МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.056.5:510.22(043.3)

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема:Система для маскування персональних даних у журналі подій сервісу

Виконавець: Олександр ЗОСИК
Керівник: к.т.н., доцент Микола БРАІЛОВСЬКИЙ
Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА
Нормоконтролер: к.т.н., доцент Микола БРАІЛОВСЬКИЙ

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«__» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

здобувача вищої освіти Зосика Олександра Олександровича

1. Тема: *Система для маскування персональних даних у журналі подій сервісу*

затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.

2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.

3. Вихідні дані: реалізувати і описати реалізацію системи для маскування персональних даних у журналі подій сервісу; проаналізувати та дослідити загальні поняття маскування персональних даних; проаналізувати та дослідити актуальні методи щодо маскування персональних даних.

4. Зміст пояснювальної записки: аналіз та дослідження загальних понять щодо маскування персональних даних в хмарних середовищах; проблеми з зберіганням та управлінням токенами безпеки; реалізація і детальний опис системи для маскування персональних даних у журналі подій сервісу.

5. КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	23.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	26.10.2023	*Виконано*
4.	Збір інформації	30.10.2023	*Виконано*
5.	Дослідження предметної області	13.11.2023	*Виконано*
6.	Архітектура системи для маскування персональних даних	20.11.2023	*Виконано*
7.	Реалізація системи для маскування персональних даних	08.12.2023	*Виконано*
8.	Охорона навколишнього середовища	10.12.2023	*Виконано*
9.	Перевірка на антиплагіат	14.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	20.12.2023	*Виконано*
11.	Оформлення презентації	21.12.2023	*Виконано*
12.	Отримання рецензій від рецензента	22.12.2023	*Виконано*

6. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Олександр ЗОСИК
(підпис, дата)

Керівник кваліфікаційної роботи (підпис, дата) Микола БРАІЛОВСЬКИЙ

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Система для маскування персональних даних у журналі подій сервісу»: 65 с., 42 рис., 18 інформаційних джерел.

КУБЕРНЕТЕС, АМАЗОН ВЕБ СЕРВІСИ, ФЛЮЄНТБІТ, ПІТОН, БЕЗПЕКА, ТОКЕН БЕЗПЕКИ, , ПЕРСОНАЛЬНІ ДАНІ

Об’єкт розробки – Система для аналізу журналів сервісу та їх маскування.

Мета роботи – створити систему, яка надає можливість автоматично аналізувати журнали сервісу на наявність персональних даних і маскувати їх в подальшому.

ABSTRACT

Explanatory note to the thesis " System for masking personal data in service logs ": 65 p. , 42 Fig. , 18 information sources.

KUBERNETES, AMAZON WEB SERVICES, FLUENTBIT, PYTHON, SECURITY, SECURITY TOKEN, , PERSONAL IDENTIFIABLE INFORMATION

Property development – system for masking personal data.

Purpose -create a system, which gives the ability to automatically analyse service logs, find pii data in it and mask it afterwards.

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 9
1.1 Поняття маскування даних 9
1.2 Аналіз інструментів та сервісів для маскування персональних даних 9
Висновок 18
РОЗДІЛ 2 АРХІТЕКТУРА СИСТЕМИ ДЛЯ МАСКУВАННЯ ПЕРСОНАЛЬНИХ ДАНИХ 19
2.1. Вибір хмарного середовища 19
2.2. Опис архітектури на базі AWS 27
Висновок 37
РОЗДІЛ 3 РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ МАСКУВАННЯ ПЕРСОНАЛЬНИХ ДАНИХ 38
3.1. Створення Авс аккаунту 38
3.2. Purchase Service розгортання 46
3.3. Elasticsearch розгортання 49
3.4. Kibana розгортання 51
3.5. Fluentbit розгортання 54
3.6. Тестування системи 60
Висновок 62
РОЗДІЛ 4 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА 64
Висновок 66
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 67
ДОДАТОК А 69

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

API – application programming interface;

PII – personal identifiable information;

AWS – amazon web services;

K8S – kubernetes;

ELK – elastic logstach kibana;

EC2 – elastic compute container;

EKS – elastic Kubernetes service;

HTTP – Hypertext Transfer Protocol;

IDE - integrated development environment;

JSON – javascript object notation;

SA – service account;

OS – operational system;

ПЗ – програмне забезпечення.

ВСТУП

У сучасному світі, захист персональної даних став першочерговим. Персональні дані охоплюють будь-яку інформацію, яка може бути використана для ідентифікації особи, наприклад імена, адреси, номери соціального страхування тощо. Із поширенням мікросервісної архітектури на базі Kubernetes, організації стикаються з проблемою ефективного захисту персональних даних, особливо коли йдеться про журнали подій(логи). Kubernetes(K8S) це, простими словами, система, яка дозволяє ефективно керувати сервісами, розміщеними на сервері та оптимізувати утилізацію ресурсів на обладнанні. Але K8S не має можливості обробляти логи, він тільки пише їх у відповідні файли у сирому вигляді, тому і потрібна надійна система для ідентифікації і маскування персональних даних.

Актуальність дослідження. На сьогоднішній день існує безліч автоматизованих способів захисту персональних даних у логах за допомогою сторонніх ресурсів. Але більшість компаній зараз використовують хмарні середовища для їх застосунків, і вони не бажають передавати персональну інформацію стороннім сервісам по причинам безпеки. Саме тому потрібно дослідити спосіб захисту конфіденційної інформації, використовуючи тільки хмарні рішення.

Аналіз останніх досліджень і публікацій. Підхід до захисту конфіденційних даних у ненадійних середовищах аналізували Анас Абу Ель Калам та Еммануель Кордоньє[18].

Вагомий внесок у висвітлення проблем маскування даних у хмарних середовищах і їх вирішення зробили Пол Джанг[1] та Шям Нандан Кумар[17].

Мета дослідження – розробити систему для маскування персональних даних у журналі подій сервісу.

Об’єкт дослідження – система маскування.

Предмет дослідження – сукупність необхідних інструментів, що забезпечують найбезпечніший підхід до маскування у хмарному середовищі.

Завдання дослідження можна сформулювати так:

- провести аналіз існуючих інструментів для маскування даних;
- дослідити особливості використання хмарного сервісу «GCP Sensitive Data Protection»

Методи дослідження в роботі використані такі: пошуковий по наявній інформації з інтернету із аналізом знайденого матеріалу, порівняння, з'ясування причинно-наслідкових зв'язків, систематизація, аналіз документації та результатів діяльності дослідників з проблеми проведеного дослідження.

Новизна дослідження. Розрозблено систему маскування, якої немає в наявності у відкритому доступі.

Джерельна база дослідження. Робота ґрунтується на аналізі технічної літератури, наукових статей, періодичних видань та напрацювань сучасних та попередніх вчених і дослідників в галузі кібербезпеки.

Теоретична та практична цінність. Розроблена система маскування виконує свою основну роль – маскує персональні дані в сервіс логах. Іншою перевагою є те, що в цій системі використовуються хмарні інструменти виключно з хмарного провайдера, де розташовані наші сервери сервісів, тобто немає потенційного витоку персональної інформації за рамки нашого середовища.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття маскування даних

Маскування даних - це важливий аспект у сфері безпеки та конфіденційності інформації. Цей процес передбачає застосування різних методів для збереження конфіденційності та безпеки даних у контексті їх обробки, передачі та зберігання. Основні аспекти маскування даних включають наступне:

- Заміна конфіденційних даних: У процесі маскування конфіденційна або особиста інформація може бути замінена псевдонімами або іншими безпечними значеннями. Це допомагає уникнути витоку справжніх даних.
- Видалення чутливих елементів: Деякі елементи інформації можуть бути видалені повністю, залишаючи тільки необхідну частину даних для обробки.
- Обфускація та шифрування: Застосування методів обфускації або шифрування може ускладнити доступ до даних без необхідних дозволів чи ключів.
- Маскування у логах: Важливим аспектом є маскування особистих даних у логах системи. Це запобігає витоку конфіденційної інформації під час ведення журналів подій.
- Забезпечення відповідності: Маскування даних може бути необхідним для відповідності законодавству щодо захисту особистих даних, такому як GDPR.

Маскування даних є невід'ємною частиною стратегії забезпечення безпеки та конфіденційності інформації у сучасних системах обробки даних та допомагає зменшити ризик витоку особистої інформації.

1.2 Аналіз інструментів та сервісів для маскування персональних даних

1.2.1 Скрипт для знаходження і заміни персональних даних

Найпростіший варіант для маскування персональних даних, це написання

власного скрипта, який буде виконувати функції пошуку патерна персональної інформації і її подальшої заміни на наприклад зірочки. Розглянемо наступний приклад скріпта, написаного на мові програмування Lua:

```
function mask_sensitive_info(tag, timestamp, record)
message = record["message"]
if message then
-- Match "aadhaarNumber:xxxx," and replace with "aadhaarNumber:****,"
local masked_message = string.gsub(message, 'aadhaarNumber=[^,]*', 'aadhaarNumber=****')
-- Match "mobileNumber:xxxx," and replace with "mobileNumber:****,"
masked_message = string.gsub(masked_message, 'mobileNumber=[^,]*', 'mobileNumber=****')

record["message"] = masked_message
end
return 2, timestamp, record
end
```

Цей скрипт по заданому патерну `'mobileNumber=[^,]*'` знаходить мобільні номери і робить заміну значення на ****.

Плюси:

- Потребує мало часу на реалізацію.
- Можна реалізувати на любій мові програмування.
- Легко тестувати.

Мінуси:

- Не гнучкий. При найменшій зміні патерна, скріпт перестає працювати.
- Статичний. При додаванні нового типу персональних даних у систему, скрипт також потрібно буде розширяти.

1.2.2 Nightfall

Nightfall - це інструмент для маскування та виявлення персональних даних (PII) в текстових документах і комунікаціях. Він призначений для забезпечення конфіденційності та безпеки даних у текстовій інформації, що зберігається в організаційних джерелах, таких як тексти повідомлень, документація, логи тощо.

Основні можливості Nightfall включають:

- Виявлення PII: Nightfall виявляє персональні дані в тексті, такі як номери соціального страхування, банківські реквізити, номери кредитних карт тощо. Він використовує машинне навчання для точного розпізнавання цих даних.
- Маскування PII: Nightfall може замаскувати (анонімізувати) PII в тексті, замінюючи його на фіктивні дані або токени, що не мають вартості.
- Аналіз комунікацій: Він може контролювати комунікації, виявляючи PII в текстових повідомленнях, електронних листах та інших формах спілкування.
- Інтеграція з іншими інструментами: Nightfall може інтегруватися з іншими інструментами безпеки та системами керування даними для забезпечення повної охорони PII в організації.
- Спостереження за даними в режимі реального часу: Він надає можливість спостерігати за даними у режимі реального часу і вживати заходів для їх маскування або блокування, коли це необхідно.
- Правила та налаштування: Nightfall дозволяє налаштовувати правила маскування та виявлення PII відповідно до потреб конкретної організації.

Цей інструмент може бути корисним для організацій, які стежать за вимогами до захисту персональних даних та забезпечення їхньої конфіденційності. Nightfall допомагає уникнути витоків інформації та порушень безпеки даних, забезпечуючи ефективний контроль над PII в текстових документах та повідомленнях.

Приклад використання NightFall на мові Python:

```
>>> from nightfall import Confidence, DetectionRule, Detector, Nightfall
```

```
>>> # By default, the client reads the API key from the environment variable NIGHTFALL_API_KEY
>>> nightfall = Nightfall()
>>> # A rule contains a set of detectors to scan with
>>> cc = Detector(min_confidence=Confidence.LIKELY, nightfall_detector="CREDIT_CARD_NUMBER")
>>> ssn = Detector(min_confidence=Confidence.POSSIBLE, nightfall_detector="US_SOCIAL_SECURITY_NUMBER")
>>> detection_rule = DetectionRule([cc, ssn])

>>> findings, _ = nightfall.scan_text( ["hello world", "my SSN is 678-99-8212", "4242-4242-4242-4242"], detection_rules=[detection_rule])
>>> print(findings)
[[], [Finding(finding='678-99-8212', redacted_finding=...)]]
```

Плюси:

- Готове рішення для маскування даних.
- Використовує машинне навчання для точного розпізнавання.

Мінуси:

- Платне рішення.
- Опрацювання персональних даних проходить на стороні NightFall, що в деяких компаніх може бути заборонено з точки зору безпеки.

1.2.3 GCP Sensitive Data Protection сервіс

Sensitive Data Protection може деідентифікувати конфіденційні дані в текстовому вмісті, включаючи текст, що зберігається в структурах контейнерів, таких як таблиці. Деідентифікація — це процес видалення ідентифікаційної інформації з даних. API виявляє конфіденційні дані, наприклад особисту інформацію (PII), а потім використовує деідентифікаційну трансформацію, щоб маскувати, видаляти або іншим чином приховувати дані. Наприклад, методи деідентифікації можуть включати будь-яке з наступного:

Маскування конфіденційних даних шляхом часткової або повної заміни символів на символ, наприклад зірочку (*) або решітку (#).

Заміна кожного екземпляра конфіденційних даних маркером або сурогатним рядком.

Шифрування та заміна конфіденційних даних за допомогою випадково згенерованого або попередньо визначеного ключа.

Ви можете передавати інформацію в API за допомогою JSON через HTTPS, а також за допомогою CLI та кількох мов програмування за допомогою клієнтських бібліотек захисту конфіденційних даних. Розглянемо приклад налаштування на мові Python:

```
from typing import List  # noqa: F811, E402, I100
import google.cloud.dlp  # noqa: F811, E402


def deidentify_with_replace(
    project: str,
    input_str: str,
    info_types: List[str],
    replacement_str: str = "REPLACEMENT_STR",
) -> None:
    """Uses the Data Loss Prevention API to deidentify sensitive data in a
    string by replacing matched input values with a value you specify.
    Args:
        project: The Google Cloud project id to use as a parent resource.
        input_str: The string to deidentify (will be treated as text).
        info_types: A list of strings representing info types to look for.
        replacement_str: The string to replace all values that match given
            info types.
    Returns:
        None; the response from the API is printed to the terminal.
    """
```

```
        # Instantiate a client
        dlp = google.cloud.dlp_v2.DlpServiceClient()

        # Convert the project id into a full resource id.
        parent = f"projects/{project}"

        # Construct inspect configuration dictionary
            inspect_config = {"info_types": [{"name": info_type} for info_type in
info_types]}

        # Construct deidentify configuration dictionary
        deidentify_config = {
            "info_type_transformations": {
                "transformations": [
                    {
                        "primitive_transformation": {
                            "replace_config": {
                                "new_value": {"string_value": replacement_str}
                            }
                        }
                    }
                ]
            }
        }

        # Construct item
        item = {"value": input_str}

        # Call the API
```

```
response = dlp.deidentify_content(
    request={
        "parent": parent,
        "deidentify_config": deidentify_config,
        "inspect_config": inspect_config,
        "item": item,
    }
)

# Print out the results.
print(response.item.value)
```

Плюси:

- Готове рішення для маскування даних.
- Використовує машинне навчання для точного розпізнавання.
- Безкоштовно опрацьовує до 1 Gb даних на місяць.
- Персональні дані не виходять за рамки хмарного середовища GCP.

Мінуси:

- Платне рішення після 1 Gb даних на місяць.
- Залежність на хмарне середовище GCP.

1.2.4. AWS Comprehend

Amazon Comprehend — це служба обробки природної мови (NLP), яка використовує машинне навчання для пошуку ідей і зв’язків у тексті. Він є частиною набору Amazon Web Services (AWS) і призначений для роботи з великими обсягами неструктурованих даних, таких як взаємодії з клієнтами, соціальні мережі та документи.

Основні характеристики Amazon Comprehend:

- Аналіз тексту:

Amazon Comprehend може аналізувати текст на настрої (позитивні, негативні, нейтральні, змішані), ключові фрази, сутності (як-от люди, місця, дати), мову та синтаксис.

- Розпізнавання сутності:

  Він може ідентифікувати названі сутності (наприклад, фізичних осіб, компанії, місцезнаходження) і пов’язувати їх із пов’язаною інформацією.
- Виявлення мови:

  Сервіс може визначити мову тексту, підтримуючи кілька мов.
- Витяг ключової фрази:

  Він виділяє ключові терміни та фрази з тексту, що полегшує розуміння основних моментів.
- Аналіз настрою:

  Comprehend визначає, чи є текст позитивним, негативним, нейтральним чи змішаним.
- Спеціальні моделі:

  Ви можете навчити власні моделі, адаптовані до конкретних вимог або термінології вашого бізнесу.
- Інтеграція з іншими сервісами AWS:

  Легко інтегрується з іншими службами AWS, такими як S3, DynamoDB і AWS Lambda, для комплексної обробки даних.

  Плюси Amazon Comprehend:
- Масштабованість:

  Як керований сервіс, він автоматично масштабується для обробки великих обсягів даних.
- Простота використання:

  Пропонує зручний інтерфейс і API, що робить його доступним для розробників без глибокого досвіду машинного навчання.
- Інтеграція:

  Повна інтеграція з іншими службами AWS спрощує розгортання комплексних рішень для аналізу даних.

- Налаштування:

  Можливість створювати власну класифікацію та моделі розпізнавання сутностей відповідно до конкретних потреб.

- Аналіз у реальному часі:

  Здатність аналізувати текст у режимі реального часу, корисна для живих додатків, таких як моніторинг соціальних мереж або підтримка клієнтів.

  Мінуси Amazon Comprehend:

- Вартість:

  Хоча він пропонує розрахункову модель, витрати можуть накопичуватися через великі обсяги даних або широке використання спеціальних моделей.

- Конфіденційність даних:

  Як і в будь-якій хмарній службі, можуть виникнути занепокоєння щодо конфіденційних даних, які обробляються та зберігаються ззовні.

