванадцятифакторного додатка [66] ефективно застосовується до програмного забезпечення, яке працюватиме в кластерному середовищі Kubernetes.

Kubernetes, також, добре підходить для керування додатками на основі мікросервісів. Кожен мікросервіс працює в окремому контейнері, і Kubernetes дозволяє легко масштабувати та керувати ним незалежно. Останніми роками Kubernetes став популярним серед організацій, які прагнуть створювати та розгортати мікросервіси, завдяки своїм потужним функціям і активній спільноті. Поєднання Kubernetes із мікросервісами може стати чудовим способом використовувати потужність обох технологій для досягнення максимальної ефективності та масштабованості [4].

Kubernetes використовується для запуску таких баз даних, як MySQL, PostgreSQL або MongoDB. StatefulSets використовуються для забезпечення

стійкості даних, і ви можете створювати репліки бази даних для високої доступності.

Також, останнім часом створюється багато операторів для розміщення та управління базами даними у Kubernetes. Зокрема, CloudNativePG – це оператор Kubernetes, який охоплює повний життєвий цикл високодоступного кластера бази даних PostgreSQL з основною/резервною архітектурою, використовуючи власну потокову реплікацію [22].

Kubernetes використовується для агрегації журналів і моніторингу за допомогою таких інструментів, як Prometheus [53] і Grafana [30]. Еластичний стек (Elasticsearch, Logstash, Kibana) [27] також можна використовувати для централізованого журналювання.

Моделі машинного навчання також розгортаються за допомогою Kubernetes. Kubernetes надає масштабовану та ресурсоефективну платформу для навчання та обслуговування моделей ML. Завдяки Kubernetes розгортання моделі ML стає більш гнучким, масштабованим і стійким. Платформа автоматично обробляє розподіл ресурсів, балансування навантаження та відмовостійкість, забезпечуючи плавне масштабування програм ML відповідно до зростаючих вимог [18].

Окрім того, Kubernetes можна використовувати для керування контейнерами на периферійних пристроях або шлюзах Інтернету речей. Це дозволяє ефективно розгортати крайові програми та керувати ними.

Kubernetes, також, використовують для виконання завдань пакетної обробки, які потребують значної обчислювальної потужності, як-от аналіз даних, перетворення даних або рендеринг. Kubernetes полегшує розгортання програм у кількох хмарних постачальниках або в гібридному середовищі. Такі інструменти, як KubeFed, дозволяють об’єднати кластери.

Kubernetes часто використовується для створення середовищ розробки та тестування, які дуже нагадують робочі налаштування. Це забезпечує узгодженість і дозволяє уникнути проблем, пов’язаних із “це працює на моїй машині”. Наприклад, Dynamic Feature Environment дозволяє інженерам

використовувати повністю функціональний незалежний кластер Kubernetes, який запускає наш повний набір мікросервісів [51].

Kubernetes можна також використовувати для керування великими обсягами даних за допомогою таких інструментів, як Apache Spark [55], Apache Hadoop [8] або Apache Flink [7]. Контейнерні робочі навантаження легше розгортати та керувати ними.

Слід згадати також, про використання Kubernetes для Edge Computing. Система, побудована на основі Kubernetes, зокрема K3S та kube edge, має численні переваги для впровадження в edge-обчислення. Її легкий бінарний об’єм (навіть менш ніж 70 МБ) та вбудована функціональність роблять її хорошим вибором для впровадження на пристроях з обмеженими ресурсами, як-от Raspberry Pi. Ця архітектура дозволяє автоматизувати процеси масштабування, забезпечити високу доступність та робити деплоймент без необхідності писати власні рішення. Інтеграція з різними пристроями дуже проста, що робить цю систему чудовою платформою для створення невеликих кластерів для експериментів або демонстрацій [72].

Це лише кілька прикладів того, як Kubernetes можна реалізувати в різних сценаріях. Kubernetes – це універсальна платформа, яку можна адаптувати до широкого діапазону випадків використання, що робить її популярним вибором для оркестровки контейнерів у сучасних IT-середовищах.

3.2. Історії успіху та отримані уроки у процесі впровадження Kubernetes

Kubernetes стає основним рішенням для компаній із великими командами розробників і складними IT-потребами. Це тому, що розробники програмного забезпечення можуть використовувати Kubernetes для автоматизації процесу створення складної IT-інфраструктури та керування нею.

Kubernetes вже використовується в багатьох галузях промисловості, і він продовжуватиме розвиватися, оскільки все більше урядів і організацій користуються перевагами його гнучкості та автоматизованих операцій [52].

Наприклад, Spotify прийняв Kubernetes для керування своєю постійно зростаючою кількістю послуг. Вони значно покращили процес випуску та час розгортання. Щоб забезпечити гнучкість і стійкість, Spotify прийняв гібридну хмарну стратегію, використовуючи послуги багатьох постачальників хмарних послуг. Kubernetes відіграв важливу роль в управлінні та оркеструванні багатохмарних розгортань Spotify, ефективно абстрагуючись від базової складності інфраструктури [3]. Spotify зрозумів переваги бути частиною великої спільноти, коли вони перейшли зі свого домашнього рішення на Kubernetes. Вони побачили цінність у використанні широко поширеної та підтримуваної платформи зі зростаючою спільнотою.

Джай Чакрабарті, директор з інженерії, інфрастуктури та експлуатації, Spotify відзначив, що “Ми побачили дивовижну спільноту, яка виросла навколо Kubernetes, і хотіли бути частиною цього. Ми хотіли отримати вигоду від додаткової швидкості та зниження вартості, а також узгодити з рештою індустрії найкращі практики та інструменти” [43].

OpenAI проводить ключові експерименти в таких сферах, як робототехніка та ігри, як в Azure, так і у власних центрах обробки даних, залежно від того, який кластер має вільну ємність. “Ми використовуємо Kubernetes переважно як систему пакетного планування та покладаємося на наш автомасштабувальник для динамічного масштабування нашого кластера”, – каже Крістофер Бернер, керівник відділу інфраструктури. “Це дозволяє нам значно скоротити витрати на неактивні вузли, забезпечуючи низьку затримку та швидку ітерацію” [43]. У 2021 році інженери OpenAI пролили світло на свою на велику подорож – масштабування Kubernetes до 7500 вузлів [57].

Аналізуючи використання OpenAI Kubernetes, слід відзначити, що Kubernetes зі своїм узгодженим API дозволяє легко переносити дослідницькі експерименти між кластерами, як у хмарі, так і у власних центрах обробки

даних. Ця мобільність забезпечує гнучкість у розміщенні робочого навантаження.

Pinterest дозволяє сотням мільйонів користувачів щомісяця ділитися та відкривати зображення на своєму веб-сайті та в мобільних додатках. З таким великим трафіком Pinterest потребувала ефективнішої та безпечнішої інфраструктури, ніж тисячі мікросервісів, які вони використовували раніше. Kubernetes допомагає Pinterest швидше розгортати проекти та створювати політики відновлення після збоїв.

“Перейшовши на Kubernetes, команда змогла розробити масштабування за вимогою та нові політики відновлення після відмови, а також спростити загальне розгортання та керування складною частиною інфраструктури, такою як Jenkins”, – говорить Майкл Бенедикт, менеджер із продуктів Cloud та групи інфраструктури даних у Pinterest. “Ми побачили не лише скорочення часу створення, але й значні переваги в ефективності. Наприклад, команда відновила понад 80 відсотків ємності в непікові години. У результаті кластер Jenkins Kubernetes тепер використовує на 30 відсотків менше робочих годин на день. у порівнянні з попереднім статичним кластером.” [43]

З аналізу бізнес-кейсу Pinterest [43] випливає, що у міру зростання організації керування кількома мікросервісами та різноманітними інфраструктурними інструментами та платформами може призвести до ускладнень. Добре продуманий перехід на нову платформу може допомогти оптимізувати інфраструктуру. Pinterest обрав поетапний підхід, починаючи з переміщення служб у контейнери Docker, а потім переходячи на Kubernetes. Такий підхід дозволяє вносити контрольовані зміни без переривання роботи.

Також, компанія Huawei, найбільший у світі виробник телекомунікаційного обладнання, зіткнулася із серйозними проблемами в управлінні величезною мережею додатків у численних центрах обробки даних [43]. Маючи понад 800 додатків, які працюють на понад 100 000 віртуальних машинах (ВМ), компанія боролася з вартістю та ефективністю керування цією розгалуженою інфраструктурою. Усвідомлюючи необхідність більш гнучкого

підходу, Huawei звернулася до контейнерної технології, зокрема Kubernetes, щоб оптимізувати свої внутрішні IT-операції.

Пейсінь Хоу, головний архітектор програмного забезпечення Huawei та директор спільноти з відкритим кодом, каже: “Kubernetes фактично вирішив більшість наших проблем. Раніше час розгортання займав близько тижня, тепер лише хвилини. Розробники задоволені. Цей відділ також цілком щасливий”.

Подорож Huawei з Kubernetes підкреслює важливість внутрішніх експериментів і впровадження нових технологій. Починаючи з ініціативи одного розробника, компанія поступово прийняла Kubernetes, що призвело до суттєвих переваг у гнучкості, масштабованості та економічній ефективності. Крім того, прихильність Huawei робити внесок у спільноту Kubernetes зміцнила її позицію як найкращого учасника, зміцнюючи принцип, що активна взаємодія зі спільнотами з відкритим кодом приносить взаємні вигоди.

Заглядаючи в майбутнє, Huawei прагне і далі використовувати Kubernetes для вдосконалення своїх пропозицій і розширення масштабу оркестровки. Завдяки постійному внеску в розробку Kubernetes та інтеграції нових функцій, таких як Container Ops, Huawei готова максимізувати потенціал хмарних технологій. Оскільки компанія продовжує відстоювати принципи використання хмарних технологій, вона очікує значних переваг у плані операційної ефективності та гнучкості як внутрішньо, так і для своїх клієнтів.

Слід зазначити, що VSCO, програма для мобільних фотографій, зіткнулася з проблемами ресурсоефективності та процесів ручного розгортання після переходу від Rackspace до AWS у 2015 році. Команда містила мікросервіси Node.js і Go в контейнери за допомогою Docker, але мала проблеми з окремими екземплярами EC2 для кожної служби, що призвело до марної витрати ресурсів. Щоб вирішити цю проблему, вони шукали рішення для консолідації та ефективності, зрештою вибравши Kubernetes замість таких альтернатив, як Mesos і Swarm.

Раніше розгортання вимагало “великої кількості налаштувань вручну, власного сценарію, який ми написали, і через наші різні екземпляри EC2 відділу

операцій доводилося доглядати за всім від початку до кінця”, – каже старший інженер програмного забезпечення Брендан Раян. “У нас насправді не було історії щодо методичного тестування та стандартизованого використання багаторазових контейнерів або збірок”. Тепер процес адаптації став швидшим. Раніше час для першого розгортання становив два дні практичного налаштування; зараз дві години. Завдяки переходу до постійної інтеграції, контейнеризації та Kubernetes швидкість значно зросла. Час від завершення коду до розгортання у виробництві на реальній інфраструктурі становив від одного-двох тижнів до двох-чотирьох годин для типової послуги. Гатту додає: “У людино-годинах це одна людина проти розробника та спеціаліста DevOps одночасно”. Зі скороченням часу для одного розгортання у виробництві на 80% кількість розгортань також зросла з 1200/рік до 3200/рік. Була також реальна економія доларів: завдяки Kubernetes VSCO працює з 2-20 разами більшою ефективністю EC2, залежно від послуги, додаючи приблизно 70% загальної економії на рахунку компанії EC2. Райан вказує на здатність компанії перейти від керування однією великою монолітною програмою до 50+ мікросервісів із “більш-менш однаковою командою розробників”. І ми змогли зробити це лише тому, що ми збільшили довіру до наших інструментів і набагато більше гнучкості, тому нам не потрібно залучати інженера DevOps для налаштування кожної служби”. Завдяки Kubernetes, gRPC і Envoy у VSCO загальний час простою в хвилинах скорочено на 88%, головним чином завдяки усуненню помилок схеми JSON і помилок інфраструктури надання послуг, а також підвищеній швидкості усунення збоїв [43] .

