ENHANCING EMAIL DATA CLEANSING ALGORITHM AND DEVELOPING MONITOR SERVICE OF APPLICATION RESOURCES

MR. NAPAS VINITNANTHARAT
MR. SUPAWUT CHANGWANGPRANG

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI
2023

Enhancing Email Data Cleansing Algorithm
and Developing Monitor Service of Application Resources

Mr. Napas Vinitnantharat
Mr. Supawut Changwangprang

A Project Submitted in Partial Fulfillment
of the Requirements for
the Degree of Bachelor of Engineering (Computer Engineering)
Faculty of Engineering
King Mongkut's University of Technology Thonburi
2023

Project Committee

..... Project Advisor
(Asst.Prof. Rajchawit Sarochawikasit)

..... Committee Member
(Asst.Prof. Dr.-Ing Priyakorn Pusawiro)

..... Committee Member
(Unchalisa Taetragool, Ph.D.)

Digitally signed by 1b075b15-feba-4ade-a557-b847270e10e7
Date: 2024.07.08 13:44:12+07'00'

..... Committee Member
(Assoc.Prof. Thumrongrat Amornraksa, Ph.D.)

Project Title	Enhancing Email Data Cleansing Algorithm and Developing Monitor Service of Application Resources
Credits	6
Member(s)	Mr. Napas Vinitnantharat Mr. Supawut Changwangprang
Project Advisor	Asst.Prof. Rajchawit Sarochawikasit
Program	Bachelor of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2023

Abstract

This research project consists of two main components aimed at enhancing the functionality and security of an Email Data Cleansing model using advanced Natural Language Processing (NLP) techniques, as well as refining the monitoring dashboard for application resources. In the first component, significant improvements are made to the Email Data Cleansing model to effectively eliminate irrelevant text or verbal content from emails, ensuring a focused analysis on the main subject matter. These enhancements also contribute to strengthening the overall security of the system. The second component focuses on enhancing the monitoring dashboard for application resources by incorporating essential metrics and implementing a robust alert notification system. This alert system is designed to promptly notify users of critical scenarios, such as the detection of error logs, instances of downtime, or any other operational errors that may impact the system's integrity. Additionally, this research project includes the implementation of an automated pod scaling feature based on kafka consumer lag events using Kubernetes Event Driven Autoscaling (KEDA). This optimization aims to efficiently allocate application resources and evaluate application performance through load testing. Overall, this research project aims to advance the field of Email Data Cleansing and application resource monitoring, contributing to the development of more efficient and secure systems.

Keywords: Natural Language Processing (NLP) / Sentiment Analysis / Observability / Kubernetes Event Driven Autoscaling (KEDA) / auto scaling pod / load testing

หัวข้อปริญญานิพนธ์	การเพิ่มประสิทธิภาพของอัลกอริทึมที่ลบอีเมล และการจัดสรรตรวจสอบทรัพยากรของแอปพลิเคชัน
หน่วยกิต	6
ผู้เขียน	นาย นภัส วินิจนันทรัตน์ นาย ศุภาวุฒิ ฉางวางปราง
อาจารย์ที่ปรึกษา	ผศ.ราชวิชช์ สโรชวิกสิต
หลักสูตร	วิศวกรรมศาสตรบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์
ปีการศึกษา	2566

บทคัดย่อ

โครงการวิจัยนี้ประกอบด้วยส่วนหลักสองส่วนที่มุ่งเน้นการเพิ่มประสิทธิภาพและความปลอดภัยของโมเดล Email Data Cleansing โดยใช้เทคนิคการประมวลผลภาษาธรรมชาติ (NLP) ขั้นสูง รวมถึงการปรับปรุงแดชบอร์ดการตรวจสอบทรัพยากรแอปพลิเคชัน ในส่วนแรก มีการปรับปรุงโมเดล Email Data Cleansing อย่างมีนัยสำคัญเพื่อลบข้อความหรือเนื้อหาที่ไม่เกี่ยวข้องออกจากอีเมล ทำให้การวิเคราะห์เน้นไปที่เรื่องหลัก การปรับปรุงเหล่านี้ยังมีส่วนช่วยเสริมความปลอดภัยของระบบโดยรวม ส่วนที่สองเน้นการปรับปรุงแดชบอร์ดการตรวจสอบทรัพยากรแอปพลิเคชัน โดยรวมเมตริกสำคัญและการนำระบบการแจ้งเตือนที่แข็งแกร่งเข้ามาใช้งาน ระบบการแจ้งเตือนนี้ถูกออกแบบเพื่อแจ้งให้ผู้ใช้ทราบทันทีเมื่อเกิดสถานการณ์สำคัญ เช่น การตรวจพบบันทึกข้อผิดพลาด การดับเพลิง หรือข้อผิดพลาดใดๆ ที่อาจส่งผลกระทบต่อความสมบูรณ์ของระบบ นอกจากนี้ โครงการวิจัยนี้รวมถึงการนำเสนอคุณลักษณะการปรับปรุงขนาด pod อัตโนมัติโดยใช้เหตุการณ์ kafka consumer lag โดยใช้ Kubernetes Event Driven Autoscaling (KEDA) การปรับปรุงนี้มุ่งเน้นการจัดสรรทรัพยากรแอปพลิเคชันอย่างมีประสิทธิภาพและประเมินประสิทธิภาพของแอปพลิเคชันผ่านการทดสอบโหลด

คำสำคัญ: การประมวลผลภาษาธรรมชาติ / การวิเคราะห์ความรู้สึก / ความสามารถในการสังเกตการณ์ / Kubernetes Event Driven Autoscaling (KEDA) / การปรับปรุงทรัพยากรของแอพพลิเคชั่น / การทดสอบโหลด

ACKNOWLEDGMENTS

Firstly, we would like to express my sincere gratitude to my advisor, Asst.Prof. Rajchawit Sarochawikasit, Ph.D. for his invaluable guidance, and steadfast support throughout the duration of this project. His guidance and feedback have been instrumental in shaping the direction of this project.

Secondly, Thank you to Mr. Nathaphop Sundarabhogin and our esteemed colleagues at ExxonMobil for their generous support and insightful guidance throughout the project lifecycle. Their expertise and collaborative efforts helped us navigate through the complexities of the project and achieve our objectives.

Furthermore, we express our sincere appreciation to ExxonMobil for entrusting us with the opportunity to embark on this project and for equipping us with the essential resources needed to realize its objectives.

Finally, we would like to extend our heartfelt gratitude to our family and friends for their unwavering support and encouragement throughout this project.

CONTENTS

PAGE

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

LIST OF TECHNICAL VOCABULARY AND ABBREVATIONS

AI	=	Artificial Intelligence
APM	=	Application Performance Monitoring
ARO	=	Azure Red Hat OpenShift
AOD	=	Automated Offer Deployment
CI/CD	=	Continuous Integration/Continuous Delivery
CFM	=	Customer Feedback Management
CPC	=	Customer Product Catalog
CSLO	=	Customer Service Level Offering
DNN	=	Deep Neural Network
HPA	=	Horizontal Pod Autoscaling
k8s	=	Kubernetes
KEDA	=	Kubernetes Event Driven Autoscaling
LSTM	=	Long Short-Term Memory
NER	=	Named Entity Recognition
NLP	=	Natural Language Processing
POC	=	Proof of Concept
REGEX	=	Regular Expression
RNN	=	Recurrent Neural Network
RU	=	Request Unit
RUM	=	Real User Monitoring
SMT	=	Service Management Tools

CHAPTER 1 INTRODUCTION

1.1 Problem Statement, Motivation, and Potential Benefits

Nowadays, customers buy products from ExxonMobil such as oil, lubricant or petrochemical products from the company there might be some problems related to the process of purchasing the product such as delay delivery or quality of product doesn't match the expectation of the customer. In case customers have those problems, they open a request ticket to the customer service team through Zendesk service. Nowadays, there are more than 65,000 tickets per month. Each of these tickets encompasses a substantial volume of associated emails. Consequently, the business user operatives are confronted with a substantial workload as they endeavor to comprehend customer concerns. To address the aforementioned challenges. The company has undertaken the initiative to develop specialized software tailored to address these issues. The proposed software solution involves the automated ingestion and aggregation of case tickets sourced from Zendesk. This software subsequently processes the acquired tickets and associated emails, employing data-cleansing methodologies to ensure their alignment with the intended format. The data cleansing procedure encompasses the elimination of extraneous textual or verbal content from the emails that may be unrelated to the central context. This process is intended to facilitate sentiment analysis and the extraction of pertinent customer requirements. The strategic approach considered by the customer service team for text and speech cleansing entails a diverse array of methodologies such as a Fuzzy matching algorithm, Nature Language Processing(NLP), Long short term memory (LSTM), and regular expression (Regex). One of the challenges that the company is facing now is the machine learning model that is used to remove unnecessary text or speech hasn't been updated for a long time. So the process of removing unnecessary text is not efficient and there are issues related to security that must be improved. Given the aforementioned challenges, the Company recognizes the urgency to implement measures that ensure the robustness and accuracy of the data cleansing procedures. This entails revisiting and updating the machine learning models and methodologies to encompass contemporary advancements. Simultaneously, heightened security information throughout the data processing lifecycle. The company aims to optimize its operational efficiency, augment customer satisfaction, and fortify its position within the industry. In the present day, the customer service IT team employs Kubernetes (k8s) as the infrastructure for deploying applications through a CI/CD pipeline. Among the challenges encountered by software developers within the team is the difficulty in effectively tracing and investigating errors occurring in applications within pods or databases. In some of the cases, software developers have taken 2-3 days to find where the problem is coming from. To tackle this concern, the company has adopted Datadog as a remedy. Datadog is a monitoring and analytics platform that provides comprehensive visibility into the performance and health of various components of an IT infrastructure. In the present scenario, the team has identified several issues with the monitoring tool in use. One notable problem is the challenge of effectively visualizing logs and errors from Datadog, indicating a need for improvements in this area. Additionally, the team has expressed a requirement for additional monitoring capabilities such as creating a dashboard for monitoring pod utilization, database usage, Redis usage, Kafka metric, etc. This will help the team visualize the resources optimize their usage and notify the team when the application detects any issues in real time. Another problem is that the application that the customer service IT team is responsible for needs humans to monitor and manage pod resources manually. So when the application has high traffic, the team needs to scale up the pod manually and when the traffic is low, the team needs to scale down the pod manually. This process is time-consuming and efficient. To address these challenges, we aim to develop a feature that will automatically manage pod resources depending on resource usage. This feature will help the team optimize pod resources. This project also aims

to test the performance of the application by using K6 as a load-testing tool. This will help the team to understand the performance of the application and make sure that the application can handle the traffic when the application is in production.

1.1.1 Objectives

1. Develop techniques to ensure consistent formatting and alignment of data from different sources,enhancing the effectiveness of subsequent processing.
2. Evaluate and optimize the methodologies used for data cleansing, including Fuzzy string matching, LSTM and Regex, to achieve higher accuracy and eliminate unnecessary content more effectively.
3. To enhance security to Email Data Cleansing service. By update related development tool to newer version.
4. To enhance way of manage pod resources automatically depends on percentage of resources usage.
5. To enhance monitoring service of application resources such as pod utilization, database usage, redis usage, Kafka metric and etc.
6. To enhance a system to notify the team when the application detects any issues in real time.
7. Implement auto scaling pod feature to help team optimize Resources
8. Implement load testing tool to test performance of the application.

1.1.2 Scope

This project has two distinct phases. The first phase is to improve the Email Data Cleansing model using NLP and associated techniques. This enhancement aims to enable the model to more effectively remove any irrelevant text or verbal content from emails, which might not be pertinent to the main subject and also more security to the system. This process is crucial for subsequent sentiment analysis. The second phase focuses on refining the dashboard used for monitoring essential metrics within Kubernetes resources and related services. Additionally, a key aspect of this phase involves implementing a system for generating alert notifications to the team. These alerts will be triggered in scenarios such as the detection of error logs, instances of downtime, or any other errors that are pertinent to the system's operation This phase also includes the development of a feature that will automatically manage pod resources based on the percentage of resource usage and a load-testing tool to test the performance of the application.

1.2 Tasks Breakdown and Schedule

Tasks	August				September				October				November				December			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Email Cleansing																				
Cleanup data - separate normal comments and footer	■	■	■	■	■															
Create Training new model					■	■	■	■	■											
Restructure code and version										■	■									
Testing new model - validated by business												■								
Learning Overall Team Tech Stack																				
Study Spring Boot Framework/ Angular.js Framework														■	■					
Study Docker/Kubernetes/Azure Red Hat OpenShift(ARO)																■	■			
Study DataDog																	■	■	■	
Research Solutions (New web app or Datadog or Tableau)																			■	■

Table 1.1: First semeter task schedule table

Tasks	January				February				March				April				May			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Create new Dashboard																				
Create new Dashboard monitor pod utilization	■	■																		
feature monitor Database Usages		■	■																	
feature monitor Redis Usages			■	■																
Kafka metrics in Dashboard																				
Total message per topic metrics					■															
Broker message in/out metrics					■															
Consume Rate per topic metrics						■														
Producing Rate Topic metrics						■														
Create a Dashboard Alert																				
Error logs patterns Alert							■	■												
Pod downtime Alert							■	■												
resource leak from database or redis alert								■	■											
number failed rate to email cleansing more than threshold								■	■											
Autoscaling Pod																				
Autoscaling Pod when pod utilization is more than threshold											■	■								
Datadog Real User Monitoring																				
Create Real User Monitoring Frontend Performance insight												■	■							
Custom Metric Prometheus																				
number of fail request vs total request of email cleansing service														■	■	■				
Load Test Performance when deploying																				
create k6 load testing when deploying k8s																		■	■	■

Table 1.2: Second semeter task schedule table

CHAPTER 2 BACKGROUND, THEORY AND RELATED RESEARCH

In this chapter, the topic of background knowledge and necessary technology will be discussed. Each tool will go into detail about how they benefit or are used during this project development as well as the information about the projects that are related to Email Data Cleansing model and monitoring service of kubernetes resources.

2.1 Algorithm

This section will directly focus algorithms that will be used throughout this project on the Email Data Cleansing part. These algorithms are used to identify disclaimer sentences and normal sentences, after the algorithm identifies the disclaimer sentence. The algorithm will eliminate the sentence.

2.1.1 Deep Neural Network (DNN)

A Deep Neural Network (DNN) is an iteration of Artificial Neural Networks (ANN) that incorporates multiple hidden layers between the input and output layers. These networks draw inspiration from the human brain and its intricate functioning, mirroring aspects of the nervous system. In traditional neural networks, there are only two layers of neurons the input layer and the output layer. DNN shave multiple hidden layers between the input and output layers [1]. Each layer contains a given number of units(neurons) that apply a certain functional transformation to the input and pass the output to the next layer. These types of models can approxiamte complex non-linear functions and are capable of learning abstract representations of the data. The application of deep neural networks spans diverse domains such as computer vision, speech recognition and natural language processing.

2.1.2 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) is a type of neural network that is designed to handle sequential data such as time series, speech, and text. The previous outputs of the network are used as inputs for the current step, allowing the network to maintain a form of memory or context. This makes RNNs particularly well-suited for tasks that involve sequential or time-series data, such as natural language processing, speech recognition, music generation, and more [2].

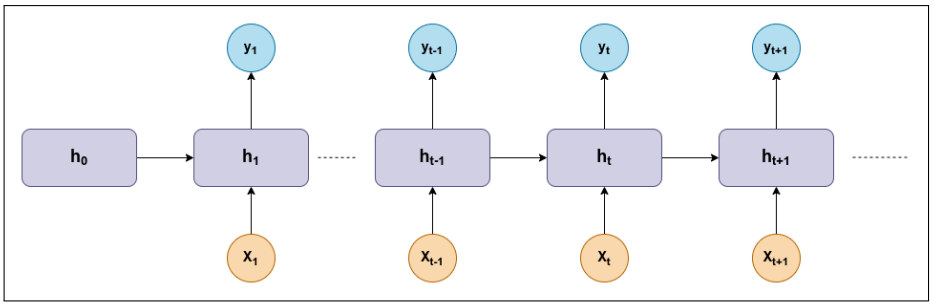


Figure 2.1: Recurrent Neural Network Architecture

Figure 2.1 shows a simple RNN architecture. The input x_t at each time step t is fed into the RNN cell along with the previous hidden state h_{t-1}. The RNN cell then computes the current hidden state h_t and the output y_t. The hidden state h_t is then passed to the next time step as the previous hidden state h_{t-1}. The output y_t

can be computed using the current hidden state h_t or the previous hidden state h_{t-1}. The RNN cell uses the same set of parameters at each time step, allowing it to maintain a form of memory or context.
for each time step t the RNN cell computes the hidden state h_t and the output y_t as follows:

$$h_t = f(Wx_t + Uh_{t-1} + b) \tag{2.1}$$

$$y_t = g(Vh_t + c) \tag{2.2}$$

where f and g are non-linear activation functions, U, V, W, b, and c are the parameters of the RNN cell. the f and g are usually the same activation function such as tanh or ReLU or sigmoid function.

2.1.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is recurrent neural network (RNN) architecture designed by Sepp Hochreiter and Jürgen Schmidhuber in 1997, aimed to deal with the vanishing gradient problem present in traditional RNNs. The LSTM enhances traditional RNNs by adding a feature to forget some information in the network if there is no useful information from other input. LSTM networks are designed to handle sequential data such as time series, speech, and text [3, 4].

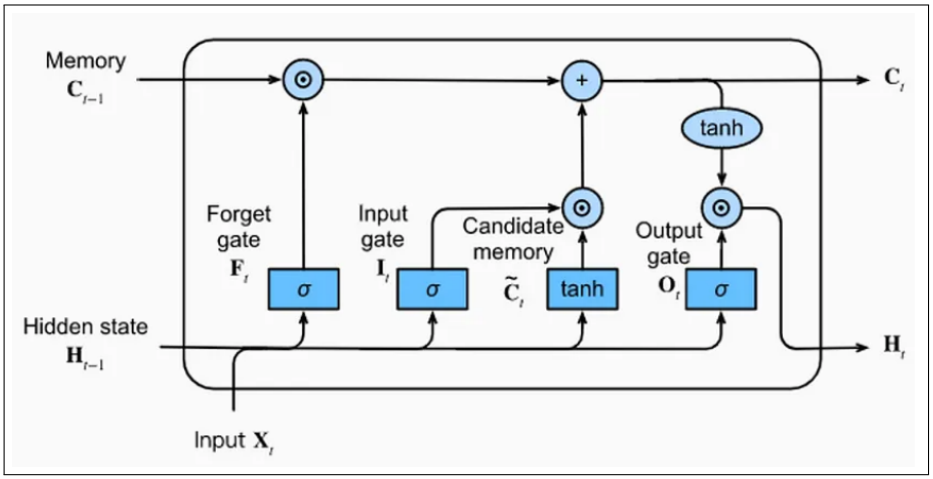


Figure 2.2: Architecture of a LSTM Unit [3]

Figure 2.2 shows a architecture of LSTM unit.The LSTM unit architecture use a series of gate to control the flow of information through the cell. There are three gates in LSTM unit architecture, the forget gate, the input gate, and the output gate [3, 4]. These three gates are used to perform the three typical memory management operations:

1. Forget gate — The forget gate is used to remove information from the cell state.
2. Input gate — The input gate is used to add information to the cell state.
3. Output gate — The output gate is used to decide what information should be outputted based on the cell state.

2.1.3.1 Forget Gate

Figure 2.3 show forget gate of LSTM unit, The forget gate of LSTM unit is used to decide what information should be thrown away or kept. The forget gate takes two inputs, the previous hidden state h_{t-1} and the current input x_t. The forget gate outputs a number between 0 and 1 for each number in the previous hidden

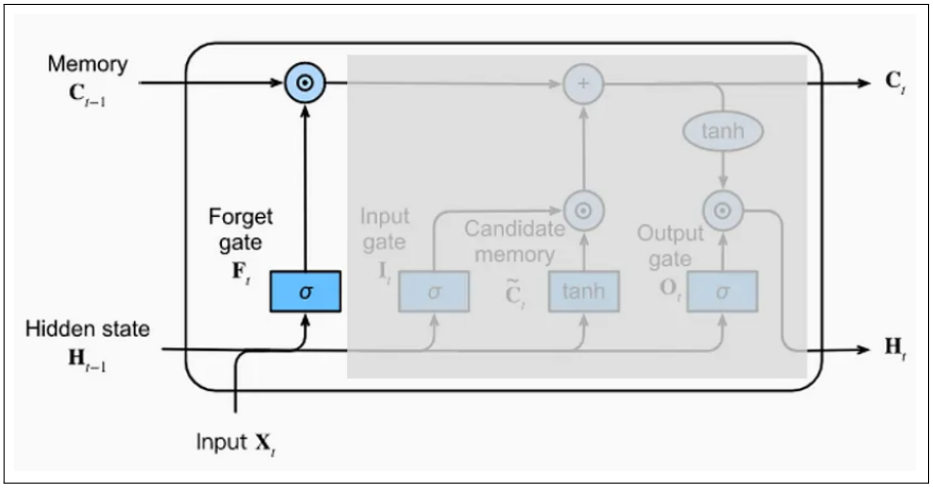


Figure 2.3: Forget Gate [3]

state h_{t-1}. A value of 0 represents that the network should forget the information completely, while a value of 1 represents that the information should be kept completely. The forget gate is computed as follows:

$$f_t = \sigma(W_{hf} \cdot h_{t-1} + W_{xf} \cdot x_t + b_f) \tag{2.3}$$

where W_{hf}, W_{xf} and b_f are the parameters of the forget gate, σ is the sigmoid function, h_{t-1} is the previous hidden state, and x_t is the current input.