- Обмежені мови:

  Хоча він підтримує кілька мов, діапазон обмежений порівняно з деякими конкуруючими службами, що потенційно може вплинути на глобальні програми.

- Залежність від екосистеми AWS:

  Максимальна вигода приносить використання в екосистемі AWS, що може бути не ідеальним для тих, хто не повністю інвестував у AWS.

- Крива навчання розширеним функціям:

  Хоча базові функції зручні для користувача, використання розширених функцій і користувацьких моделей може потребувати крутішого навчання.

Таким чином, Amazon Comprehend — це потужний інструмент для завдань NLP, який пропонує масштабованість, легкість використання та інтеграцію з екосистемою AWS. Однак при оцінці придатності AWS для конкретної програми слід враховувати міркування щодо вартості, конфіденційності даних, мовної підтримки та потенційної потреби в спеціальному досвіді AWS.

Висновок

В першому розділі було проаналізовано існуючі інструменти для маскування персональних даних. Проаналізувавши переваги і недоліки кожного, було прийтято рішення взяти за основу Comprehend сервіс, так як він інтегрований в хмарне середовище AWS(де і буде відбуватись побудова цілої системи) і має безкоштовний план на цілий рік.

РОЗДІЛ 2
АРХІТЕКТУРА СИСТЕМИ ДЛЯ МАСКУВАННЯ ПЕРСОНАЛЬНИХ ДАНИХ

2.1. Вибір хмарного середовища

Основними факторами для вибору будуть: наявність сервісу, який використовуючи нейромережу, буде аналізувати логи і маскувати персональну інформацію, ціна, доступні регіони.

2.1.1. AWS

Amazon має довгу історію використання децентралізованої IT-інфраструктури. Це дозволило командам розробників

отримати доступ до обчислювальних ресурсів і ресурсів зберігання на вимогу, і це підвищило загальну продуктивність і гнучкість. До 2005 р.

Компанія Amazon витратила більше десяти років і витратила мільйони доларів на створення й управління великомасштабною, надійною та ефективною IT-системою.

Інфраструктура, яка працювала над однією з найбільших у світі платформ онлайн-роздрібної торгівлі. Amazon запустив Amazon Web Services

(AWS), щоб інші організації могли скористатися досвідом та інвестиціями Amazon у ведення великомасштабної компанії розподілена транзакційна IT-інфраструктура. AWS працює з 2006 року і сьогодні обслуговує сотні

тисячі клієнтів по всьому світу. Сьогодні Amazon.com керує глобальною веб-платформою, яка обслуговує мільйони клієнтів і керуючи торгівлею на мільярди доларів щороку.

Використовуючи AWS, ви можете отримати обчислювальну потужність, сховище та інші послуги за лічені хвилини та мати гнучкість вибору

платформу розробки або модель програмування, яка має найбільший сенс для проблем, які вони намагаються вирішити.

Ви платите лише за те, що використовуєте, без попередніх витрат або довгострокових зобов’язань, що робить AWS економічно ефективним способом

для доставки заявок.

Ось кілька прикладів того, як організації, від дослідницьких фірм до великих підприємств, використовують AWS сьогодні:

Велике підприємство швидко та економічно розгортає нові внутрішні додатки, такі як HR-рішення, розрахунок заробітної плати

програми, рішення для управління запасами та онлайн-навчання для розподіленої робочої сили.

Веб-сайт електронної комерції задовольняє раптовий попит на «гарячий» продукт, викликаний вірусним шумом від Facebook

і Twitter без необхідності оновлювати свою інфраструктуру.

Фармацевтична дослідницька фірма проводить масштабне моделювання за допомогою обчислювальної потужності, наданої AWS.

Медіакомпанії пропонують необмежену кількість відео, музики та інших медіа для своїх клієнтів по всьому світу.

AWS легко вирізнити з-поміж інших постачальників у традиційному середовищі IT-обчислень, оскільки це:

1. гнучкий. AWS дозволяє організаціям використовувати моделі програмування, операційні системи, бази даних і архітектури, з якою вони вже знайомі. Крім того, ця гнучкість допомагає організаціям комбінувати та поєднувати архітектури, щоб задовольнити різноманітні бізнес-потреби.
2. Економічно ефективним. Завдяки AWS організації платять лише за те, що вони використовують, без авансу чи довгострокових платежів зобов'язання.
3. Масштабований і еластичний. Організації можуть швидко додавати та віднімати ресурси AWS до своїх програм, щоб задовольняти попит клієнтів і керувати витратами.
4. Безпечний. Щоб забезпечити наскрізну безпеку та наскрізну конфіденційність, AWS створює служби відповідно до найкращі практики

безпеки, надає відповідні функції безпеки в цих службах і документує, як їх використовувати ці особливості.

5. Досвідчений. Використовуючи AWS, організації можуть використовувати більш ніж п’ятнадцятирічний досвід Amazon надання великомасштабної глобальної інфраструктури надійним і безпечним способом.

Amazon Comprehend — це служба обробки природної мови (NLP), яка використовує машинне навчання (ML) для пошуку інформації та зв’язків, як-от людей, місць, настроїв і тем, у неструктурованому тексті. Тепер ви можете використовувати можливості Amazon Comprehend ML для виявлення та редагування ідентифікаційної інформації (PII) в електронних листах клієнтів, заявках у службу підтримки, оглядах продуктів, соціальних мережах тощо. Досвід ML не потрібен. Наприклад, ви можете проаналізувати запити служби підтримки та статті бази знань, щоб виявити ідентифікаційні сутності та відредагувати текст, перш ніж індексувати документи в пошуковому рішенні. Після цього пошукові рішення не містять ідентифікаційних даних у документах. Редагування ідентифікаційних даних допомагає вам захистити конфіденційність і дотримуватися місцевих законів і правил.

Випадок використання замовником: рішення TeraDact. TeraDact Solutions уже запустила цю нову функцію. Програмне забезпечення TeraDact Solutions пропонує надійну альтернативу для безпечного обміну інформацією у світі постійно зростаючих проблем щодо відповідності та конфіденційності. Завдяки фірмовим можливостям ідентифікації та представлення інформації (IIaP™) інструменти TeraDact надають користувачеві безпечне середовище обміну інформацією. «Використання Amazon Comprehend для редагування ідентифікаційної інформації з нашою системою токенізації не тільки допомагає нам охопити більшу кількість наших клієнтів, але й допомагає нам подолати недоліки виявлення ідентифікаційної інформації на основі правил, що може призвести до помилкових тривог або пропущених деталей. Виявлення ідентифікаційної інформації має вирішальне значення для бізнесу, і за

допомогою потужності контекстно-залежних моделей NLP від Comprehend ми можемо підтримувати довіру клієнтів до нас своєю інформацією. Amazon впроваджує інновації, щоб просувати наш бізнес вперед, додаючи нові функції, які є критично важливими для нашого набору продуктів». сказав Кріс Шріхте, генеральний директор TeraDact Solutions, Inc.

2.1.2. GCP

Google Cloud Platform (GCP), запропонована Google, — це набір хмарних обчислювальних сервісів, який надає низку модульних хмарних сервісів, включаючи обчислення, зберігання даних, аналітику даних і машинне навчання, а також набір інструментів керування.[2] Він працює на тій самій інфраструктурі, яку Google використовує внутрішньо для своїх продуктів для кінцевих користувачів, таких як Google Search, Gmail і Google Docs, згідно з Верма та іншими.[3] Для реєстрації потрібні дані кредитної картки або банківського рахунку.[4] Google Cloud Platform надає інфраструктуру як послугу, платформу як послугу та безсерверні обчислювальні середовища.

У квітні 2008 року Google анонсувала App Engine, платформу для розробки та розміщення веб-додатків у керованих Google центрах обробки даних, яка була першою послугою хмарних обчислень від компанії. Сервіс став загальнодоступним у листопаді 2011 року. Після анонсу App Engine Google додав до платформи кілька хмарних сервісів.

Google Cloud Platform є частиною[5] Google Cloud, яка включає інфраструктуру загальнодоступної хмари Google Cloud Platform, а також Google Workspace (G Suite), корпоративні версії Android і ChromeOS та інтерфейси прикладного програмування (API) для машинного навчання. і картографічні послуги підприємства.

У Google Cloud Platform (GCP) основною службою, призначеною для ідентифікації, перевірки та потенційного маскування або редагування особистої ідентифікаційної інформації (PII), є Cloud Data Loss Prevention (DLP) API. Ця

служба пропонує набір інструментів для виявлення, класифікації та захисту конфіденційних даних.

Cloud Data Loss Prevention (DLP) API. Cloud DLP API надає кілька функцій для обробки ідентифікаційних даних:

Виявлення та класифікація даних:

Автоматично виявляйте та класифікуйте ідентифікаційну інформацію та інші конфіденційні дані в таких службах GCP, як BigQuery, Cloud Storage та Datastore.

Виявляйте широкий діапазон типів даних, у тому числі загальну ідентифікаційну інформацію, як-от імена, номери кредитних карток, номери соціального страхування тощо.

Маскування та редагування даних:

Деідентифікуйте конфіденційні дані, застосовуючи різні методи перетворення, такі як редагування (видалення даних), маскування (приховування даних) і токенізація (заміна даних маркерами).

Пропонує гнучкість у способах деідентифікації даних, дозволяючи часткове редагування, хешування та інші методи.

Аналіз ризиків:

Оцініть ризик розголошення конфіденційних даних, проаналізувавши, як дані поширюються, і визначивши потенційні області концентрації ідентифікаційної інформації.

Перевірка та звітність:

Скануйте та перевіряйте дані, щоб знайти потенційні випадки ідентифікаційної інформації та створюйте звіти про результати.

Корисно для перевірок відповідності та конфіденційності.

Випадки використання Cloud DLP

Захист даних у хмарному сховищі: автоматично скануйте та редагуйте ідентифікаційну інформацію у файлах, що зберігаються у хмарному сховищі.

Відповідність у BigQuery: аналізуйте конфіденційні дані в наборах даних BigQuery та видаляйте їх ідентифікацію, особливо корисно для відповідності GDPR, HIPAA або CCPA.

Маскування даних у потоках і експорті: застосовуйте перетворення деідентифікації в реальному часі до потокових даних або під час експорту даних із GCP.

Cloud DLP можна інтегрувати з іншими службами GCP для безперебійного захисту даних.

Його можна використовувати разом із Cloud Pub/Sub для обробки даних у реальному часі та Cloud Functions або Cloud Run для безсерверної трансформації даних.

Cloud DLP — це потужний інструмент у GCP для організацій, які хочуть керувати та захищати ідентифікаційну інформацію та інші конфіденційні дані. Це допомагає не лише дотримуватися різноманітних нормативних актів щодо захисту даних, але й підтримувати довіру клієнтів, забезпечуючи безпечну та конфіденційну обробку їхніх даних.

2.1.3. Azure

Служби Azure стосуються колекції різноманітних хмарних служб, що надаються платформою Microsoft Azure. Azure пропонує широкий спектр хмарних служб, у тому числі для обчислень, аналітики, зберігання даних і мереж, допомагаючи компаніям керувати своїми програмами та послугами через центри обробки даних, якими керує Microsoft.

Деякі з основних служб Azure включають:

Віртуальні машини (VM) Azure: масштабовані обчислювальні ресурси на вимогу для розміщення програм і служб.

База даних SQL Azure: служба керованої реляційної бази даних на основі SQL Server.

Azure Cosmos DB: Глобально розповсюджена багатомодельна служба бази даних.

Azure Active Directory (AD): служба керування ідентифікацією та доступом.

Azure Blob Storage: масштабоване сховище об’єктів для будь-якого типу неструктурованих даних.

Служба Azure Kubernetes (AKS): служба оркестровки керованого контейнера Kubernetes.

Функції Azure: безсерверна обчислювальна служба, керована подіями.

Програми Azure Logic: хмарний сервіс, який допомагає автоматизувати та оркеструвати завдання, бізнес-процеси та робочі процеси. Переваги Azure перед іншими провайдерами:

Інтеграція з продуктами Microsoft: Azure пропонує повну інтеграцію з широким спектром програмного забезпечення та служб Microsoft, включаючи Windows Server, Active Directory, SQL Server, SharePoint і Dynamics, що може бути особливо корисним для організацій, які вже закріпилися в екосистемі Microsoft.

Гібридні можливості: Azure приділяє значну увагу можливостям гібридної хмари, що дозволяє організаціям інтегрувати локальні центри обробки даних із хмарою, що є перевагою для компаній, яким потрібні гібридні середовища.

Орієнтація на підприємство: Azure відомий своєю сильною орієнтацією на підприємство, надаючи комплексні рішення для великих компаній, включаючи безпеку та відповідність корпоративного рівня.

Інструменти розробника та екосистема: Azure надає багатий набір інструментів і служб розробки, таких як Visual Studio та інтеграція .NET, що може бути особливо привабливим для розробників, знайомих з екосистемою Microsoft.

Глобальне охоплення: Azure має велику мережу центрів обробки даних по всьому світу, що забезпечує краще глобальне покриття та доступність послуг у багатьох регіонах.

Розширені послуги штучного інтелекту та машинного навчання: Azure пропонує надійні пропозиції щодо штучного інтелекту та машинного навчання з такими службами, як Azure Machine Learning, Azure Cognitive Services і Azure Bot Service.

Сильна підтримка IaaS і PaaS: Azure забезпечує збалансовану пропозицію між інфраструктурою як послугою (IaaS) і платформою як послугою (PaaS), надаючи користувачам гнучкість у розгортанні програм і керуванні ними.

Відповідність і безпека: Azure має серйозну прихильність до безпеки та відповідності, з повним набором пропозицій відповідності для задоволення різноманітних нормативних вимог.

Кожен постачальник хмарних послуг, включаючи AWS, Google Cloud Platform (GCP), IBM Cloud та інші, має власний набір функцій і переваг. Вибір між ними часто залежить від конкретних вимог проекту, існуючої інфраструктури та стратегічних міркувань бізнесу.

В Microsoft Azure сервіс, який можна використовувати для виявлення та приховання PII (особистих ідентифікаційних інформацій) даних, це Azure Purview. Раніше цей сервіс був відомий як Azure Data Catalog.

Azure Purview надає можливості класифікації даних, які дозволяють автоматично виявляти та класифікувати конфіденційні дані в різних джерелах даних, таких як Azure Data Lake Storage, Azure Blob Storage, SQL Data Warehouse та інші.

Основні функції Azure Purview для роботи з PII даними:

Автоматичне виявлення та класифікація PII даних: Azure Purview дозволяє виявляти конфіденційну інформацію, таку як імена, адреси, номери телефонів, соціальні номери безпеки та інші PII дані.

Управління даними: Засоби управління даними в Purview дозволяють створювати політики, які регулюють доступ до даних та їх використання.

Маскування даних: Можливість деідентифікувати конфіденційні дані за допомогою маскування або інших технік зменшення чутливості даних.

Моніторинг та звітність: Надає можливості для моніторингу та створення звітів про статус конфіденційних даних у вашому середовищі.

Дотримання нормативних вимог: Допомагає організаціям відповідати глобальним та регіональним нормативним вимогам щодо конфіденційності даних.

Azure також пропонує інші інструменти та сервіси, які можуть бути використані у поєднанні з Purview для розширеного захисту та управління PII даними, такі як Azure Information Protection для класифікації та захисту електронної пошти та інших документів.

Користування цими сервісами дозволяє забезпечити захист PII даних у відповідності до корпоративних політик та нормативних вимог, тим самим знижуючи ризики, пов'язані з даними в Azure.

2.2. Опис архітектури на базі AWS

Проаналізувавши вищеописані провайдери хмарних сервісів, було прийняте рішення будувати систему на базі AWS, так як сервіс Comprehend найбільш гнучкий та дешевий, а сам AWS дає безкоштовний доступ до своїх сервісів на цілий рік.

2.2.1. EKS

Всі компоненти системи будуть встановлені на кубернетіс сервіс від AWS EKS. K8s — це скорочення для Kubernetes, яке походить від того, що між буквами "K" та "s" розміщено 8 букв. Kubernetes — це відкритий платформа для автоматизації розгортання, масштабування та управління контейнеризованими додатками. Ось основні аспекти Kubernetes:

- Оркестрація контейнерів

  Kubernetes дозволяє управляти контейнерами, які виконують додатки, забезпечуючи необхідні інструменти для розгортання та масштабування додатків, а також для підтримки їх стану.

- Висока наявність

Kubernetes гарантує, що додатки завжди доступні та відмовостійкі. Він автоматично розміщує контейнери на основі їх вимог до ресурсів і доступності, перерозподіляє ресурси, якщо сервери зазнають збоїв, і автоматично перезапускає контейнери, які не відповідають.

- Масштабування

  Kubernetes дозволяє автоматично масштабувати системи на основі використання ресурсів або інших метрик, що є ключовим для оптимізації вартості та ефективності.

- Балансування навантаження

  Kubernetes може розподіляти мережевий трафік так, щоб навантаження було рівномірно розподілене між контейнерами, забезпечуючи таким чином оптимальну роботу додатків.

- Самовідновлення

  Kubernetes забезпечує самовідновлення додатків, автоматично замінюючи контейнери, які не працюють, перезапускаючи невдачі контейнери, вбиваючи контейнери, які не відповідають визначеним користувачем здоров'ям перевірок, і відмовляючись від розгортань, які не вдаються.

- Управління конфігурацією та секретами

  Kubernetes дозволяє управляти конфігурацією та секретами додатків окремо від образів контейнерів, що підвищує безпеку та гнучкість розгортання.

Kubernetes став стандартом де-факто для оркестрації контейнерів у виробничих середовищах, завдяки своїй гнучкості, потужним можливостям і широкій підтримці спільноти та індустрії.

Amazon Elastic Kubernetes Service (EKS) — це керований сервіс оркестрації контейнерів від Amazon Web Services (AWS), який дозволяє легко запускати, масштабувати та управляти додатками на базі Kubernetes на AWS. Ось основні аспекти Amazon EKS:

- Керований Kubernetes

  Amazon EKS автоматично керує складними завданнями, такими як управління кластером Kubernetes, налаштування, патчінг та оновлення.