Успіх VSCO з проектами CNCF, включаючи внесок у gRPC і Envoy, вселив упевненість у експериментуванні з додатковими технологіями, такими як CNI і Prometheus. Команда цінує підтримку спільноти CNCF і передбачає продовження активної участі. Загалом застосування Kubernetes і пов’язаних технологій змінило процеси розробки та розгортання VSCO, що призвело до підвищення ефективності, економії коштів і підвищення надійності системи.

Отже, шлях VSCO підкреслює важливість узгодження вибору технологій із конкретними потребами, впровадження автоматизації для підвищення ефективності та активного залучення спільнот із відкритим кодом для стимулювання постійного вдосконалення та інновацій.

BlaBlaCar, у свою чергу, повністю використовує свою інфраструктуру на Google Cloud Platform, спираючись на сервіс GKE. Усі програми розгорнуті в контейнерах, включаючи бази даних [58]. “Коли ви переходите на цю хмарну модель і запускаєте все в контейнерах, ви повинні переконатися, що в будь-який момент ви можете перезавантажитися без будь-яких простоїв і без втрати трафіку. [З Kubernetes] наша інфраструктура набагато стійкіша, і ми мати кращу доступність, ніж раніше”, – вказує Саймон Лаллемант, інфраструктурний інженер BlaBlaCar [43].

“Перш ніж використовувати контейнери, для створення нового сервісу знадобився іноді день, іноді два”, – каже Лаллеманд. “З усіма інструментами, які ми створили для контейнерів, тепер копіювання нової служби займає лічені хвилини. Це справді величезна перевага. Ми краще плануємо потужність у нашому центрі обробки даних, оскільки у нас менше обмежень через цю абстракцію між послуги та апаратне забезпечення, на якому ми працюємо. Для розробників це також означає, що вони можуть зосередитися лише на функціях, які вони розробляють, а не на інфраструктурі”[43].

З прикладу BlaBlaCar [43] можна отримати кілька цінних уроків. BlaBlaCar вдалося масштабувати свою інфраструктуру, щоб не відставати від експоненційного зростання без переходу до хмарної віртуалізації. Такий підхід дозволив їм підтримувати продуктивність на власних “голих” серверах. BlaBlaCar першим запровадив контейнеризацію, визнаючи переваги цієї технології для ефективного масштабування та підтримки продуктивності. Контейнерізація в поєднанні з інструментами та робочими процесами значно підвищила ефективність розгортання сервісу. Час, необхідний для створення нових сервісів, скорочено з днів до хвилин.

Таким чином, ці історії успіху Kubernetes демонструють переваги Kubernetes з точки зору масштабованості, ефективності та швидшого розгортання. Уроки, отримані з цього досвіду, включають важливість навчання, моніторингу, тестування та обміну знаннями всередині організації для забезпечення успішного впровадження Kubernetes. Шлях кожної організації з Kubernetes унікальний, і ключ до успіху полягає в узгодженні Kubernetes з вашими конкретними бізнес-цілями та технічними цілями.

Висновки до розділу 3

У цьому розділі розглянути приклади реалізації Kubernetes у різних сценаріях, таких як веб-програми, мікросервіси, бази даних, моніторинг і журналювання, машинне навчання, Edge Computing, пакетна обробка даних, середовища розробки і тестування.

Практики та приклади використання Kubernetes проаналізовані на базі компаній Spotify, OpenAI, Pinterest, BlaBlaCar, VSCO та Huawei. Так, Spotify прийняв Kubernetes для управління своєю постійно зростаючою кількістю послуг. Перехід на Kubernetes дозволив Spotify покращити процес випуску та час розгортання, забезпечивши гнучкість і стійкість. Spotify відзначив переваги приєднання до великої спільноти Kubernetes. В той же час, OpenAI використовує Kubernetes для ключових експериментів в робототехніці. Здатність Kubernetes ефективно оркеструвати багатохмарні розгортання дозволяє їм гнучко переміщати дослідницькі експерименти між різними кластерами. Використання Kubernetes сприяє швидкості та зниженню витрат в інфраструктурі.

Окрім того, Pinterest використовує Kubernetes для швидкого розгортання проектів та політик відновлення після збоїв. Компанія відзначає, що Kubernetes допомагає відновити понад 80% ємності в непікові години, забезпечуючи ефективну та безпечну інфраструктуру. А от, BlaBlaCar успішно використовує Google Kubernetes Engine (GKE) для повного використання інфраструктури на Google Cloud Platform. Контейнеризація всіх програм дозволила підтримувати

продуктивність на власних серверах, уникнувши переходу до хмарної віртуалізації. Подорож Huawei з Kubernetes підкреслює важливість внутрішніх експериментів і впровадження нових технологій. Починаючи з ініціативи одного розробника, компанія поступово прийняла Kubernetes, що призвело до суттєвих переваг у гнучкості, масштабованості та економічній ефективності. Також, VSCO, популярний мобільний фотосервіс, здолав проблеми ресурсоефективності та ускладнення процесів ручного розгортання завдяки Kubernetes. Інженери використовували Kubernetes для контейнеризації та оркестрації мікросервісів, що призвело до значного покращення швидкості розгортання та збільшення числа розгортань у виробництві.

Загалом, історії успіху демонструють, що Kubernetes є потужним інструментом для покращення інфраструктури та допомагає компаніям досягати високої ефективності та гнучкості. Уроки, отримані з цих історій, можуть бути використані іншими організаціями для успішного впровадження Kubernetes у своєму середовищі.

РОЗДІЛ 4. РОЗРОБЛЕННЯ ТА ІМПЛЕМЕНТАЦІЯ KUBERNETES КЛАСТЕРА НА БАЗІ AMAZON ELASTIC KUBERNETES SERVICE

4.1. Аналіз вимог до Kubernetes кластера

Допустимо, що бізнес використовує застосунки з наступним технологічним стеком: 1) back-end – Python 3 із фреймворком Django; 2) front-end – Next.Js; 3) Databases – PostgreSQL, Redis; 4) WebServer – Nginx. Для ефективного використання такого стеку бізнес виносить наступні вимоги по інфраструктури:

- Інфраструктура бізнесу повинна легко масштабувати свої додатки вгору або вниз в залежності від попиту. Це важливо для обробки стрибків трафіку і забезпечення безперервного користувацького досвіду, забезпечуючи оптимальну продуктивність без ручного втручання.
- Автоматичне балансування навантаження та стійкість до відмов, забезпечуючи доступність додатків навіть у випадку відмов обладнання або інших проблем.
- Система повинна підтримувати розгортання нових версій програмного забезпечення та оновлень, зменшуючи простроченість та дозволяючи легко повертатися до попередніх версій в разі виникнення проблем. Вказане повинне зменшити час і зусилля, необхідні для виведення нових функцій та послуг на ринок.
- Інфраструктура повинна забезпечити ефективне використання ресурсів. Слід ефективно виділяти ресурси різним командам або відділам, забезпечуючи справедливе використання та контроль витрат.
- Інфраструктура повинна бути розміщена у хмарному середовищі, з потенціалом міграції в різні хмарні середовища, забезпечуючи гнучкість та уникнення залежності від постачальника.
- Архітектура мікросервісів повинна підтримуватися.

- Система повинна підтримувати технологію контейнерів, що спрощує упакування, розгортання та управління додатками.
- Конфігурації інфраструктури слід визначити як код (інфраструктура як код), що полегшить автоматизацію розгортання та управління інфраструктурою.
- Система повинна поєднуватися із практиками DevOps, сприяючи співпраці між командами розробки та експлуатації та підтримці безперервної інтеграції та безперервного розгортання (CI/CD).
- Інфраструктура повинна забезпечувати безпеку, включаючи ізоляцію між додатками та можливість безпечно керувати конфіденційними даними, а, також, підтримувати дотримання стандартів безпеки відповідної галузі.

Для задоволення цих бізнес-вимог може використовувати Kubernetes, що допоможе покращити можливість компанії надавати надійні, масштабовані та ефективні послуги, сприяти інноваціям, зменшувати оперативне навантаження та забезпечувати конкурентну перевагу в цифровому середовищі.

Розгортання кластера Kubernetes слід здійснити на Amazon EKS за допомогою Terraform із відповідними системними компонентами і використанням Argo CD для розгортання додатків. Так, Terraform [62] – це інструмент із відкритим вихідним кодом “Інфраструктура як код” (IaC), який надає спосіб визначення та надання інфраструктури, зокрема хмарних ресурсів, декларативним способом із керуванням версіями. Його основна мета – автоматизувати та оптимізувати процес надання інфраструктури та управління нею. В той же час як Argo CD [9] – це декларативний інструмент безперервної доставки GitOps, розроблений для Kubernetes. Він використовується для керування та автоматизації розгортання, конфігурації та синхронізації додатків, що працюють у кластерах Kubernetes.

Схематичне зображення інфраструктури AWS хмарного середовища для розгортання Kubernetes кластера на базі Amazon EKS зображене на рисунку 4.1.

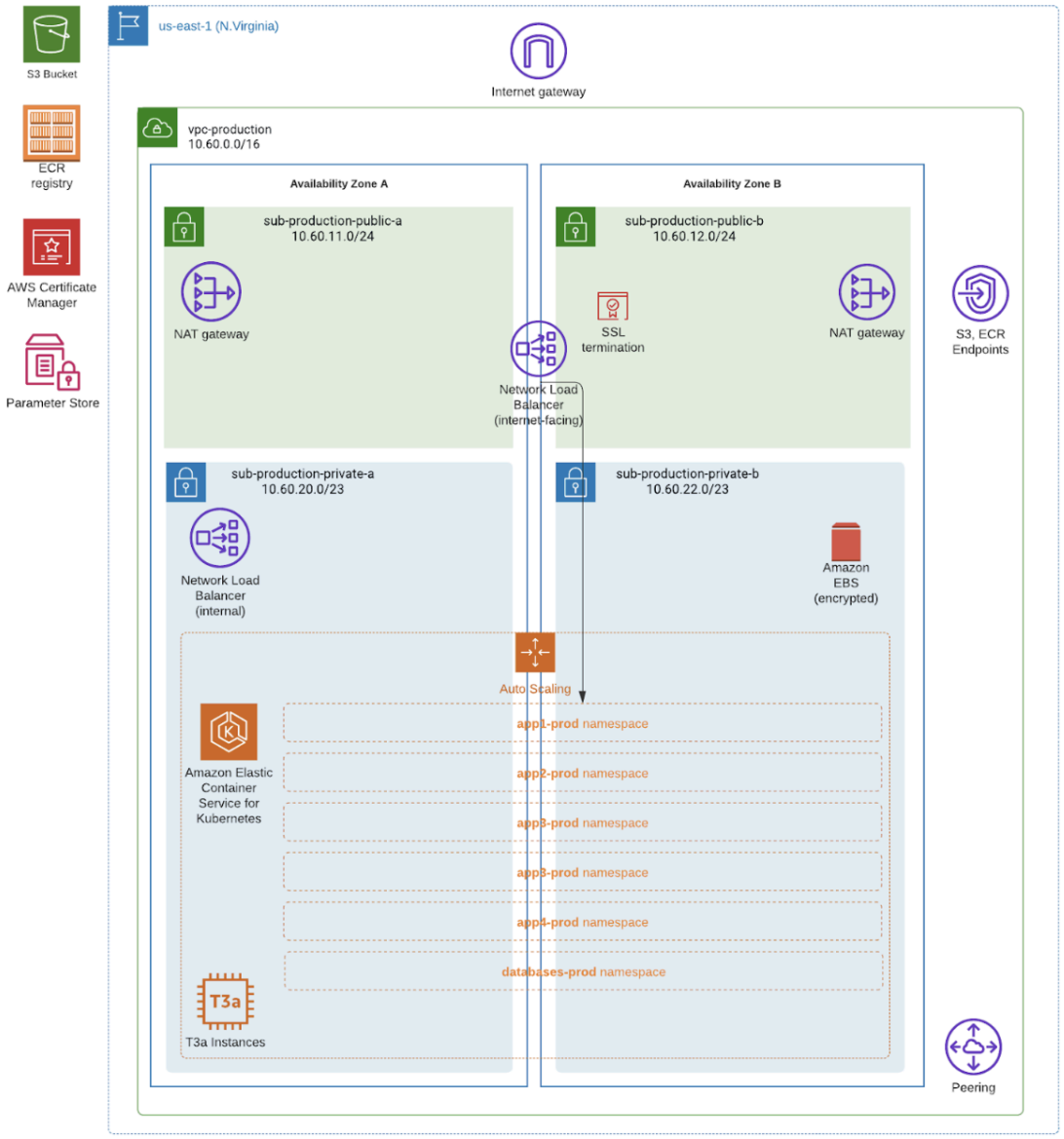


Рисунок 4.1 – Діаграма інфраструктури AWS для розгортання Kubernetes кластера на базі Amazon EKS