2.1.3.2 Input Gate and Candiadate Memory

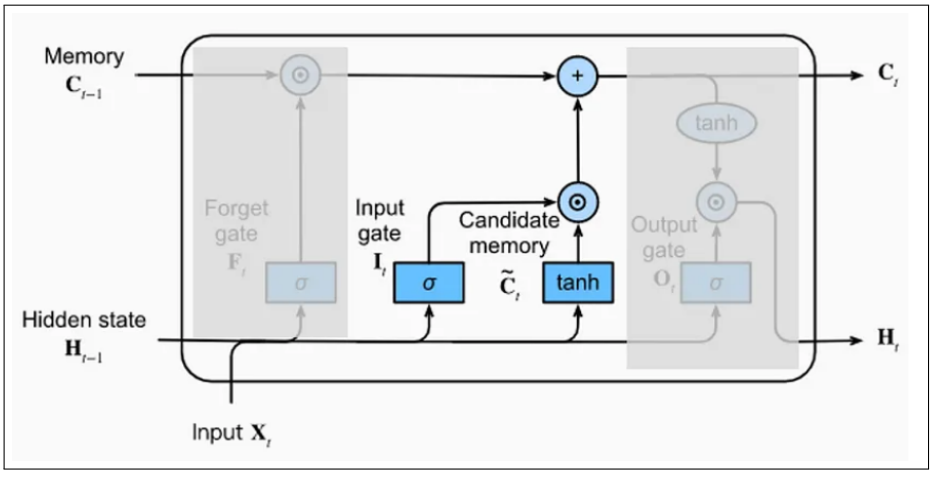


Figure 2.4: Input Gate and Candiadate memory [3]

Figure 2.4 show Input Gate and Candiadate memory of LSTM unit, The input gate of LSTM unit is used to decide what new information should be stored in the cell state. The input gate takes two inputs, the previous hidden state h_{t-1} and the current input x_t. The input gate outputs a number between 0 and 1 for each number in the previous hidden state h_{t-1}. A value of 0 represents that the network should not store the information at all, while a value of 1 represents that the information should be stored completely. The input gate is computed as follows:

$$i_t = \sigma(W_{hi} \cdot h_{t-1} + W_{xi} \cdot x_t + b_i) \tag{2.4}$$

where W_{hi}, W_{xi} and b_i are the parameters of the input gate, σ is the sigmoid function, h_{t-1} is the previous hidden state, and x_t is the current input.

The candidate memory is used to store new information that the input gate decides to store. The candidate memory is computed as follows:

$$\tilde{C}_t = tanh(W_{hC} \cdot h_{t-1} + W_{xC} \cdot x_t + b_C) \tag{2.5}$$

where f_t is the forget gate, C_{t-1} is the previous cell state, i_t is the input gate, and $\tilde{C}_t$ is the candidate memory.

2.1.3.3 Cell State

The cell state is used to store information over a long period of time. The cell state is computed as follows:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2.6}$$

where f_t is the forget gate, C_{t-1} is the previous cell state, i_t is the input gate, $\tilde{C}_t$ is the candidate memory. and $\odot$ is represent as element-wise multiplication.

2.1.3.4 Output Gate

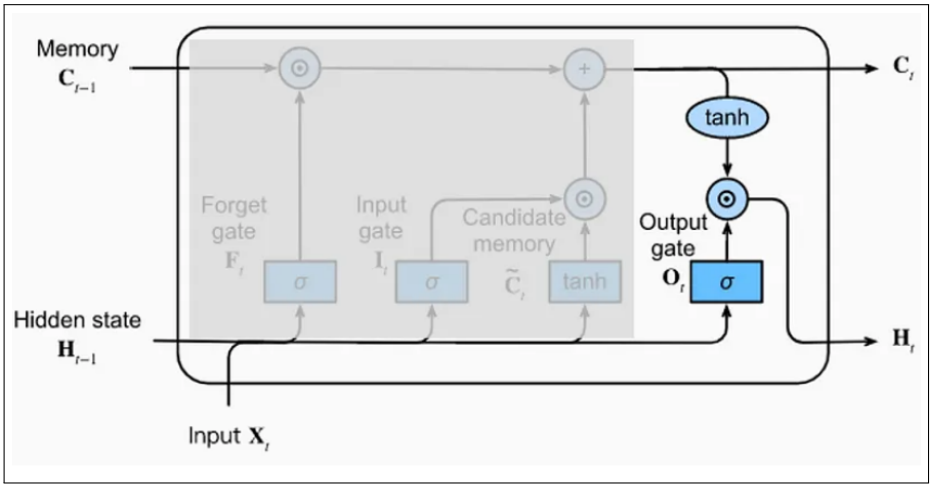


Figure 2.5: Output Gate [3]

Figure 2.5 show output gate of LSTM unit, The output gate is used to decide what information should be outputted from the cell state. The output gate takes two inputs, the previous hidden state h_{t-1} and the current input x_t. The output gate outputs a number between 0 and 1 for each number in the previous hidden state h_{t-1}. A value of 0 represents that the network should not output the information at all, while a value of 1 represents that the information should be outputted completely. The output gate is computed as follows:

$$o_t = \sigma(W_{ho} \cdot h_{t-1} + W_{xo} \cdot x_t + b_o) \tag{2.7}$$

where W_{ho}, W_{xo} and b_o are the parameters of the output gate, σ is the sigmoid function, h_{t-1} is the previous hidden state, and x_t is the current input.

2.1.4 Softmax Function

Softmax Function is a activation function that has a input of vector of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers.

The output of Softmax Function will represent as a vector of K real numbers between 0 and 1 that add up to 1. The softmax function is computed as follows:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.8}$$

where z is the input vector, j is the index of the output neuron, and K is the total number of output neurons.

2.1.5 Regular Expression (Regex)

Regular Expression (called REs, or regexes, or regex patterns) is a sequence of characters that forms a search pattern. It is used to locate or validate specific strings or patterns of text in a sentence, document. Regex helps programmers identify which word of the sentence is phone number or email pattern. After text that matchs the search pattern is found, a decision can be made to either eliminate or modify that text to fit the desired pattern [5]. Regular Expression can be used in many cases such as validate email or phone number whether it is in the correct format or not. Regular Expression can be used in many programming languages such as Python, Java, Javascript, C++, etc.

```python
import re

# Define the regular expression pattern
phone_number_pattern = r'^\d{3}-\d{3}-\d{4}$'

# Test string
test_string = '123-456-7890'

# Use the re.match() function to check if the string matches the pattern
if re.match(phone_number_pattern, test_string):
    print("Valid phone number")
else:
    print("Invalid phone number")
```

Figure 2.6: Regular Expression usecase for identify phone number format

For example, To identified phone number in a sentence, The regex can be used to find the pattern of phone number in the sentence. The pattern of phone number is 3 digits, a hyphen, 3 digits, a hyphen, and 4 digits (ex. 123-456-7890) can represented in term regex pattern as `\d{3}-\d{3}-\d{4}` as shown in the figure 2.6.

2.1.6 Name Entity Recognition (NER)

Name Entity Recognition (NER) is a Nature Language Process task use to identify and categorize word from sentence. The example of labeling entity via using name entity recognition is shown in figure 2.7. Name Entity Recognition can be used in business case such as speed up the process of scanning resume via NER , improve speed of search engine result and etc. Nowadays, there are many open-source tool for implement name entity recognition such as Stanford NLP, Spacy, NLTK, etc. There are some of common name entities such as PERSON(find name person in sentence), ORGANIZATION(find organization name in sentence), LOCATION(find location word in sentence) [6].

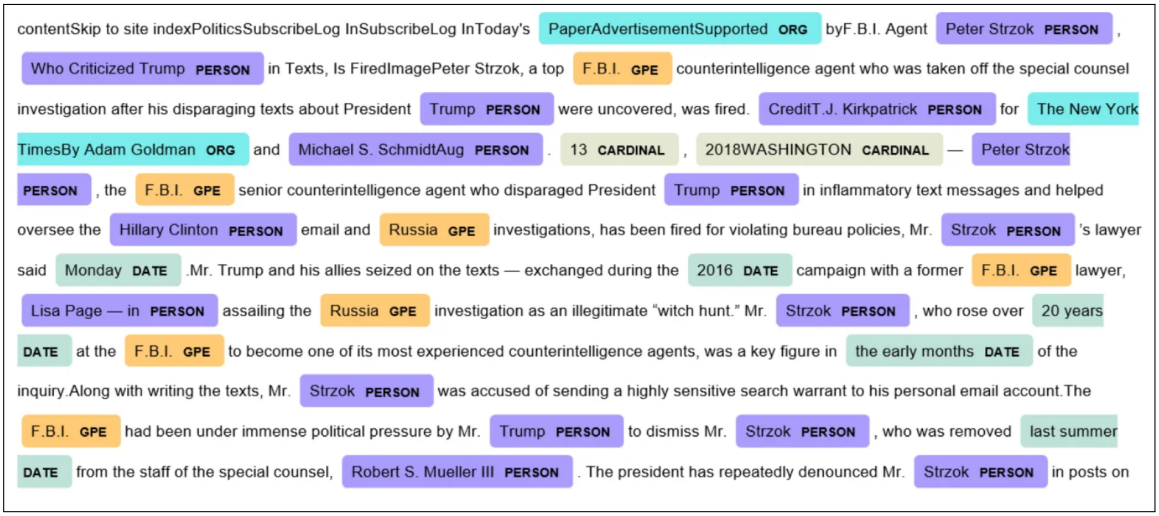


Figure 2.7: Name Entity Recognition Example

2.2 Development Tools

2.2.1 Natural Language Processing Tools

This section will focus on Python libraries and modules that are used to process Natural Language Processing (NLP) Task. Natural Language Processing tools help computers to process and analyze large amounts of natural language data. NLP enables computers to understand human language and derive meaning from it. Their applications span diverse domains such as machine translation, sentiment analysis, speech recognition, and more.

2.2.1.1 Tensorflow

Tensorflow is a free and open-source software library developed by the Google Brain team. It is designed for machine learning and artificial intelligence(AI) applications such as image processing,natural language processing, etc. Tensorflow uses a single dataflow graph to represent all computation and state in a machine Learning and use all data as tensors(dense n-dimensional arrays) [7]. This framework supports multiple programming languages, including Python, Javascript, C++ and Java [8].

2.2.1.2 FuzzyWuzzy

Fuzzy Wuzzy is a python library for processing fuzzy string-matching algorithms, which are essential for effectively dealing with text data that might have inconsistencies, typos, or variations. In computer science, fuzzy string matching involves the process of locating strings that bear a close resemblance to a given pattern, even if they are not exact matches. Fuzzy Wuzzy functionality uses the Levenshtein Distance algorithm to measure the difference between two strings. This algorithm calculates the minimum number of single-character edits required to transform one string into another. Fuzzy Wuzzy provides several methods for comparing pairs of strings with varying levels of granularity. There are four common methods of comparing two strings. The first one method is the "simple ratio", which assesses the similarity of two strings as a whole. It returns a similarity score ranging from 0 to 100, indicating the percentage of similarity between the strings. The second method is "partial ratio". This approach matches by using the best match substrings. The third method is "token sort ratio". This approach tokenizes the strings and sorts them alphabetically and ignores word order before matching and the last method is "token set ratio". This approach tokenizes the strings and compares the intersection and the remainder [9, 10].

2.2.1.3 Stanza

Stanza is a python natural language analysis package. Developed by Stanford NLP Group. It contains tools that can be used to perform tasks such as tokenization, multi-word token expansion, lemmatization, part-of-speech tagging, dependency parsing, and named entity recognition. Stanza help developers to convert string containing human language text into lists of sentences and words to generate base forms of those words, their parts of speech and morphological features. Stanza is built on top of the PyTorch deep learning framework, making it easy to train custom models on top of the existing models [11].

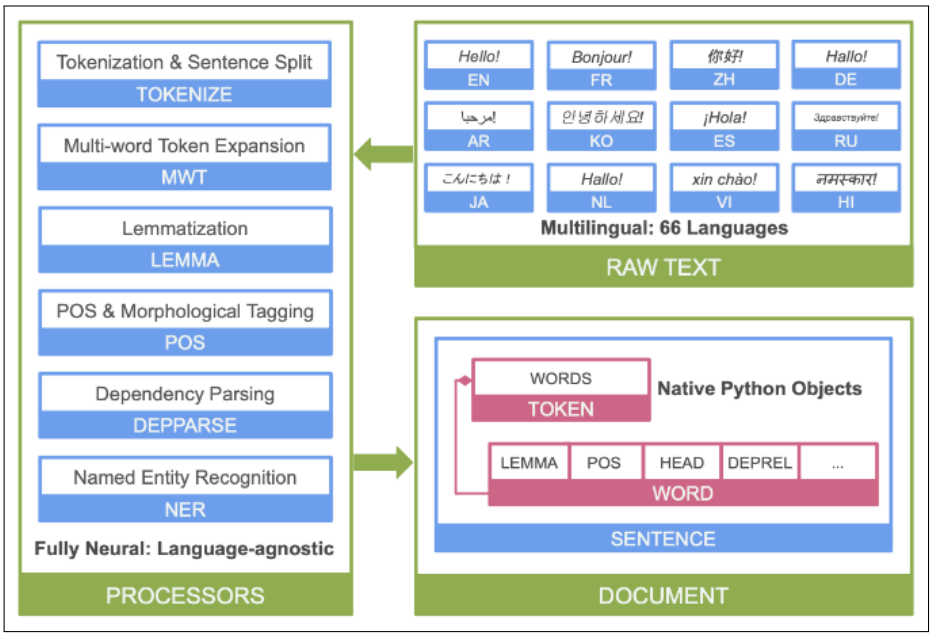


Figure 2.8: Stanza's neural network NLP pipeline [11]

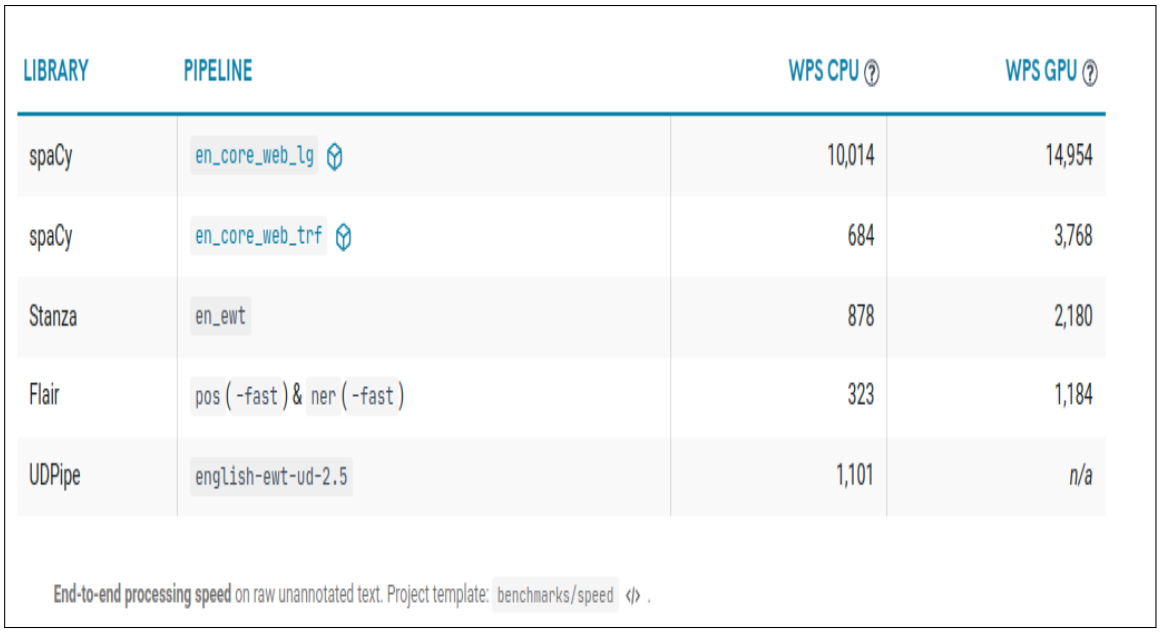
To intiate annotaing text with Stanza, a Stanza pipeline need to be constructed, comprising processors as shown in Figure 2.8. Each of processor in the pipeline performs a specific NLP task, such as tokenization, multi-word token expansion, lemmatization, part-of-speech tagging, dependency parsing, and named entity recognition. The Stanza pipeline takes in raw text or a Document object that contains partial annotations, runs the specified processors that user selected and returns an annotated Document.

The following processors are available in Stanza:

1. tokenization: Tokenize a document into sentences and words.
2. multi-word token expansion: Expand multi-word tokens into single words.
3. lemmatization: Generate the base forms (lemmas) of words. For example, the lemma of "walking" is "walk", and the lemma of "rats" is "rat".
4. part-of-speech tagging: Tag words with their parts of speech. This processor helps to identify whether a word is a noun, verb, adjective, etc.
5. named entity recognition: Recognize and classify named entities. This processor helps to identify whether a word is a person, location, organization, etc.

2.2.1.4 Spacy

Spacy is a open source python library for Natural Language Processing (NLP) task release in 2015 [12]. Spacy come with pre-trained statistical models for NLP tasks such as tokenization, part-of-speech tagging, named entity recognition, text classification, and dependency parsing. To start using Spacy, a Spacy processing pipeline need to be constructed, This pipeline is a sequence of steps that process the text and produce an annotated document. The doc object in spacy contain information about the text such as tokens, part-of-speech tags, named entities, and etc [13]. Spacy is designed to use in production environment due to it speed performance, while other NLP library such as NLTK is designed to use in research environment [12, 14].



LIBRARY	PIPELINE	WPS CPU	WPS GPU
spaCy	en_core_web_lg	10,014	14,954
spaCy	en_core_web_trf	684	3,768
Stanza	en_ewt	878	2,180
Flair	pos (-fast) & ner (-fast)	323	1,184
UDPipe	english-ewt-ud-2.5	1,101	n/a

End-to-end processing speed on raw unannotated text. Project template: benchmarks/speed .

Figure 2.9: Speed Comparsion of different NLP libraries [14]

Figure 2.9 shows the End-to-end processing speed performance of Spacy and other nlp library by test with 10,000 reddit comment dataset. The result shows that Spacy with `en_core_web_lg` pipeline has the fastest processing speed. Spacy with `en_core_web_lg` pipeline can process with 10,014 word per second on CPU and 14,954 word per second on GPU.

2.2.1.5 Presidio

Presidio is a open source python library help developer to detect Personally Identifiable Information(PII) in text. Presidio also provide a way to masking or changing personal information in text to protect user privacy. In Presidio, There are two major parts, Presidio Analyzer and Presidio Anonymizer [15].

1. Presidio Analyzer

   Presidio Analyzer is a service used to detect Personally Identifiable Information(PII) in text. It apply Named Entity Recognition ML model, regular expressions, rule-based logic, and checksum validation to detect Personally Idntifiable Information(PII) in text.

2. Presidio Anonymizer

   Presidio Anonymizer is a service used to masking or changing personal information in text to protect user privacy.

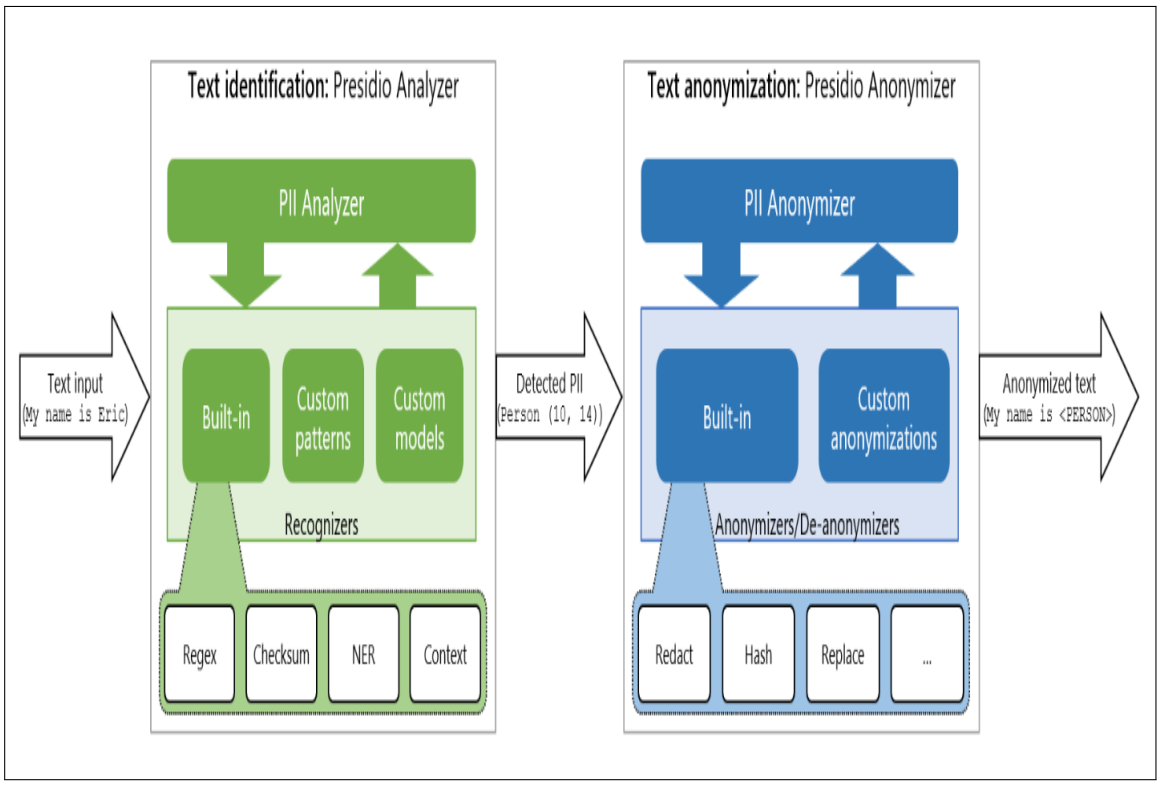


Figure 2.10: Example of Presidio Detection Flow [16]

Figure 2.10 shows Example of Presidio Detection Flow that use to detect Personally Identifiable Information(PII) in text. The First step start with text input, In this case is “My name is Eric” and then the text will be pass to Presidio Analyzer. The Presidio Analyzer will detect the Personally Identifiable Information(PII) in text and return the result to user. After that, Presidio Anonymizer will masking the data base on Presidio Analyzer result and return Anonymizer text to user.