- Висока наявність

  Amazon EKS забезпечує високу доступність для Kubernetes кластерів, розгортаючи кластери у декількох зонах доступності (Availability Zones) для забезпечення відмовостійкості.

- Безпека

  Amazon EKS інтегрується з AWS Identity and Access Management (IAM), надаючи детальний контроль доступу до сервісів на рівні користувача.

- Інтеграція з AWS сервісами

  EKS можна інтегрувати з багатьма іншими сервісами AWS, включаючи Amazon Elastic Load Balancing (ELB), Amazon EBS, Amazon EFS, AWS Identity and Access Management (IAM) та Amazon VPC.

- Сумісність з Kubernetes

  Amazon EKS повністю сумісний з додатками, розробленими для самостійних Kubernetes кластерів, що означає, що існуючі додатки на Kubernetes можуть легко мігрувати до EKS без необхідності внесення змін.

- Спільнота та підтримка

  Як і інші продукти AWS, EKS підтримується широкою спільнотою користувачів та партнерів, а також має доступ до обширної документації та підтримки від AWS.

EKS використовується організаціями, які бажають використовувати переваги Kubernetes без необхідності самостійно управляти інфраструктурою Kubernetes.

2.2.2. Fluentbit

Fluent Bit — це відкритий і легкий логгер та збирач даних для Linux, Windows і macOS. Це інструмент, створений для збору, обробки та надсилання журналів та метрик до різних виходів, як-от централізовані системи управління журналами, бази даних чи індексатори (наприклад, Elasticsearch, Splunk, Datadog).

Fluent Bit особливо оптимізований для контейнерів і мікросервісних архітектур, надаючи функціоналітет для швидкого збору та агрегації даних у таких середовищах, як Kubernetes. Ось деякі ключові особливості Fluent Bit:

- Написаний на C, Fluent Bit має маленький розмір і високу продуктивність.
- Підтримує різноманітні вхідні плагіни (inputs), фільтри (filters), та виходи (outputs), що робить його дуже гнучким для збору та маршрутизації журналів.
- Збирає дані з різних джерел, таких як файли, системні журнали, HTTP, TCP та інші.
- Обробляє та збагачує дані в реальному часі, використовуючи фільтри для модифікації або додавання нової інформації до потоку журналів.
- Відправляє журнали до різних приймачів або служб, таких як Elasticsearch, Kafka, HTTP end points, файлові системи тощо.
- Працює на Linux, Windows і macOS, дозволяючи збирати та обробляти дані з різних операційних систем.
- Підтримує моніторинг та обробку надійності, забезпечуючи відстеження та збереження даних навіть у випадку збоїв мережі чи інших непередбачених обставин.

Fluent Bit часто використовується у комбінації з іншим інструментом відкритого коду — Fluentd. Fluentd забезпечує більш важкий і повний функціонал для обробки та маршрутизації журналів, тоді як Fluent Bit є більш оптимізованим для швидкого збору даних і використовується в основному для пересилання даних до Fluentd або інших систем.

2.2.2.1. Lua скрипт

Fluent Bit дозволяє використовувати Lua скрипти для розширення його можливостей по обробці даних. Lua — це потужна, швидка, легка та вбудовувана мова сценаріїв. У контексті Fluent Bit, Lua скрипти можуть використовуватися для створення користувацьких фільтрів, які дозволяють виконувати різноманітні операції з обробки журналів під час їхньої передачі через потік обробки.

Як працює Lua скрипт у Fluent Bit

Ви можете написати Lua скрипт, який визначає функції для модифікації або фільтрації записів журналів. Сценарій Lua потім може бути підключений до конфігурації Fluent Bit за допомогою фільтра Lua. Ось приклад:

```lua
-- my_filter.lua
function my_filter(tag, timestamp, record)
    -- Новий запис, який буде повернутий
    new_record = record

    -- Додаємо нове поле до запису
    new_record["new_field"] = "new_value"

    -- Вказуємо, що запис потрібно зберегти
    return 1, timestamp, new_record
end
```

Рис.1. Приклад Lua скрипта

Тепер ви могли б використати цей скрипт у вашому Fluent Bit конфігураційному файлі таким чином:

```
[FILTER]
    Name   lua
    Match  *
    script /path/to/my_filter.lua
    call   my_filter
```

Рис.2. Приклад використання Lua скрипта

У цьому конфігураційному файлі:

[FILTER] блок вказує, що ми хочемо використовувати фільтр.

Name lua вказує, що ми використовуємо Lua фільтр.

Match * вказує на те, що фільтр застосовується до всіх записів.

script вказує шлях до Lua скрипту.

call вказує функцію у Lua скрипті, яку потрібно викликати для кожного запису.

Lua скрипти у Fluent Bit можуть бути використані для:

- Зміни структури записів.
- Збагачення записів додатковими даними.
- Вилучення або маскування конфіденційної інформації.
- Відкидання записів, які не відповідають певним умовам.
- Розподілу записів на основі динамічних умов.
- Використання Lua скриптів у Fluent Bit надає велику гнучкість та потужність для користувацької обробки даних, дозволяючи розробникам тонко налаштовувати процес обробки журналів згідно з їх конкретними вимогами.

2.2.3. Elasticsearch

Elasticsearch — це відкритий пошуковий та аналітичний двигун, який широко використовується для всіляких завдань, пов'язаних з індексацією, пошуком та аналізом великих обсягів даних в реальному часі. Він базується на бібліотеці Lucene і забезпечує розподілену мульти-тенантну здатність до повнотекстового пошуку з відкритим API, що працює через веб-інтерфейси та схеми JSON.

Ось декілька ключових особливостей Elasticsearch:

- Горизонтальне Масштабування: Elasticsearch дозволяє легко масштабувати систему, додаючи більше вузлів, і він автоматично розподіляє дані і запити по вузлах.
- Відмовостійкість: Дані автоматично реплікуються по вузлах кластера, що забезпечує високу доступність і відновлення від помилок.

- Аналізатори та Токенізатори: Elasticsearch має потужну систему аналізу тексту, яка підтримує багато мов та може обробляти складні текстові дані для пошуку.
- Швидкий Відгук: Дані в Elasticsearch індексуються і стають доступними для пошуку майже в реальному часі, що робить його ідеальним для задач, які вимагають швидкого доступу до даних.
- RESTful API: Elasticsearch можна використовувати за допомогою простих HTTP запитів, і він відповідає у форматі JSON.
- Клієнтські Бібліотеки: Існує багато клієнтських бібліотек для різних мов програмування, що полегшує інтеграцію з Elasticsearch.
- Схеми: Elasticsearch не вимагає жорстко визначених схем, дозволяючи динамічно додавати поля до документів і змінювати індекси.
- Aggregations: Elasticsearch дозволяє виконувати складні агрегаційні операції для аналізу даних, що робить його потужним інструментом для виведення інсайтів з ваших даних.

Elasticsearch широко використовується в таких сферах, як лог-аналітика, пошук всередині веб-сайтів, сек'юріті-інтелігенс, бізнес-аналітика, а також як ключовий компонент в стеку ELK (Elasticsearch, Logstash, Kibana), який є популярним рішенням для моніторингу і аналізу журналів.

2.2.4. Kibana

Kibana — це безкоштовний відкритий веб-інтерфейс для візуалізації та керування даними в Elasticsearch. Вона є частиною Elastic Stack, до якого також входять Elasticsearch, Logstash і Beats, і часто використовується разом з ними для аналізу та моніторингу логів, метрик та інших видів часових даних. Основні можливості Kibana:

- Візуалізація Даних: Kibana дозволяє створювати різноманітні візуалізації для даних, які зберігаються в Elasticsearch, включаючи графіки, карти, гістограми та багато іншого.

- Дашборди: Користувачі можуть комбінувати візуалізації в дашборди, які надають багатогранний огляд даних та можуть бути налаштовані для відображення різних метрик.
- Розширений Пошук: Kibana надає можливість проводити гнучкі та складні запити для пошуку конкретних даних в Elasticsearch.
- Моніторинг: Користувачі можуть використовувати Kibana для моніторингу здоров'я кластерів Elasticsearch, стану вузлів та продуктивності.
- Управління Elasticsearch: Kibana також забезпечує інтерфейс для управління Elasticsearch, дозволяючи користувачам індексувати дані, налаштовувати вузли та виконувати інші адміністративні завдання.
- Машинне Навчання: У Kibana є вбудовані засоби для машинного навчання, які можуть виявляти аномалії в часових рядах даних.
- Canvas: Функція Canvas у Kibana дозволяє створювати індивідуальні динамічні інфографіки, що включають візуалізації в реальному часі.
- Elastic Maps: Користувачі можуть створювати карти для візуалізації геопросторових даних.

Kibana часто використовується для лог-аналітики, де вона допомагає інженерам та аналітикам розуміти великі обсяги логів, вироблених серверами та додатками. Завдяки своїй інтуїтивно зрозумілій інтерфейсу та могутнім аналітичним можливостям, Kibana є популярним рішенням для багатьох організацій, які хочуть легко візуалізувати та отримати цінні інсайти зі своїх даних.

2.2.5. Purchase сервіс

Простий сервіс написаний на nodejs використовуючи express фреймоворк. Буде мати один ендпоінт /purchase, основна функція якого – це прологувати тіло запроса в консоль у вигляді JSON строки.

2.2.6. Helm

Helm — це менеджер пакетів для Kubernetes, який спрощує розгортання та управління додатками на Kubernetes. Він дозволяє розробникам та системним адміністраторам легко пакувати, налаштовувати та розподіляти додатки Kubernetes.

Основні аспекти Helm:

- Чарти Helm. Чарти Helm — це пакети, які містять всю необхідну інформацію для розгортання додатка, сервісу або інструменту на Kubernetes. Чарти включають YAML конфігураційні файли та шаблони, які дозволяють налаштувати розгортання для різних середовищ або сценаріїв використання.
- Репозиторії. Репозиторії Helm дозволяють розробникам зберігати та ділитися чартами Helm. Вони сприяють спільному використанню чартів всередині спільноти або організації.
- Установка та Управління. Helm спрощує установку та управління Kubernetes додатками. Команди, такі як helm install, helm upgrade, і helm delete, дозволяють легко керувати додатками.
- Шаблонізація. Helm використовує систему шаблонів для створення конфігурацій Kubernetes. Це дозволяє використовувати один чарт для генерації відповідних конфігурацій для різних середовищ або варіантів використання.
- Управління Залежностями. Helm дозволяє визначати та управляти залежностями між чартами, що спрощує складне розгортання, яке містить багато частин.
- Атомарні Операції. Helm розроблений для підтримки атомарних операцій, що забезпечує надійне розгортання, таким чином, що можна легко відкатити зміни до стабільного стану у випадку невдач.

Helm є важливою частиною екосистеми Kubernetes і використовується для забезпечення більш простого та управлінням процесу розгортання, який традиційно міг би бути складним та схильним до помилок. Helm стандартизує

підхід до управління ресурсами Kubernetes, що робить процес розгортання більш предбачуваним та легким для автоматизації.

2.2.7. Terraform

Terraform — це інструмент для побудови, зміни та версіонування інфраструктури безпечним і ефективним способом. Він використовує декларативний код для визначення і управління ресурсами різних сервісів хмарних провайдерів, таких як AWS, Google Cloud Platform, Microsoft Azure, а також інших провайдерів.

Основні характеристики Terraform:

- Інфраструктура як Код (IaC): Terraform дозволяє описувати інфраструктуру за допомогою високорівневої конфігураційної мови HCL (HashiCorp Configuration Language) або JSON.
- Управління Різними Провайдерами: Інструмент підтримує різноманітні технологічні платформи і сервіси, не обмежуючись лише одним хмарним провайдером.
- Ідемпотентність: Terraform операції є ідемпотентними, що означає виконання одних і тих самих конфігураційних файлів кілька разів дасть один і той самий результат і не призведе до змін у вже існуючій інфраструктурі, якщо ці зміни не були описані у конфігурації.
- Планування та Застосування Змін: Terraform виконує планування змін перед їх застосуванням, надаючи користувачам можливість побачити, які зміни будуть внесені в інфраструктуру, перш ніж вони стануться.
- Залежності Ресурсів: Terraform автоматично визначає залежності між ресурсами і виконує операції в правильному порядку для досягнення бажаного стану інфраструктури.
- Модульність: Terraform дозволяє використовувати модулі для організації і повторного використання коду, що полегшує управління складною інфраструктурою та сприяє кращому управлінню кодом.

- Версіонування: Terraform дозволяє версіонувати інфраструктуру, використовуючи системи контролю версій, такі як Git, що полегшує відстеження змін і відкат до попередніх версій.

Terraform широко використовується у DevOps для автоматизації управління інфраструктурою та підтримки інфраструктури як коду, що дозволяє швидко та ефективно запускати і оновлювати інфраструктурні компоненти.

2.2.8. Github actions

GitHub Actions — це платформа автоматизації розробки програмного забезпечення, яка інтегрована безпосередньо в GitHub. GitHub Actions дозволяє автоматизувати, налаштовувати та виконувати ваші робочі процеси розробки прямо у вашому репозиторії.

Основні характеристики GitHub Actions:

- Автоматизація робочих процесів. Ви можете створювати робочі процеси, що автоматично будують, тестують, публікують, розгортають та оновлюють ваші проекти, використовуючи YAML-файли у директорії .github/workflows у вашому репозиторії.
- Підтримка різних мов та платформ. GitHub Actions підтримує багато мов програмування та платформ, дозволяючи запускати робочі процеси на різних операційних системах, включаючи Linux, macOS та Windows.
- Спільнота та Перевикористання. Ви можете використовувати дії, створені спільнотою, або створювати свої власні. Це дозволяє легко перевикористовувати код, інтегруватися з іншими інструментами та сервісами.
- Тригери подій. Робочі процеси можуть бути запущені різними подіями, такими як push до репозиторію, створення pull request, публікація релізу, зміна у визначеній гілці або тегу, та багатьма іншими.
- Секрети та змінні середовища. Ви можете використовувати секрети для зберігання конфіденційної інформації та змінні середовища для налаштування робочих процесів.

- Гнучкість та Розширюваність. GitHub Actions надає велику гнучкість, дозволяючи вам писати складні робочі процеси для задоволення вашого конкретного процесу розробки та потреб доставки.

GitHub Actions став популярним інструментом для CI/CD (Continuous Integration та Continuous Deployment), оскільки він дозволяє розробникам значно спростити процеси тестування та розгортання програмного забезпечення.

Висновок

У цьому розділі були проаналізовані різні хмарні середовища, і був вибраний найкращий із них за декількома параметрами. Також було глибоко проаналізовано Fluenbit як технологію, яка дозволяє під час пересилання логів до бази даних, оброблювати їх, а саме – знаходити персональну інформацію і маскувати її. Тому я дійшов висновку, що цей компонент є найважливішим і в подальшій реалізації, я зроблю найбільший фокус саме на ньому.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ МАСКУВАННЯ ПЕРСОНАЛЬНИХ ДАНИХ

3.1. Створення Авс аккаунту

Крок 1. Спочатку відкрийте веб-переглядач і перейдіть до сторінки безкоштовного рівня AWS

Крок 2: натисніть середню кнопку Створити безкоштовний обліковий запис

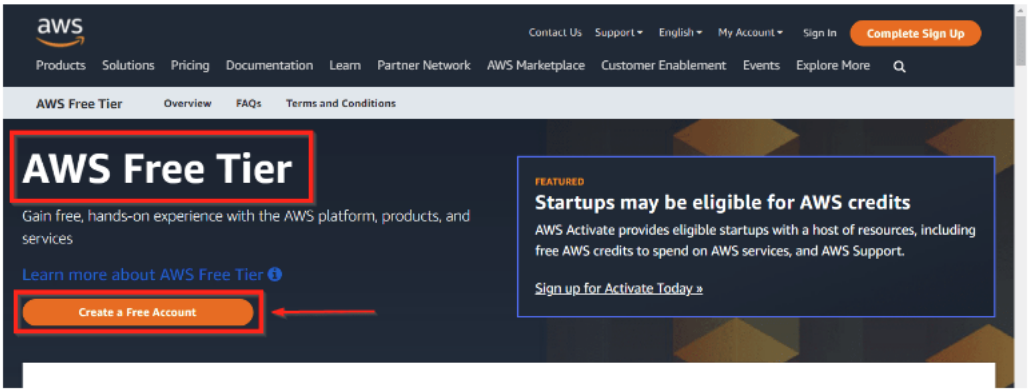

Рис.3. Кнопка створення акаунту

Крок 3: введіть дані, які ви хочете використовувати для входу в обліковий запис AWS, і натисніть «Продовжити»

- Адреса електронної пошти: надайте ідентифікатор електронної пошти, який ще не зареєстровано в Amazon AWS.
- Пароль: введіть свій пароль.
- Підтвердити пароль: автентифікація пароля.
- Назва облікового запису AWS: виберіть ім’я для свого облікового запису. Ви можете змінити це ім’я в налаштуваннях облікового запису після реєстрації.