Для реалізації вказаного проєкту потрібні наступні попередні умови:

- AWS-акаунт з необхідними дозволами для створення кластерів EKS, ролей IAM, VPC та інших необхідних ресурсів;
- переконайтеся, що Terraform встановлено на вашому локальному комп’ютері або в середовищі CI/CD;
- налаштуйте ролі IAM з відповідними дозволами для ваших вузлів, контрольної площадки та інших ресурсів;

- створіть і налаштуйте репозиторій AWS Elastic Container Registry (ECR) для зберігання зображень контейнерів Docker, які використовуються вашими програмами;
- використовуйте сховище параметрів сховище параметрів AWS Systems Manager (SSM) для безпечного зберігання параметрів конфігурації, секретів та інших конфіденційних даних, необхідних для ваших програм, і керування ними.

Розгортання і налаштування кластера Kubernetes на Amazon EKS можна здійснити за допомогою Terraform в наступні етапи:

1. створення VPC та мережевих ресурсів (віртуальної приватної мережі (VPC), підмереж, шлюзу NAT, тощо);
2. створення кластера EKS, а саме визначте ресурси кластера EKS, вказавши бажану версію Kubernetes, ролі IAM, налаштування мережі та інші параметри;
3. конфігурація група вузлів і автомасштабування, в тому числі групи автомасштабування для робочих вузлів;
4. інсталяція утиліти для автомасштабування кластера для AWS за допомогою менеджера пакерів для Kubernetes – Helm;
5. встановлення служби метрик (Metrics Server) за допомогою Helm, що включає розгортання служби метрик Kubernetes для надання метрик використання ресурсів;
6. розгортання контролера для балансувальника навантаження за допомогою Helm (AWS Load Balancer Controller);
7. встановлення менеджера сертифікатів (Cert Manager) в кластері EKS через Helm;
8. встановлення Argo CD за допомогою Helm;
9. розгортання додатків за допомогою Argo CD, а саме розробка конфігураційних файлів Kubernetes об’єктів додатків Argo CD, які вказують, як розгортати ваші додатки, розгортання додатки за допомогою Argo CD.

Вказані етапи будуть детально розглянути в наступних підрозділах.

4.2. Автоматизація розгортання кластера за допомогою Terraform

Розгортання Terraform повинне здійснюватися із використанням найкращих практик для автоматизації Terraform, таких як безпечне зберігання файлів стану, використання віддалених інфраструктурних носіїв, версіонування та використання автоматичного конвеєра розгортання коду Terraform. Так, рисунок 4.2 містить діаграму конвеєра розгортання для Terraform за допомогою GitHub Actions.

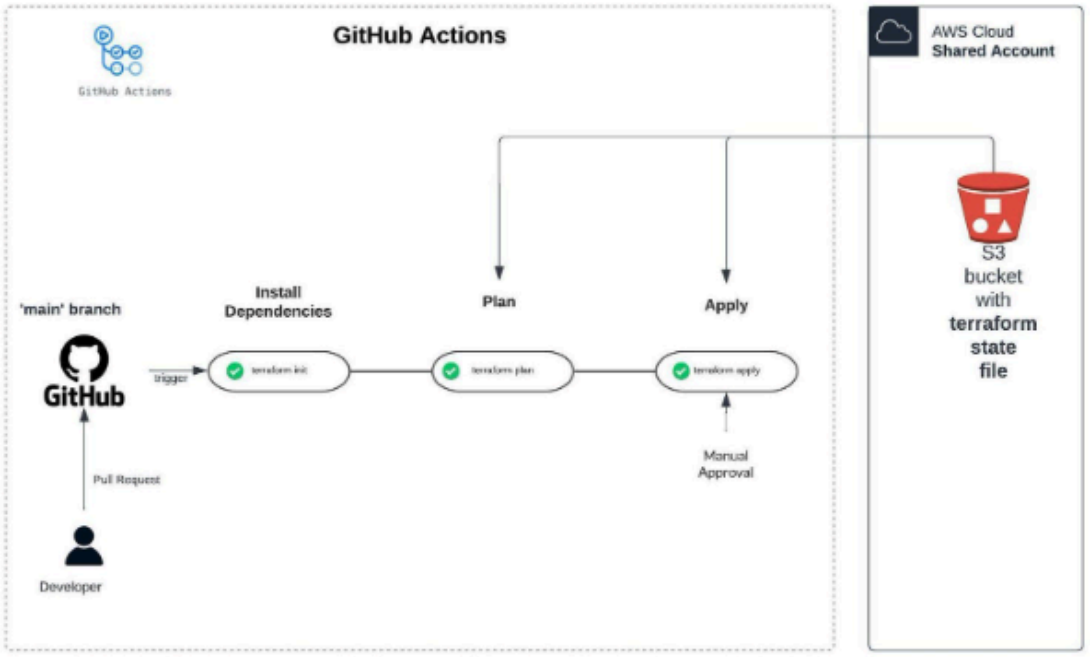


Рисунок 4.2 – Діаграма конвеєра розгортання для Terraform за допомогою GitHub Actions

Цей конвеєр GitHub автоматизує процес розгортання за допомогою Terraform для надання інфраструктури на хмарній платформі AWS. Розберемо його компоненти:

1. тригери конвеєра, де конвеєр запускається за двома подіями: push до головної гілки та pull_request;

2. визначення дозволів для дій у конвеєрі, зокрема запис маркерів ідентифікатора, читання вмісту та написання запитів на отримання:

```
permissions:
    id-token: write
    contents: read
    pull-requests: write
```

3. встановлення змінних середовища, необхідних для конвеєра, наприклад TF_LOG для журналювання Terraform і AWS_REGION, отримані з секретів GitHub:

```
env:
 TF_LOG: INFO
 AWS_REGION: ${{ secrets.AWS_REGION }}
```

4. конвеєр включає наступні кроки:

- “git checkout” перевіряє код сховища:

```
- name: Git checkout
      uses: actions/checkout@v3
```

- налаштування облікових данних AWS за допомогою ролей AWS IAM, отриманих із секретів GitHub:

```
- name: Configure AWS credentials from AWS account
      uses: aws-actions/configure-aws-credentials@v1
      with:
        role-to-assume: ${{ secrets.AWS_ROLE }}
        aws-region: ${{ secrets.AWS_REGION }}
        role-session-name: GitHub-OIDC-TERRAFORM
```

- налаштування Terraform із зазначеною версією:

```
- name: Setup Terraform
      uses: hashicorp/setup-terraform@v2
      with:
        terraform_version: 1.5.4
```

- “terraform fmt” форматує файли Terraform і перевіряє синтаксичні помилки:

```
- name: Terraform fmt
    id: fmt
    run: terraform fmt -check
    continue-on-error
```

- “terraform init” ініціалізує Terraform із зазначеною конфігурацією серверної частини:

```
- name: Terraform Init
    id: init
    env:
      AWS_BUCKET_NAME: ${{ secrets.AWS_BUCKET_NAME }}
      AWS_BUCKET_KEY_NAME: ${{ secrets.AWS_BUCKET_KEY_NAME }}
    run: terraform init -backend-config="bucket=${AWS_BUCKET_NAME}"
-backend-config="key=${AWS_BUCKET_KEY_NAME}"
-backend-config="region=${AWS_REGION}"
```

- “terraform validate” перевіряє конфігураційні файли Terraform:

```
- name: Terraform Validate
    id: validate
    run: terraform validate -no-color
```

- “terraform plan” генерує план виконання змін, які будуть застосовані (тільки для запитів на отримання):

```
- name: Terraform Plan
    id: plan
    run: terraform plan -no-color
    if: github.event_name == 'pull_request'
    continue-on-error: true
```

- “actions/github-script” генерує коментар до запиту на отримання з результатами форматування, ініціалізації, перевірки та плану виконання (якщо доступний):

```
- uses: actions/github-script@v6
    if: github.event_name == 'pull_request'
    env:
      PLAN: "terraform\n${{ steps.plan.outputs.stdout }}"
    with:
      github-token: ${{ secrets.GITHUB_TOKEN }}
      script: |
        const output = `#### Terraform Format and Style 🖌\`${{ steps.fmt.outcome }}\`
        #### Terraform Initialization ⚙️\`${{ steps.init.outcome }}\`
        #### Terraform Validation 🤖\`${{ steps.validate.outcome }}\`
        <details><summary>Validation Output</summary>
```

- статус плану Terraform перевіряє, чи успішно виконано план; якщо ні, завдання завершується з помилкою:

```
- name: Terraform Plan Status
    if: steps.plan.outcome == 'failure'
    run: exit 1
```

- “terraform apply” автоматично застосовує зміни Terraform лише для надсилання до головної гілки:

```
- name: Terraform Apply
    if: github.ref == 'refs/heads/main' && github.event_name == 'push'
    run: terraform apply -auto-approve -input=false
```

Цей конвеєр забезпечує послідовне розгортання інфраструктури з використанням Terraform, надаючи зворотний зв’язок щодо запитів на отримання щодо форматування, перевірки та запланованих змін. Він також забезпечує додаткові перевірки, такі як вихід у разі помилки під час виконання плану та обмеження автоматичного застосування до основної гілки.