2.2.2 Database and Data Processing Tools

This section aims to provide a deeper exploration of databases and data processing tools, explaining their core characteristics, functionalities, and significance within the technological environment. Readers can develop a heightened awareness of their pivotal role in the management and utilization of data across a wide array of applications.

2.2.2.1 MongoDB

MongoDB is a popular open-source developed by DoubleClick (now owned by Google), a NoSQL (non-relational) database management system that is designed to store, manage, and retrieve large volumes of data in a flexible and scalable manner. Unlike traditional relational databases, which use tables with predefined schemas, MongoDB uses a document-oriented model, where data is stored in JSON-like documents with dynamic schemas. MongoDB is designed for horizontal scalability, which means it can be used to distribute data across multiple servers or clusters to handle large volumes of data and high traffic loads [17].

2.2.2.2 Cosmos DB

Cosmos database is a Paas (Platform as a Service) database service provided by Microsoft Azure. It can be used to store and query data in a fiexible and scalable manner. Azure Cosmos DB is support multiple

data models such as documentDB, MongoDB, Cassandra, Graph API, and Table API. Azure Cosmos DB is designed for horizontal scalability, which means it can be used to distribute data across multiple servers or clusters to handle large volumes of data and high traffic loads [18].

2.2.2.3 PostgreSQL

PostgreSQL is a powerful open-source relational database management system (RDBMS). It is known for its robustness, extensibility, and advanced features. PostgreSQL is designed to store and manage structured data in a way that supports complex queries, transactions, and data integrity. It supports a wide range of advanced data types, including arrays, hstore (key-value pairs), JSON, and more, allowing for flexible data modeling. PostgreSQL also provides a framework for creating custom data types, operators, functions, and extensions, enabling developers to tailor the database to specific needs. Its versatility and stability make it a popular choice among developers and enterprises who are looking for a reliable and powerful relational database [19].

2.2.2.4 Redis

Redis (Remote Dictionary Server) is an open-source, in-memory data structure store that can be used as a caching mechanism, message broker, and key-value database. It is designed for high-performance applications that require fast data retrieval and manipulation. Redis stores data entirely in memory, allowing for extremely fast read and write operations. It operates as a key-value store, where data is associated with a unique key, enabling efficient data retrieval. Redis is categorized as a NoSQL database due to its non-relational data storage model. It supports various data structures, including strings, lists, sets, sorted sets, hashes, bitmaps, and geospatial indexes, providing versatility for different use cases [20].

2.2.2.5 Confluent Kafka

Confluent Kafka is a leading platform designed to streamline and enhance the use of Apache Kafka. Confluent provides a comprehensive ecosystem of tools and services built around Kafka, making it easier for organizations to harness the power of real-time data streaming. With Confluent, users can efficiently manage, store, and process vast volumes of data in motion, facilitating data integration across various applications and systems. This platform offers a range of features, including data pipelines, event-driven microservices, and connectors to external data sources. Confluent also simplifies Kafka deployment , monitoring, and management, making it a valuable choice for businesses seeking to build robust and scalable event-driven architectures [21].

2.2.3 Software Frameworks

This section will focus on software frameworks, integral tools that provide developers with structured blueprints for constructing a diverse range of applications. It will further explain the essential characteristics, significance, and specific examples of software frameworks. There will be a discussion on the software frameworks such as Angular, Spring Boot, and Flask, highlighting their distinctive features and applications in modern development contexts.

2.2.3.1 Angular

Angular is an open-source front-end web application framework founded and upheld by the Google team, operates on typescript. This framework is usually used for sizable enterprises due to its backing by Google, which minimizes the potential risks associated with transitioning to new technologies. Angular also comes equipped with a robust dependency injection system. This system allows for efficient management and sharing of various resources and services across the application, promoting code reusability and maintaining a separation of concerns [22]. Angular is written base on typescript, which is a superset of javascript. Angular

Architecture is based on components, which are the fundamental building blocks of an Angular application called NgModules. Each component consists of a template, which is an HTML-based view, and a component class that controls the behavior of the view. Angular also provides a Router service that allows developers to define routes for navigating between different components [23].

2.2.3.2 Spring Boot

Spring Boot is an open-source Java-based framework, created by the Pivotal Team, dedicated to crafting microservices and web applications. It empowers developers to construct autonomous, production-ready Spring applications. One of its distinguishing characteristics is its stand-alone capability, enabling applications to run independently on various devices, free from the constraints of a specific platform or the need for an internet connection or additional services. The primary objective of Spring Boot is to streamline the development of production-quality applications, minimizing the time and effort required. It offers an array of indispensable features for modern application development, including embedded servers, security measures, performance metrics, health checks, and the ability to externalize configuration settings. Additionally, Spring Boot provides a selection of plugins to enhance the development process, facilitating integration with build tools such as Maven and Gradle [24,25]. Spring Boot is remarkably versatile, compatible with a wide array of integrated development environments (IDEs) including Eclipse, NetBeans, and IntelliJ IDEA. It is constructed atop the popular Spring framework, known for its effectiveness in building enterprise-level applications [26].

2.2.3.3 Flask

Flask is a micro web framework written and developed on python. It is designed as a lightweight and flexible framework for developing web applications with python. Flask is consider a good choice for small to medium-sized applications that require a high degree of customization due to its minimalistic nature [27].

2.2.4 DevOps tools

This section will directly focus towards an array of essential devops tools. DevOps tools can be categorized into several groups, each serving a specific purpose within the DevOps workflow, including Datadog, Kubernetes, Kubernetes lens, Helm, Azure Red Hat Openshift (ARO), K6, Github Action, and Micrometer, each playing a pivotal role in streamlining and enhancing modern development and operational processes.

2.2.4.1 Datadog

Datadog is a cloud-based monitoring and analytics platform that helps organizations gain insights into the performance and health of their applications, infrastructure, and services as well as troubleshooting the problem. Datadog supports a wide range of platforms, including cloud environments (AWS, Azure, GCP), on-premises servers, containers, and more. It uses lightweight agents installed on hosts and containers to collect and send performance data to the Datadog platform. It also provides flexible alerting and notification capabilities, allowing users to set up alerts based on predefined thresholds or anomaly detection. It is commonly used by DevOps team to monitor the performance of complex, distributed systems, allowing them to proactively identify and resolve issues, optimize resource usage, and improve the overall user experience [28].

2.2.4.2 Kubernetes

Kubernetes is an open-source container orchestration platform developed by Google that automates the deployment, scaling, management, and operation of containerized applications. Kubernetes clusters consist of a master node and multiple worker nodes. The master node controls and manages the cluster, while the worker nodes run the containers. The smallest deployable units in Kubernetes are pods, which can contain

one or more containers that share the same network and storage resources. Kubernetes enables automatic scaling of pods to handle increased load and can maintain a desired number of pod replicas for high availability. It provides a powerful and flexible way to manage containerized workloads and services in a distributed computing environment making it easier to manage complex microservices architectures and cloud-native applications [29].

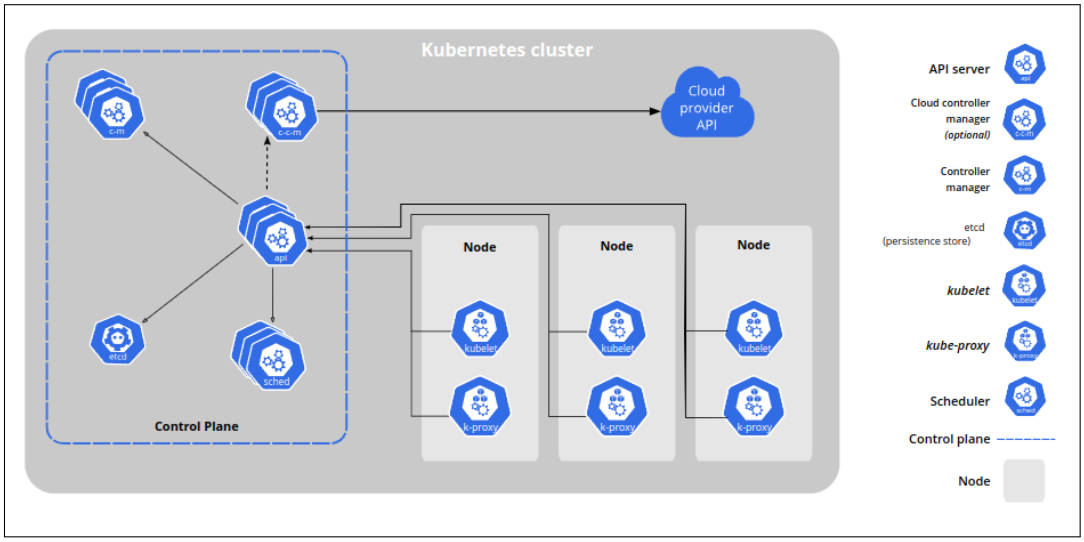


Figure 2.11: Kubernetes Architecture and Components [29]

Figure 2.11 shows a Kubernetes architecture. The components of Kubernetes (K8s) can be categorized into two main groups: Control Plane Components and Node Components. The control plane's components are responsible for controlling Kubernetes clusters and managing the state of the cluster. The node components are responsible for running pods and communicating with the control plane to receive instructions and report the node's status [30].

- Control Plane Components

  1. kube-apiserver:

     kube-apiserver is the front-end component of the Kubernetes control plane. It exposes the Kubernetes API, which is used to interact with the cluster, manage resources, and perform various operations. It is mainly used for Authentication, authorization, admission control, and validation of API requests.

  2. etcd:

     etcd is a distributed key-value store that serves as the cluster's backing store for all cluster data, including configuration, secrets, and the current state of all objects in the system. The key features of etcd are Consistency, reliability, and high availability. It allows multiple Kubernetes master nodes to maintain a consistent view of the cluster's state.

  3. kube-scheduler:

     The scheduler assigns pods to nodes in the cluster. It will assign pods to node by condition for example like resource requirements, node capacity, and user-defined constraints. It is mainly used for balancing, affinity/anti-affinity rules, and resource allocation.

  4. kube-controller-manager:

kube-controller-manager run controller processes that regulate the state of the system. Each controller specializes in managing a particular type of resource, such as pods, nodes, or services. It is mainly used to ensure that the desired state of resources matches the actual state by reconciling any differences.

5. cloud-controller-manager:

   cloud-controller-manager is responsible for interacting with the underlying cloud infrastructure (AWS, Azure) to manage resources and services that are specific to that cloud provider. It is used to integrate cloud-specific features like load balancers, block storage, and networking with Kubernetes.

- Node Components

1. kubelet:

   kubelet runs on each node and is responsible for ensuring that the containers are running in a Pod. It communicates with the control plane to receive Pod specifications and report the node's status. It is used for container management, health checks, and status updates.

2. kube-proxy:

   kube-proxy is a network proxy that runs on each node. It maintains network rules on the node and enables network communication to and from Pods. It is used for load balancing across Pods, service discovery, and network routing.

3. Container Runtime:

   Container Runtime is responsible for pulling container images, running containers, and managing their life cycles. It is used for container isolation, resource management, and image management.

2.2.4.3 Kubernetes HPA

Kubernetes HPA is a Kubernetes component that allows user to automatically scale the number of pods in a replication controller, deployment, replica set or stateful set based on observed CPU/Memory utilization (or, with custom metrics support, on some other application-provided metrics). Figure 2.12 show Kubernetes Horizontal Pod Autoscaling (HPA) flow. After setup the HPA, The HPA controller will periodically query the resource utilization from the metric server and compare the resource utilization whether it is above or below the target utilization. If the resource utilization is above the target utilization, the HPA controller will increase the number of pods. If the resource utilization is below the target utilization, the HPA controller will decrease the number of pods [31]. The number of pods is automatically adjusted based on scaling algorithm as shown in equation 2.9.

$$\text{DesiredReplicas} = ceil[\text{CurrentReplicas} \times \frac{\text{CurrentMetricValue}}{\text{DesiredMetricValue}}] \tag{2.9}$$

where DesiredReplicas is the number of pods that should be running, CurrentReplicas is the current number of pods, CurrentMetricValue is the current value of the metric, and DesiredMetricValue is the target value of the metric.

2.2.4.4 Azure Red Hat Openshift (ARO)

Azure Red Hat OpenShift (ARO) is a managed container platform offered by Microsoft Azure in partnership with Red Hat. ARO combines the benefits of both Red Hat OpenShift, a popular Kubernetes distribution, and the capabilities of the Azure cloud platform. ARO takes care of the underlying Kubernetes infrastructure, allowing developers to focus on deploying and managing containerized applications without having much expertise on the Kubernetes. ARO handles cluster creation, scaling, and maintenance tasks, ensuring that

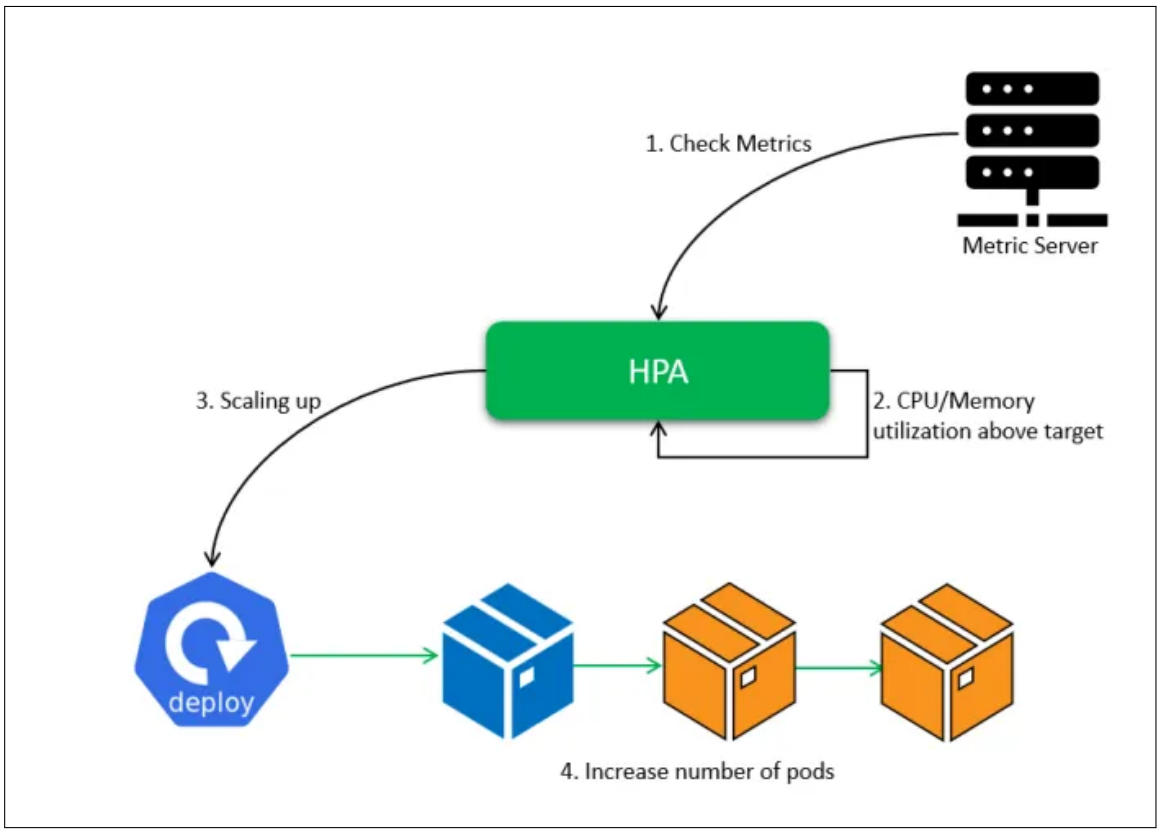


Figure 2.12: Kubernetes Horizontal Pod Autoscaling (HPA) flow [31]

Kubernetes clusters are always up to date and secure. ARO provides multi-cloud support, allowing developers to deploy Kubernetes clusters consistently across Azure regions and, in the future, potentially on other cloud providers. It provides an enterprise-grade environment for deploying, managing, and scaling containerized applications using Kubernetes. ARO can be easily integrated with Azure services, making it easier to use Azure resources like Azure Monitor, Azure Active Directory, and Azure Policy. It provides an enterprise-grade environment for deploying, managing, and scaling containerized applications using Kubernetes [34].

2.2.4.5 Kubernetes Event-Driven Autoscaling (KEDA)

Kubernetes Event-Driven Autoscaling (KEDA) is a Kubernetes-based event-driven autoscaling component developed by Microsoft and Red Hat. In August 2023, Cloud Native Computing Foundation (CNCF) announced that KEDA has been accepted as an graduated project, which mean KEDA project is consider stable and production ready. [32] KEDA allows users to scaling the pod based on external metrics such as Azure Queue, Azure Service Bus, Azure Event Hub, Kafka, Prometheus and etc [33]. KEDA acts as metrics server for HPA as shown in figure 2.13. It gathers metrics from external sources and trigger the HPA to scale the pod based on the metrics. KEDA is especially useful when application workload is event-driven and requires scale on-demands that are not covered by the Kubernetes default horizontal pod autoscaler (HPA).

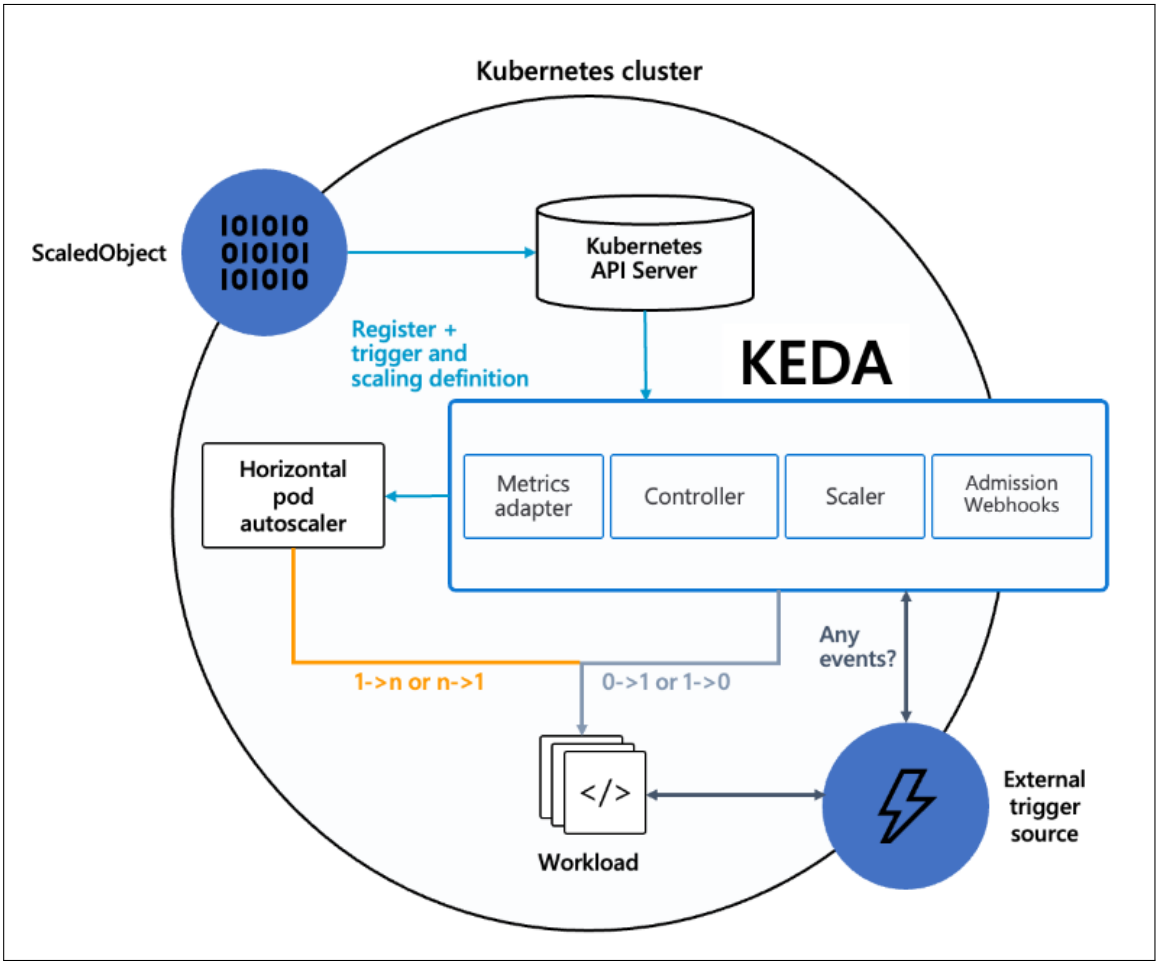


Figure 2.13: KEDA Architecture [33]

2.2.4.6 Helm

Helm is a package manager for Kubernetes which is a tool designed to simplify the deployment, management, and scaling of containerized applications on Kubernetes clusters. Helm packages are called "charts," which are collections of pre-configured Kubernetes resources (YAML files) needed to run an application or service. Charts include deployment configurations, services, secrets, and more. Helm uses Go templates to enable parameterization and customization of charts, allowing users to adjust configurations dynamically during installation. Helm charts can be stored and distributed through Helm repositories, which are collections of packaged charts that can be shared with others. Helm can be integrated into CI/CD pipelines to automate the deployment of Kubernetes applications [35].