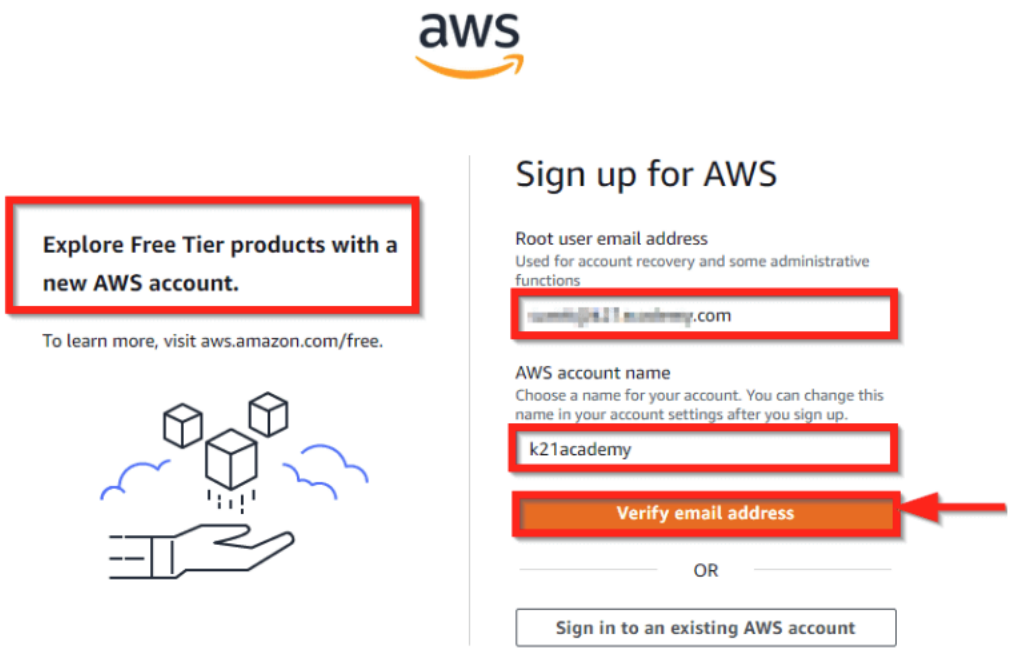

Рис.4. Кнопка верифікації імейл адресу

Крок 4: Контактна інформація

Виберіть тип AWS (професійний/персональний). Заповніть правильну інформацію, щоб підтвердити свій обліковий запис, якщо ви збираєтеся створити особисте використання, а потім натисніть «Особистий обліковий запис», інакше використовуйте «Обліковий запис компанії», прийміть умови та

умови, а потім натисніть «Створити обліковий запис і продовжити».

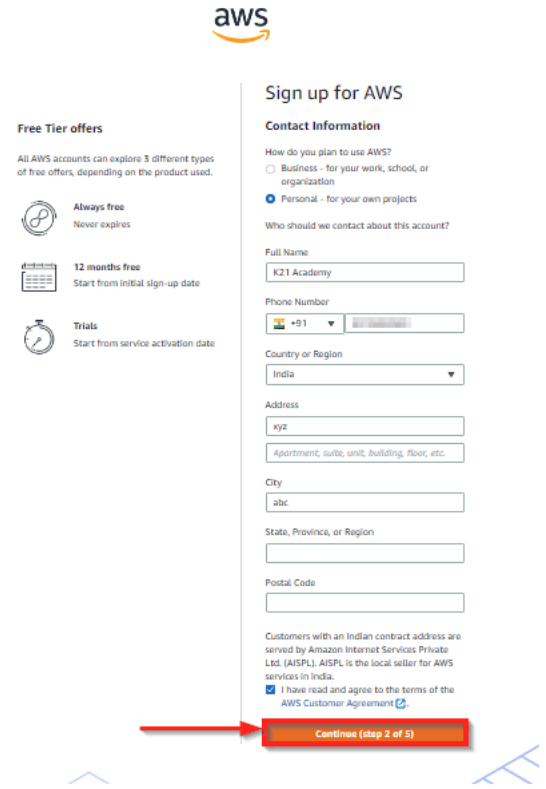

Рис.4. Заповнення контактної інформації

Примітка: переконайтеся, що ви вказали належні контактні дані та номер мобільного телефону, щоб отримати код підтвердження від AWS.

Крок 5: Інформація про платіж і PAN: на цьому кроці ви повинні ввести інформацію про свою кредитну/дебетову картку та платіжну адресу та натиснути «Безпечне надсилання».

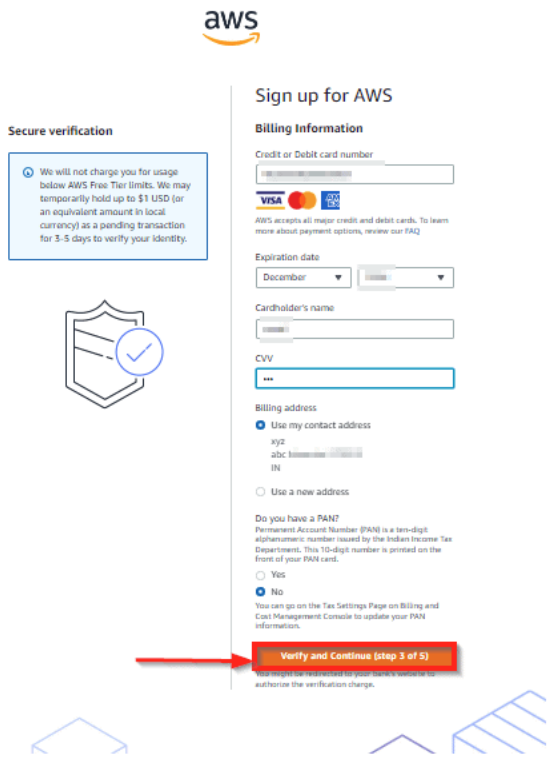

Рис.5. Заповнення контактної інформації

Крок 6. На цьому етапі ви перейдете до платіжного шлюзу, щоб перевірити вашу платіжну інформацію, а для підтвердження вашої кредитної картки Amazon стягуватиме мінімальну ціну залежно від країни. Тут я надав Індію, тому Amazon стягнув 2 INR.

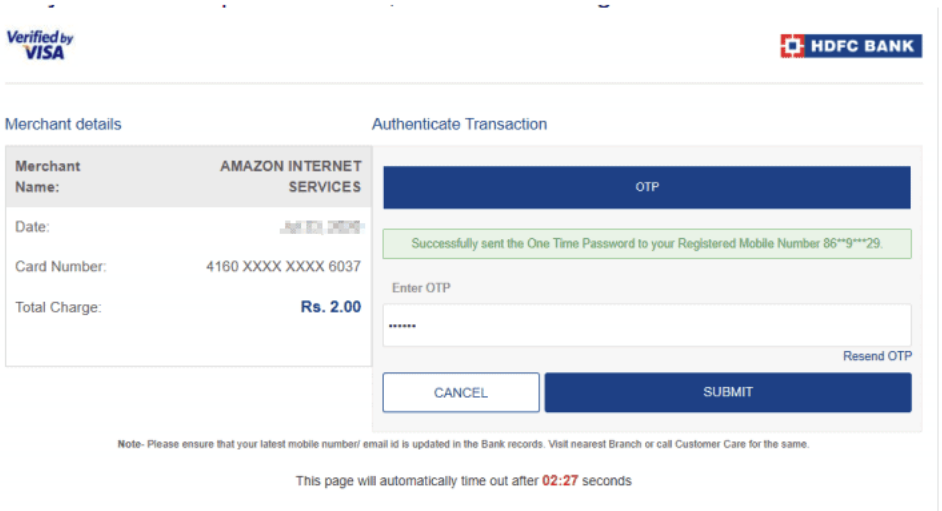

Рис.6. Заповнення платіжної інформації

Крок 7: Перевірка телефону: тут ви потрапите на сторінку підтвердження особи, на якій уже буде ваш номер телефону, тому вам потрібно просто вибрати «Текстове повідомлення або голосовий дзвінок». Укажіть дійсний номер телефону, введіть кодову перевірку, а потім натисніть «Надіслати SMS» або «Зателефонуйте мені зараз» (залежно від вашого вибору).

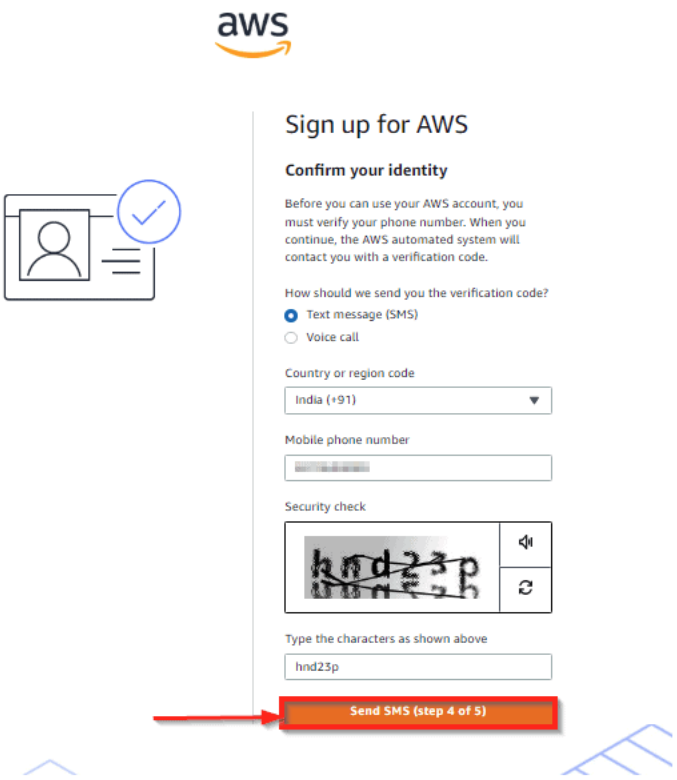

Рис.7. Форма верифікації через смс

Крок 8: Після натискання «Надіслати SMS» або «Зателефонувати мені зараз» ви негайно отримаєте дзвінок або SMS від Amazon, щоб отримати код підтвердження. Введіть свій код, а потім натисніть «Підтвердити код».

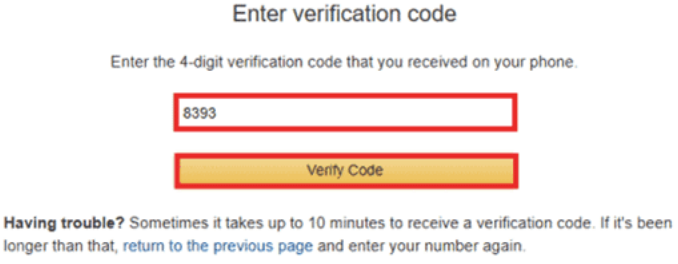

Рис.7. Форма ведення смс кода

Крок 9: План підтримки. Підтримка AWS пропонує вибір планів для задоволення потреб вашого бізнесу.

Виберіть відповідний план і натисніть «Продовжити».

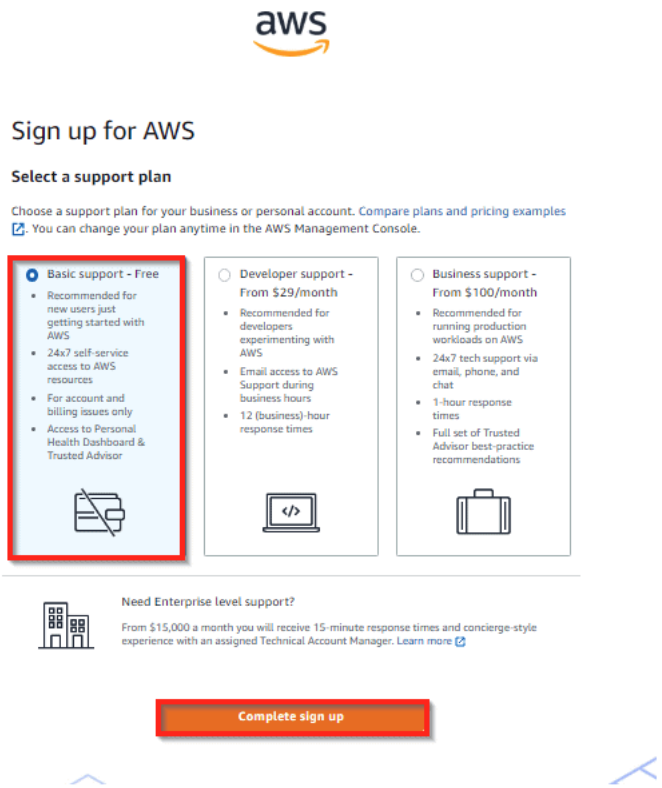

Рис.8. Форма вибору плану

Примітка. Усі клієнти отримують безкоштовну базову підтримку.

Крок 10: Сторінка підтвердження реєстрації.

Після виконання всіх вищезазначених кроків і процесів. Нижче ви побачите сторінку підтвердження. Тепер ваш обліковий запис буде оброблено для активації. Вам може знадобитися від 30 хвилин до 1 години, перш ніж ви отримаєте підтвердження електронною поштою про те, що ваш обліковий запис Amazon Cloud Services активовано.

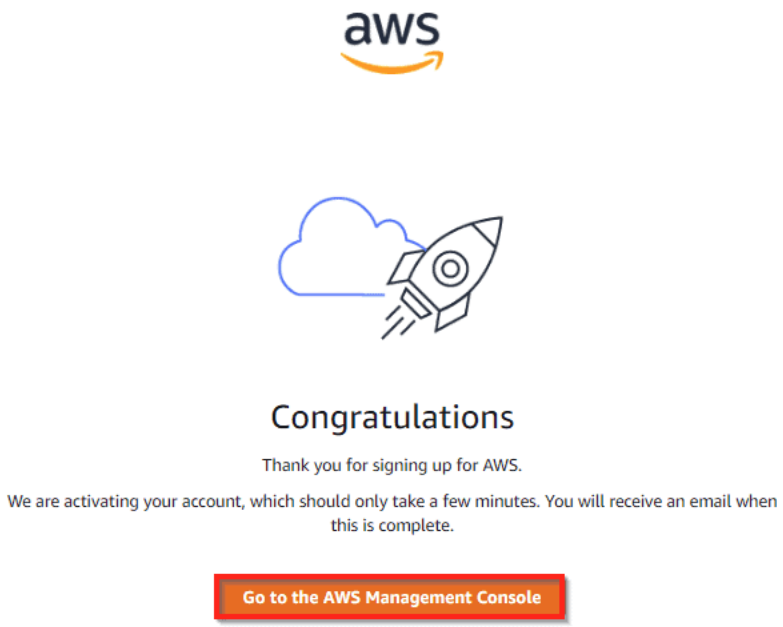

Рис.8. Сторінка підтвердження реєстрації

ми успішно створили обліковий запис безкоштовного рівня AWS.

Безкоштовні обмеження рівня

- Щомісяця 750 годин хмарних обчислень Amazon EC2 з можливістю масштабування.
- 5 ГБ основного сховища на Amazon S3, інфраструктура для масштабованого, надійного та безпечного зберігання об’єктів.
- Щомісячне використання бази даних, призначене Amazon RDS 750 годин (для відповідних механізмів баз даних) SQL Server, MariaDB, PostgreSQL і служби керованих реляційних баз даних MySQL.
- 5 ГБ пам’яті Amazon EFS. Рішення для спільного зберігання файлів, яке просте у використанні та масштабується для примірників Amazon EC2.
- 30 ГБ загального призначення (SSD) або Magnetic Elastic Block Storage від Amazon Elastic Store — це довговічні, надійні сховища на блочному рівні з низькою затримкою для екземплярів EC2.

3.2. EKS розгортання

Для розгортання EKS ми будемо використовувати Terraform скрипти та Github Actions пайплайн, щоб автоматизувати весь процес. Почнемо зі структури

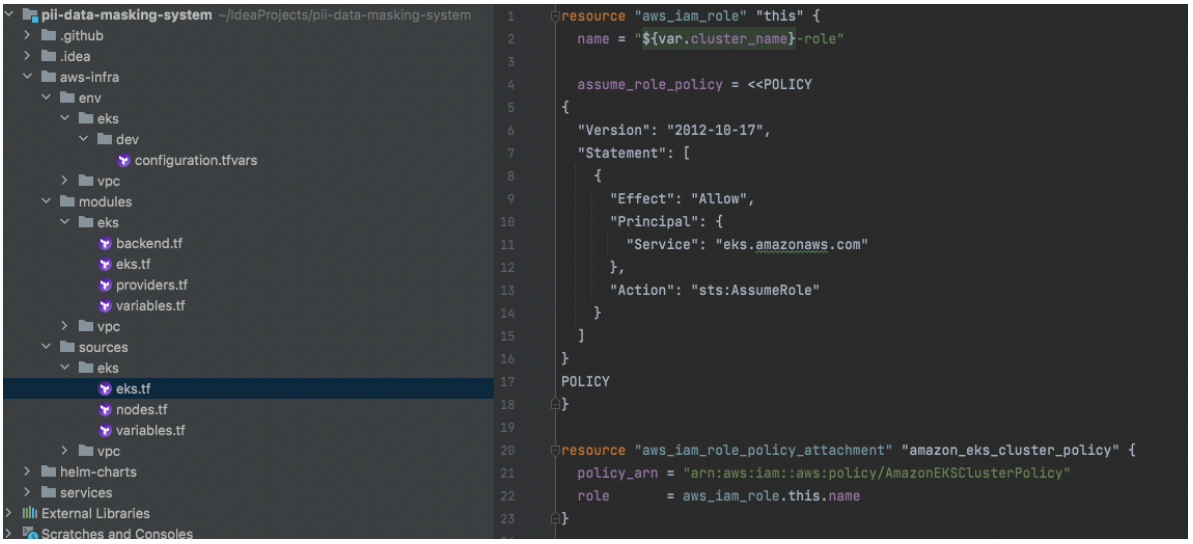

Рис.9. Тераформ скрипт для створення кластеру

Після успішного завершення пайплайну, ми можемо переконатись, що кластер вже працює:

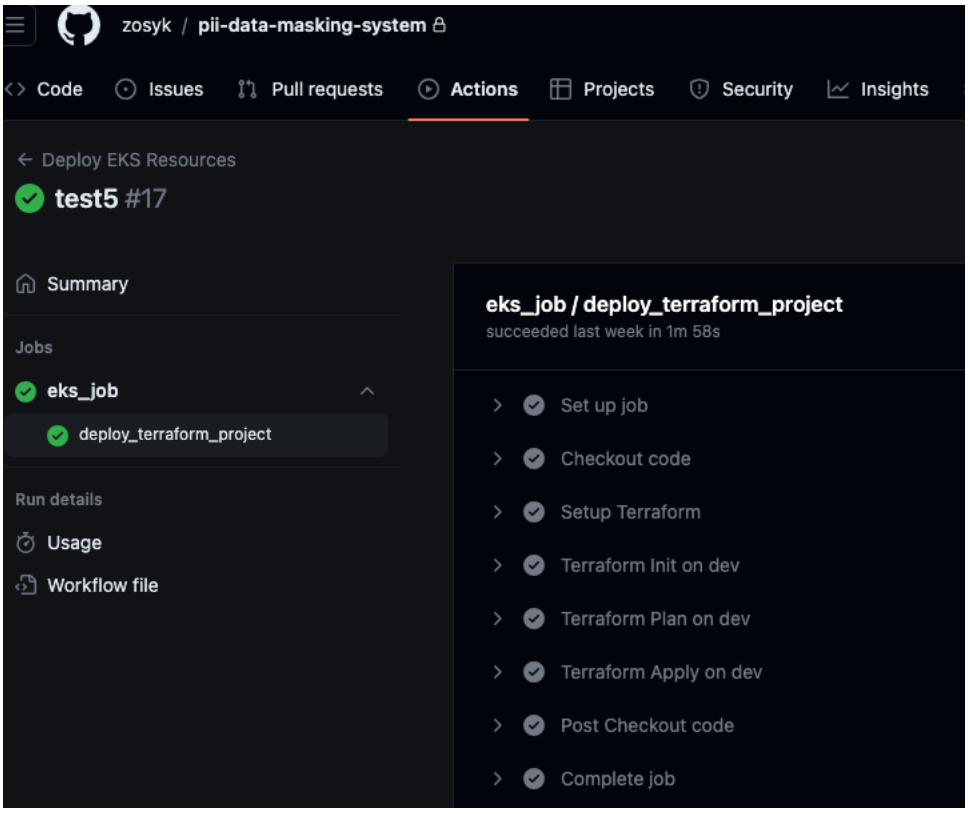

Рис.10. Сторінка успішного завершення пайплайна для створення кластера

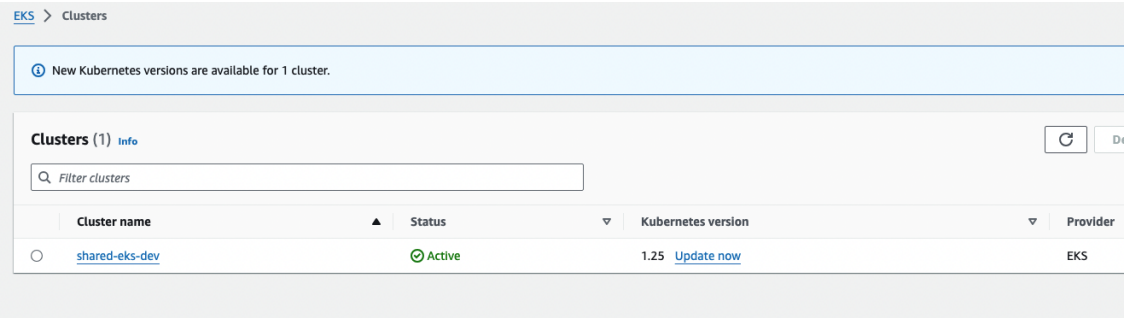

Рис.11. Спикок активних кластерів

```
oleksandrzosyk@WIO-OZOSYK IdeaProjects % ku arn:aws:eks:eu-central-1:205286308224:cluster/shared-eks-dev
Switched to context "arn:aws:eks:eu-central-1:205286308224:cluster/shared-eks-dev".
oleksandrzosyk@WIO-OZOSYK IdeaProjects % k get pods
No resources found in default namespace.
```

Рис.12. Список доступних подів

3.2. Purchase Service розгортання

Для тестування нам буде потрібен простий сервіс з базовим функціоналом. Я вирішив створити сервіс покупки з одним відкритим ресурсом /purchase, який є POST запитом, приймає пейлоад і просто логує його в консоль у вигляді JSON строки.

```
const express = require('express');

const app = express();
const port = process.env.PORT || 3000;

app.use(express.json());

app.post( path: '/purchase', handlers: (req : Request<P, ResBody, ReqBody, Req
    console.log(JSON.stringify(req.body));

    res.status( code: 200).send( body: 'Received the request.');
});

app.listen(port, hostname: () => {
    console.log(`Server running on port ${port}`);
});
```

Рис.13. Імплементація сервісу

Для розгортування на кубернетес, нам потрібен буде докер контейнер.

```dockerfile
# Use an official Node runtime as a parent image
FROM node:14

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json (or npm-shrinkwrap.jso
COPY package.json ./

# Install any needed packages
RUN npm install

# Bundle the app's source code inside the Docker image
COPY ../.. .

# Make the container's port 3000 available to the outside
EXPOSE 3000

# Define the command to run the app
CMD [ "node", "index.js" ]
```

Рис.14. Докер імплементація сервісу покупки

Також необхідно додати кубернетес ресурси.

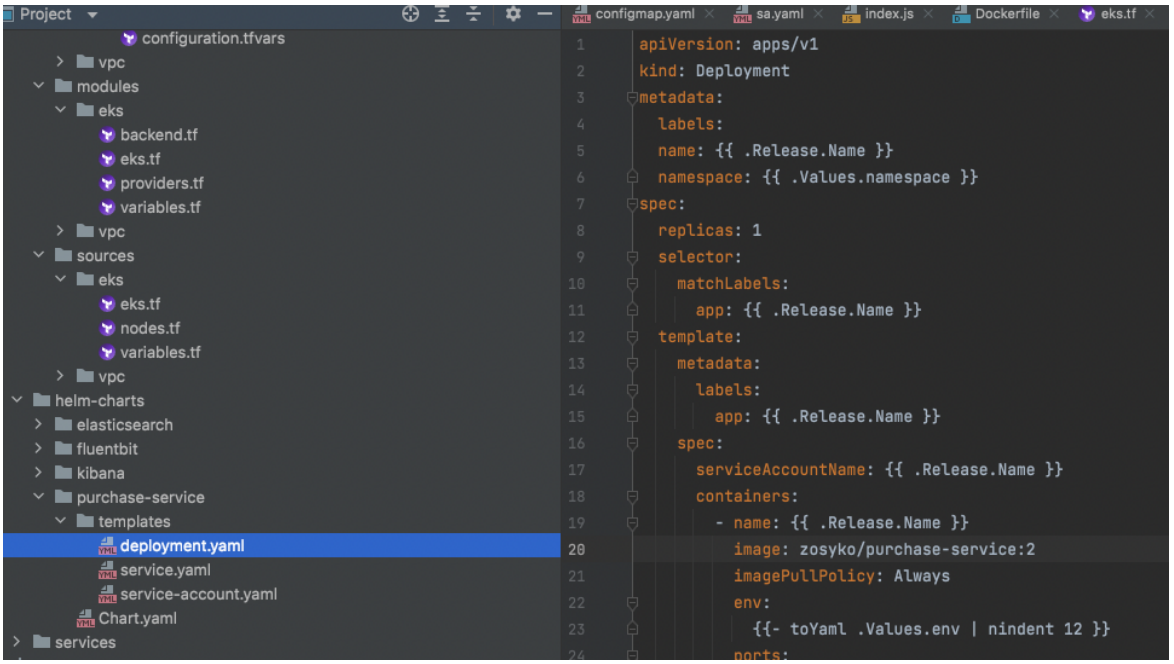

Рис.15. Хелм чарти сервісу покупки

І сам пайплайн, який автоматизує процес розгортання на сам кластер

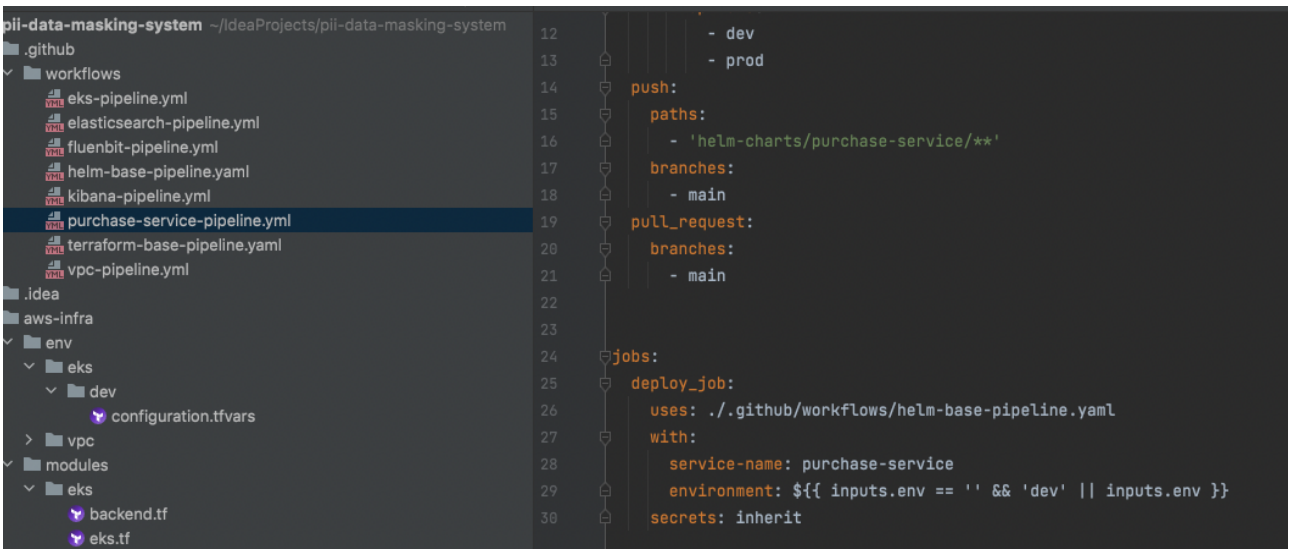

Рис.16. Пайплайн імплементація для сервісу покупки

Після успішного завершення пайплайну, перевіряємо наявність сервісу на самому кубернетес кластері:

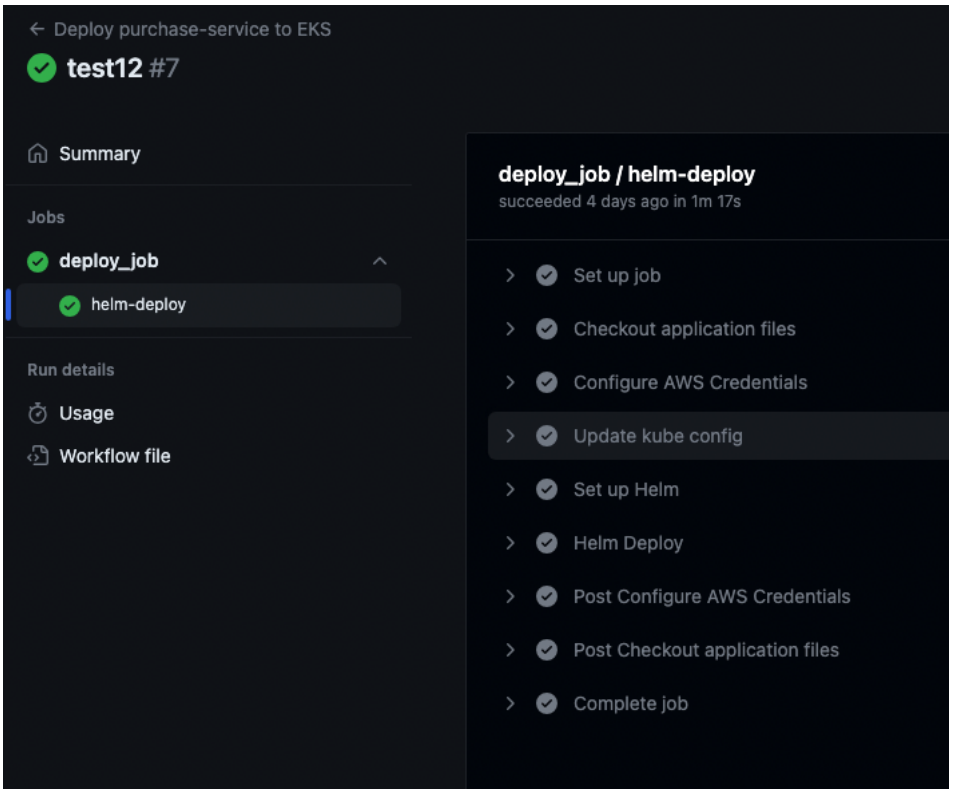

Рис.17. Результат успішно завершеного пайплайну для сервісу покупки

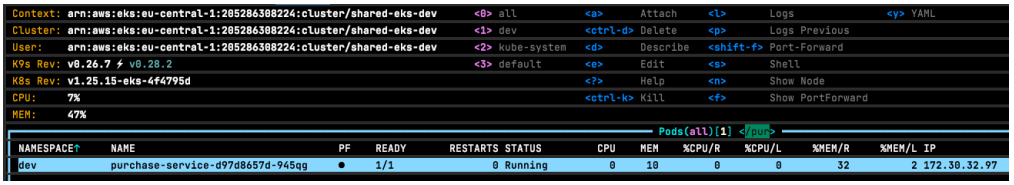

Рис.18. Спикок доступних подів на кластері

А тепер перевіримо саму роботу сервісу:

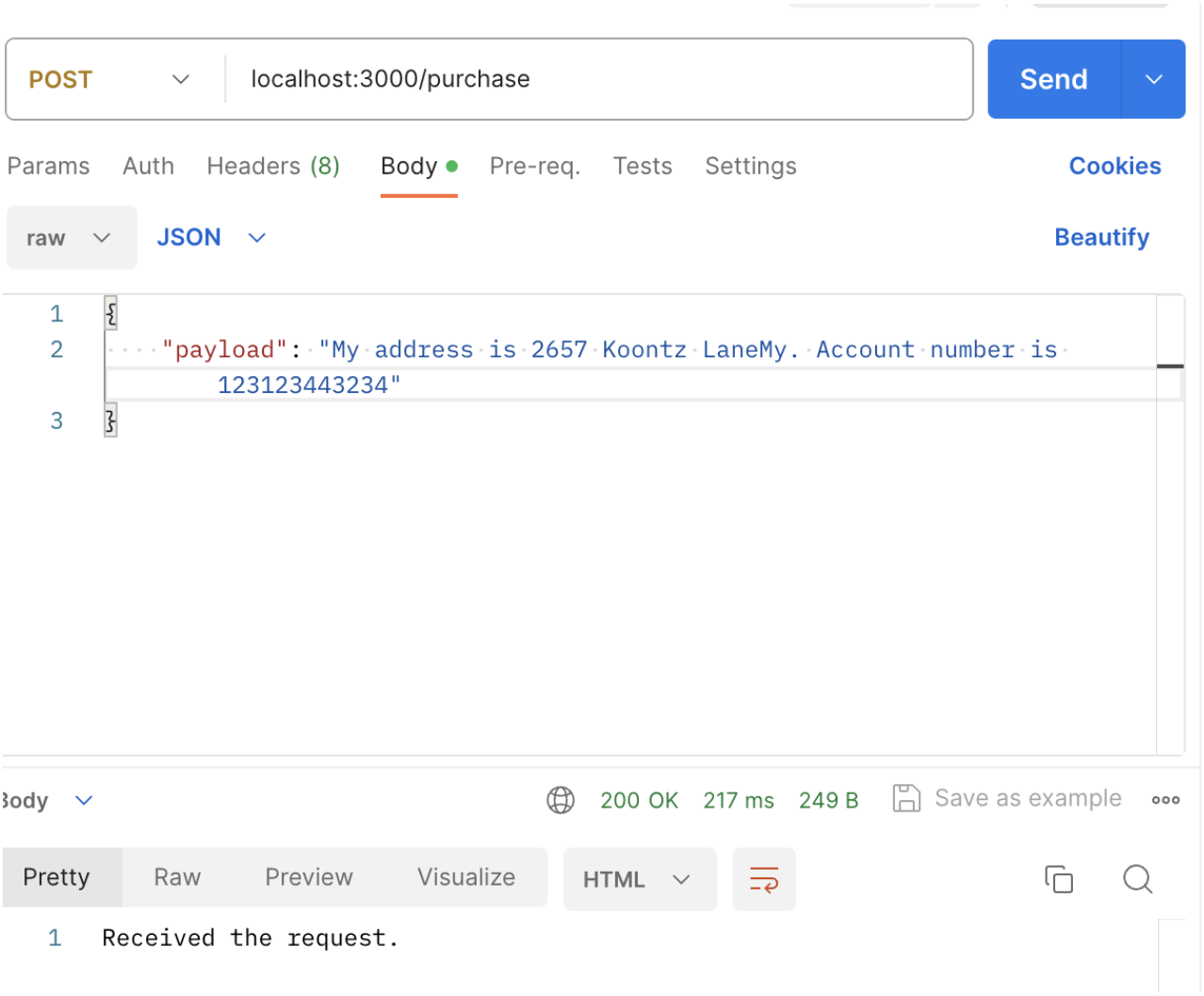

Рис.19. Успішно виконаний реквест до сервісу покупок

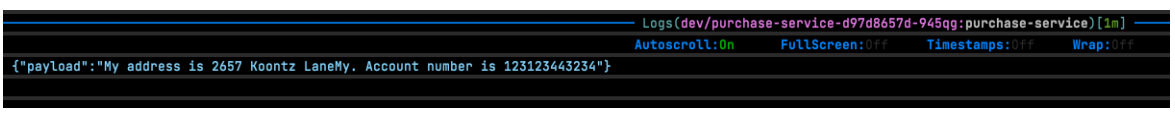

Рис.20. Логи сервісу покупок

Як бачимо, сервіс працює. В логах ми бачимо персональну інформацію, яку вподальшому будемо відправляти на лог сховище.