Для розгортання мережевих ресурсів в AWS може використовуватися модуль Terraform “AWS VPC Terraform module” [14]. ‘local values’ були використані у коді Terraform. Локальне значення призначає ім’я виразу, тому ви можете використовувати ім’я кілька разів у межах модуля замість повторення виразу:

```
locals {
 aws_region      = "us-east-1"
 az_for_compute  = "${local.aws_region}c"
 environment     = "development"
 cluster_name    = "eks-${local.environment}-1"
 cluster_version = "1.28"
 whitelisted_ip = [
   "1.1.1.1/32", #Yurii Home
 ]
 azs = formatlist("${local.aws_region}%s", ["b", "c"])
 argocd_namespace_name = "argocd"
}
```

Нижче наведений фрагмент коду Terraform визначає модуль Terraform для створення віртуальної приватної хмари (VPC) в AWS за допомогою модуля “terraform-aws-modules/vpc/aws”, зокрема версії “5.1.1”:

```
module "vpc" {
 source  = "terraform-aws-modules/vpc/aws"
 version = "5.1.1"
 name = local.vpc_name
 cidr = local.vpc_cidr
```

Цей код Terraform визначає VPC з публічною та приватною підмережами в AWS. Він використовує вказаний модуль для створення та налаштування VPC на основі наданих параметрів і налаштувань. Код також активує функції DNS і шлюзи NAT, водночас вказуючи, що певні ресурси за замовчуванням не повинні керуватися цим модулем:

```
 manage_default_network_acl    = false
 manage_default_route_table    = false
 manage_default_security_group = false
 enable_dns_hostnames = true
 enable_dns_support   = true
 enable_nat_gateway   = true
```

VPC має тег для ідентифікації та асоціації з кластером Kubernetes і різними мережевими ролями:

```
public_subnet_tags = {
  "kubernetes.io/cluster/${local.eks_cluster_name}" = "shared"
  "kubernetes.io/role/elb"                      = "1"
  "Tier"                                    = "public"
 }
 private_subnet_tags = {
  "kubernetes.io/cluster/${local.eks_cluster_name}" = "shared"
  "kubernetes.io/role/internal-elb"             = "1"
  "Tier"                                    = "private"
 }
```

Amazon EKS кластер розгорнутий за допомогою модуля Terraform “AWS EKS Terraform module” [11]. Фрагмент коду Terraform із модулем Terraform для створення кластера Amazon Elastic Kubernetes Service (EKS) наведений в нижче Зокрема, цей код встановлює та конфігурує різні аспекти кластера EKS, такі як параметри мережі, групи вузлів, безпеку та дозволи. Код містить модуль “eks”, отриманий із модуля “terraform-aws-modules/eks/aws” у реєстрі Terraform. Використана версія “19.16.0”. Ключові компоненти цього коду включають:

- конфігурація кластера включає “cluster_name” ім’я кластера EKS походить від змінної local.cluster_name та “cluster_version” визначає версію Kubernetes для кластера EKS на основі змінної local.cluster_version:

```
module "eks" {
 source          = "terraform-aws-modules/eks/aws"
 version         = "19.16.0"
 cluster_name    = local.cluster_name
```

```
cluster_version = local.cluster_version
```

- VPC і підмережі включають vpc_id, чим посилається на ідентифікатор віртуальної приватної хмари (VPC), де буде створено кластер EKS. ідентифікатор VPC отримується з джерела даних віддаленого стану, subnet_ids – ідентифікує приватні підмережі в межах VPC, де будуть розміщені робочі вузли кластера EKS, та control_plane_subnet_ids, що ідентифікує підмережі для площини керування кластером EKS:

```
vpc_id                = data.terraform_remote_state.data_core.outputs.vpc_id
subnet_ids            = data.terraform_remote_state.data_core.outputs.private_subnets
control_plane_subnet_ids = data.terraform_remote_state.data_core.outputs.intra_subnets
```

- “enable_irsa” вказує, чи потрібно ввімкнути ролі IAM для облікових записів служб (IRSA) для кластера EKS.

```
enable_irsa = true
```

- конфігурація кінцевої точки кластера включаються cluster_endpoint_private_access, що визначає, чи повинен кластер EKS мати приватну кінцеву точку сервера API, cluster_endpoint_public_access, який визначає, чи повинен кластер EKS мати публічну кінцеву точку сервера API, а також cluster_endpoint_public_access_cidrs, що перелічує блоки IP CIDR із білого списку, які можуть отримати доступ до загальнодоступної кінцевої точки сервера API:

```
cluster_endpoint_private_access      = true
cluster_endpoint_public_access       = true
cluster_endpoint_public_access_cidrs = local.whitelisted_ip
```

- додатки кластера визначаються різні надбудови кластера, такі як CoreDNS, kube-proxy, VPC CNI та драйвер AWS EBS CSI, що забезпечені конфігураціями та відомостями про роль IAM:

```
cluster_addons = {
  coredns = {
    most_recent       = true
    resolve_conflicts = "OVERWRITE"
  }
  kube-proxy = {
    most_recent       = true
    resolve_conflicts = "OVERWRITE"
  }
  vpc-cni = {
    most_recent       = true
    resolve_conflicts = "OVERWRITE"
  }
}
```

- адміністратори ключів KMS визначають користувачів і ролі IAM, які є адміністраторами для ключа KMS, який використовується для шифрування даних:

```
kms_key_administrators = [
  "arn:aws:iam::**********:user/yurii@gmail.com",
  "arn:aws:iam::**********:role/develop-terraform-role"
 ]
```

- правила групи безпеки кластера розширюють правила для групи безпеки кластера EKS, зокрама у цьому коді є правило дозволу входу з вузлів на тимчасових портах:

```
cluster_security_group_additional_rules = {
  ingress_nodes_ephemeral_ports_tcp = {
    description              = "Nodes on ephemeral ports"
    protocol                 = "tcp"
    from_port                = 1025
    to_port                  = 65535
    type                     = "ingress"
    source_node_security_group = true
  }
```

```
}
```

- правила групи безпеки вузла розширюють правила для групи безпеки вузла, дозволяючи весь трафік між вузлами.

```
node_security_group_additional_rules = {
  ingress_self_all = {
    description = "Node to node all ports/protocols"
    protocol    = "-1"
    from_port   = 0
    to_port     = 0
    type        = "ingress"
    self        = true
  }
```

- групи керованих вузлів EKS визначають одну або кілька груп керованих вузлів для кластера EKS, де кожна група вузлів визначає такі деталі, як типи екземплярів, розмір диска та бажану ємність:

```
eks_managed_node_groups = {
  nodes1 = {
    name           = "${local.environment}-sys-mng"
    use_name_prefix = true

    subnet_ids = [data.terraform_remote_state.data_core.outputs.private_subnets[0]]
    disk_size  = 30
    min_size     = 2
    max_size     = 3
    desired_size = 2
    instance_types = ["t3a.medium"]
    capacity_type  = "ON_DEMAND"
```

Ці групи вузлів налаштовані за допомогою пов’язаних тегів, ролей та інших параметрів:

```
tags = {
     "k8s.io/cluster-autoscaler/enabled"              = "true"
```

```
    "k8s.io/cluster-autoscaler/${local.cluster_name}" = "owned"
    "kubernetes.io/cluster/${local.cluster_name}"     = "owned"
    Name                                              = "${local.environment}-sys-mng"
  }
 }
```

- конфігурація автентифікації AWS, включає manage_aws_auth_configmap, який визначає, чи потрібно керувати AWS Auth ConfigMap для налаштування доступу Kubernetes, aws_auth_roles і aws_auth_users, які використовуютсья для визначення ролі IAM і користувачів із пов’язаними ролями Kubernetes для доступу до кластера:

```
manage_aws_auth_configmap = true
aws_auth_roles = [
{
    rolearn  = module.eks_admins_iam_role.iam_role_arn
    username = "EKS Admin One Role"
    groups   = ["system:masters"]
  },
]
aws_auth_users = [
  {
    userarn  = "arn:aws:iam::${local.aws_account}:user/yurii@company.com"
    username = "Yurii Vatsyk"
    groups   = ["system:masters"]
  }
 ]
```

- теги застосовуються до різних ресурсів, створених у складі кластера EKS, забезпечуючи ідентифікацію та метадані:

```
tags = {
  Owner       = "Terraform"
  Environment = "${local.environment}"
  Source      = "Terraform"
 }
```

Важливим елементом розгортання кластера є конфігурація доступів для Kubernetes сервіс акаунтів до AWS через використання постачальника IAM OIDC. У 2014 році AWS Identity and Access Management додала підтримку федеративних ідентифікацій за допомогою OpenID Connect (OIDC). Ця функція дозволяє автентифікувати виклики AWS API за допомогою підтримуваних постачальників ідентифікаційної інформації та отримувати дійсний веб-токен OIDC JSON (JWT). Цей маркер операції може бути переданий до AWS STS AssumeRoleWithWebIdentity API та отримати облікові дані тимчасової ролі IAM. Ці облікові дані можуть бути використані для взаємодії з будь-яким сервісом AWS, включаючи Amazon S3 і DynamoDB.

Нижче наведений код, що є частиною конфігурації Terraform, яка використовується для створення ролі IAM (Identity and Access Management) для сервісних облікових записів в Amazon Web Services (AWS) для керування контролером балансування навантаження. Цей код використовує Terraform модуль “IAM Role for Service Accounts in EKS”:

```
module "load_balancer_controller_irsa_role" {
 source = "terraform-aws-modules/iam/aws//modules/iam-role-for-service-accounts-eks"
 role_name                              = "load-balancer-controller"
 attach_load_balancer_controller_policy = true
 oidc_providers = {
   ex = {
     provider_arn               = module.eks.oidc_provider_arn
     namespace_service_accounts = ["kube-system:aws-load-balancer-controller"]
   }
 }
 tags = local.tags
}
```

Основні параметри цього модуля є:

- source – шлях до модуля Terraform, який використовується для створення ролі IAM;
- role_name – назва ролі, яку буде створено;

- attach_load_balancer_controller_policy – параметр, який вказує, чи потрібно прикріпити політику для керування контролером балансування навантаження;
- oidc_providers – параметр, який визначає OIDC (OpenID Connect) провайдера, що використовується для аутентифікації користувачів;
- tags – мітки, які будуть назначені створеній ролі.

4.3. Налаштування системних компонентів Kubernetes кластера (Cluster Autoscaler, Metrics Server, AWS Load Balancer Controller, Cert Manager)

Для ефективної роботи та виконання бізнес вимог необхідно встановити такі системні компоненти та утиліти для Kubernetes як утиліту для автомасштабування кластера для AWS (Cluster Autoscaler on AWS) [23], службу метрик (Kubernetes Metrics Server) [42], контролер для балансувальника (AWS Load Balancer Controller) [13] та менеджер сертифікатів (Cert Manager) [19].

Утиліта для автомасштабування кластера для AWS (Cluster Autoscaler on AWS) відповідає за те, щоб у вашому кластері було достатньо вузлів для планування ваших модулів, не витрачаючи ресурси. Він спостерігає за модулями, які не виконують планування, і за вузлами, які недостатньо використовуються. Потім він моделює додавання або видалення вузлів перед застосуванням змін до вашого кластера. Реалізація AWS Cloud Provider у Cluster Autoscaler контролює поле DesiredReplicas ваших груп автоматичного масштабування EC2. Рисунок 4.3 містить схему роботи Cluster Autoscaler в AWS, яка зображує принцип роботи Cluster Autoscaler із моніторингу об’єктів із недостатньо кількістю ресурсів і додаванням вузлів до груп вузлів кластера.