2.2.4.7 K6

K6 is an open-source load testing tool developed by using golang. It is designed for performance testing, load testing, and stress testing web applications. K6 uses JavaScript as its scripting language, allowing users to write test scenarios and customize load tests. K6 scripts are easy to create and maintain, making it accessible to developers with various levels of experience. K6 is well-suited for HTTP load testing, enabling developers to stimulate thousands of virtual users (VUs) making requests to the application. K6 focuses on providing developers and DevOps teams with a user-friendly and scriptable way to simulate user traffic and interactions on the applications to assess how well the application performs under various conditions. K6 is capable of

generating high loads and can be used as a tool for assessing the performance and scalability of the web applications [36].

2.2.4.8 Github Action

GitHub Actions is an automation and workflow platform provided by GitHub. GitHub Actions allows developers to automate various tasks, workflows, and processes within the repositories. It enables the user to define custom workflows using YAML configuration files, which are triggered by events such as code pushes, pull requests, releases, and more. It allows developers to automate repetitive tasks, streamline workflows, and enhance collaboration within the repositories. Developers can create custom actions, encapsulating specific processes or tasks, and reuse them across different workflows or repositories. GitHub Actions allows the secure storage and use of secrets, such as API keys and access tokens, to interact with external services or APIs. GitHub Actions can be used for CI/CD pipelines, automating code testing, building, and deployment processes. It is highly customizable, extensible, and well-integrated with the GitHub platform [37].

2.2.4.9 Micrometer

Micrometer is an application metrics facade that provides a simple and consistent way to instrument application code for monitoring and observability [38]. It provide a way to measure the performance of the application and gather metrics such as response time, throughput, error rate, and more. Micrometer provide user to create a custom metrics using various type of meter such as Counter, Gauge, Timer, DistributionSummary, and LongTaskTimer. This metric can publish to various monitoring system such as Prometheus, Datadog, New Relic, and etc [39].

2.2.5 Software as a Service Tools (Saas)

This section will be discussed on Software as a service tools that have been used in the Email Data Cleansing part.

2.2.5.1 Zendesk

Zendesk is a cloud-based customer service software. This software provides tools for businesses to manage and improve customer interactions. Zendesk offer products and services designed to help companies communicate with customers across various channels including email, chat, social media, phone calls, and etc. There are some of key feature that Zendesk provide such as Ticketing System, Multi - Channel Support,Ticket Prioritization and etc [40].

2.2.5.2 Clarabridge

Clarabridge is Software as a Service (SaaS) customer experience management platform. This platform is designed to help organizations understand and improve the customer experience. Clarabridge use Natural Language Processing (NLP) to analyze customer feedback from various sources such as social media, emails, surveys, and more [41]. Clarabridge provide some of key feature such as Text Analytics, Sentiment Analysis, Text Mining, Text Classification and etc.

2.2.6 Integrated Development Environment Tools

This section will directly focus towards programming tools, with a particular emphasis on Integrated Development Environments (IDEs). IDEs are comprehensive software suites that provide a unified environment for coding, testing, debugging, and deploying software. It typically include code editors, syntax highlighting, code completion, debugging tools, version control integration.

2.2.6.1 Intellij IDEA

IntelliJ IDEA is an integrated development environment (IDE) developed by JetBrains. It's primarily focused on Java development but also supports a wide range of other programming languages and technologies. It provides features like intelligent code editing, refactoring, version control integration, debugging, and more. IntelliJ IDEA focuses on productivity, code quality, and developer-friendly features makes it a preferred choice for many developers [42].

2.2.7 Programming Language

This section will directly focus towards programming languages, with a particular emphasis on Java, Python, and JavaScript. Java is a versatile, object-oriented programming language that is known for its platform independence. Python is a high-level, dynamically typed programming language known for its simplicity and readability. JavaScript is a dynamic, interpreted scripting language primarily used for building interactive and dynamic web pages. Each of these programming languages has its own strengths and weaknesses that make them a popular choice among developers.

2.2.7.1 Java

Java is a programming language and computing platform. When the programmer is writing Java, The bytecode will run on most operating systems without the need for recompilation, including Windows, Linux, and Mac OS. Java comes from most syntax from C and C ++ programming languages [43].

2.2.7.2 Python

Python is a high-level programming language interpreter. It known as a programming language that is easy for beginner developers. Python also provides a variety of libraries that can be used with various software development tasks such as web deployment, scientific computing, data analysis and artificial intelligence tasks [44].

2.2.7.3 TypeScript

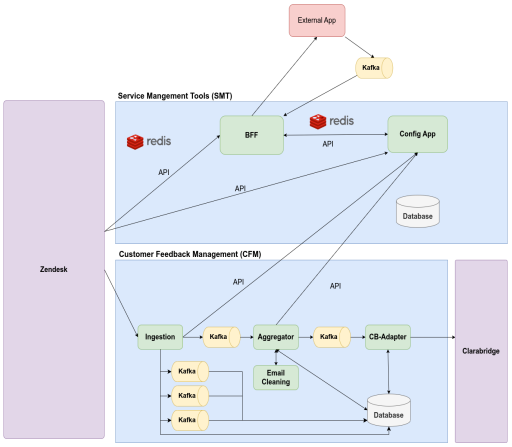
Typescript is a programming language developed by Microsoft. It additional feature to the JavaScript programming language. This mean Typescript has a similar syntax to JavaScript and can be compiled to JavaScript. Among the additional features are static typing, type annotations, and type inference which can help developers to find errors in their code more quickly and easily [45].

CHAPTER 3 RELATED APPLICATION

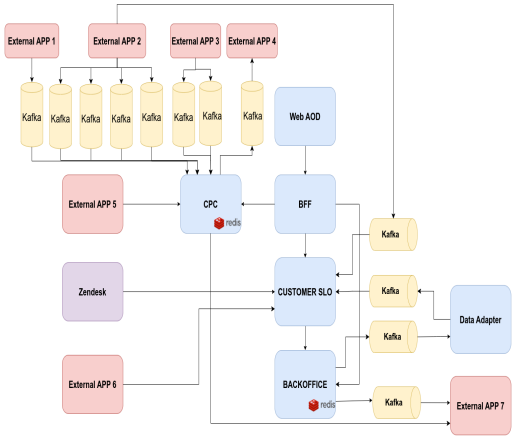
In this chapter, The application that related to this project will be discussed. This chapter will help the reader to understand the application that related to this project and understand the scope of work that has been used to monitoring application resources. Noted that all of the application that mention on this chapter has been developed by Customer Service IT Team.

3.1 System Architecture

In this section, the System Architecture of each application for which the team is responsible will be discussed. Various business use cases will be explored, to understand the scope of work in this project and propose work implementation. Currently, Customer Service IT Team is responsible for 4 applications as shown in Figure 3.1a and Figure 3.1b. Which are Service Management Tools (SMT), Customer Feedback Management (CFM), Automated Offer Deployment (AOD), and Customer Product Catalog (CPC). This 4 applications are developed by the team but there are also some connection with the external applications that are developped by external teams.



(a) SMT and CFM Architecture



(b) Automated Offer Deployment and Customer Product Catalog(AOD and CPC)

Figure 3.1: System architecture of SMT, CFM, AOD and CPC

3.1.1 Customer Feedback Management System

The business case started when customer have a problem or question related to the product or service. Customer normally contact the organization via email or website which is provided by the organization. After customer contact to the organization, the ticket will be created in Zendesk system. There are a lot of tickets that created by customer every month. customer service team has faced the difficult to understand the customer feedback and classify the feedback into the category that organization provide. So the organization want to develop the system that can help customer service team to classify the customer feedback whether it is a complaint, suggestion, or praise. Also analysis the sentiment of customer feedback whether it is positive, negative, or neutral. The Customer Feedback Management (CFM) System is established to help customer service team analysis sentiment and classify customer feedback. Figure 3.2 shows the architecture of Customer

Feedback Management System. The CFM System is divided into 4 main components: Ingestion, Aggregator, Email Data Cleansing Model, and Clarabridge Adapter.

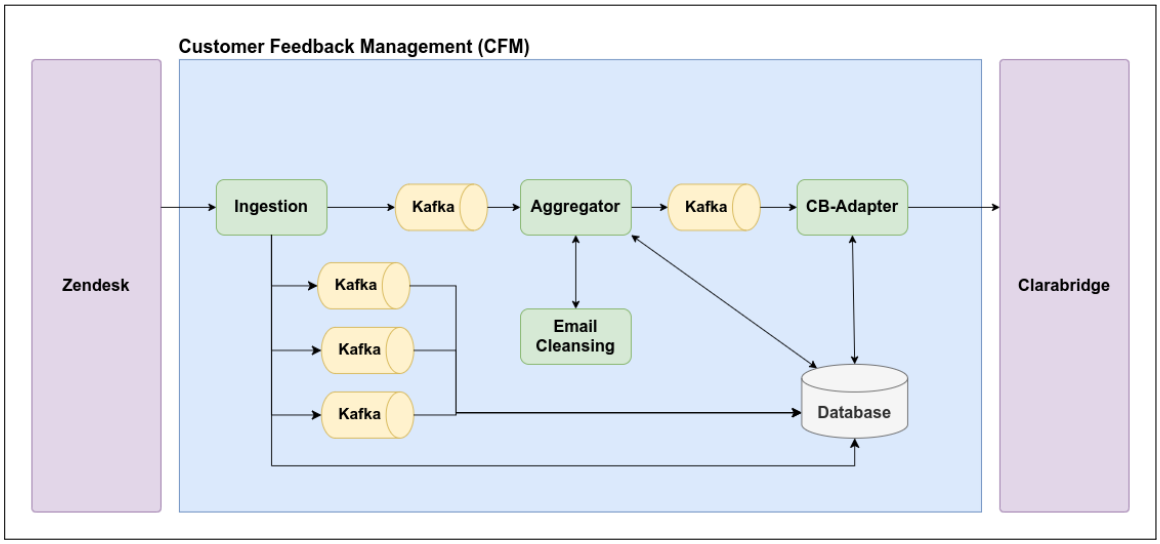


Figure 3.2: Customer Feedback Managemt Architecture

3.1.1.1 Ingestion

Ingestion service is a service developed by using Spring Boot framework that responsible for ingestion message from Zendesk system using Zendesk API. The process is running by using cronjob. The data from zendesk can separate into 3 types, namely ticket, comment, and user information. This data was be sent to Kafka topic by using Kafka producer, then it was consumed by Ingestion service and load into database. After being ingested the email message from Zendesk system, Ingestion service checked that the ticket has already closed or not. If the ticket has already closed, The ingestion service sent trigger to Aggregator service to inform that the ticket has already closed. by sending message to another Kafka topic using Kafka producer and waiting for aggregate service to consume the message. The process of checking ticket status is running by the use of cronjob.

3.1.1.2 Aggregator

After Aggregator service consume the message from Kafka topic from Ingestion service, The comments and user information and email message from Ingestion service was aggregated in to a ticket. The process of aggregation email message was done by reading the email message from database and then mapping ticket. After aggregate the email message into one ticket, the ticket will be sent to Email Data Cleansing service to removing unrelated content such as header, footer, signature, disclaimer and etc. Noted that the aggregate service is communicate with Email Data Cleansing service by using Restful API with one POST method endpoint.

3.1.1.3 Email Data Cleansing

Email Data Cleansing service is a service that responsible for eliminate disclaimer and noise sentence from email messages. This will leave only main context of email messages, which is the customer feedback. There are some of business acceptance criteria that the Email Data Cleansing service must eliminate, The criteria is discussed in more details in the session 3.2. This service is developed by using Flask framework and deployed

on Azure Red Hat Openshift (ARO) cluster. Figure 3.3 shows the previous process eliminate disclaimer and noise sentence from email messages.

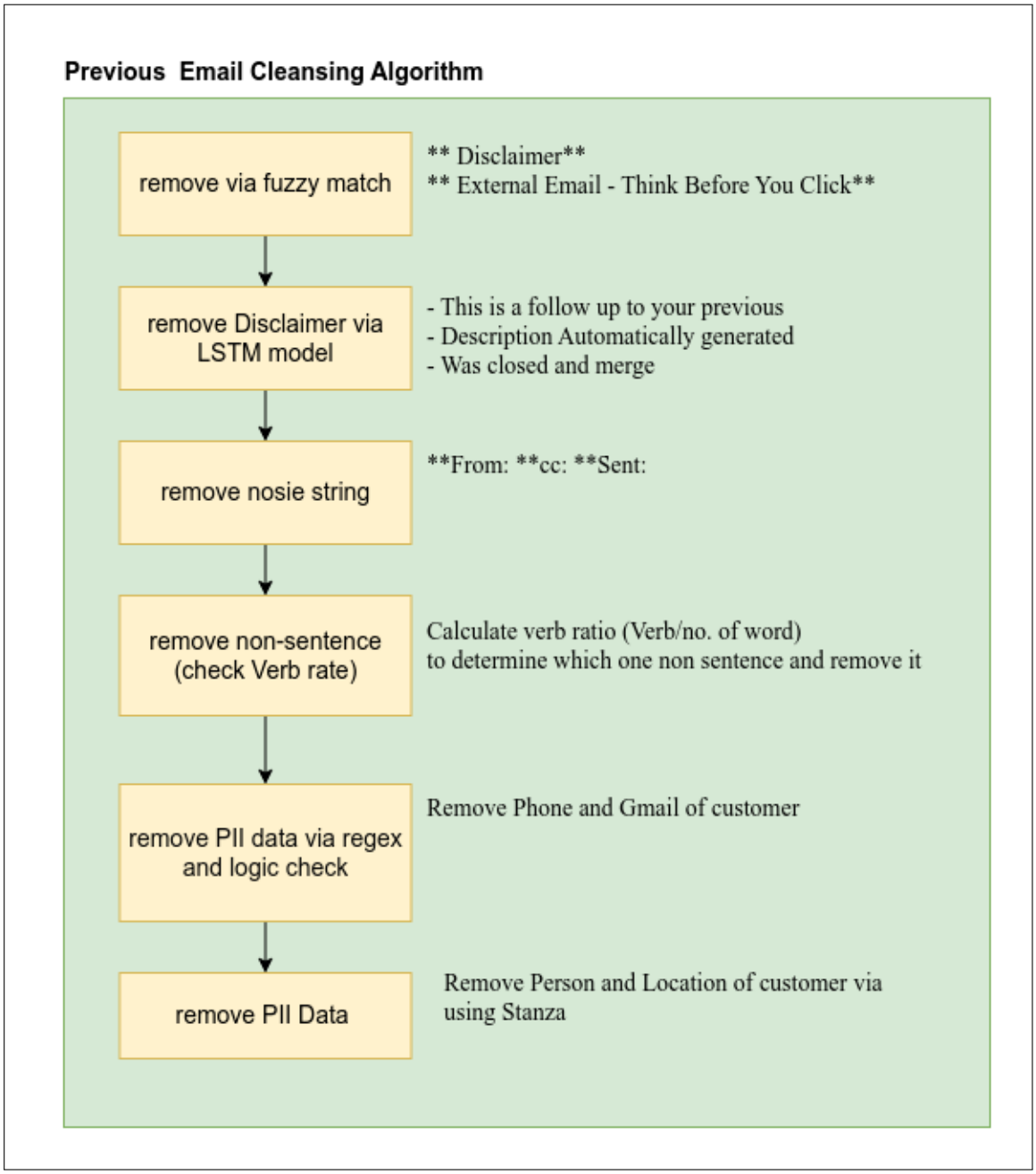


Figure 3.3: Process of Current Email Data Cleansing Service

The flow starts with remove common short disclaimer sentence pattern such as

- **External Email - Think Before You Click**
- **Disclaimer**
- Noted: THIS MAILBOX IS NOT MONITORED. DO NOT REPLY TO THIS EMAIL

In this procedure, a fuzzy string matching algorithm is employed to eliminate typical short sentence patterns. After cleaning these common short sentence patterns with the fuzzy string matching algorithm, the next step

involves the removal of disclaimer sentences from email messages using machine learning techniques. At present, an LSTM model is utilized for the purpose of eliminating disclaimer sentences from email messages. The ultimate goal of this process is to effectively identify and remove disclaimer sentences that exhibit irregular sentence structures.
For Example, the following disclaimer sentence is removed from the email message using the LSTM model:

- This email and any files transmitted with it are confidential and intended solely for the use of the individual or entity to whom they are addressed. If you have received this email in error please notify the system manager.
- This email is confidential and may be privileged. If you are not the intended recipient please accept our apologies. Please do not disclose, copy, or distribute information in this email nor take any action in reliance on its contents: to do so is strictly prohibited and may be unlawful.

After eliminate disclaimer sentence from email messages, the next step is to remove noise sentence from email messages such as **Sent:, **Subject, Call from, **From and etc. In this stage, a combination of a fuzzy string matching algorithm and pattern checks is employed to filter out noisy sentences from email messages. The subsequent step involves eliminating non sentence in email message. To achieve this, a Part-of-Speech (POS) tagging process using Stanza is utilized to calculate the verb rate for each sentence within the email messages. The algorithm then considers removing sentences with a verb rate lower than 0.10 (10%). Following the removal of unrelated content from email messages, a dedicated Email Data Cleansing service is used to strip away Personal Identifiable Information (PII). This process begins by employing Regular Expressions to identify and remove phone numbers and email patterns within the email messages. The second step involves utilizing Stanza's named entity recognition (NER) to identify and remove patterns related to individuals, locations, organizations, and dates within the email messages. In more detail of the process of remove personal identifiable information is discussed in the session 4.2.3.

3.1.1.4 Clarabridge Adapter

After beging aggregated the data in to the ticket and eliminate disclaimer sentence and unrelated content from email messages, the ticket will be sent to Clarabridge Adapter service via consume the message from Kafka topic of aggregator service. The Clarabridge Adapter service convert the email message data into Clarabridge json format and send to Clarabridge system via using REST API with POST Method. Then, ClaraBridge analyzed the sentiment and classify the customer feedback into the category that organization provide.

3.1.2 Service Management Tools

Service Management Tools (SMT) is a system designed for the efficient management of customer service requests, particularly in the aftermarket services industry. It is designed to manage customer inquiries submitted through email, classifying them based on predefined criteria, and routing them to the responsible teams for a faster and more accurate solution.
Figure 3.4 shows Architecture of Service Management Tools (SMT). SMT contains 2 main components including Backend for Frontend (BFF) and Config App. BFF stands for backend for frontend. It is a service mainly used as a API gateway.The connection between the Zendesk Application and BFF has redis cache in the middle, same as the connection between BFF and Config App. These redis cache is used for improve the speed of the application. BFF allows the services or application to communicate with either external applications or the Databases or config app where all config has been stored in the config databases. Config App is a service mainly used to be called API for the application in Zendesk to access the config Databases which

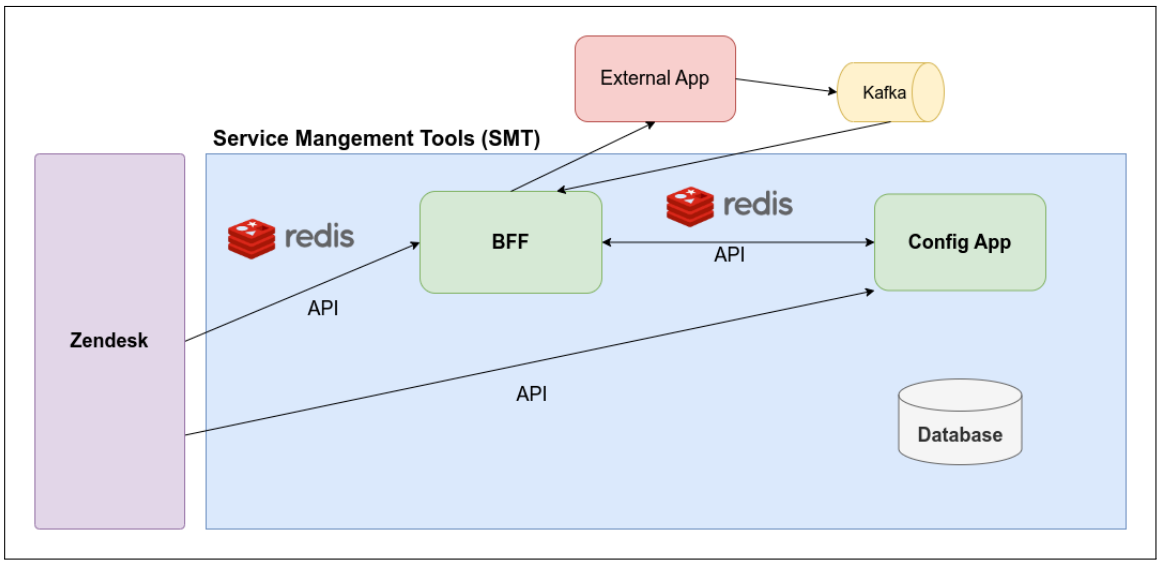


Figure 3.4: Service Management Tools (SMT)

holds the configuration for the services that are in the Zendesk. Config App services uses Redis cache as a connection from internal services of Zendesk.