3.3. Elasticsearch розгортання

Elasticsearch як вже було описано з Розділу 2, це база даних для логів, а також це пошукова система по цим логам. Я збираюсь розгорнути Elasticsearch на тому ж кубернетес кластері що і всі інші компоненти. Єдиний недолік тут, це те що нам потрібен постійний диск для запису і зберігання логів. Так як відомо, що у випадку рестарту кубернетес пода, всі дані які є на цьому поді будуть знищені. Для вирішення цієї проблеми, нам допоможуть наступні компоненти: PersistentVolumeClaim і PersistentVolume.

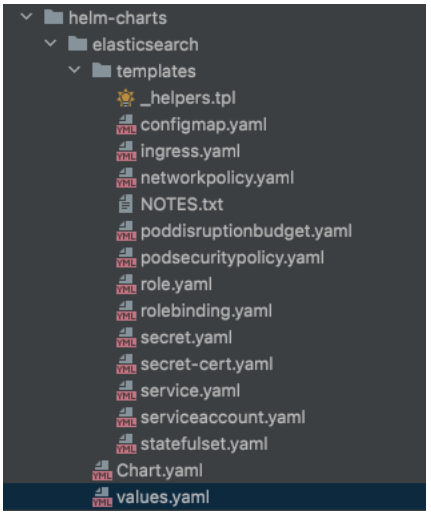

Рис.20. Хелм файли для розгортання elasticsearch

```yaml
name: Deploy elasticsearch to EKS

on:
  workflow_dispatch:
    inputs:
      env:
        description: 'Environment'
        required: true
        default: 'dev'
        type: choice
        options:
          - dev
          - prod
  push:
    paths:
      - 'helm-charts/elasticsearch/**'
    branches:
      - main
  pull_request:
    branches:
      - main


jobs:
  deploy_job:
    uses: ./.github/workflows/helm-base-pipeline.yaml
    with:
      service-name: elasticsearch
      environment: ${{ inputs.env == '' && 'dev' || inputs.env }}
    secrets: inherit
```

Рис.21. Імплементація пайплайну для розгортання elasticsearch

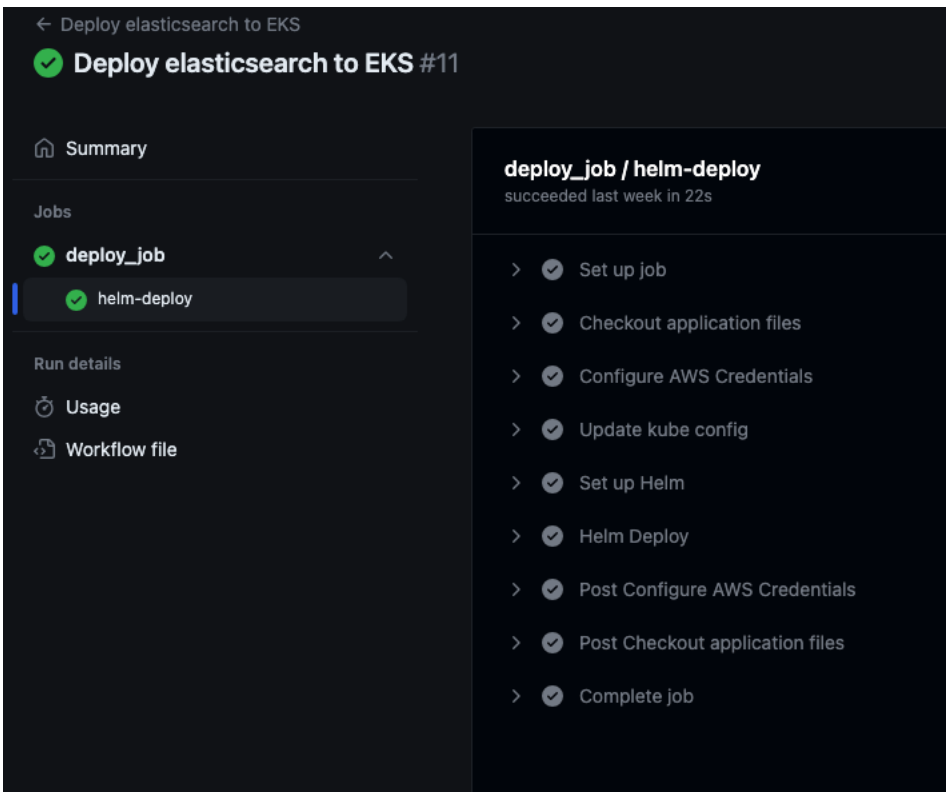

Рис.22. Результат завершеного пайплайну для elasticsearch

Рис.23. Elasticsearch у списку активних подів

3.4. Kibana розгортання

Kibana буде використовуватись для візуального аналізу і пошуків логів. Розгортання буде аналогічним до інших компонентів на кубернетес кластері. Для цього нам потрібні helm chart файли:

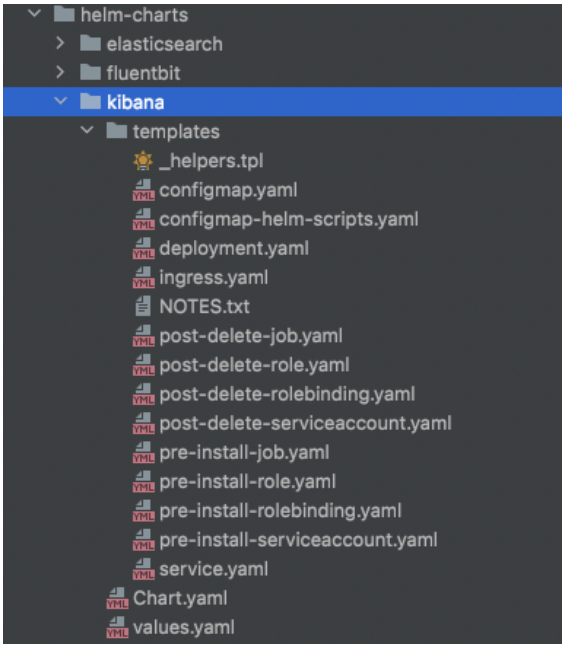

Рис.24. Хелм чарт файли для розгортання kibana

А також сам пайплайн, який ці всі конфігурації віднесе на кластер:

```yaml
on:
  workflow_dispatch:
    inputs:
      env:
        description: 'Environment'
        required: true
        default: 'dev'
        type: choice
        options:
          - dev
          - prod
  push:
    paths:
      - 'helm-charts/kibana/**'
    branches:
      - main
  pull_request:
    branches:
      - main


jobs:
  deploy_job:
    uses: ./.github/workflows/helm-base-pipeline.yaml
    with:
      service-name: kibana
      environment: ${{ inputs.env == '' && 'dev' || inputs.env }}
    secrets: inherit
```

Рис.25. Пайплай для розгортання kibana

Запускаємо пайплайн і перевіряємо наявність компонента на самому кластері:

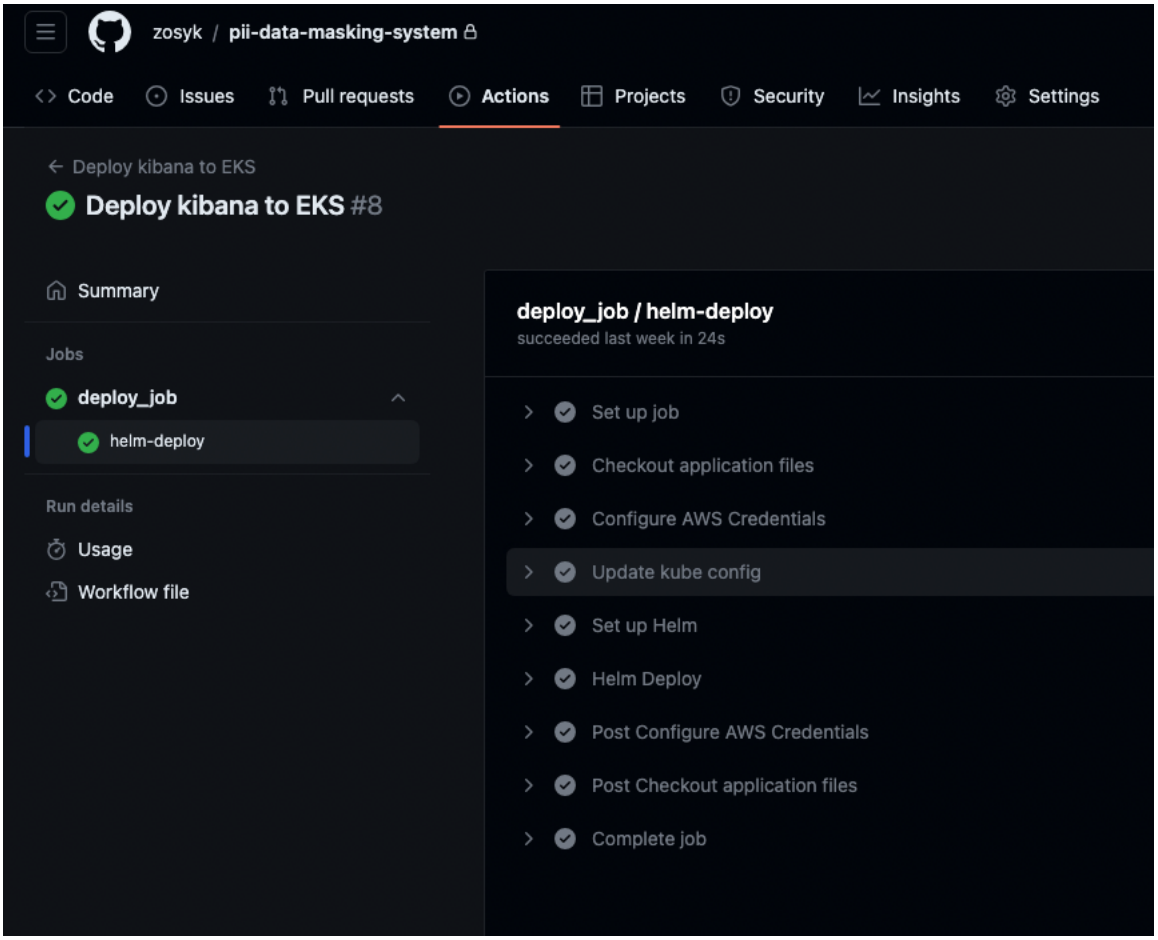

Рис.26. Пайплайн успішно розгорнув Kibana

Рис.26. Kibana серед запущених сервісів

Спробуємо відкрити веб сторінку:

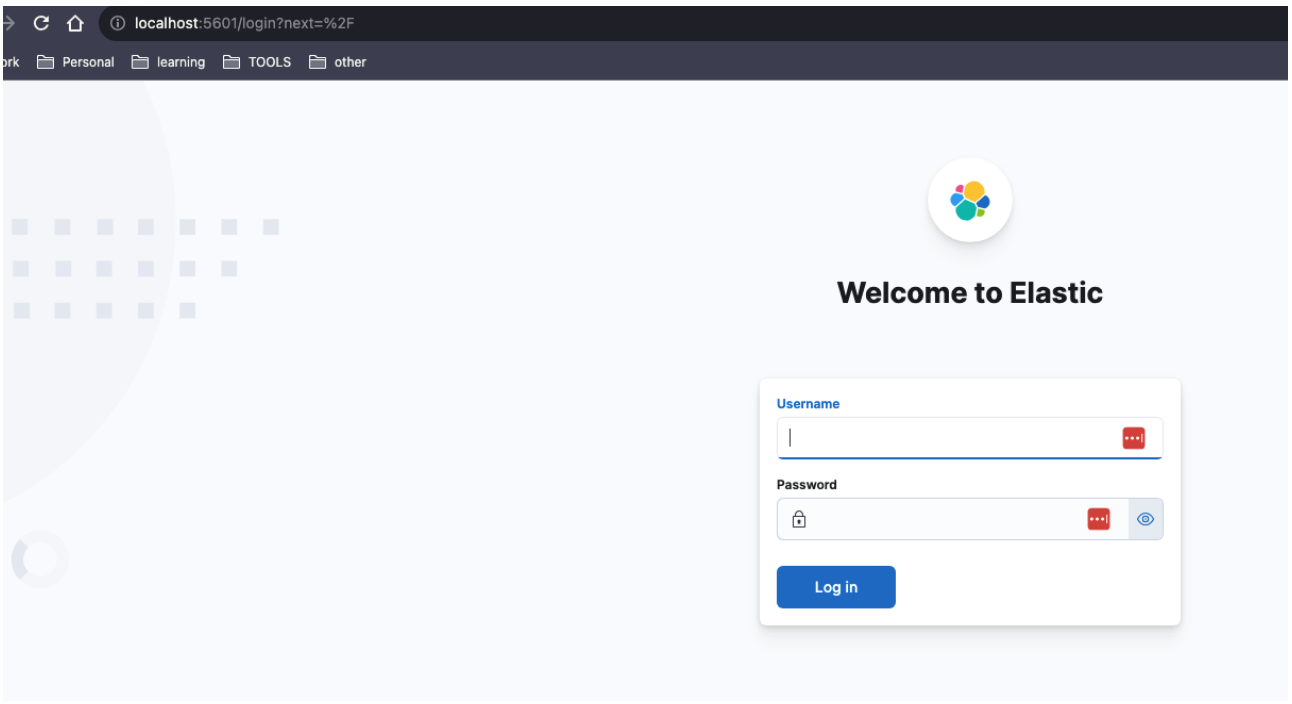

Рис.27. Сторінка логіну в сервіс kibana

3.5. Fluentbit розгортання

Розгортаємо fluentbit, використовуючи helm чарти та Github Actions пайплайн:

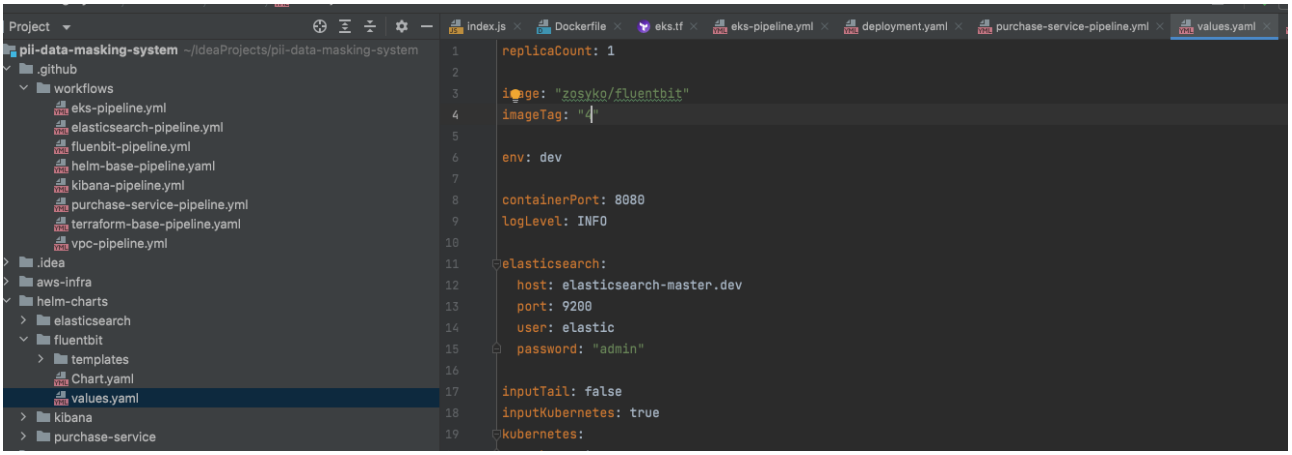

Рис.28. Список змінних для сервісу fluentbit

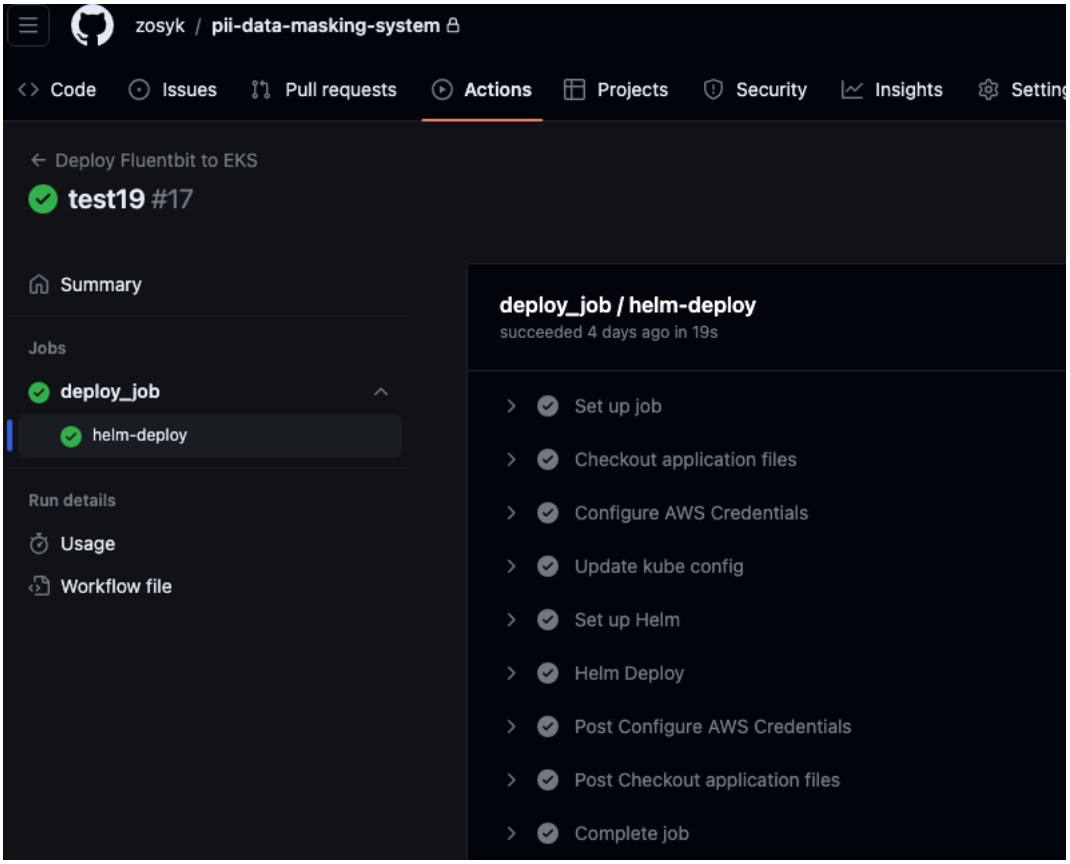

Рис.29. Успішно завершений пайплайн для розгортання fluentbit

Перевіряємо, що компонент вже працює на кластері:

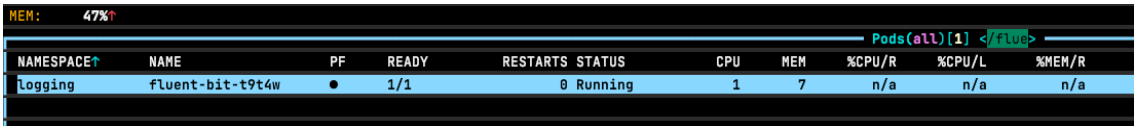

Рис.30. Fluentbit на кубернетес кластері

Розглянемо сам воркфлоу цього компонента:

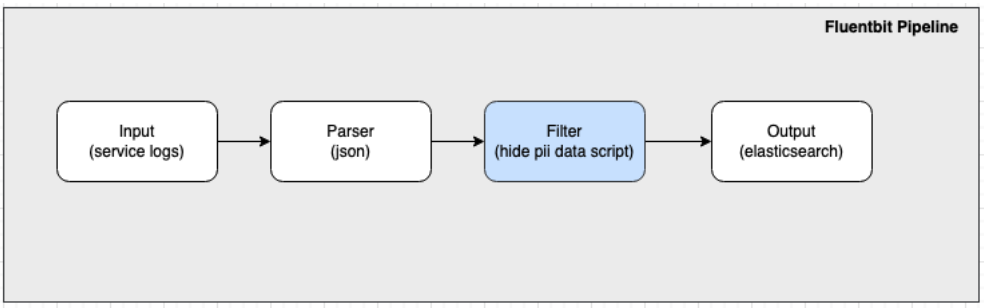

Рис.31. Fluentbit воркфлоу

Як видно з малюнку, fluentbit вичитує логи сервісу(за це відповідає input компонент), потім передає наступному компоненту вже самі текстові логи, парсер намагається розпарсити текст в json формат і потім передає далі, фільтер відповідає на зміну або доповнення логів. На цьому компоненті зупинимось більш детально. В цьому компоненті ми і будемо аналізувати логи на наявність персональний даних. Це ми будемо робити за допомогою AWS Comprehend сервісу.

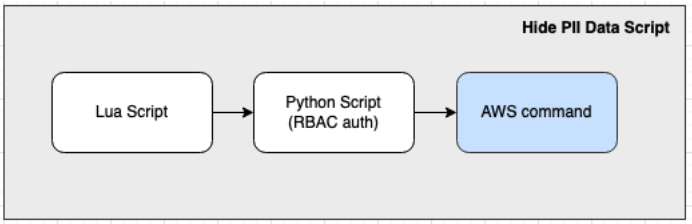

Рис.32. Алгоритм скрипта

Як видно з малюнку, для цього нам потрібен буде Lua скрипт, так як fluentbit може працювати тільки з цими скриптами, а також Python скрипт, який є більш багатим і комфортним для роботи, якщо роботи порівняння з Lua.

```
hide_pii.lua: |
  function hide_pii(tag, timestamp, record)
      local command = "python3 /fluent-bit/etc/call_comprehend.py --text " .. "\"" .. record['payload'] .. "\""

      os.execute(command)
      local handle = io.popen(command, "r")
      local result = handle:read("*a")  -- Read the command's entire output
      handle:close()

      record['payload'] = result

      return 1, timestamp, record
  end
```

Рис.33. Lua скрипт

```
call_comprehend.py: |
  import boto3
  import json
  import sys
  import os

  if "--text" not in sys.argv:
    print("text parameter is required!")
    exit(1)

  aws_region = 'eu-central-1'

  # Initialize the Comprehend client
  comprehend = boto3.client('comprehend', region_name=aws_region)

  # Detect PII entities in the text
  text_arg = sys.argv[sys.argv.index("--text") + 1]
  response = comprehend.detect_pii_entities(Text=text_arg, LanguageCode='en')

  # Print the JSON result
  entities = response.get('Entities', [])

  result = []
  for i, c in enumerate(text_arg):
    hidden = False
    for entity in entities:
      beginOffset = entity.get("BeginOffset")
      endOffset = entity.get("EndOffset")
      if i >= beginOffset and i < endOffset:
        result.append("*")
        hidden = True
    if not hidden:
      result.append(c)

  print(''.join(result))
```

Рис.34. Python скрипт

Як бачимо з самих скриптів, там немає і згадки про якусь авторизацію. Ми для цього будемо використовувати вже вбудувані механізми самого AWS, а саме RBAC.

Для цього нам потрібно додати відповідно анотацію на ServiceAccount кубернетес ресурс fluentbit компонента:

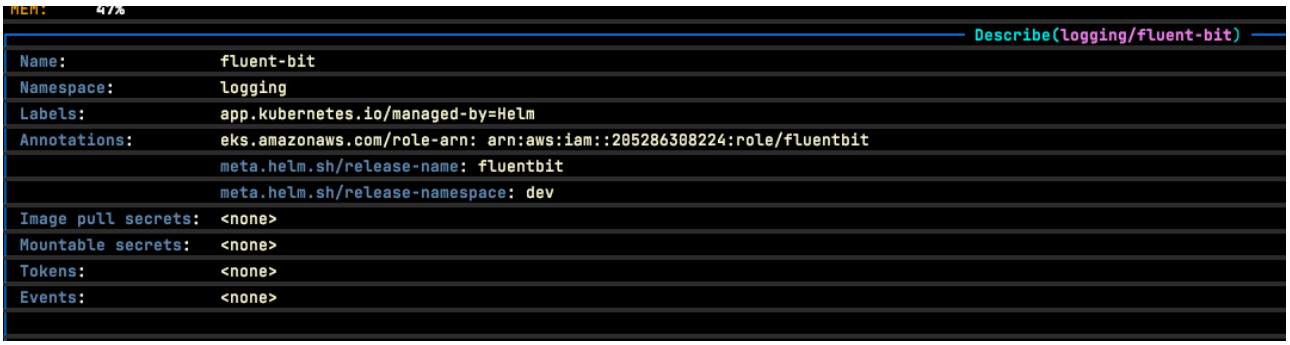

Рис.35. Список fluentbit SA анотацій

Де ми і вказуємо, яка роль буде використана для авторизації до AWS сервісів.

А коли ми зайдемо на сам под fluentbit компонента, то ми побачимо що кубернетес автоматично додав нам декілька змінних(AWS_ROLE_ARN & AWS_WEB_IDENTITY_TOKEN_FILE) в середовищі пода для подальшої авторизації:

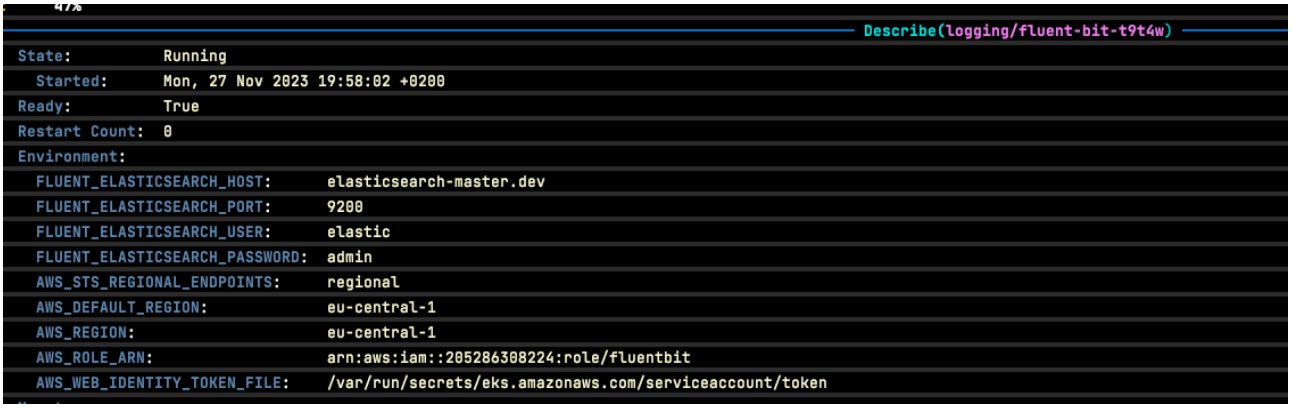

Рис.36. Список fluentbit змінних середовища

Єдине що нам залишилось, це створення самої ролі, де потрібно додати доступи до Comprehend сервісу

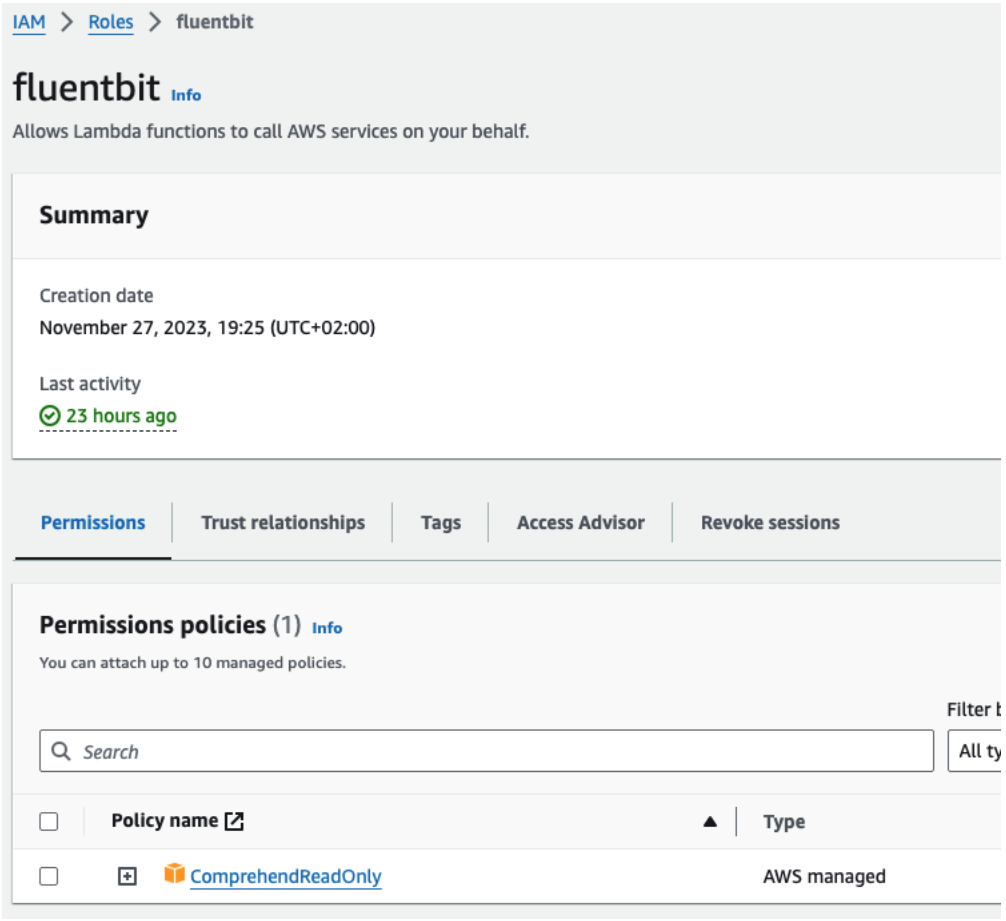

Рис.37. Fluentbit роль

Fluentbit був останнім і найважливішим компонентом, який ми розгорнули на кубернетес кластері. Відповідно, наступний рисунок покаже всі компоненти і їх взаємодію всередині кластера:

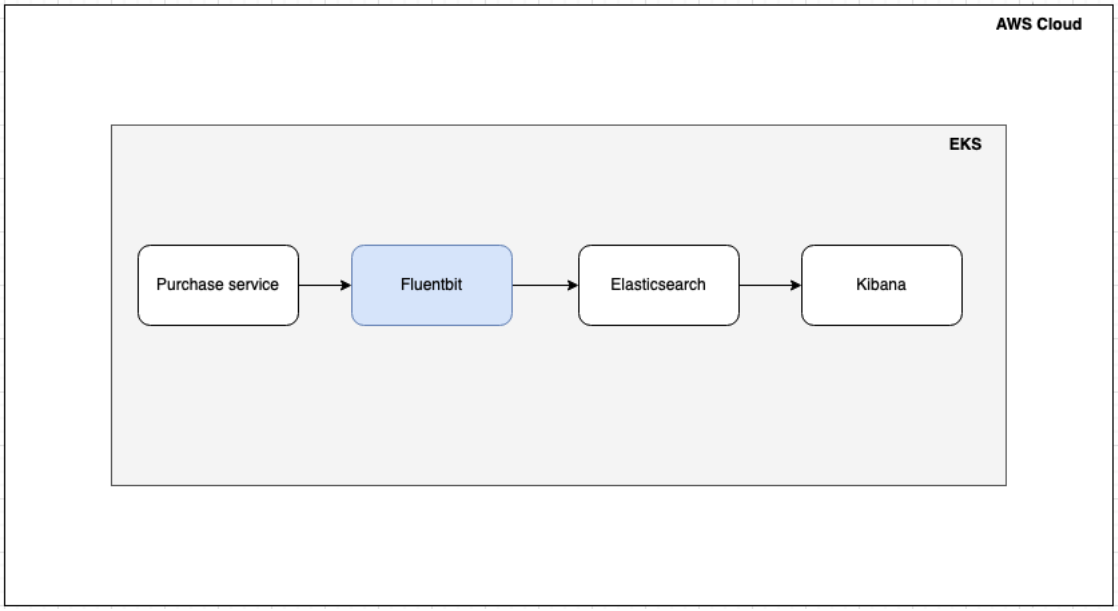

Рис.38. Всі компоненти K8S кластера

3.6. Тестування системи

Для початку зробимо декілька запитів через Postman:

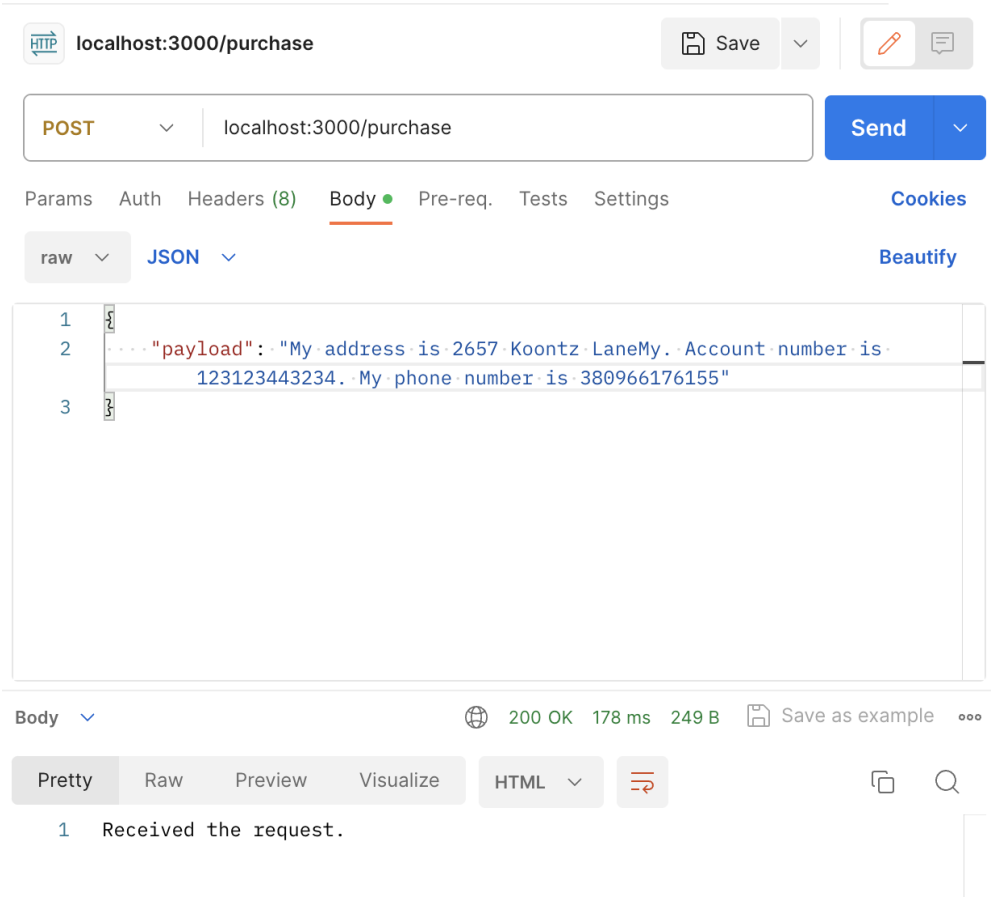

Рис.39. Перший запит до сервісу покупок

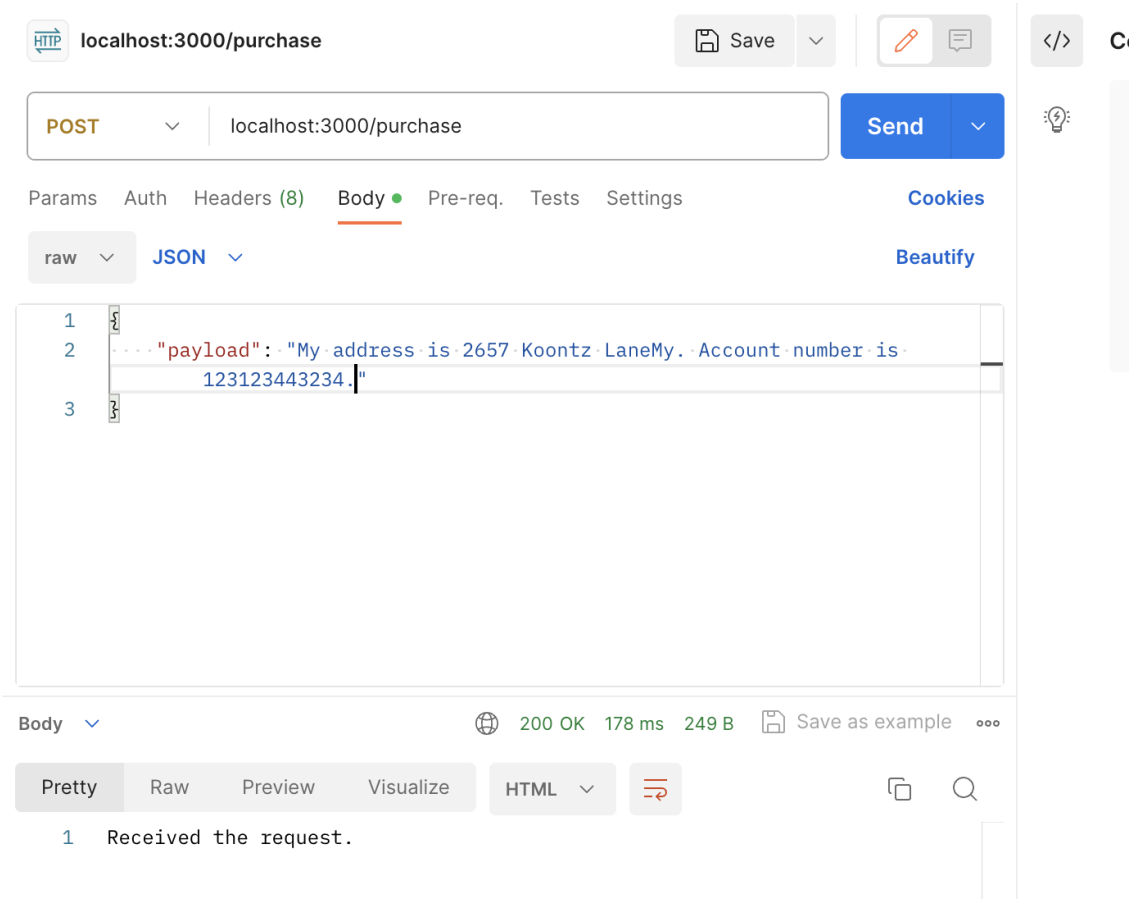

Рис.40. Другий запит до сервісу покупок

Тепер залогінимось на кібану і перевіримо які логи ми там бачимо:

Бачимо, що прийшло два лога:

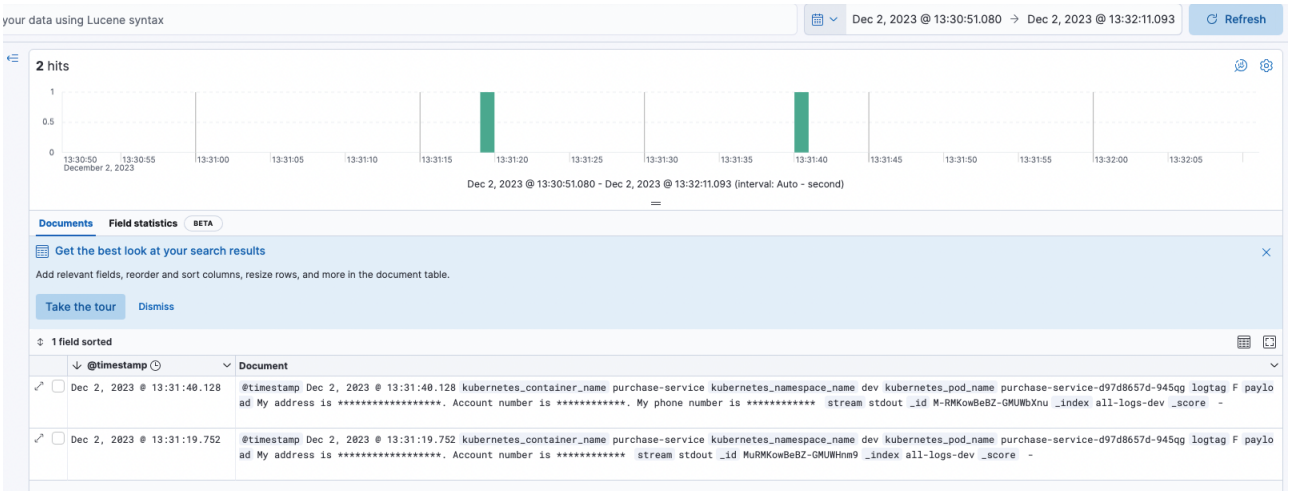

Рис.41. Кібана логи

І якщо ми відкриємо кожен із них для більш детальної інформації, то побачимо що персональні дані були замасковані під зірочки:

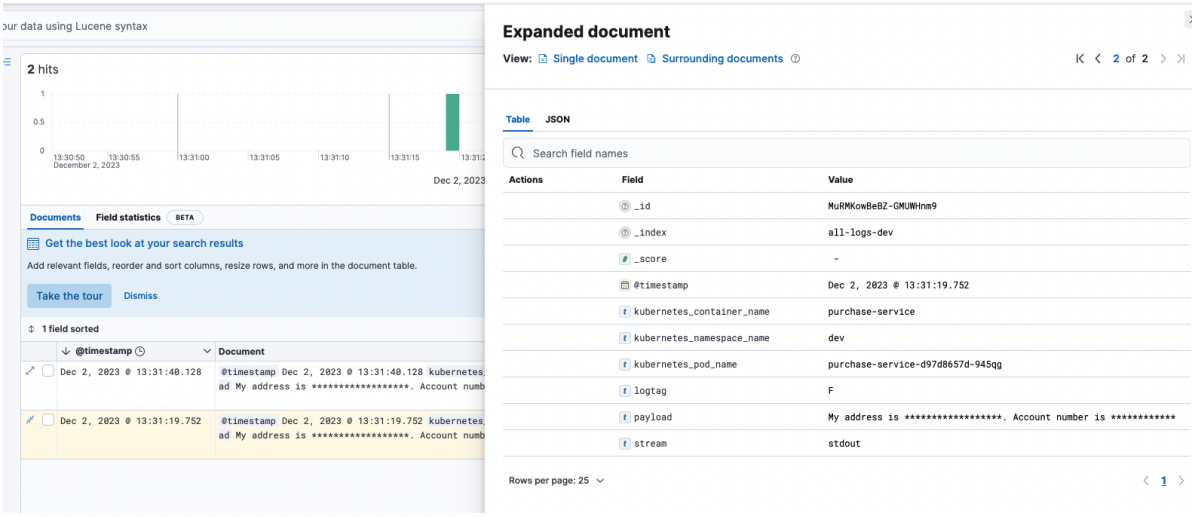

Рис.41. Детальна інформація першого лога

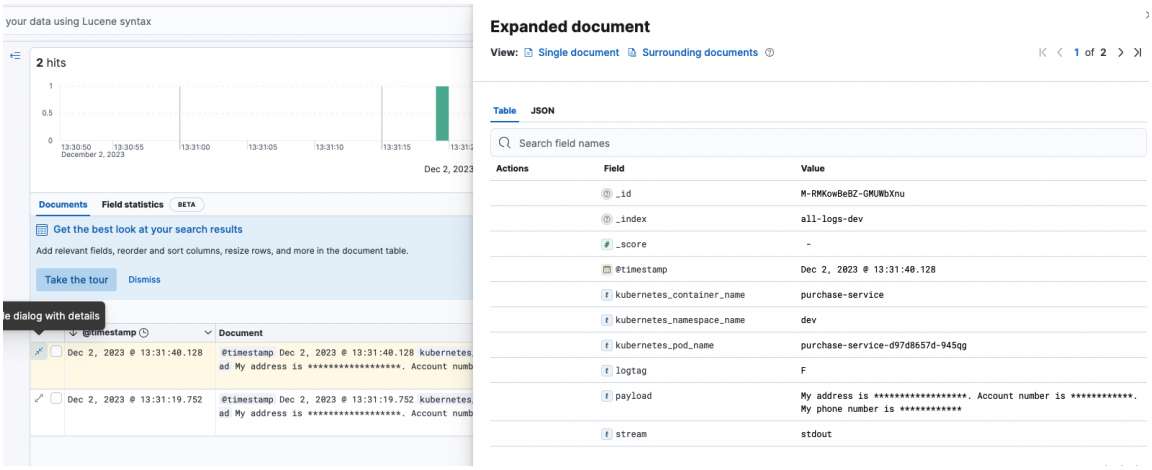

Рис.42. Детальна інформація другого лога

Висновок

У третьому розділі було розгорнуто кубернетес кластер, а також необхідні компоненти для створення системи по маскуванню персональних даних. Було проаналізовано і протестовано можливості fluenbit компонента, а саме як за допомогою його фільтра і скриптів аналізувати і маскувати в подальшому персональні дані. Під час тестування було доведено що ця система працездатна і безпечна, так як не містить жодних приватних ключів або токенів у відкритому вигляді. Також було використано принцип IoC – infrastructure as a code, опис

всієї бажаної інфраструктури було розміщено на Github, після чого за допомогою пайпланів було розгорнуто в самій хмарній інфраструктурі AWS. Тому кожен хто має доступ до цього коду, зможе відтворити систему на своєму акаунті майже автоматично.

РОЗДІЛ 4
ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Екологічні фактори - всі складові (елементи) природного середовища, які впливають на існування й розвиток організмів і на які живі істоти реагують реакціями пристосування (за межами здатності пристосування настає смерть)

Раніше виділяли три групи екологічних факторів:

абіотичні (неорганічні умови: хімічні й фізичні, такі, як склад повітря, води, ґрунтів, температура, світло, вологість, радіація, тиск тощо), біотичні (форми взаємодії між організмами - хазяїн - паразит) та антропогенні (форми діяльності людини).

Сьогодні розрізняють декілька груп екологічних факторів (загальна кількість - близько шістдесяти), об'єднаних у спеціальну класифікацію:

- за часом - фактори часу (еволюційний, історичний, діючий), періодичності (періодичний і неперіодичний), первинні та вторинні;
- за походженням (космічні, абіотичні, природноантропогенні, техногенні, антропогенні);
- за середовищем виникнення (атмосферні, водні, геоморфологічні, фізіологічні, генетичні, екосистемні);
- за характером (інформаційні, фізичні, хімічні, енергетичні, термічні, біогенні, комплексні, кліматичні);
- за об'єктом впливу (індивідуальні, групові, видові, соціальні);
- за ступенем впливу (летальні, екстремальні, обмежуючі, мутагенні, тератогенні);
- за умовами дії (залежні чи незалежні від щільності);
- за спектром впливу (вибіркової чи загальної дії).

Одній й ті ж екологічні фактори неоднаково впливають на організми різних видів, які живуть разом. Для деяких вони можуть бути сприятливими, для інших - ні. Важливим елементом є реакція організмів на силу впливу екологічного фактора, негативна дія якого може виникати у разі надлишку або

нестачі дози. Тому є поняття сприятлива доза, або зона оптимуму фактора й зона песимуму (доза фактора, за якої організми почуваються пригнічено).

Діапазон зон оптимуму й песимуму є критеріями для визначення екологічної валентності - здатності живого організму пристосовуватися до змін умов середовища. Кількісно вона виражається діапазоном середовища, в межах якого вид нормально існує. Екологічна валентність різних видів відрізняється одна від одної (північний олень витримує коливання температури повітря від -55 до 25-30оС, а тропічні корали гинуть вже при зміні температури на 5-6оС).

За екологічною валентністю організми поділяють на: стенобіоти - з малою пристосованістю до змін середовища (орхідеї, форель, далекосхідний рябчик, глибоководні риби) та еврибіонти - з великою пристосованістю до змін довкілля (колорадський жук, миші, пацюки, вовки, таргани, очерет, пирій). У межах еврибіонтів і стенобіонтів залежно від конкретного фактора організми поділяють на евритермні та стенотермні (за реакцією на температуру), евригалінні й стеногалінні (за реакцією на солоність водного середовища), еврифоти та стенофоти (за реакцією на освітлення).

Слід наголосити, що в природі екологічні фактори діють комплексно. Особливо важливо пам'ятати це, оцінюючи вплив хімічних забруднювачів, коли "сумаційний" ефект (на негативну дію однієї речовини накладається негативна дія інших, до чого додається вплив стресової ситуації, шумів, різних фізичних полів - радіаційного, теплового, гравітаційного чи електромагнітного) дуже змінює умовні значення ГДК, наведені в довідниках. Це питання на сьогодні ще мало вивчене, але через актуальність і велике значення перебуває в стані активного дослідження в усіх розвинених країнах.

Важливим є також поняття лімітуючи фактори, тобто такі, рівень (доза) яких наближається до межі витривалості організму, концентрація якого нижча або вища оптимальної. Це поняття започатковане законами мінімуму Лібіха (1840 р.) і толерантності Шелфорда (1913 р.). Найчастіше лімітуючи ми

факторами є температура, світло, біогенні речовини, течії та тиск у середовищі, пожежі тощо.

Найбільше поширені організми з широким діапазоном толерантності щодо всіх екологічних факторів. Найвища толерантність характерна для бактерій і синьо-зелених водоростей, які виживають у широкому діапазоні температур, радіації, солоності, pH.

Екологічні дослідження, пов'язані з вивченням впливу екологічних факторів на існування й розвиток окремих видів організмів. взаємозв'язків з довкіллям, е предметом науки аутекології.

Розділ біоекології, що вивчає умови формування структури й динаміки популяцій якогось виду, називається демекологією, а розділ, який досліджує асоціації популяцій різних видів рослин, тварин, мікроорганізмів (біоценозів), шляхи їх формування й взаємодії з довкіллям, - синекологією. У межах синекології виділяють фітоценологію, або геоботаніку (об'єкт вивчення - угруповання рослин), біоценологію (угруповання тварин).

Наступним важливим поняттям є ланцюг живлення (трофічний ланцюг) - це взаємовідносини між організмами під час перенесення енергії їжі від її джерела (зеленої рослини) через ряд організмів (шляхом поїдання) на більш високі трофічні рівні. На цьому шляху перенесення діють автотрофи - представники рослинного світу та гетеротрофи різного ступеня. Спинимося на цьому понятті детальніше.

Висновок

У четвертому розділі були розглянуті екологічні фактори як складові природного середовища. Були проаналізовані три групи екологічних факторів, їх особливості та відмінності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Маскування даних у хмарних обчисленнях [Електронний ресурс]. - Режим доступу: https://www.academia.edu/41858074/Data_Masking_in_Cloud_Computing
2. Маскування даних для захисту конфіденційності даних [Електронний ресурс]. - Режим доступу: https://ieeexplore.ieee.org/document/6699820
3. Техніка безпечного маскування даних за допомогою алгоритмів шифрування [Електронний ресурс]. - Режим доступу: https://www.researchgate.net/publication/282147206_Secure_Data_Masking_Technique_using_Encryption_Algorithms
4. Підхід до маскування даних [Електронний ресурс]. - Режим доступу: https://www.researchgate.net/publication/305152501_An_Approach_to_Data_Masking
5. Маскування конфіденційних журналів за допомогою k-Anonymity [Електронний ресурс]. - Режим доступу: https://dl.acm.org/doi/10.1145/1501750.1501758
6. Fluent Bit [Електронний ресурс]. - Режим доступу: https://docs.fluentbit.io
7. Amazon Comprehend - Natural Language Processing (NLP) [Електронний ресурс]. - Режим доступу: https://aws.amazon.com/comprehend/
8. Архітектура логування Kubernetes [Електронний ресурс]. - Режим доступу: https://kubernetes.io/docs/concepts/cluster-administration/logging/
9. Kubernetes логування за допомогою Fluentd та Elasticsearch [Електронний ресурс]. - Режим доступу: https://docs.fluentd.org/v/1.0/architecture/kubernetes-overview
10. Elasticsearch for Kubernetes [Електронний ресурс]. - Режим доступу: https://www.elastic.co/guide/en/cloud-on-k8s/current/index.html
11. Introduction to AWS Comprehend [Електронний ресурс]. - Режим доступу: https://aws.amazon.com/comprehend

12. AWS Comprehend Medical: Аналіз медичного тексту на основі машинного навчання AWS [Електронний ресурс]. - Режим доступу: https://aws.amazon.com/comprehend/medical
13. Найкращі методи безпеки Kubernetes [Електронний ресурс]. - Режим доступу: https://kubernetes.io/docs/concepts/security/overview
14. Створення масштабованих і безпечних мультитенантних кластерів Kubernetes [Електронний ресурс]. - Режим доступу: https://www.youtube.com/watch?v=53Sj-DoI_v0
15. Аналіз тексту із збереженням конфіденційності за допомогою Amazon Comprehend [Електронний ресурс]. - Режим доступу: https://aws.amazon.com/blogs/machine-learning/privacy-preserving-text-analysis-with-amazon-comprehend/
16. Розширене логування Kubernetes із Fluent Bit [Електронний ресурс]. - Режим доступу: https://fluentbit.io/blog/advanced-kubernetes-logging-fluentbit
17. Знаходження персональної інформації за допомогою AWS Comprehend [Електронний ресурс]. - Режим доступу: https://docs.aws.amazon.com/comprehend/latest/dg/how-pii.html
18. Амазон хмарні сервіси [Електронний ресурс]. - Режим доступу: https://media.amazonwebservices.com/AWS_Overview.pdf

ДОДАТОК А

ТЕКСТ ПРОГРАМИ

https://github.com/zosyk/pii-data-masking-system