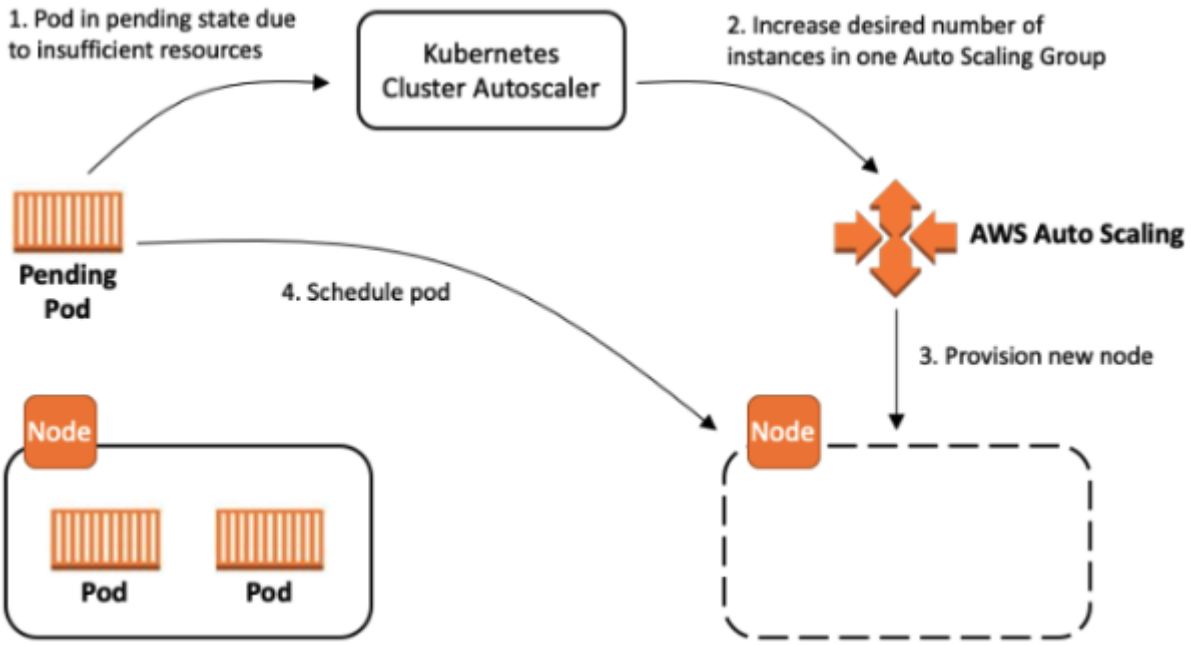


Рисунок 4.3 – Схема роботи Cluster Autoscaler в AWS

Cluster Autoscaler встановлюється за допомогою менеджера пакетів для Kubernetes – Helm [31] та Terraform. Зокрема, фрагмент коду Terraform із ресуром випуску Helm для розгортання Cluster Autoscaler у кластері Kubernetes нижче:

```
resource "helm_release" "cluster_autoscaler" {
 name       = "cluster-autoscaler"
 repository = "https://kubernetes.github.io/autoscaler"
 chart      = "cluster-autoscaler"
 version    = "9.29.3"
 namespace  = "kube-system"
  set {
   name  = "image.tag"
   value = "v1.28.0"
 }
 set {
   name  = "autoDiscovery.clusterName"
   value = local.cluster_name
 }
 set {
   name  = "awsRegion"
   value = local.aws_region
```

```
 }
```

Цей код Terraform відповідає за розгортання Cluster Autoscaler як випуску Helm у просторі імен “kube-system” кластера Kubernetes. Він містить діаграму Helm та її параметри конфігурації, що дозволяє налаштовувати поведінку автомасштабувальника:

```
set {
  name  = "rbac.serviceAccount.name"
  value = "cluster-autoscaler"
 }
 set {
  name  = "rbac.serviceAccount.annotations.eks\\.amazonaws\\.com/role-arn"
  value = module.clusterautoscalerrole.iam_role_arn
 }
 depends_on = [
  module.eks
 ]
}
```

Розгортання залежить від створення кластера EKS, гарантуючи, що засіб автоматичного масштабування кластера розгортається в повністю підготовленому середовищі.

Окрім того, служба метрик (Kubernetes Metrics Server) є агрегатором даних про використання ресурсів у згаданому кластері. Вона не розгортається за замовчуванням у кластерах Amazon EKS. Metrics Server зазвичай використовується іншими додатками Kubernetes, такими як Horizontal Pod Autoscaler або Kubernetes Dashboard. Нижче наведений фрагмент коду Terraform із ресуром випуску Helm для розгортання Metrics Server у кластері Kubernetes:

```
resource "helm_release" "cert_manager" {
 name            = "cert-manager"
 repository      = "https://charts.jetstack.io"
 chart           = "cert-manager"
 version         = "v1.12.3"
```

```
 namespace        = "cert-manager"
 create_namespace = true
 values = [
   "${file("values/cert-manager-values.yaml")}"
 ]
depends_on = [
   module.eks,
 ]
}
```

AWS Load Balancer Controller – це контролер, який допомагає керувати еластичними балансувальниками навантаження для кластера Kubernetes. Він задовольняє ресурси Kubernetes Ingress, надаючи балансувальники навантаження програм (Application Load Balancers) та ресурси служби Kubernetes, надаючи балансувальники мережевого навантаження (Network Load Balancers). Зокрема, ось зразок коду Terraform із ресуром випуску Helm для розгортання AWS Load Balancer Controller у кластері Kubernetes:

```
resource "helm_release" "aws-load-balancer-controller" {
 name = "aws-load-balancer-controller"
 repository = "https://aws.github.io/eks-charts"
 chart      = "aws-load-balancer-controller"
 version    = "1.4.5"
 namespace  = "kube-system"
 values = [
   "${file("values/aws-lb-values.yaml")}"
 ]
  depends_on = [helm_release.cert_manager]
}
```

На рисунку 4.4 зображена діаграма, яка детально описано компоненти AWS, які створює цей контролер. Він також демонструє маршрут вхідного трафіку від ALB до кластера Kubernetes [33].

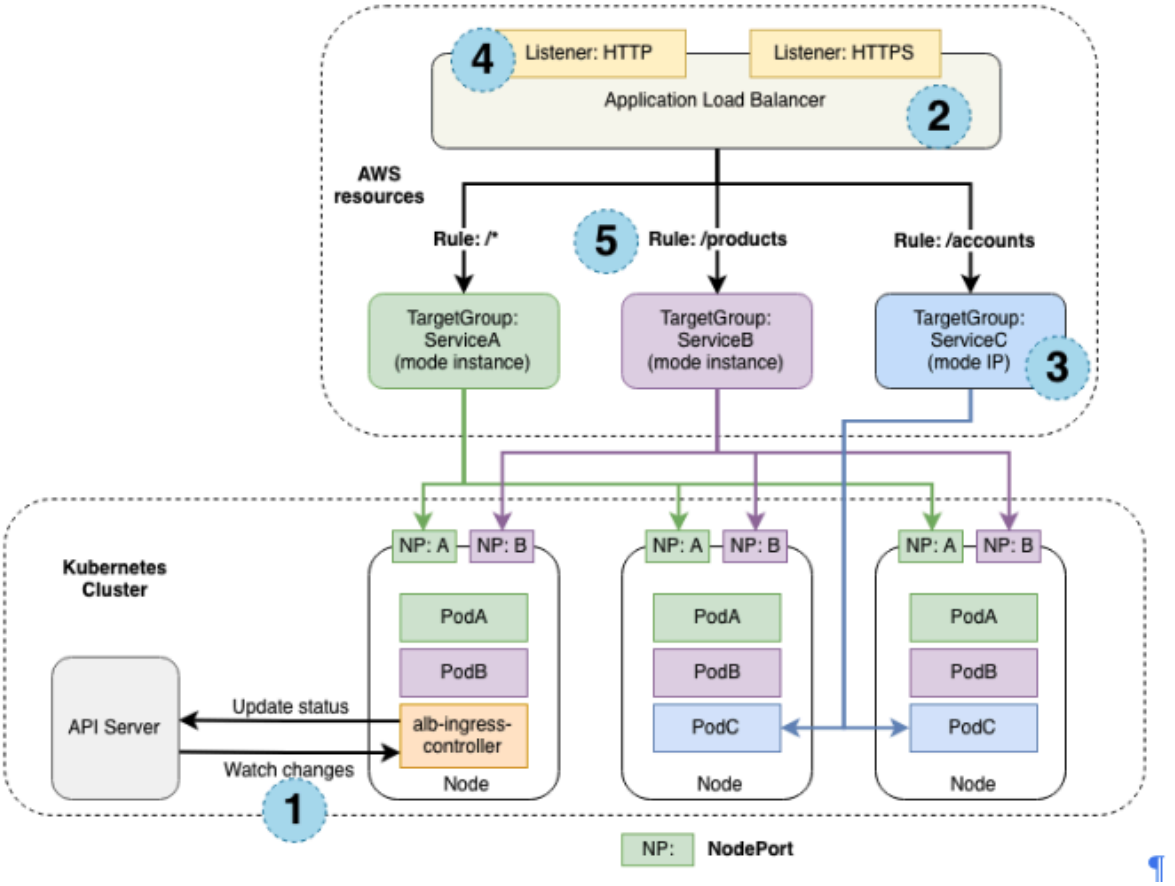


Рисунок 4.4 – Діаграма із компоненти AWS, які створені AWS Load Balancer Controller

Також, досить часто використовується NGINX Ingress Controller – це реалізація Ingress Controller для NGINX і NGINX Plus, яка може балансувати навантаження додатків Websocket, gRPC, TCP і UDP. Він підтримує стандартні функції Ingress, такі як маршрутизація на основі вмісту та завершення TLS/SSL. Кілька функцій NGINX і NGINX Plus доступні як розширення для ресурсів Ingress через анотації та ресурс ConfigMap. Для прикладу, ще код Terraform із ресуром випуску Helm для розгортання NGINX Ingress Controller у кластері Kubernetes:

```
resource "helm_release" "ingress_nginx_external" {
 name       = "ingress-nginx-external"
 repository = "https://kubernetes.github.io/ingress-nginx"
 chart      = "ingress-nginx"
 version    = "4.7.0"
```

```
 namespace        = "ingress"
 create_namespace = true
 values = [
   "${file("values/nginx-controller-values.yaml")}"
 ]
 depends_on = [
   helm_release.aws-load-balancer-controller,
 ]
}
```

Також, Helm конфігураційний файл, який використовується для налаштування розгортання додатка і налаштування його взаємодії з сервісами AWS Load Balancer в середовищі Kubernetes включає наступні елементи:

- kind: Deployment – вказує на те, що цей об’єкт Kubernetes є типом “Deployment”, який використовується для розгортання і керування репліками додатків:

```
controller:
 kind: Deployment
```

- ingressClass: external-nginx – задає клас входу для інгресу, який буде використовуватися для маршрутизації трафіку на цей додаток:

```
 ingressClass: external-nginx
```

- ingressClassByName: true – вказує на те, що інгрес буде використовувати класи входу за назвою:

```
 ingressClassByName: true
```

- ingressClassResource.enabled: true – увімкнення ресурсу інгресу, який буде використовувати вказаний клас входу:

```
ingressClassResource:
   enabled: true
   name: external-nginx
```

- replicaCount: 2 – кількість реплік, які будуть розгортані для цього додатка:

```
replicaCount: 2
```

- admissionWebhooks.enabled: true – увімкнення веб-хуків допуску, які дозволяють перехоплювати та змінювати запити до API Kubernetes перед тим, як вони будуть збережені в системі:

```
admissionWebhooks:
  enabled: true
```

- “service.annotations” використовуються для налаштування властивостей AWS Load Balancer, таких як таймаут з’єднання, режим перехресного балансування, сертифікат SSL, політика переговорів SSL, тип та схема балансувальника навантаження:

```
service:
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-connection-idle-timeout: "60"
    service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled: "true"
    service.beta.kubernetes.io/aws-load-balancer-ssl-cert:
"arn:aws:acm:us-east-1:********:certificate/**********"
    service.beta.kubernetes.io/aws-load-balancer-ssl-negotiation-policy:
"ELBSecurityPolicy-TLS13-1-2-2021-06"
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
```

- service.targetPorts – порти, на які буде направлено вхідний трафік в сервісі Kubernetes:

```
  service:
      targetPorts:
          https: http
```

Cert Manager додає сертифікати та видавців сертифікатів як типи ресурсів у кластери Kubernetes і спрощує процес отримання, оновлення та використання цих сертифікатів. Він може видавати сертифікати з різних підтримуваних джерел, включаючи Let's Encrypt, HashiCorp Vault і Venafi, а також приватні PKI. Також, він забезпечує дійсність і актуальність сертифікатів і спробує оновити сертифікати в налаштований час до закінчення терміну дії [19]. Наприклад, ось код Terraform із ресуром випуску Helm для розгортання Cert Manager у кластері Kubernetes:

```
resource "helm_release" "metrics_server" {
 name       = "metrics-server"
 repository = "https://kubernetes-sigs.github.io/metrics-server/"
 chart      = "metrics-server"
 version    = "3.11.0"
 namespace  = "kube-system"
 values = [
  "${file("values/metrics-server-values.yaml")}"
 ]
 depends_on = [
  module.eks,
 ]
}
```

4.4. Використання Argo CD для Continuous Delivery в Kubernetes кластері

Argo CD – це декларативний інструмент безперервної доставки GitOps для Kubernetes. Argo CD автоматизує розгортання бажаних станів додатків у вказаних цільових середовищах. Розгортання додатків може відстежувати оновлення гілок, тегів або прикріплення до певної версії маніфестів у коміті Git [9]. Наприклад, на рисунку 4.5 зображена архітектура та основні принципів роботи Argo CD.