3.1.3 Automated Offer Deployment

Automatic Offer Deployment (AOD) is a system that is designed to generate personalized offers for each individual customer by calculating multiple factors. The AOD system is accessible to customers through the company's application, offering them a user-friendly way to access special promotions unique to their needs and preferences. It is also used by salesmen in order to give the offer to each customer. AOD system is contain 5 main components including Web Automated Offer Deployment (AOD), Customer Service Level Offer (SLO), Backend for Frontend (BFF), BackOffice and Data Adapter. This system is developed by using Angular framework for frontend, Spring Boot framework for backend and deployed on Azure Red Hat Openshift (ARO) cluster.

Figure 3.5 shows Automated Offer Deployment and Customer Product Catalog (AOD and CPC) architecture. For the AOD system, team has been responsible 3 Kafka topics which are connected to the AOD system. The first topic connect between AOD and customer SLO system.

3.1.3.1 Web Automated Offer Deployment (Web AOD)

It is the frontend part of the website. It is where all the offers will be shown for each customer. It is mainly used by salesmen in order to give the offer to customers. It also allows the sales team to change the offer of the customers by uploading the new offer list in the system. Then it will communicate with the BFF by using the API.

3.1.3.2 Backend for Frontend (BFF)

BFF stands for backend for frontend. It is a service that acts like a gateway for the frontend page to be routed to the desired destination. It is used by the Web AOD by calling the API. Then, It will either be routed to CustomerSLO where all the data of customers and offers for each customer is stored or the back office where it will relay the new offer made by the sales team.

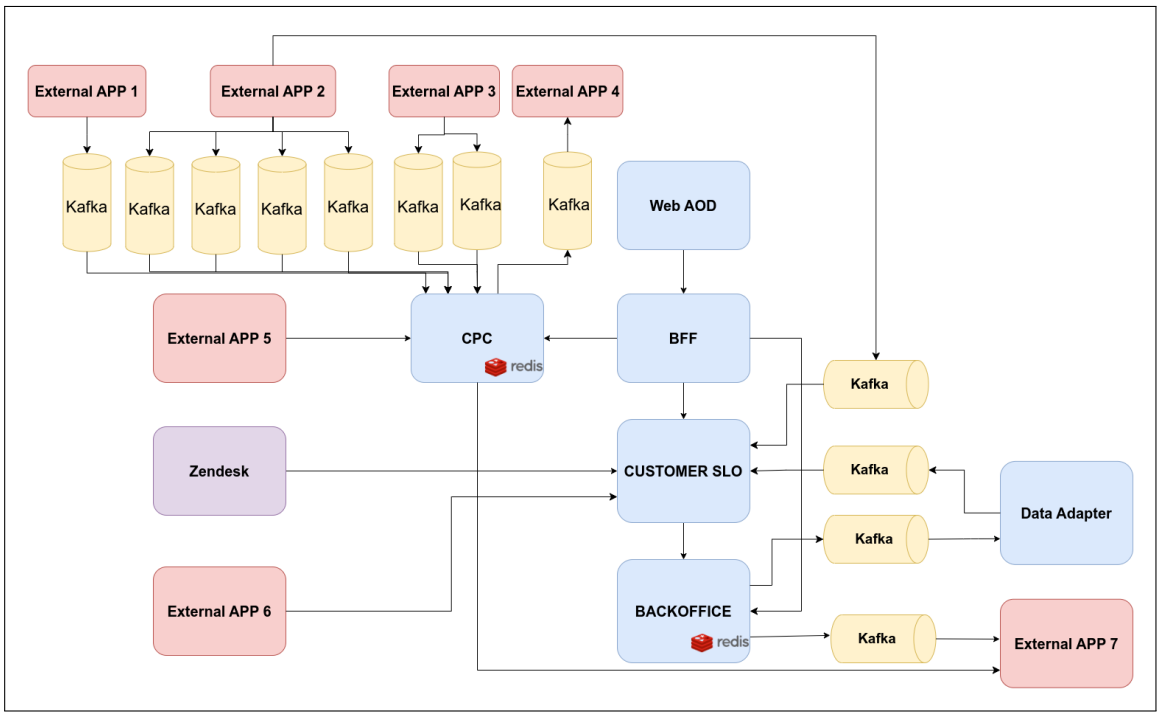


Figure 3.5: Automated Offer Deployment and Customer Product Catalog(AOD and CPC)

3.1.3.3 Customer Service Level Offer (CSLO)

Customer SLO (CSLO) stands for customer service level offer. It is a service that gets the customer offer list from the backoffice service then the Customer SLO will calculate the customer offer for each customer. The customer offer includes the shorter delivery time, aftermarket support for example, allowing face to face meetings with us or will close the issue in a shorter period of time etc. The bigger the customer the better the offer the customer will get. Zendesk called Customer SLO to find the remaining time available for solving the customer issue. Customer SLO also consumes the data from Kafka where Data Adapter is the publisher. Data Adapter will publish all customers who got affected by the new offer. Then, Customer SLO will calculate the new offer for the affected customers and save on the Databases.

3.1.3.4 BackOffice

Backoffice is a service that is mainly used to either save the new offer or retrieve the offer from the Database. When there is a new offer uploaded by the sales team the BFF will send API to backoffice then backoffice will save the new offer as well as publishing the offer to Kafka topics for external services or the Data Adapter.

3.1.3.5 Data Adapter

Data Adapter is a service that consumes the data from the Kafka topics Then it will connect with Database that contains all the customer information and publish the customers who got affected by the new offer in the Kafka. Then, Customer SLO will consume the data.

3.1.4 Customer Product Catalog

Customer Product Catalog (CPC) is backend service application using for checking the product catalog that customer can purchased. This application also provide a information related to location plant where customer can purchased the product. This application consume customer product catalog data via Kafka topic from external application.

3.2 Acceptance Criteria of Email Data Cleansing Service

While the email is sent from the customer to the organization, The email contains a lot of information such as header, footer, signature, disclaimer and so on. So the standard of sentence elimination is setup to ensure that the Email Data Cleansing service could eliminate content that meet the business acceptance criteria.

1. if the sentence is header of email message such as

   - **External Email - Think Before You Click** or **Disclaimer**
   - Noted: THIS MAILBOX IS NOT MONITORED.
   - DO NOT REPLY TO THIS EMAIL,

   this type of sentence will be eliminated from email message.

2. if the sentence is footer of email message such as

   - **This email and any files transmitted with it are confidential and intended solely for the use of the individual or entity to whom they are addressed.
   - If you have received this email in error please notify the system manager.**
   - **This email is confidential and may be privileged. If you are not the intended recipient please accept our apologies.
   - Please do not disclose, copy, or distribute information in this email nor take any action in reliance on its contents: to do so is strictly prohibited and may be unlawful.**

   this type of sentence will be considered to be a disclaimer sentence and will be removed from email message.

3. if the sentence is signature such as

   - **Best Regards, **Regards, **Sincerely

   this type of sentence will be considered to be a signature sentence and will be removed from email message.

4. if the sentence is noise sentence such as

   - (**Sent:, **Subject, Call from, **From, **To, **Cc,**Sent) by folow by email address or name

   this type of sentence will be considered to be a noise sentence and will be removed from email message.

5. if the email message uses the language other than English, the email message will be removed from email message.

6. if the email contain reply messages from previous email, This reply message is considered to keep in from email message.

CHAPTER 4 PROPOSED WORK

This chapter presents the methodologies and design enhancements for the email data cleansing algorithm. It provides a comprehensive overview of the data collection process for training the model, including a discussion on the data source for training machine learning. Additionally, this chapter introduces a monitoring service for tracking the resource usage of the application, an alert system for notify team when error occur in team application resource, the implementation of auto pod scaling which allows the application resource to scaling up and down base on traffic that come to application and API load testing feature.

4.1 Project Overview and Scope

For the scope of this project, the work is divided into two main components: enhancing the email data cleansing algorithm (Section 4.2) and developing a monitoring service for application resources (Section 4.3). Figure 4.1 provides an overview of the project's scope and company work scope. The development environment is designed to automate the process of deployment applications into an openshift cluster using a CI/CD pipeline. The production environment is designated for hosting the application and setup infrastructure. The team has chosen Openshift Container Platform as the hosting platform for the four application (AOD, CPC, CFM, SMT). The infrastructure such as the database, redis cache, etc is setup on the Microsoft Azure environment. It is important to note that development environment and production environment are maintained by the team and are considered not to be the scope of the project.

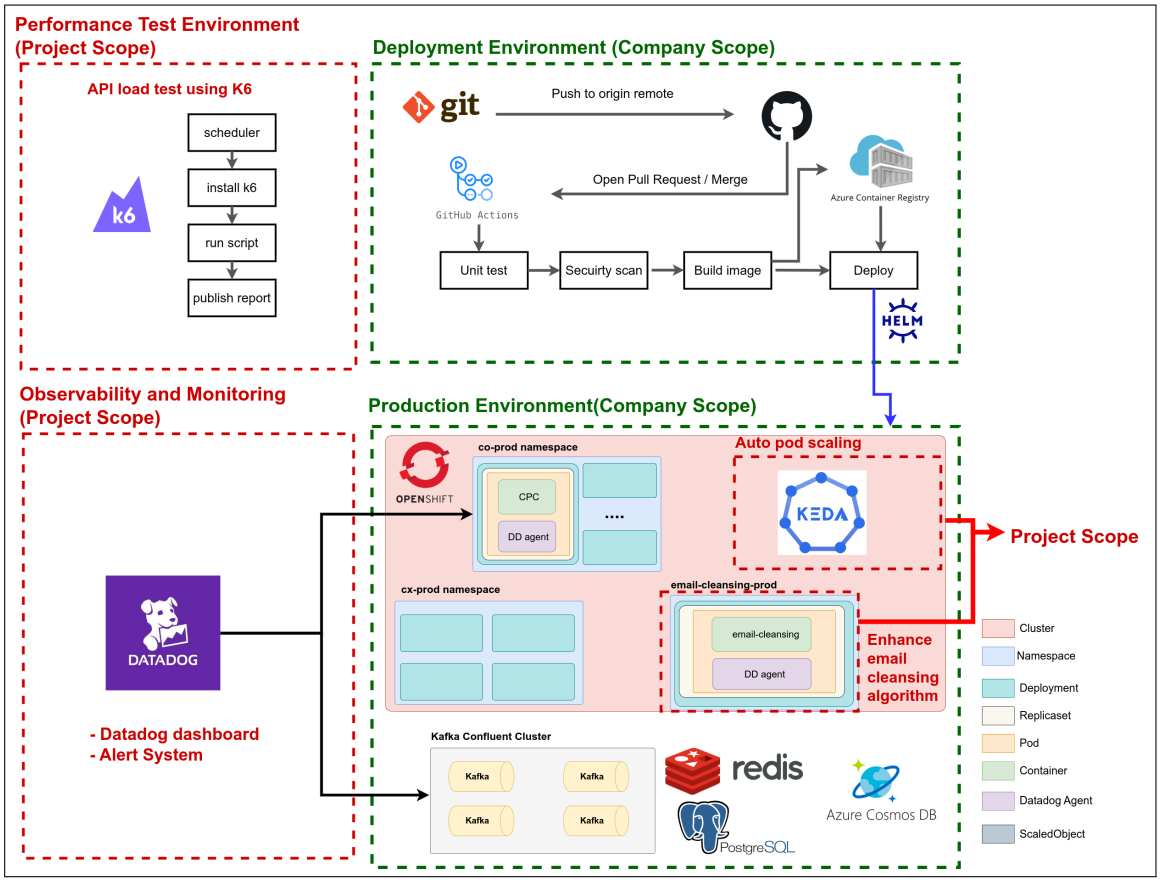


Figure 4.1: Overview Scope of Project

The project scope encompasses the following four features:

1. Enhancement of the Email Data Cleansing Algorithm

   This section focuses on enhancing and improving the algorithm to eliminate the data in the email message that is not related to the main content of the email messages such as disclaimer text and personally identifiable information (PII) data

2. Implementation of Observability and Monitoring Tools

   This section focuses on implementing a tool to assist the team in monitoring application resources across all four applications: AOD, CPC, CFM, and SMT. The resources to be monitored include pod utilization, database metrics, cache metrics, and other relevant parameters.

3. Development of an Auto Pod Scaling Feature

   This section focuses on implementing a feature that enables the application to scale up and down based on incoming traffic. Within the scope of this project, the emphasis is on scaling the application using Kubernetes Event-Driven Autoscaling (KEDA), triggered via consumer lag offset.

4. Performance API Testing

   This section addresses the implementation of a tool designed to evaluate the performance of four specific applications: AOD, CPC, CFM, and SMT, wherein each application must handle substantial traffic, thereby assessing whether that application can handle high traffic when launched in a production environment.

4.2 Enhancement of Email Data Cleansing Algorithm

After being analyzed the previous email data cleansing algorithm, it determined that the effectiveness of eliminating disclaimers and noisy sentences from email messages is not sufficient. There are four main issues that need to be addressed in order to improve the effectiveness of the email data cleansing algorithm.

1. time-coming and ineffective fuzzy string matching algorithm

   The first issue of previous email data cleansing algorithm pertains to the time-consuming and ineffective nature of the fuzzy string matching algorithm. After research and exploration, it has been discovered that some sentences can be removed on remove non sentence process or the LSTM model processing.

2. inefficiency of LSTM model in eliminating disclaimers from email messages

   The second issue involves the inefficiency of the LSTM model in eliminating disclaimers from email messages. This is because the model is unable to identify and remove disclaimers with irregular sentence structures due to the limited amount of training data. Additionally, the previous LSTM model's use of duplicate training data hampers its effectiveness.

3. inefficiency in eliminating non sentence

   The third issue involves the inefficiency of the Part of Speech (POS) tagging process in eliminating non sentence. Currently use Stanza library the model is computationally expensive and time-consuming [12, 14].

4. hard to analyze customer feedback when remove PII data

   The fourth issue arises from feedback received from business users, who have found that removing Personal Identifiable Information (PII) from email messages makes it difficult to analyze customer feedback from the ClaraBridge system.

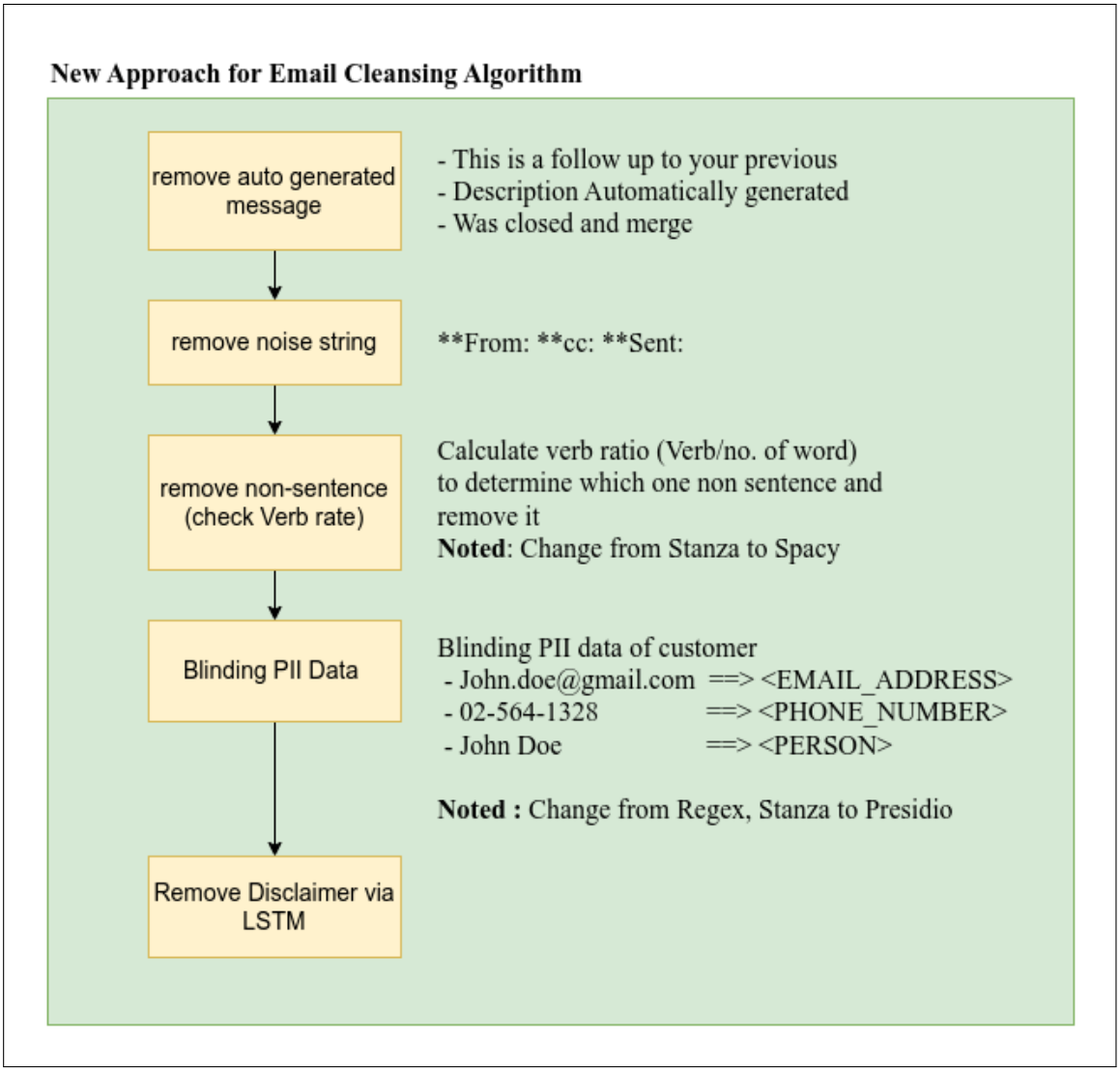


Figure 4.2: New approach for email data cleansing algorithm

These issues will be addressed by proposing a new algorithm to eliminate disclaimers and noisy sentences from email messages. For the first issue, It was observed that most of the sentences in the previous fuzzy string matching algorithm are deleted on the remove non-sentence process or LSTM model process. Therefore, There is no need to add this fuzzy string-matching algorithm. To tackle the problem of a lack of dataset for training the LSTM model, The decision was made to retrain the LSTM model with updated data and add more data for training the LSTM model. Also, the duplicate data will be removed from the training data. There are further preprocess data for training the LSTM model which will be discussed in section 4.2.2. The process of collecting data for the training model and data source will be discussed in section 4.2.1. for the third issue, The previous Python library that was used to calculate the verb rate in a sentence is the Stanza library, Which is not efficient and time-consuming. So decision was made to change python library from Stanza to Spacy large web-based pre-trained model pipeline (`en_core_web_lg`) model as result by testing speed comparison between Spacy `en_core_web_lg` model and Stanza model base on the spacy documentation [14]. It was found that the Spacy `en_core_web_lg` model is faster than the Stanza model. The result of the speed comparison is shown in Figure 2.9. The fourth problem is related to the process of eliminating PII data. To solve this problem, The process has been changing from eliminating the PII data to blinding the PII data into the placeholder. To summarize The new approach of the email cleaning algorithm as shown in Figure 4.2, the flow begins with

removing autogenerated sentences such as “Description automatically generated” and “This is a follow up to your previous” from an email message. In this process, It was done by checking if the sentence matched with the list of autogenerated sentences. If the sentence matches the list of the autogenerated sentence then the sentence will be removed from an email message. Then the next step is to remove the noise string from an email message. This process is similar to the previous approach. The noise strings such as **Sent: **Subject, Call from, **From, **To, **Cc,**Sent by follow by email address or name will be removed from an email message. After removing the noise string from an email message, the next step is to remove the non sentence from an email message. This process is applied by using the Spacy `en_core_web_lg` model to calculate the verb rate of each sentence. If the verb rate of the sentence is less than 0.1 then the sentence will be removed from an email message. the next step is to blind Personal Identifiable Information (PII) data. This process is done by applying a presidio engine to blind PII data into the placeholder. After blinding PII data into a placeholder, the next step is to eliminate the disclaimer sentence from an email message. The process of eliminating the disclaimer sentence is done by using the LSTM model.

4.2.1 Data Collecting Algorithm and Data Source

In this project, data was collected from real message conservation between customer service and company customer. This message data is stored on Zendesk application. After collecting message data from Zendesk, these sentences from email message body will be labeled into 2 groups, disclaimer sentence and normal sentence for training the LSTM model. Each month, the company receives more than 100,000 email messages from customers. In order to handle this large amount of data, it needs a process to filter the data. Normally, the disclaimer sentences are likely to be found multiple time than normal sentence. By this assumption, The hash table can be used to collect the number of occurance of sentence as shown in figure 4.3.

A hash table was created and the number of occurrences of each sentence was collected, then the next step was to remove non-sentence from the hash table. The process of removing non-sentence is implemented by using Stanza Part of Speech (POS) tagging to calculate the verb rate of each sentence. If the verb rate is less than `verbrate_threshold` then the sentence will be removed from a hash table. After removing the non-sentence from the hash table, The next step is to classify into two groups which are sentences that are likely to be disclaimer sentences and sentences that are likely to be normal sentences. The process of classifying sentences into two groups is based on the occurrence of the sentence if the sentence has an occurrence more than `occurance_threshold` then it will be considered to be a sentence that is likely to be a disclaimer sentence. If the sentence occurrence has less than `occurance_threshold` then it was considered to be a sentence that is likely to be a normal sentence. The `occurance_threshold` are used just to classify the disclaimer sentences from the normal sentences because disclaimer sentences will likely appear more in the whole month email message where there is a record of email messages between the agents and customers. However, It is not truly accurate to consider the email that has more than `occurance_threshold` to be the disclaimer sentences. This process was used to briefly classify the sentences. After classifying sentence into two groups, , the next step is to further extract the message by hand before using the data to train the model.