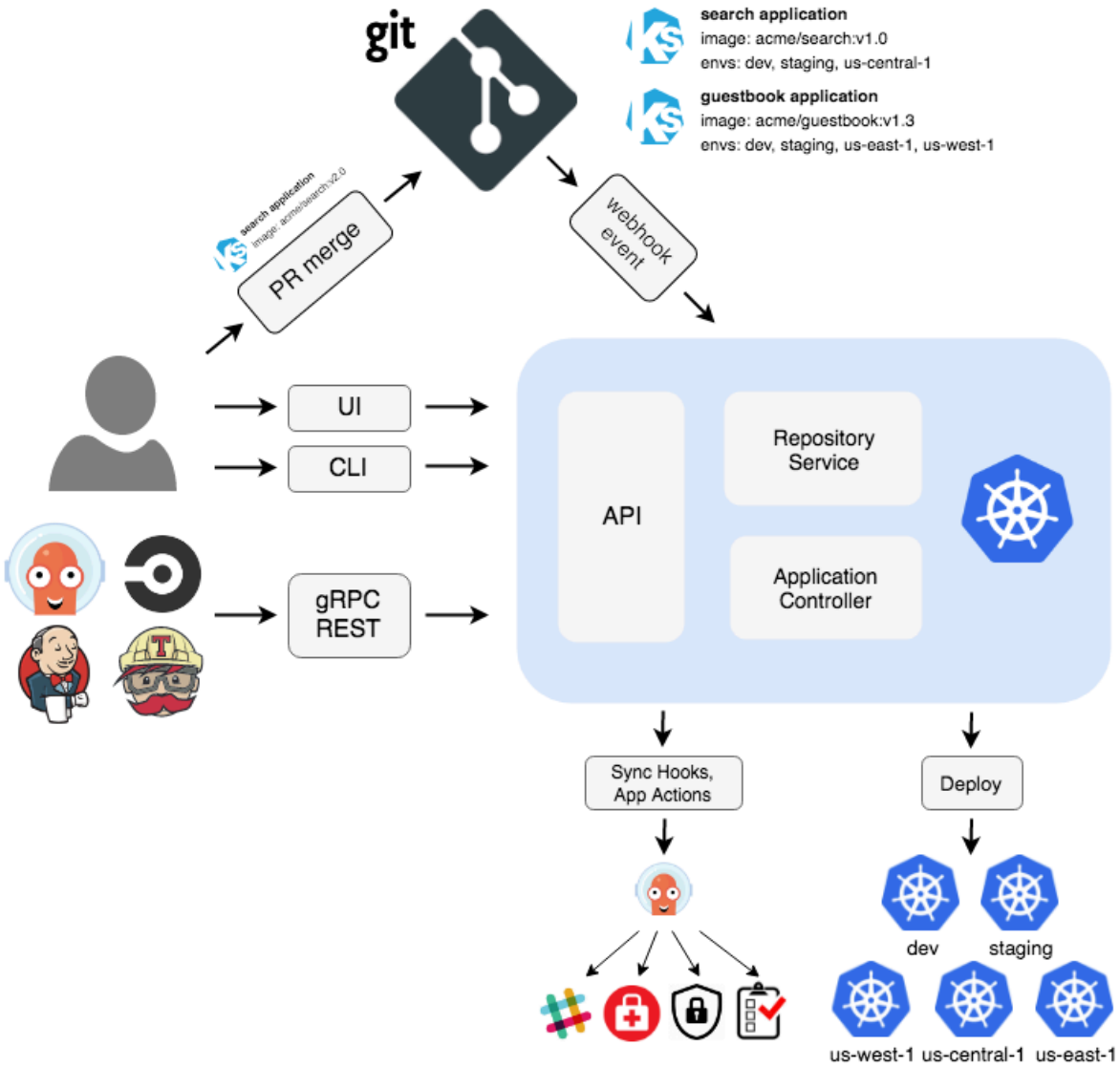


Рисунок 4.5 – Архітектура Argo CD

Встановлення та налаштування Argo CD у Kubernetes здійснюється через застосування Terraform та Helm. Зокрема, наведений нижче фрагмент коду Terraform із ресуром випуску Helm для розгортання Argo CD у кластері Kubernetes:

```
resource "helm_release" "argocd" {
 name = "argocd"
 repository       = "https://argoproj.github.io/argo-helm"
 chart            = "argo-cd"
 namespace        = local.argocd_namespace_name
 create_namespace = true
```

```
 version          = "5.44.0"
 values = [file("values/argocd-values.yaml")]
 depends_on = [
   module.eks,
   helm_release.cluster_autoscaler,
   helm_release.ingress_nginx_external
 ]
}
```

Цей код Terraform відповідає за розгортання Argo CD у кластері Kubernetes. Він визначає діаграму Helm, користувацькі значення конфігурації та залежності, щоб гарантувати, що розгортання відбувається в правильному порядку та з дотриманням необхідних передумов.

Нижче наведені значення Helm значення, які використовувалися для розгортання Argo CD у кластері Kubernetes.

- “global” визначає глобальні параметри, такі як теги зображень і правила спорідненості. Тут він встановлює тег для зображення ArgoCD і визначає правила спорідненості для керування розміщенням модулів.
- “configs” налаштовує різні параметри та політики RBAC для ArgoCD, включаючи параметри безпеки, конфігурації RBAC і області для автентифікації:

```
global:
 image:
   tag: "v2.8.2"
configs:
 params:
   server.insecure: true
 rbac:
   create: true
   policy.csv: |
     g, argocd-dev-admin@company.com, role:admin
     g, argocd-dev-view@company.com, role:readonly
   scopes: "[groups, email]"
```

- “cm” визначає параметри для ConfigMaps, пов’язані з ArgoCD, наприклад конфігурації URL-адрес і конфігурації для інтеграції Dex (провайдера OpenID Connect):

```
cm:
  create: true
  url: https://argocd.dev.company.com
  dex.config: |
    connectors:
    - type: google
      id: google
      name: Google
      config:
        clientID: $argocd-google-oauth-creds:auth0.clientID
        clientSecret: $argocd-google-oauth-creds:auth0.clientSecret
        redirectURI: https://argocd.dev.company.com/api/dex/callback
        serviceAccountFilePath: /tmp/oidc/googleAuth.json
        adminEmail: yurii@company.com
```

- “secret” керує налаштуваннями, пов’язаними з секретами Kubernetes, зокрема створенням, мітками та анотаціями. Він також визначає секрет GitHub для ArgoCD:

```
secret:
  createSecret: true
  labels: {}
  annotations: {}
  githubSecret: "****************"
```

- “dex” конфігурує параметри, специфічні для постачальника Dex OpenID Connect, зокрема підключення томів для секретів і конфігурації для підключення до Google OAuth:

```
dex:
 volumeMounts:
 - mountPath: /tmp/oidc
   name: google-json
```

```
  readOnly: true
```

- “volumes” налаштовує томи для монтування, як-от том для зберігання конфігурацій Google OAuth:

```
 volumes:
 - name: google-json
   secret:
     defaultMode: 420
     secretName: argocd-google-groups-json
```

- “controller” надає параметри для контролера ArgoCD, включаючи метрики та конфігурації ServiceMonitor для моніторингу:

```
controller:
 metrics:
   enabled: true
   serviceMonitor:
     enabled: true
```

- “server” налаштовує параметри, пов’язані з серверним компонентом ArgoCD, у тому числі конфігурації Ingress для зовнішнього показу служби ArgoCD та дає змогу Ingress із анотаціями для перенаправлення SSL і заголовків CORS:

```
server:
 ingress:
   enabled: true
   annotations:
     nginx.ingress.kubernetes.io/force-ssl-redirect: "false"
     nginx.ingress.kubernetes.io/ssl-redirect: "false"
     nginx.ingress.kubernetes.io/rewrite-target: /
     nginx.ingress.kubernetes.io/cors-expose-headers: "*, X-CustomResponseHeader"
   ingressClassName: "external-nginx"
   hosts:
     - argocd.dev.company.com
   tls:
```

```
- hosts:
  - argocd.dev.company.com
```

Підсумовуючи, ці значення Helm створюють комплексну конфігурацію для розгортання ArgoCD у кластері Kubernetes, що охоплює теги зображень, правила спорідненості, параметри безпеки, політики RBAC, ConfigMaps, секрети, інтеграцію Dex, томи, налаштування входу та метрики контролера.

Для розгортання програми в Kubernetes слід розробити yaml-файл із відповідними конфігураціями та створити об’єкт “Application” у Argo CD.

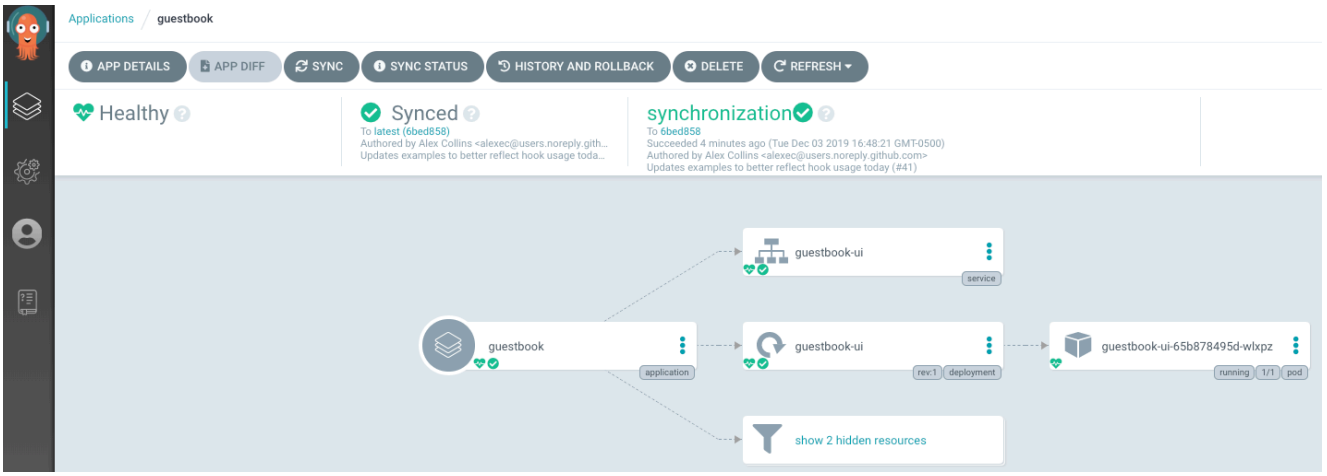


Рисунок 4.6 – Результат розгортання програми “QuestBook” у Argo CD

Зокрема, нижче наведені yaml-файли для розгортання програми “QuestBook” у кластері Kubernetes. код описує дві Kubernetes об'єкти: Service та Deployment. Також, він описує конфігурацію для розгортання фронтенду “guestbook” з трьома репліками, які слухають на порту 80 та використовують образ “gcr.io/google-samples/gb-frontend:v4”:

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
```

```
spec:
  type: NodePort
  ports:
  - port: 80
  selector:
    app: guestbook
    tier: frontend
```

Об’єкт Service включає:

- “metadata” налаштовує метадані сервісу, такі як його ім’я та мітки (labels);
- “type” визначає тип доступу до сервісу. У поточному вигляді, сервіс налаштований на тип NodePort, що означає, що кожен вузол в кластері буде відображений на певний порт на вузлі для доступу до сервісу;
- “ports” визначає список портів, на яких сервіс буде доступний. У цьому випадку, сервіс прослуховуватиме порт 80;
- “selector” визначає мітки (labels), за якими буде вибрано потрібні екземпляри додатків для маршрутизації трафіку до них. В цьому випадку, сервіс буде відправляти трафік до контейнерів з мітками “app: guestbook” та “tier: frontend”.

Об’єкт Deployment визначає:

- “metadata” налаштовує метадані розгортання, такі як його ім’я;
- “selector” визначає мітки (labels), за якими будуть вибрані потрібні репліки додатків для управління;
- “replicas” вказує кількість реплік, які мають бути запущені:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  selector:
    matchLabels:
      app: guestbook
```

```
    tier: frontend
replicas: 3
```

- “template” описує шаблон контейнера, який буде використовуватися для створення реплік;
- “metadata” налаштовує мітки (labels) контейнера;
- “spec” налаштовує специфікацію контейнера;
- “containers” описує список контейнерів, які мають бути запущені в кількості, вказаній в replicas:

```
template:
  metadata:
    labels:
      app: guestbook
      tier: frontend
  spec:
    containers:
```

- “name” – ім’я контейнера;
- “image” вказує образ, який має бути використаний для контейнера;
- “resources” налаштовує обмеження ресурсів, які контейнер може використовувати;
- “env” встановлює змінні середовища для контейнера;
- “ports” визначає список портів, які контейнер повинен відкрити для комунікації з іншими контейнерами або зовнішніми додатками:

```
    - name: php-redis
      image: gcr.io/google-samples/gb-frontend:v4
      resources:
        requests:
          cpu: 100m
          memory: 100Mi
      env:
      - name: GET_HOSTS_FROM
        value: dns
```

```
ports:
- containerPort: 80
```

Вказані вище кроки забезпечують ефективне і автоматизоване розгортання кластера Kubernetes на Amazon EKS із відповідними системними компонентами за допомогою Terraform і використанням Argo CD для розгортання програм.

Висновки до розділу 4

Цей розділ наводить детальні приклади коду для розгортання кластера Kubernetes із відповідними системними компонентами на Amazon EKS за допомогою Terraform і Helm і використанням Argo CD для розгортання додатків. Таке розгортання може забезпечити задоволення вимог бізнесу до кластера включають можливість легко масштабувати додатки та забезпечувати оптимальну продуктивність.

Автоматизація розгортання кластера за допомогою Terraform продемонструвала себе як ефективний спосіб створити і керувати інфраструктурою, забезпечуючи безпечне зберігання файлів стану, використання віддалених інфраструктурних носіїв та версіонування. Використання GitHub Actions для автоматичного конвеєру розгортання коду Terraform спрощує процес розгортання та забезпечує швидкий та надійний розгортання інфраструктури. Використання Argo CD для безперервної доставки у Kubernetes продемонструвало себе як ефективність методології GitOps для автоматизації розгортання додатків із декларативним підходом до управління станами додатків. Також, слід виділити, те що у процесі розгортання кластера важливо правильно налаштувати системні компоненти та інструменти Kubernetes, такі як утиліту для автомасштабування кластера для AWS (Cluster Autoscaler on AWS), службу метрик (Kubernetes Metrics Server), контролер для балансувальника (AWS Load Balancer Controller) та менеджер сертифікатів (Cert Manager).

Отже, вказані у цьому розділі методологічні та практичні рекомендації та висновки можуть допомогти компаніям забезпечити безперебійну та ефективну роботу Kubernetes кластера на базі AWS EKS, допомагаючи виконувати бізнес-вимоги та забезпечувати масштабованість та безпеку інфраструктури.

ВИСНОВКИ

У роботі проведено теоретичні узагальнення та розв’язано науково-практичне завдання щодо дослідження та аналізу актуальності застосування Kubernetes в умовах сучасного IT ринку.

Основні висновки та результати, отримані у процесі проведеного дослідження, полягають у наступному:

1. У процесі вивчення найбільш поширених прикладів розгортання кластерів Kubernetes в хмарних середовищах було підтверджено тенденцію до збільшення використання Kubernetes в хмарних середовищах. Зокрема, керовані сервіси Kubernetes, надані провідними постачальниками хмарних послуг (Amazon Elastic Kubernetes Service, Google Kubernetes Engine, Azure Kubernetes Service, Red Hat OpenShift, DigitalOcean Kubernetes, тощо) демонструються стійку динаміку росту. Дослідження підтверджує те, що Kubernetes у хмарних середовищах стає все більш зручним та інтегрованим, що робить його надійним вибором для розвитку та оркестрації контейнерних додатків у сучасних умовах.

2. На основі аналізу прикладів реалізації Kubernetes у компаніях Spotify, OpenAI, Pinterest, BlaBlaCar, VSCO та Huawei виділено такі ключові висновки: 1) Kubernetes стає основним рішенням для компаній із складними IT-потребами, дозволяючи автоматизувати керування інфраструктурою та забезпечувати гнучкість в розгортанні; 2) Kubernetes використовується в багатьох галузях, включаючи розваги, дослідження та технології, а також уряди та організації; 3) багато компаній використовують поетапний підхід до впровадження Kubernetes, мігруючи додатки у контейнери та поступово переходячи до оркестрації з використанням Kubernetes; 4) Kubernetes допомагає компаніям масштабувати інфраструктуру та забезпечувати стійкість, знижувати час відновлення після збоїв; 5) впровадження Kubernetes сприяє підвищенню продуктивності, швидшому розгортанню проектів та ефективному керуванню інфраструктурою; 6) Kubernetes дозволяє оптимізувати інфраструктуру,

знижуючи витрати та покращуючи доступність; 7) важливим уроком є постійне навчання та обмін знаннями всередині організації для успішного впровадження Kubernetes; 8) Kubernetes показує свою цінність через використання автоматизації для підвищення продуктивності та роль “open source” спільнот для стимулювання постійного удосконалення та інновацій.

3. У процесі написання цієї роботи було розроблено та імплементовано Kubernetes кластер на базі Amazon Elastic Kubernetes Service, із врахування бізнес вимог, автоматизації розгортання, та налаштування системних компонентів. За результатами його впровадження можна виділити наступні методологічні та практичні рекомендації та висновки: 1) поєднання Terraform та Helm для розгортання кластера Kubernetes на Amazon EKS продемонструвало себе як ефективний спосіб створення та керування інфраструктурою; 2) використання GitHub Actions для автоматизації конвеєру розгортання коду Terraform спростило та прискорило процес розгортання інфраструктури; 3) використання модулів Terraform для AWS дозволило суттєво зменшити час на створення і управління AWS ресурсами; 5) Cluster Autoscaler on AWS, Kubernetes Metrics Server, AWS Load Balancer Controller та Cert Manager було розгорнуто через Terraform у поєднанні із Helm, що суттєво спростило їхню інсталяцію; 6) Argo CD було використано як інструмент для безперервної доставки, автоматизуючи розгортання додатків та гарантуючи декларативний підхід до управління станами у Kubernetes на основі GitOps методології.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 2021 Cloud Native Survey. URL: https://www.cncf.io/wp-content/uploads/2022/02/CNCF-AR_FINAL-edits-15.2.21.pdf (Last accessed: 12.10.2023).
2. Abhinav Dubey. How to Execute Canary Deployments Using NGINX Ingress. URL: https://devtron.ai/blog/how-to-execute-canary-deployments-using-nginx-ingress/ (Last accessed: 02.10.2023).
3. Abhishek. Unleashing the Power of Kubernetes: A Comprehensive Case Study of Spotify's Transformation Journey. URL: https://www.linkedin.com/pulse/unleashing-power-kubernetes-comprehensive-case-study-spotifys-/ (Last accessed: 21.10.2023).
4. A Guide to Using Kubernetes for Microservices. URL: https://loft.sh/blog/a-guide-to-using-kubernetes-for-microservices/ (Last accessed: 18.10.2023).
5. Amazon ECR Image scanning. URL: https://docs.aws.amazon.com/AmazonECR/latest/userguide/image-scanning.html (Last accessed: 10.10.2023).
6. Amazon Elastic Kubernetes Service. URL: https://aws.amazon.com/eks/ (Last accessed: 06.10.2023).
7. Apache Flink Kubernetes Setup. URL: https://nightlies.apache.org/flink/flink-docs-master/docs/deployment/resource-providers/standalone/kubernetes/ (Last accessed: 20.10.2023).
8. Apache Hadoop Official Website. URL: https://hadoop.apache.org/ (Last accessed: 20.10.2023).
9. ArgoCD Official Website. URL: https://argo-cd.readthedocs.io/en/stable/ (Last accessed: 24.10.2023).

10. AWS App Mesh. URL: https://aws.amazon.com/app-mesh/ (Last accessed: 11.10.2023).

11. AWS EKS Terraform module. URL: https://registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest (Last accessed: 25.10.2023).

12. AWS Fargate for EKS. URL: https://docs.aws.amazon.com/eks/latest/userguide/fargate.html (Last accessed: 08.10.2023).

13. AWS Load Balancer Controller. URL: https://github.com/kubernetes-sigs/aws-load-balancer-controller (Last accessed: 25.10.2023).

14. AWS VPC Terraform module. URL: https://registry.terraform.io/modules/terraform-aws-modules/vpc/aws/latest (Last accessed: 25.10.2023).

15. Azure Kubernetes Service (AKS). URL: https://learn.microsoft.com/en-us/azure/aks/ (Last accessed: 06.10.2023).

16. Bob Ganley. Container Adoption Trends: Why, How and Where. URL: https://www.dell.com/en-us/blog/container-adoption-trends-why-how-and-where/ (Last accessed: 23.09.2023).

17. Borg: The Predecessor to Kubernetes. URL: https://kubernetes.io/blog/2015/04/borg-predecessor-to-kubernetes/ (Last accessed: 28.09.2023).

18. Brindha Jeyaraman. Managing ML Model Deployments with Kafka and Kubernetes. URL: https://www.linkedin.com/pulse/managing-ml-model-deployments-kafka-kubernetes-brindha-jeyaraman/ (Last accessed: 18.10.2023).

19. Cert Manager. URL: https://cert-manager.io/ (Last accessed: 26.10.2023).

20. Charles Betz, Cortney Eldridge, Kathryn Bell. The State Of DevOps 2022. URL: https://www.forrester.com/report/the-state-of-devops-2022/RES177685 (Last accessed: 14.10.2023).

21. Claus Pahl, Antonio Brogi, Jacopo Soldani, Pooyan Jamshidi. Cloud Container Technologies: a State-of-the-Art Review. URL: https://www.researchgate.net/publication/316903410_Cloud_Container_Technologies_A_State-of-the-Art_Review (Last accessed: 26.09.2023).