4.2.2 LSTM model Enhancement

Before apply data for training the model.There are a few process of data preparation process that need to be done for improve the performance of the model. Firstly, the sentence that has PII data such as email address, phone number and etc will be replace into placeholder. For example, The email address ’Napas@cpe.com’ was replaced into ’emailentity’. The phone number ’0812345678’ was replaced into ’phoneentity’ and etc. Secondly, the special character in the sentence was be removed such as comma, period, colon and etc. Lastly,

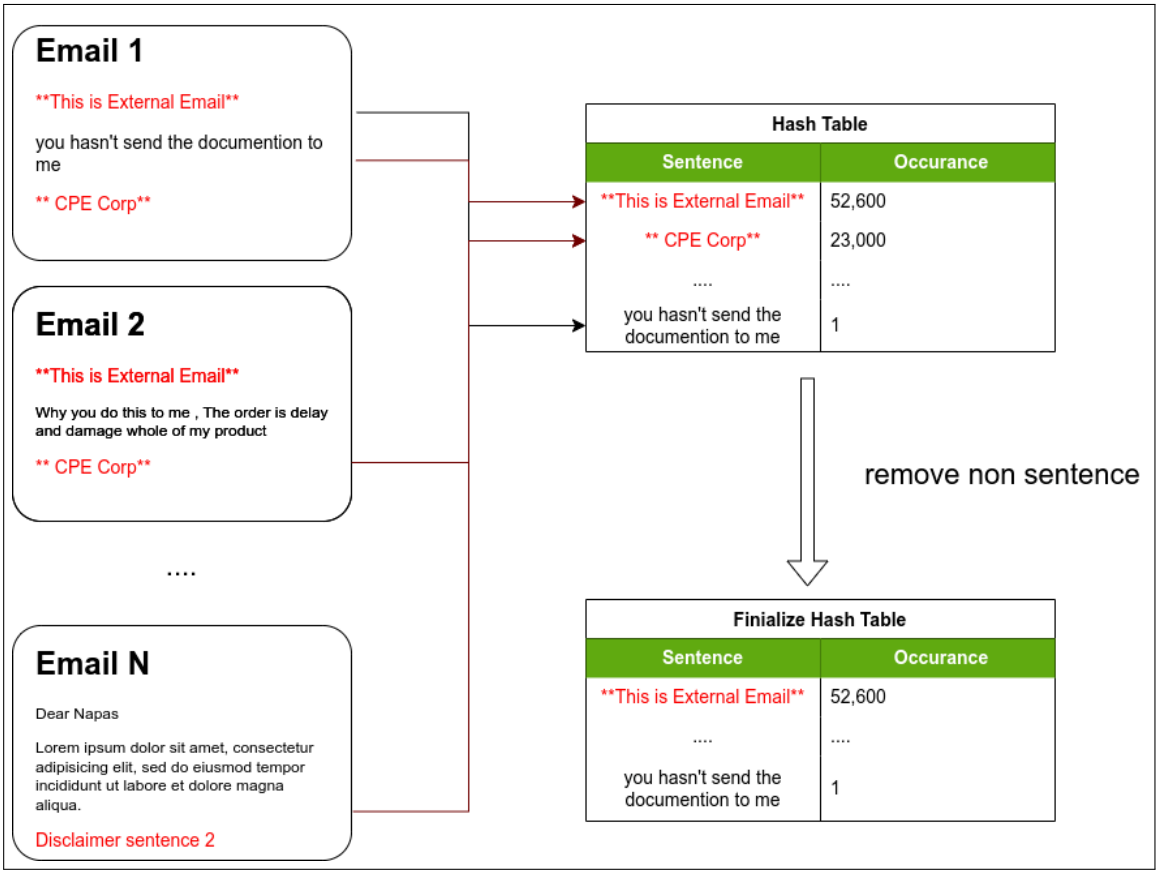


Figure 4.3: Collecting data step for training model

the sentence was tokenized into word and convert into lower case. After data preparation process is done, the data was converted into vector by using Keras tokenizer texts to sequence method. The neural network model is constructed by using Keras with several layers as shown in figure 4.4. The first layer is the embedding layer, which is responsible for reducing the dimension of the vector. The embedding layer has input dimensions of (700, 1850) with output dimensions of (700, 128). The second layer is an LSTM layer with 50 units. This layer is designed to let the model learn the sequence of the text in the sentence. To protect the model from overfitting, The dropout layer has been added after the LSTM layer with a 0.25 dropout rate. The last 2 layers are dense layers with 2 units and softmax activation function. These layers are used for sentence classification. This model was trained using Adam optimizer as a optimize and binary cross entropy as a loss function. The training process was run for 7 epochs. Each of the epoches process 32 batch sizes. The early stopping callback with the patience 3 was setup to protect the model from overfitting. This mean if the model is not improve three consecutive epoches, the trainning process will stop train the model. For the validation phase, the dataset has been split into two parts, a train dataset and a validation dataset. 70% of whole data has used for training LSTM and 30% of whole dataset using for validate the performance of the algorithm in various metric such as accuracy, precision, recall, F1-score.

4.2.3 Personal Identifiable Information masking

The previous approach of PII data detection is using a diver process to detect phone numbers, email addresses, and named entities. Algorithm 1 is used to detect phone number patterns via using logic check and algorithm 2 is used to detect email patterns via using regular expression. from line 1 to line 8 of algorithm 1 is used to check whether the character is a phone character or not. If the character is a phone character including numbers

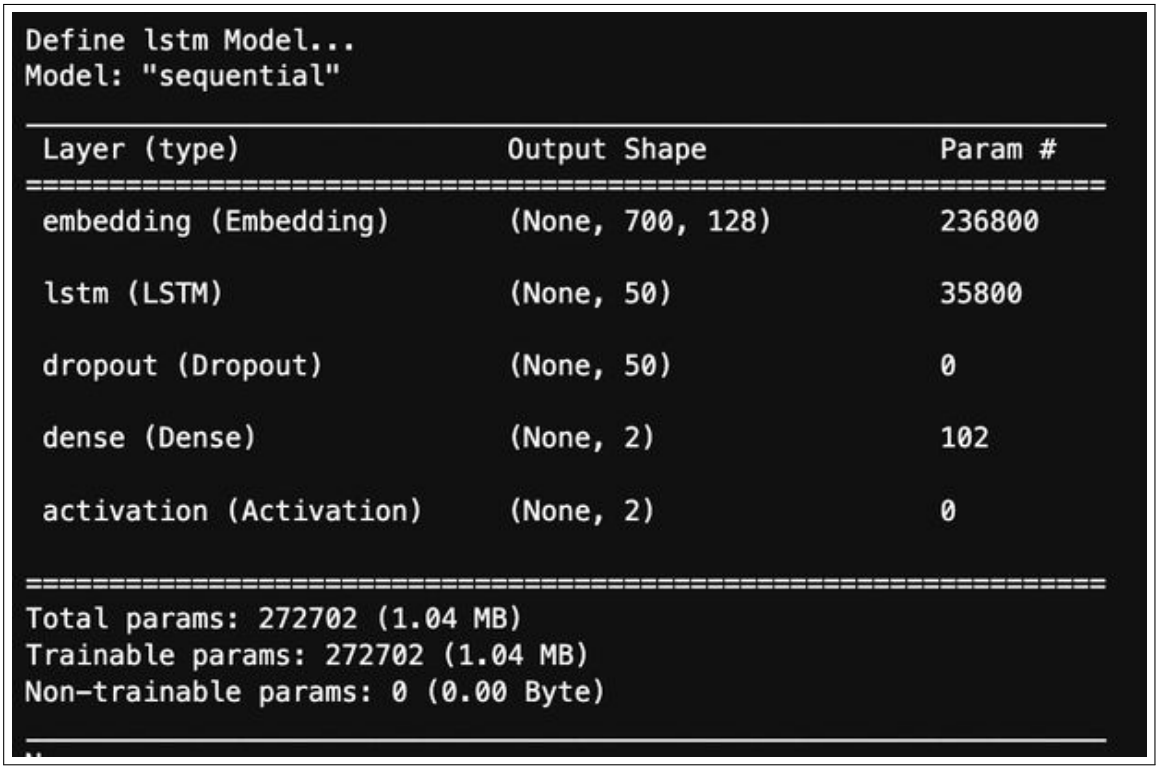


Figure 4.4: The LSTM layer

0-9,+,- and () then the isPhoneCharacter function will return true. If the character is not a phone character then the isPhoneCharacter function will return false. from line number 9 to line number 22 the algorithm will read the message character by character. If the character is a phone character then the character will be added to the phone variable. So If the character is not a phone character then the phone variable will be checked to ensure the length of the phone variable is between 7 and 20 or not. if the length of the phone variable is between 7 and 20 then the phone variable is considered to be the phone number and append the phone number into phoneList. For the algorithm 2, the email pattern detection is detected by using regular expressions to detect email patterns. Recently Spacy Name Entity Recognition(NER) has proven that documentation is easier to read, and the speed of NER is faster than Stanza. and Presidio Library has been using more specific on masking PII data [14, 15]. Thus, This project tests 3 approaches to PII data detection.

1. Stanza NER + algorithm 1,2

   The first approach is the previous approach that apply in Email Data Cleansing service. The process of PII data detection is apply algorithm 1, algorithm 2 to detect phone number and email pattern and Stanza NER to detect location, person and organization pattern.

2. Spacy NER + algorithm 1,2

   The second approach apply algorithm 1, algorithm 2 to detect phone number and email pattern and Spacy NER to detect location, person and organization pattern.

3. Use only Presidio Library

   The third approach apply only Presidio Library to detect all of PII data pattern. The performance will be valuate in two aspect which are correctness and satisfaction of business user.

Algorithm 1 Current approach for finding phone pattern

function isPhoneCharacter(schar)
    $phoneCharacter \leftarrow$ `'+-0123456789()'` ▷ This function check that the character is phone character or not
    if schar in $phoneCharacter$ **then**
        return True
    else
        return False
    end if
end function
Initialize $phoneList \leftarrow []$
Initialize $phone \leftarrow''$
for char in range(len(message)) **do** ▷ read message character by character
    if isPhoneCharacter(line[char]) **then**
        $phone \leftarrow phone + line[char]$
    else
        iLen $\leftarrow$ len(phone)
        if iLen $>$ 7 and iLen $<$ 20 **then** ▷ if the length is between 7 and 20 then it must be phone number
            remove last character from phone if character is spcieal character
            $phoneList$.append(phone)
        end if
    end if
    $phone \leftarrow''$
end for

Algorithm 2 Current approach for finding email pattern

$emailPattern \leftarrow$ `r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+'`
for each line in email message **do** ▷ read email message line by line
    for each word in line **do** ▷ read word in line
        if word match $emailPattern$ **then** ▷ check word match email pattern with regex
            remove word from line
        end if
    end for
end for

For the evaluation of the performance of 3 process of Blinding PII data into placeholder. The evaluation will be valuate in two aspect which are correctness and satisfaction of business user.

4.3 Monitoring Service of Application Resources

The monitoring service task involves overseeing all four applications detailed in section 3.1 : Service Management Tools (SMT), Customer Feedback Management (CFM), Automated Offer Deployment (AOD), and Customer Product Catalog (CPC). Each application demands the monitoring of diverse resources, including CPU, Memory, Redis usage, Kafka metrics, and more. This holistic monitoring initiative will be facilitated through the implementation of the Datadog service. Currently, all of the applications have been deployed on the Azure Red Hat Openshift (ARO) cluster. After interviewing with Backend developers, Frontend developers, Business Analysts, and Quality Assurance professionals. Certainly, the pain points can be succinctly listed as follows: the absence of an alerting system in the event of API failures, resulting in delayed problem detection. Additionally, the error exception setup lacks coverage for all error types, complicating error investigations when errors fall outside defined cases, leading to default errors and potential confusion. The current reliance on decentralized monitoring tools necessitates the use of multiple tools—one for service status logs and another for real-time message payloads. Furthermore, manual pod scaling is required during specific high-traffic periods throughout the day. The possible solutions can be outlined as follows: implementing a monitoring dashboard and alerting system to notify the team, with customizable thresholds to manage excessive alerts. Enhancing the system with more error exception cases, consolidating monitoring tools into a single platform capable of monitoring both logs and message payloads, and implementing an autoscaling pod system are the proposed solutions to address these issues. The tools that will be mainly used are the Datadog and AKHQ. The main approach will focus on the Datadog. The features will include the APM, log, RUM, monitor, metrics, and Dashboard. Custom metrics in Prometheus will be used so that it is easier to monitor metrics related to business logic cases. This custom metric will help understand the business insight. This metric is sent to Datadog for visualizing or alerting the team via the monitor function of Datadog. The approach for consolidating the tools will be combining the AKHQ Kafka message payload feature and displaying it in Datadog along with some integrations. The approach for the autoscaling pod will be using the horizontal pod autoscaling feature of the ARO along with some integrations. Figure 4.5 shows the cluster and namespace of CFM, SMT, AOD, and CPC applications. The clusters are split into 2 clusters which are the production (Prod) cluster and the non-production (Non-prod) cluster. For the Non-prod cluster, there are 7 namespaces which are co-dev, co-qa, co-uat, cx-dev, cx-uat, email-cleansing-dev, email-cleansing-uat. For the Prod cluster, there are 3 namespaces which are co-prod, cx-prod, and email-cleansing-prod. Each namespace will contain the pod of the application. For co-dev, co-qa, co-uat, and co-prod namespace will contain the pod of application of Automated Offer Development (AOD) and Customer Product Catalog (CPC). For cx-dev, cx-uat, cx-prod, email-cleansing-dev, email-cleansing-uat, email-cleansing-prod namespace will contain the pod of application of Customer Feedback Management (CFM) and Service Management Tools (SMT). The environments of applications are split into 4 environments which are development (DEV), quality assurance (QA), user acceptance testing (UAT), and production (PROD). The DEV environment is used for the developer to test the application. The QA environment is used for the quality assurance team to test the application. The UAT environment is used for business users to test the application. The PROD environment is used for customers using the application.

To start monitoring the application, The first step is to set up the Datadog agent in each pod of the application as shown in figure 4.6. The second step is to set up the application to throw logs and metrics to the Datadog agent. The setup can be done by adding the Datadog agent configuration file into each pod of the application. The

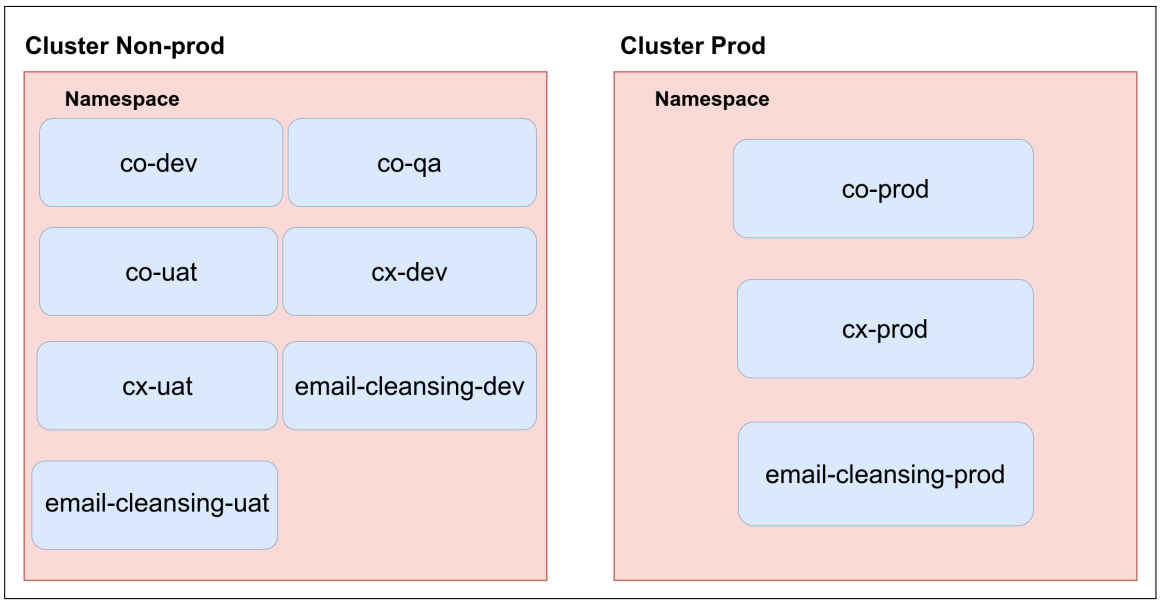


Figure 4.5: Cluster and namespace of CFM,SMT,AOD and CPC applications

application in the container will throw a log to the file that has been set up in the Datadog agent configuration file. The Datadog agent will read the log from the file and send it to Datadog SaaS. For Custom metrics, The custom metrics will be sent to the Datadog agent by using a Micrometer. The Datadog agent will read the metrics from the endpoint of application name /actuator/prometheus/* and send it to Datadog SaaS. The last step is to set up the dashboard and alerting system in Datadog Saas. The dashboard is split into 2 parts which are the dashboard. The first dashboard will be used for monitoring the resources of SMT, and CFM applications. The second dashboard will be used for monitoring the resources of the AOD, CPC application.

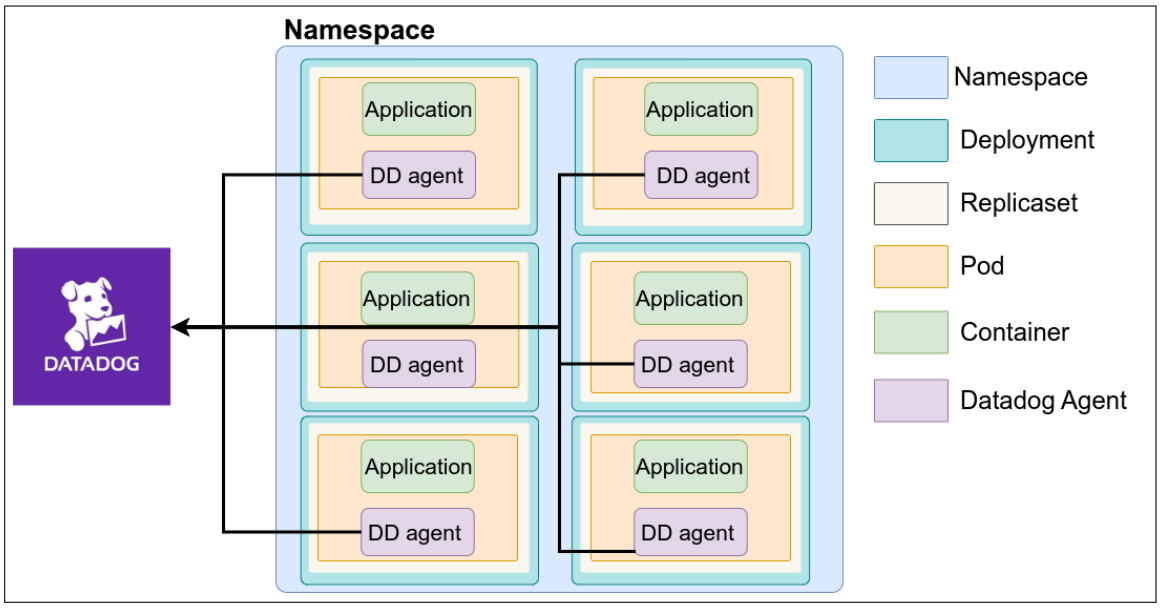


Figure 4.6: Implementation of Datadog Agent in each pod application

4.3.1 Dashboard Feature

For the Dashboard feature of monitoring service of application resources, The feature will be spilt into 8 parts which are Pod utilization monitoring, API Request monitoring, Redis monitoring, PostgreSQL Server Monitoring Azure comosDB Monitoring, Kafka monitoring, Real User Monitoring and custom metrics.

1. Pod Utilization Monitoring

   For Pod utilization monitoring, The focus is on the CPU and memory utilization of the pod. This feature will help the team understand the usage of resource over time and optimize the resource usage. Currently, team has more 50+ pods in the cluster.

2. API Request Monitoring

   For API Request monitoring, The focus is on the request rate, request throughput, request latency and request error rate. This let the team understand the performace of each endpoint of application and optimize throughput and latency of each endpoint.

3. Redis Monitoring

   For Redis monitoring, The focus is on the redis memory usage, redis hit rate, redis latency and redis memory used. If the number of redis hit rate is low, there might be some problem related to redis cache such as time to live (ttl) of cache is too short.

4. PostgreSQL Server Monitoring

   For PostgreSQL Server monitoring, the focus is on the database storage usage, database query latency and input/output operations per second (IOPs). There are 3 services that use PostgreSQL server which are CPC application, CFM application and SMT application.

5. Azure comosDB Monitoring

   For Azure comosDB monitoring, the focus is on number of request that sent to comosDB and request unit (RU) usage. request unit (RU) is performance currency of abstracting the system resources calculate from CPU, memory and IOPs. So If the operation to query data from database is hard to operates, the request unit (RU) usage will be high. Azure comosDB calculate cost of database based on the request unit (RU) usage. Thus the team need to monitor the request unit (RU) usage to optimize the cost of database.

6. Kafka Monitoring

   For Kafka monitoring, The focus is on the kafka message rate, kafka message latency and kafka message error rate. There are some of flow in CPC application that need to be monitor to ensure that the message CPC application receive from external application is correct, No message loss and No message duplication saved in CPC database.

7. Real User Monitoring

   For Real User monitoring, The focus on metric such as time spent loading the page, the largest contentful paint, the first input delay and etc. This let the team understand the user experience of application and optimize the user experience of application.

8. Custom Metrics (depends on the business logic)

   This feature is used for monitoring the metric that related to business logic. The implementation of custom metrics will be done by using micrometer library. The micrometer will be add to all of the

service. The metrics from micrometer will be sent to Datadog agent from endpoint of application name /actuator/prometheus/*.

4.3.1.1 Resources in Each Application

Table 4.1, 4.2 and show the resources in AOD application, CPC application in single environment, Currently each application has 4 environments which are development (DEV), quality assurance (QA) and user acceptance testing (UAT) and production (PROD). The resources in each environment are the same.

Table 4.1 Resources in AOD application within a single environment

Service	Resources Type	Resources Amount
Web AOD	-	-
Backend for frontend (BFF)	-	-
CustomerSLO	MongoDB	1
CustomerSLO	Kafka consumer group	3
BackOffice	Redis cache	5
BackOffice	In memory cache	1
BackOffice	PostgreSQL Server	1
BackOffice	Kafka publisher	2
Data Adapter	Kafka consumer group	1
Data Adapter	Kafka publisher	1

Table 4.2 Resources in CPC application within a single environment

Service	Resources Type	Resources Amount
CPC	Kafka consumer	6
CPC	Kafka publisher	1
CPC	Redis cache	1
CPC	In memory cache	1
CPC	PostgreSQL Server	1

Table 4.3 Resources in SMT application within a single environment

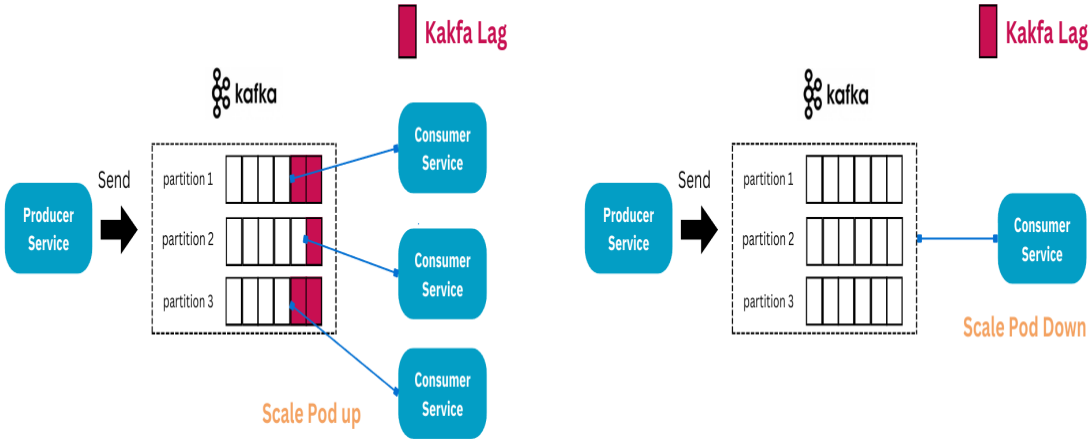
Service	Resources Type	Resources Amount
Backend for frontend (BFF)	Redis cache	1
Backend for frontend (BFF)	Kafka consumer external application	1
Config Application	Redis cache	1
Config Application	PostgreSQL Server	1

Table 4.4 Resources in CFM application within a single environment

Service	Resources Type	Resources Amount
Ingestion	Kafka publisher	4
Ingestion	Kafka consumer	3
Ingestion	PostgreSQL Server	1
Aggregator	Kafka consumer	1
Aggregator	Kafka publisher	1
Aggregator	PostgreSQL Server	1
Email Cleansing	-	-
CB-Adapter	Kafka consumer	1
CB-Adapter	PostgreSQL Server	1

4.3.2 Auto Pod Scaling Feature

After analyzing the use case of each application, there are 3 applications that need to implement auto pod scaling features which are CFM, SMT, and AOD applications. For the team use case, the auto pod scaling feature will be used when there is Kafka consumer lag in applications. So when Kafka consumer lag occurs in the application, It means that the application cannot consume the message from the Kafka topic fast enough. The Kafka consumer lag must come from the case that other applications publish a huge amount of messages to the Kafka topic and consumers of applications cannot consume the message from the Kafka topic fast enough. To speed up the consuming message from the Kafka topic. The Kubernetes should increase the number of a pod of application to consume the message from the Kafka topic faster as shown in Figure 4.7a and when there is no message to consume from the Kafka topic, It is better to let Kubernetes cluster to scale down the number of pods of application as shown in figure 4.7b.



(a) Pod scaling up when no kafka consumer lag

(b) Pod scaling down when no kafka consumer lag

Figure 4.7: Pod scaling up/down base on kafka consumer lag event

The maximum number of scale-up pods of the application will depend on the number of partitions of the Kafka topic. For example, If the Kafka topic has 3 partitions, The number of pods of the application will be scaled up to 3 pods. This is because the number of partitions of the Kafka topic is the maximum number of consumers that can consume the message from the Kafka topic. To implement the auto pod scaling feature, The KEDA will be used to implement horizontal pod autoscaling based on the kafka consumer lag metric. The autopod scaling feature will be implemented in all environments of the application except the development environment, This decision was made on team agreement. For the deployment process, GitHub Actions will be used as a process for Continuous Delivery (CD) to deploy the auto pod scaling feature to the ARO cluster.

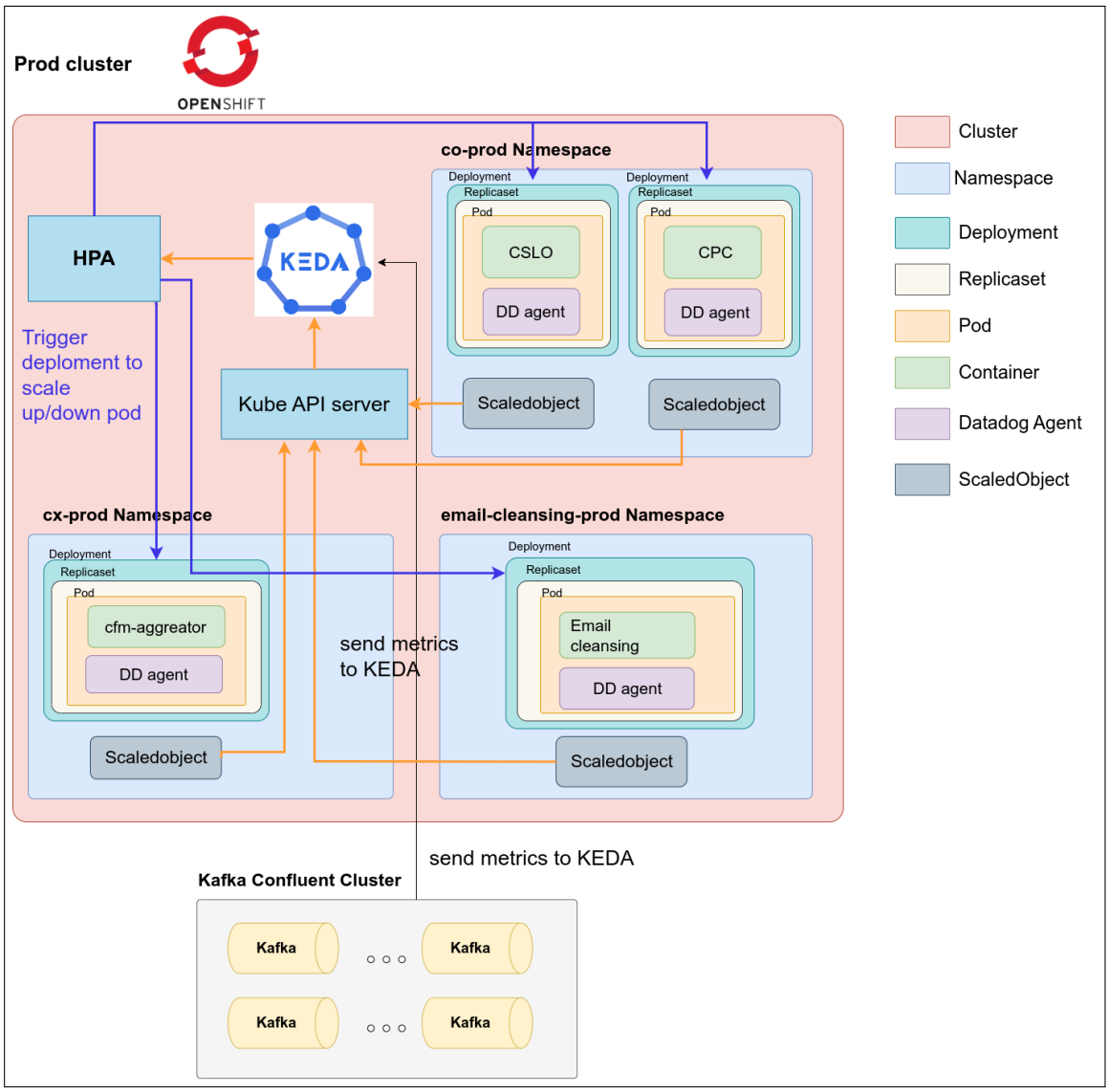


Figure 4.8: Plan implement KEDA to ARO cluster

Figure 4.8 shows plan to implement KEDA in ARO cluster and setup horizontal pod autoscaling base on kafka consumer lag metric. The following steps will be used to implement KEDA in ARO cluster and setup horizontal pod autoscaling base on kafka consumer lag metric.

1. Install the KEDA in ARO cluster by using helm
2. Setup KEDA Controller in KEDA namespace
3. Create ScaledObject in namespace of application
4. Create TriggerAuthentication in namespace of application
5. Deploy the KEDA ScaledObject and TriggerAuthentication to ARO cluster via using kubectl command.

To create scaledobject in namespace of application, the yaml file need to be config as shown in listing A.1. scaledobject is used to define the scaling behavior of the application. The scaling behavior is defined by the triggers. In this case, The kafka trigger is used to define the kafka consumer lag metric. For this example,

The scaledobject try to scale up the number of pod of deployment name `example-deployment` by polling the kafka consumer lag metric every `pollingInterval` seconds.

```yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: example-scaledobject
spec:
  scaleTargetRef:
    deploymentName: example-deployment
  pollingInterval: 15
  cooldownPeriod: 30
  minReplicaCount: 1
  maxReplicaCount: 10  # Max can go upto num of partitions you have in the topic
  fallback:
    failureThreshold: 10
    replicas: 1
  advanced:
    restoreToOriginalReplicaCount: true
  triggers:
  - type: kafka
    metadata:
      topic: example-topic
      bootstrapServers: kafka-bootstrap:9092
      consumerGroup: example-consumer-group
      lagThreshold: "1"
      activationThreshold: "0"
      sasl: plaintext
      tls: enable
      authenticationRef:
        name: keda-kafka-credentials
```

Listing 4.1: KEDA ScaledObject yaml file

The follow explain the meaning of each field in the scaledobject yaml file.

1. `scaleTargetRef`: The deployment name that need to be scale up or scale down
2. `pollingInterval`: The interval time to poll the kafka consumer lag metric
3. `cooldownPeriod`: The time to wait before the next scaling operation
4. `minReplicaCount`: The minimum number of pod of deployment
5. `maxReplicaCount`: The maximum number of pod of deployment to scale up
6. `failureThreshold`: The number of failure before the fallback to the original number of pod
7. `replicas`: The number of pod of deployment to fallback
8. `restoreToOriginalReplicaCount`: The flag to restore the number of pod of deployment to original number of pod when ScaledObject is deleted
9. `triggers`: The trigger that define the scaling behavior of the application
10. `type`: type of trigger
11. `topic`: kafka topic name
12. `bootstrapServers`: kafka bootstrap server
13. `consumerGroup`: kafka consumer group

14. `lagThreshold`: kafka consumer lag threshold
15. `activationThreshold`: kafka consumer lag activation threshold
16. `sasl`: The kafka sasl type
17. `tls`: The kafka tls type
18. `authenticationRef`: The kafka authentication reference

To let KEDA access consumer lag metric from Confluent Kafka cluster, The TriggerAuthentication need to be created in the namespace of application to authenticate the KEDA to access the kafka cluster. The yaml file need to be config as shown in listing 4.2.

```yaml
kind: TriggerAuthentication
apiVersion: keda.sh/v1alpha1
metadata:
  name: secret-triggerauthentication
  namespace: my-namespace
spec:
  secretTargetRef:
  - parameter: user-name
    name: my-secret
    key: USER_NAME
  - parameter: password
    name: my-secret
    key: USER_PASSWORD
```

Listing 4.2: KEDA TriggerAuthentication yaml file

The follow explain the meaning of each field in the TriggerAuthentication yaml file.

1. `metadata.name` : The name of TriggerAuthentication object
2. `secretTargetRef` : The secret reference that need to be used to authenticate the KEDA to access the kafka cluster
3. `parameter` : The parameter that need to be used to authenticate the KEDA to access the kafka cluster
4. `name` : The name of secret that need to be used to authenticate the KEDA to access the kafka cluster
5. `key` : The key of secret that need to be used to authenticate the KEDA to access the kafka cluster

4.3.2.1 Implement Auto Pod Scaling Plan on CFM Application

For the CFM application, There is a scenario in which the number of messages that messages ingest to the CFM application is higher than it used to be. In this case, There will be a consumer lag that occurs in Kafka consumers. In order to solve the problem of consumer lag faster, The KEDA will be used to implement horizontal pod autoscaling based on the Kafka consumer lag metric at aggregator service and email service (Figure 4.9). So If there is a consumer lag more than the threshold, The KEDA will trigger the horizontal pod autoscaling to increase the number of pods of aggregator service and email service. This will help the aggregator service consume the message from the Kafka topic faster. For ingestion service, there is no need to implement horizontal pod autoscaling because it might be a case that the ingestion service ingests the duplicate message to the CFM application, and for the cb-adapter service there is no need to implement horizontal pod autoscaling because the cb-adapter service has rate limits to post messages with ClaraBridge API.

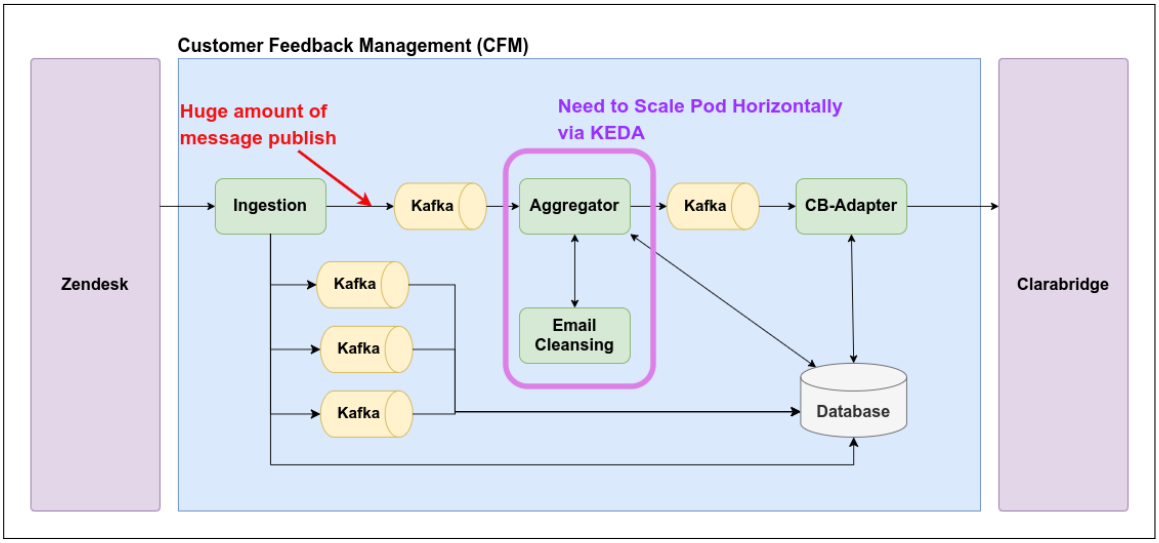


Figure 4.9: Scenario to scaling pod horizontally in CFM application

4.3.2.2 Implement Auto Pod Scaling Plan on CPC Application

For the CPC application, There is a flow in that external application 1 publishes a huge amount of messages to the Kafka topic. This flow occurs on a daily basis but the amount of messages that external application 1 publishes to the Kafka topic is unpredictable, So there might be some times when the number of messages that external application 1 publishes to the Kafka topic is higher than the number of messages that CPC application can consume. When this case occurs, The consumer lag will occur in the kafka consumer at CPC service. In order to solve this problem, The KEDA will be used to implement horizontal pod autoscaling based on the kafka consumer lag metric at CPC service (Figure 4.10). for messages published from external applications 2,3, There is no need to implement horizontal pod autoscaling because the number of messages that external applications 2,3 publish to the kafka topic is not high. On a daily basis, The number of messages that external application 2,3 publish to the Kafka topic is less than 1,000 messages per day while messages published from external application 1 is about 100,000-10,000,000 messages per day.

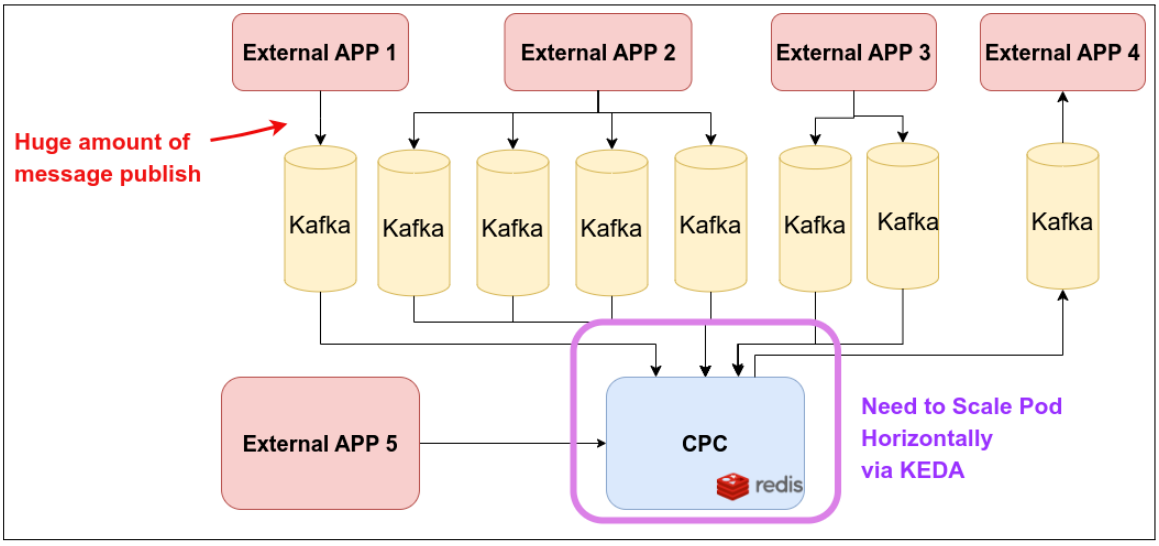


Figure 4.10: Scenario to scaling pod horizontally in CPC application

4.3.2.3 Implement Auto Pod Scaling Plan on AOD Application

For the AOD application, There is a flow in the Data Adapter service that publishes a huge amount of messages to the Kafka topic between the CSLO and Data Adapter. This flow occurs based on the action that the sales team creates or updates the customer offer from the web application (Web-AOD). These actions trigger from web-AOD to BFF service. Then BFF service will send the API to the backoffice service. The back office service will publish the message to the Kafka topic. The Data Adapter service will consume the message from the Kafka topic and publish the message to the Kafka topic to the CSLO service. The number of messages that the Data Adapter service will be based on the number of customers that were affected by the new offer. So there might be some time when the number of messages that the Data Adapter service publishes to the Kafka topic is higher than the number of messages that the CSLO application can consume. When this case occurs, The consumer lag will occur in the Kafka consumer at CSLO service. In order to solve this problem, The KEDA will be used to implement horizontal pod autoscaling based on the Kafka consumer lag metric at CSLO service (Figure 4.11).