22. CloudNativePG Official Website. URL: https://cloudnative-pg.io/ (Last accessed: 18.10.2023).

23. Cluster Autoscaler on AWS. URL: https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/cloudprovider/aws/README.md (Last accessed: 25.10.2023).

24. CNCF Projects 1/1/2022 - 1/1/2023. URL: https://docs.google.com/spreadsheets/d/1NifFdE45BlscO6j7vlDspccQ-i9Ksgip6yx2bmhmCCs/edit#gid=976519966 (Last accessed: 12.10.2023).

25. CNCF survey 2019. URL: https://www.cncf.io/wp-content/uploads/2020/03/CNCF Survey Report.pdf (Last accessed: 04.10.2023).

26. DigitalOcean Kubernetes. URL: https://www.digitalocean.com/products/kubernetes (Last accessed: 09.10.2023).

27. Elastic Stack Official Website. URL: https://www.elastic.co/elastic-stack (Last accessed: 18.10.2023).

28. GKE Autopilot overview. URL: https://cloud.google.com/kubernetes-engine/docs/concepts/autopilot-overview (Last accessed: 09.10.2023).

29. Google Kubernetes Engine (GKE). URL: https://cloud.google.com/kubernetes-engine?hl=en (Last accessed: 06.10.2023).

30. Grafana Official Website. URL: https://grafana.com/ (Last accessed: 18.10.2023).

31. Helm Official Website. URL: https://helm.sh/ (Last accessed: 26.10.2023).

32. Horizontal Pod Autoscaling. URL: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ (Last accessed: 02.10.2023).

33. How AWS Load Balancer controller works. URL: https://kubernetes-sigs.github.io/aws-load-balancer-controller/v2.6/how-it-works/ (Last accessed: 26.10.2023).

34. Identity and access management for Amazon EKS. URL: https://docs.aws.amazon.com/eks/latest/userguide/security-iam.html (Last accessed: 10.10.2023).

35. Infrastructure security in Amazon EKS. URL: https://docs.aws.amazon.com/eks/latest/userguide/infrastructure-security.html (Last accessed: 10.10.2023).

36. Johannes Berggren, Jens Karlsson. Differences in performance between containerization & virtualization with a focus on HTTP requests. URL: https://www.diva-portal.org/smash/get/diva2:1665577/FULLTEXT01.pdf (Last accessed: 26.09.2023)

37. Justin Ellingwood. Architecting Applications for Kubernetes. URL: https://www.digitalocean.com/community/tutorials/architecting-applications-for-kubernetes (Last accessed: 17.10.2023).

38. Jyoti Sahoo. What is a CI/CD Pipeline for Kubernetes? URL: https://devtron.ai/blog/what-is-a-ci-cd-pipeline-for-kubernetes/ (Last accessed: 02.10.2023).

39. Kubernetes: Up and Running by Brendan Burns, Joe Beda, Kelsey Hightower, and Lachlan Evenson Copyright. Published by O'Reilly Media, 2022. URL: https://books.google.com.ua/books?hl=uk&lr=&id=KeB-EAAAQBAJ&oi=fnd&pg=PT17&dq=kubernetes+usage+research&ots=V9VUBNlkV8&sig=Zkqxop1kvAyu-i_

wUGMR2_iQ490&redir_esc=y#v=onepage&q=kubernetes%20usage%20research&f=false (Last accessed: 30.09.2023).

40. Kubernetes in the Enterprise. URL: https://3855032.fs1.hubspotusercontent-na1.net/hubfs/3855032/DZone-TR-kubernetes-enterprise-2022-spotlight-Kasten.pdf?__hstc=78384331.3e433beb2ff2e36d3761698573424ed5.1697454544847.1697454544847.1697454544847.1&__hssc=78384331.2.1697454544847&__hsfp=4225138836&hsCtaTracking=42f75c86-fb79-4ac7-af05-ddc2c08f3d42%7C5d33645e-95f7-4028-a6d3-3a8f73b5eaed (Last accessed: 12.10.2023).

41. Kubernetes in the wild 2023 Report. URL: https://assets.dynatrace.com/en/docs/wp/bae3218-wp-kubernetes-in-the-wild-en.pdf?_ga=2.51528748.42413689.1697455033-554709570.1697455033&_gl=1*166qadx*_ga*NTU0NzA5NTcwLjE2OTc0NTUwMzM.*_ga_1MEMV02JXV*MTY5NzQ1NTAzMi4xLjEuMTY5NzQ1NTg2NS4wLjAuMA (Last accessed: 12.10.2023).

42. Kubernetes Metrics Server. URL: https://github.com/kubernetes-sigs/metrics-server (Last accessed: 25.10.2023).

43. Kubernetes User Case Studies. URL: https://kubernetes.io/case-studies/ (Last accessed: 22.10.2023).

44. Lewis, J., Fowler, M. Microservices. URL: http://martinfowler.com/articles/microservices.html (Last accessed: 26.09.2023).

45. Lionel Sujay Vailshery. Kubernetes leading challenges for businesses worldwide 2022. URL: https://www.statista.com/statistics/1248435/worldwide-kubernetes-challenges-organizations/ (Last accessed: 14.10.2023).

46. Lionel Sujay Vailshery. Most popular environments for running Kubernetes clusters worldwide 2022. URL: https://www.statista.com/statistics/1248448/worldwide-run-kubernetes-environments/ (Last accessed: 06.10.2023).

47. Miles S. Kubernetes: A Step-By-Step Guide For Beginners To Build, Manage, Develop, and Intelligently Deploy Applications By Using Kubernetes (2020

Edition). Independently Published, 2020. URL: https://books.google.com/books?id=M4VmzQEACAAJ (Last accessed: 06.10.2023).

48. NGINX Ingress Controller. URL: https://docs.nginx.com/nginx-ingress-controller/ (Last accessed: 16.10.2023).

49. Oracle Container Engine for Kubernetes. URL: https://www.oracle.com/cloud/cloud-native/container-engine-kubernetes/ (Last accessed: 06.10.2023).

50. Overview of Kubernetes. URL: https://kubernetes.io/docs/concepts/overview/ (Last accessed: 20.10.2023).

51. Peter Wong. Dynamic Kubernetes based Feature Environment on AWS URL: https://medium.com/@peteryhwong/dynamic-kubernetes-based-feature-environment-on-aws-b911eec92e7 (Last accessed: 20.10.2023).

52. Priya Kumari. Why Spotify Migrated From Its Homegrown Orchestration Tool (Helios) to Kubernetes? URL: https://hackernoon.com/why-spotify-migrated-from-its-homegrown-orchestration-tool-helios-to-kubernetes (Last accessed: 20.10.2023).

53. Prometheus Official Website. URL: https://prometheus.io/ (Last accessed: 20.10.2023).

54. Red Hat OpenShift. URL: https://www.redhat.com/en/technologies/cloud-computing/openshift (Last accessed: 20.10.2023).

55. Running Spark on Kubernetes. URL: https://spark.apache.org/docs/latest/running-on-kubernetes.html (Last accessed: 20.10.2023).

56. Samantha Kaylee. Kubernetes: The Current and Future State of K8s in the Enterprise. URL: https://cloudacademy.com/blog/kubernetes-the-current-and-future-state-of-k8s-in-the-enterprise/ (Last accessed: 14.10.2023).

57. Scaling Kubernetes to 7,500 nodes. URL: https://openai.com/research/scaling-kubernetes-to-7500-nodes (Last accessed: 22.10.2023).

58. Sebastien Doido. Be Lean, Go Far: leveraging Kubernetes for an elastic right-sized platform. URL: https://medium.com/blablacar/be-lean-go-far-leveraging-kubernetes-for-an-elastic-right-sized-platform-bc1179c4c784 (Last accessed: 22.10.2023).

59. State of Cloud Native Development. URL: https://www.cncf.io/wp-content/uploads/2020/08/CNCF-The-State-of-Cloud-Native-Development_Q419.pdf (Last accessed: 26.09.2023).

60. State of Kubernetes Security Report 2023. URL: https://www.redhat.com/rhdc/managed-files/cl-state-kubernetes-security-report-262667-202304-en.pdf (Last accessed: 14.10.2023).

61. Sysdig 2022 Cloud-Native Security and Usage Report. URL: https://sysdig.com/content/c/pf-2022-cloud-native-security-and-usage-report?x=u_WFRi&mkt_tok=MDY3LVFaVC04ODEAAAGO2-_GG4zdwJLUkPeWbaZwRRSnIGqE7-PYEwyq0_HMS39j7hySJv9bU-WSWX8B-vuXenap-p7TT3krYUpGPzzjflpX8zS9UcLNKeSfWdyVreLt&_pfses=kwLHBKnWS4WG3nXpnY6FcwnG (Last accessed: 14.10.2023).

62. Terraform Official Website. URL: https://www.terraform.io/ (Last accessed: 24.10.2023).

63. Tesla cloud resources are hacked to run cryptocurrency-mining malware. URL: https://arstechnica.com/information-technology/2018/02/ (Last accessed: 05.10.2023).

64. The State of Application Development in 2022 and Beyond. URL: https://www.docker.com/wp-content/uploads/2022/04/Docker-Report-AppDev-Trends-2022.pdf (Last accessed: 20.10.2023).

65. The State of Kubernetes 2023. URL: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/docs/vmware-ebook-state-of-kubernetes.pdf (Last accessed: 20.10.2023).

66. The Twelve-Factor App. URL: https://12factor.net/ (Last accessed: 16.10.2023).

67. VMware Tanzu. URL: https://tanzu.vmware.com/tanzu (Last accessed: 08.10.2023).

68. What is container orchestration? URL: https://www.redhat.com/en/topics/containers/what-is-container-orchestration (Last accessed: 28.09.2023).

69. What is Kubernetes Architecture? URL: https://www.vmware.com/topics/glossary/content/kubernetes-architecture.html (Last accessed: 05.10.2023).

70. With Kubernetes, the U.S. Department of Defense Is Enabling DevSecOps on F-16s and Battleships. URL: https://www.cncf.io/case-study/dod/ (Last accessed: 30.09.2023).

71. Анастасія Кравчук. Навіщо нам контейнеризація: переваги для DevOps. URL: https://blog.iteducenter.ua/articles/navishho-nam-kontejnerizaciya-perevagi-dlya-devops/ (дата звернення: 23.09.2023).

72. Михайло Майдан. Використання Kubernetes для оркестрації пристроїв в edge computing. URL: https://dou.ua/forums/topic/45709/ (дата звернення: 24.10.2023).

метадані

Заголовок

Дослідження та аналіз актуальності застосування Kubernetes в умовах сучасного IT ринку

Автор

Вацик Ю.Ю. Науковий керівник / Експерт

підрозділ

King Danylo University

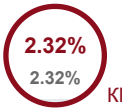
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

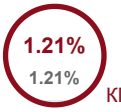
Заміна букв	ß	5
Інтервали	A→	0
Мікропробіли	·	2
Білі знаки	ß	0
Парафрази (SmartMarks)	a	12

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

10539
Кількість слів

82580
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/388/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0%20%D0%9B%D0%B8%D1%82%D0%B2%D0%B0%D0%BA.pdf?sequence=1	56	0.53 %
2	2023_Фel_44_Ракітін_ВЕ_робота.pdf [redacted] 6/9/2023 The Ivan Franko National University (Факультет електроніки та комп'ютерних технологій)	39	0.37 %
3	РОЗРОБКА КЛАСТЕРУ МЕРЕЖЕВИХ ЗАСТОСУНКІВ НА ОСНОВІ KUBERNETES [redacted] 5/31/2021 State University of Telecommunications (НННІТ)	33	0.31 %