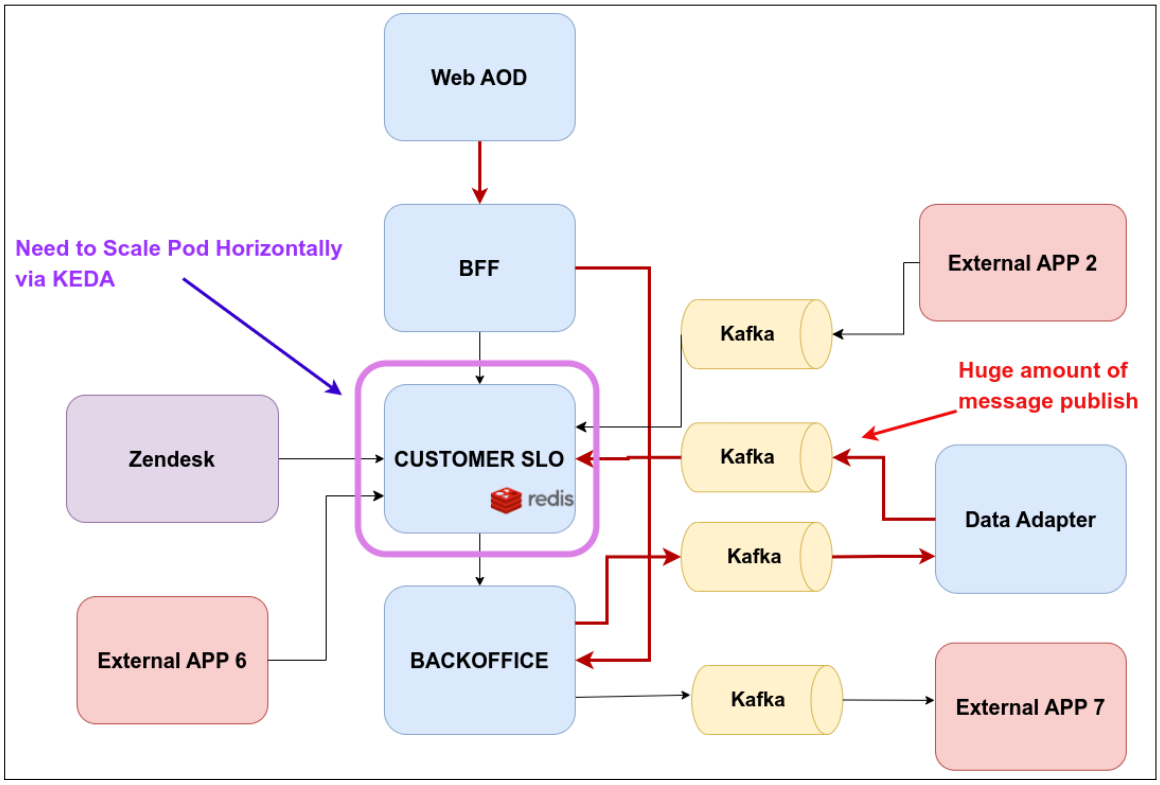


Figure 4.11: Scenario to scaling pod horizontally in AOD application

4.3.3 Alerting System Feature

The alert system help the team to detect the problem of application. So when error occurs in some of the application, the alert system will send the alert to the developer in team via email message to notify the situations. The alert system will be setup by using Datadog monitoring feature. The alerts have been setup base on the flaw of each application. Table 4.5 show the case of alerting system and the application that need to be alert.

Table 4.5 Case of alerting system

Type Alert	Application and Service	Event to Alert Description
Pod down	All application	If the pod is down
PostgreSQL Server	CPC	If the storage usage of PostgreSQL Server is more than 80%
PostgreSQL Server	AOD - Backoffice	If the storage usage of PostgreSQL Server is more than 80%
CosmosDB	AOD - CSLO	If the comosDB Request Units eceed the threshold
Kafka consumer lag	CFM	If the consumer lag at CFM-aggregator exceed the threshold
Kafka consumer lag	CPC	If the consumer lag from external application 1 is more than 1,000,000 messages
Kafka consumer lag	AOD - CSLO	Error on receive message from kafka topic.
Kafka error from publish message	AOD - Backoffice	Error on publish message to kafka topic
Kafka error from receive message	CPC	Error on receive message from kafka topic.
fail request between email cleansing and aggregator service	CFM - email cleansing and aggregator	Request timeout and error from clean email message from email cleansing service.
fail request cb-adapter	CFM - Cb-adapter	Request fail to send the message to ClaraBridge.

4.3.4 Load Test Feature

To test the performance of the application, The load test will be conducted by using K6 as a load testing tool. This evaluation will focus exclusively on API load testing, with the top 10 most frequently used APIs in each service being the primary target. The load test will be conducted only in the UAT environment of each service. After analysis service in each application, It has been determined that 6 services need to implement load test feature which include aod-bff, aod-backoffice, aod-cslo, cpc, smt-bff, smt-configapp.

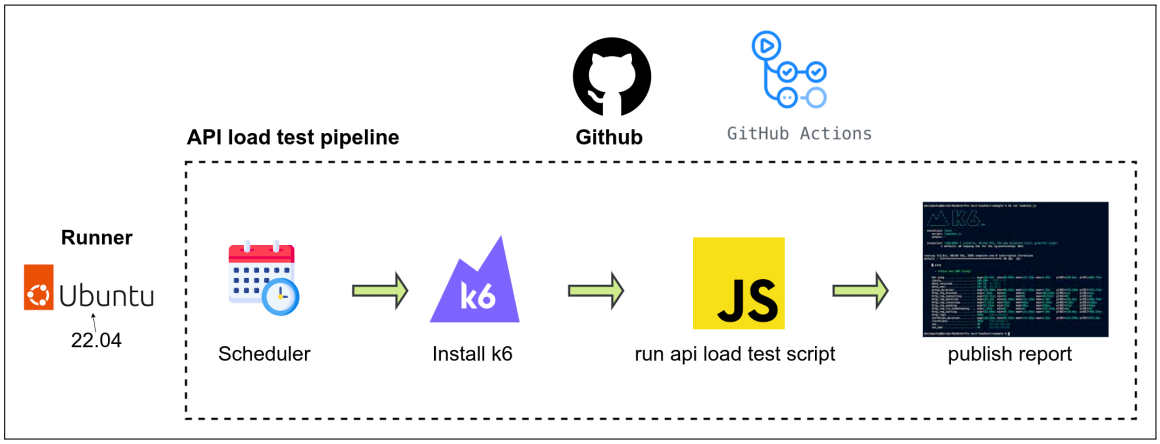


Figure 4.12: Pipeline design for running API load test using K6 on Github Actions

To execute the load test, GitHub Actions will be used as a process for Continuous Integration (CI) to run the load test on a biweekly basis. This setup will allow the team to monitor the performance of the application and detect the problems of the application faster. The result of the load test will be uploaded to the artifact of GitHub Actions. Figure 4.12 shows the plan to run an API load test using K6 on GitHub Actions. For the API load test pipeline, Ubuntu version 22.04 will be used as a runner for the load test. The following steps will be used to run the API load test using K6 on GitHub Actions.

1. Schedule trigger workflow on Github Actions
2. Install K6 in Github Actions snd setup the K6 configuration file
3. Run the load test script on Github Actions
4. upload the result of load test to the artifact of Github Actions

4.4 Evaluation Plan

4.4.1 Email Data Cleansing Evaluation

for Email Data Cleansing evaluation, The evaluation will be divided into 3 parts:

1. The correctness of the email data cleansing algorithm.

   The correctness of the email data cleansing algorithm. the algorithm remove the disclaimer and noise sentence from email messages correctly. This correctness is evaluated by using 1000 email messages from real data record on august 2023. The result of email data cleansing algorithm will be compare between previous email data cleansing algorithm and new approach of email data cleansing algorithm. This include correctness of LSTM model and correctness of masking PII data.

2. The speed of the email data cleansing algorithm.

   To evaluate the speed performance of new approach and previous email data cleansing algorithm, The test has been done by using 1,000 email messages from real data record on august 2023. The test by conducted on a MacBook Pro 2018 with 2.2 GHz 6-Core Intel Core i7 and 16 GB 2400 MHz DDR4.

3. Vulnerability alert.

   The number of vulnerability alert from dependabot from Github. The dependabot is a service that automatically check the security vulnerability of the project. If the project has security vulnerability, the dependabot show vulnerability alert on Github. The number of vulnerability alert will be compare between previous email data cleansing algorithm and new approach of email data cleansing algorithm.

4.4.2 Monitoring Service of Application Resources Evaluation

In order to evaluate the monitoring service of application resources, The evaluation is considered to satisfaction of the developer and business analyst in the customer service team who use the dashboard to monitor the application. The evaluation will be conducted by using a questionnaire to collect the feedback from developer and business analyst. For the auto pod scaling feature, The evaluation will be completed that the pod is scaled up and scaled down correctly based on the Kafka consumer lag metric. For the load test feature, The evaluation will be completed that the load test is conducted correctly and the result of the load test is correct.

CHAPTER 5 IMPLEMENTATION RESULTS

5.1 Implementation of Data Collecting Algorithm

After implementing the data collecting algorithm by writing python script to collect data from Zendesk system, The parameters of data collecting algorithm including `verbrate_threshold` and `occurance_threshold` has been tested to find the best parameters for data collecting algorithm. The result show that at parameters `verbrate_threshold` = 0.1 and `occurance_threshold` = 10 fit to the data. when `verbrate_threshold` is less than 0.1, the sentence tends to be unsentence such as url link, email address and etc. The distribution of sentence occurance won't be shown in this report because the confidential of data. The result shows that most of the disclaimer sentences have a higher occurrence than normal sentences, but there are some normal sentences that have a high occurrence as well. In order to collect the data for training the model, the data needs to be filtered by hand to ensure that the data is collected correctly. There are some issues related to business requirements, such as identifying whether a sentence is a normal sentence or a disclaimer sentence is difficult. For example a sentence related to instructions located on the footer of email message. So it can be identified as both a disclaimer sentence and a normal sentence at the same time. When this issue occurs, the team has to contact to business user to confirm that this sentence is a disclaimer sentence or a normal sentence. Another issue is the lack of disclaimer sentence data for the training model, This is because Normally, when people send emails to the organization, they will not change their disclaimer footer or signature. This led to few amount of disclaimer sentence data for the training model. After apply Data Collecting Algorithm on raw email message on may-june 2023. There are only 4,700 disclaimer sentences and 110,000 normal sentences. To solve the imbalance data problem, the undersampling method has been used to decrease the number of normal sentences to approximately 5,000. After collecting the data for training the model, the data was split into two parts: the train dataset and the validation dataset. 70% of whole data has used for training LSTM and 30% of whole dataset using for validate the performance of the algorithm.

5.1.1 Exploratory Data Analysis (EDA)

After collecting the data for the training model by applying the data collecting algorithm that is mentioned in section 5.1, The exploratory data analysis (EDA) has been conducted to understand the data. Figure 5.1 shows the ratio between several disclaimers and normal sentences. The result shows that the number of disclaimer sentences is a little bit less than a normal sentence. Figure 5.2 shows the distribution number of the word in one single line of the email message. The result shows that most of the words in one single line of the email message are between 0 to 200 words. The number of words in one single line of the email message that is a normal sentence is less than the disclaimer sentence. It was found that a single line of email message, containing 700 words, was the longest. There is also another explore data analysis (EDA) that has been done but due to the confidentiality of the data it won't be shown in this report.

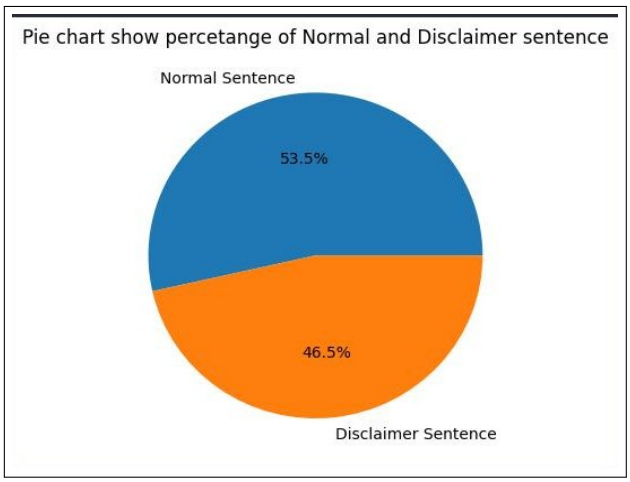


Figure 5.1: Pie chart shows ratio between number of disclaimer and normal sentence

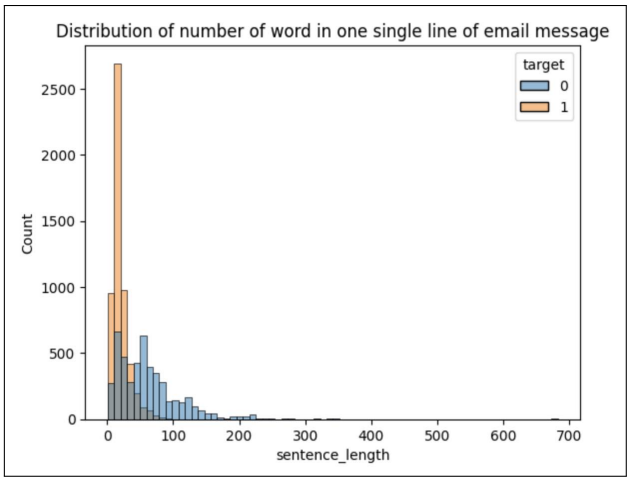
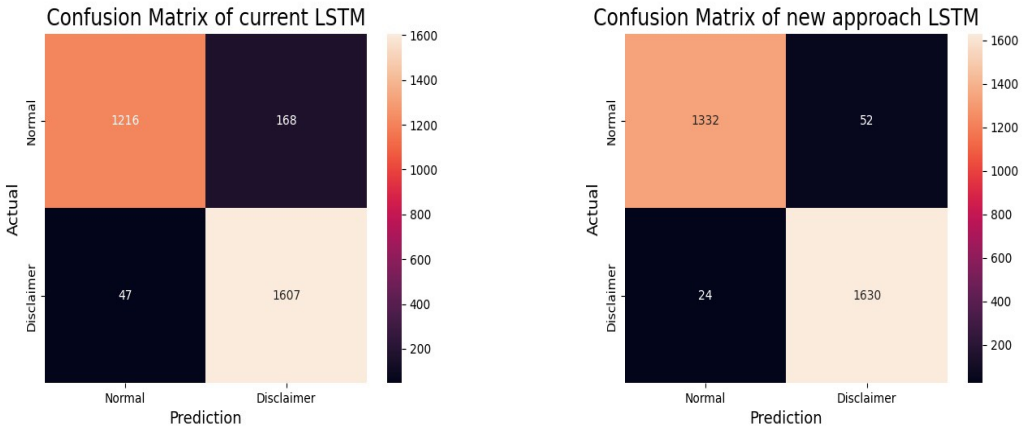


Figure 5.2: Distubition number of word in one single line of email message

5.2 Implementation of Email Data Cleansing Algorithm

After the implementation of the new email data cleansing algorithm, The result shows that the a new approach to Email Data Cleansing has better performance in terms of detecting disclaimers. But there are some cases where the new approach of email data cleansing algorithm cannot noise and disclaimer sentences from email messages. In most cases that model can not detect the noise or disclaimer are Obuscated disclaimer sentence. For example, some sentences contain the word “This email” followed by instructions that are not related to the disclaimer sentence. Figure 5.4 shows the example result of the new approach of email data cleansing algorithm with mock data. From the example result Figure 5.4, It can be seen that the new approach of email data cleansing algorithm can detect The sentence “If you have any problem please contact Tel 081-234-5678” is a disclaimer sentence and remove it from an email message. This is because the sentence is an Obsucated disclaimer sentence makes the model confused to detect whether the sentence is a disclaimer sentence or not. Another problem that makes the algorithm detect the false result comes from the normal sentence is removed in the process of removing the non-sentence, This occurs because the verb rate of the sentence is less than 0.1. To solve this problem, the verb rate threshold has been changed from 0.1 to 0.06. This lets the email message keep a more normal sentence than the previous version of the email data cleansing algorithm. To evaluate the

speed performance of the new approach and previous email data cleansing algorithm, The test was conducted by using 1,000 email messages from real data records in August 2023. The test by conducted on a MacBook Pro 2018 with 2.2 GHz 6-Core Intel Core i7 and 16 GB 2400 MHz DDR4. The result shows that the new approach to the email data cleansing algorithm is faster than the previous version of the email data cleansing algorithm. The previous version of the email data cleansing algorithm took 4,058.46 seconds to process 1,000 email messages (4.05 seconds per email message), while the new approach of the email data cleansing algorithm takes 590.45 seconds to process 1,000 email messages (0.59 seconds per one email message). The reason that the new approach to the email data cleansing algorithm is faster than the previous version of the email data cleansing algorithm is because the new approach of the email data cleansing algorithm employed Spacy to calculate verb rate in sentences while the previous version of the email data cleansing algorithm employed Stanza to calculate verb rate in sentences. This transition improves speed performance due to Spacy name entity recognition being faster than Stanza [14]. Also, the new approach of email data cleansing algorithm removes processes that redundant



(a) Current LSTM

(b) New approach LSTM

Figure 5.3: Confusion matrix of LSTM model

Figure 5.4: Mock email example of runing new approach of email data cleansing algorithm

5.2.1 Personal Identifiable Information Masking Performance

The evaluation of the effectiveness of masking Personal Identifiable Information (PII) data involves comparing the accuracy of PII data masking and the satisfaction of business users between the existing email data cleansing algorithm and a new approach to email data cleansing algorithm. figure 5.5 shows a comparsion result of masking PII data between stanza, spacy and presidio library. The test data is a mock data that created

for testing purpose. by testing with 15 sentences that contain PII data. The result shows that the presidio library has better performance in term of correctness masking PII data. This is because presidio library consider the context, regex, name entity recognition while spacy and stanza consider only pattern check and name entity recognition perspective. Some of phone number is not detected by algorithm 1 because the phone number is not in the pattern of phone number. For example that phone number using . (dot) instead of - (dash) to separate the number.

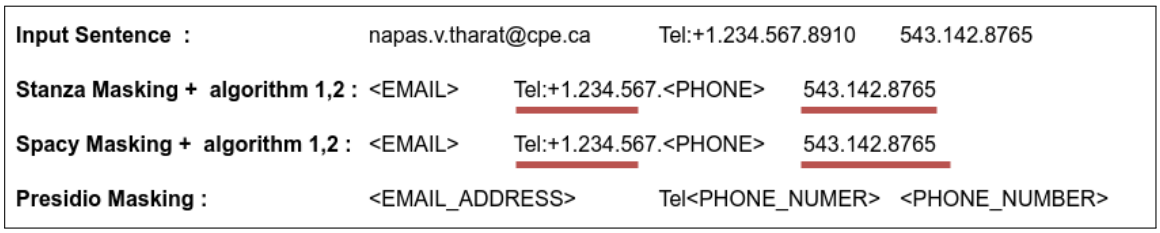


Figure 5.5: Comparsion of masking pii data between stanza, spacy and presidio library

In terms of business users satisfaction, it has been observed that masking PII data using Stanza and Spacy is perceived as difficult to understand by business users. For example, when some words of the sentence are masked as a GPE placeholder, they are often confused and unsure about its meaning. This is because the business users are not familiar with the concept of Named Entity Recognition (NER) and the meaning of the placeholder. On the other hand, business users are more satisfied with the result of masking PII data using the Presidio library because the Presidio library has placeholder labels that are easier to understand. Thus, the Presidio library is selected as the algorithm for masking PII data.

5.2.2 Implement of LSTM Model

After implementing the new version of the LSTM model, 30The results show that the new LSTM model has satisfactory results, as shown in Table 5.1 and Figure 5.3. The new LSTM model exhibits improved performance in accuracy, precision, recall, and F1 score compared to the previous LSTM model. However, the enhancements are not significantly pronounced when compared to the previous LSTM model. The reason for the better performance of the new LSTM model compared to the previous LSTM model is that the new LSTM model has more new data for training than the previous LSTM model. By investigating the results after testing with data from August 2023.

Table 5.1 Comparison of performance of LSTM models

Metric result	Previous LSTM Model	New Approach LSTM Mode
Accuracy	0.9292	0.9750
Precision	0.8786	0.9624
Recall	0.9628	0.9823
F1 Score	0.9188	0.9723

It has been observed that in some cases, the model cannot detect disclaimer sentences. Most of the cases where the model fails to detect the disclaimer are obfuscated disclaimer sentences. For example, some sentences contain the phrase “This email” followed by instructions that are not related to the disclaimer sentence. This confuses the model and makes it difficult to classify the sentence as a disclaimer or a normal sentence. Some normal sentences are also being removed by the new LSTM model. To address this issue, the threshold has been adjusted to be slightly biased towards normal sentences. This allows the email message to retain more normal sentences.

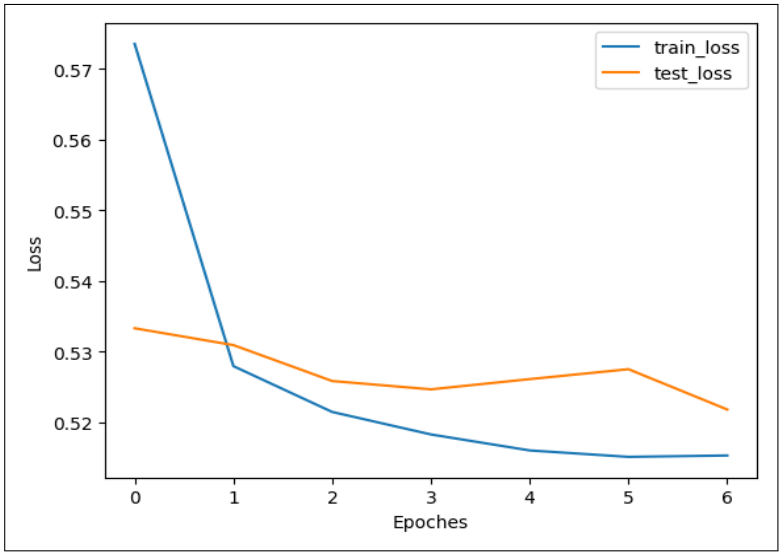


Figure 5.6: Loss of LSTM model training in each epoch

Figure 5.6 shows the loss of LSTM model training in each epoch. In the training loss graph, the loss of the model is decreasing in each epoch. While the validation loss graph, the loss of the model is decreasing in each epoch until epoch 4 and then the loss is increasing in epoch 5 and decrease in epoch 6. Figure 5.7 shows the accuracy of LSTM model training in each epoch. In the training accuracy graph, the accuracy of the model is increasing in each epoch. While the validation accuracy graph, the accuracy of the model is increasing in each epoch until epoch 4 and then the accuracy is decreasing in epoch 5 and increase in epoch 6.

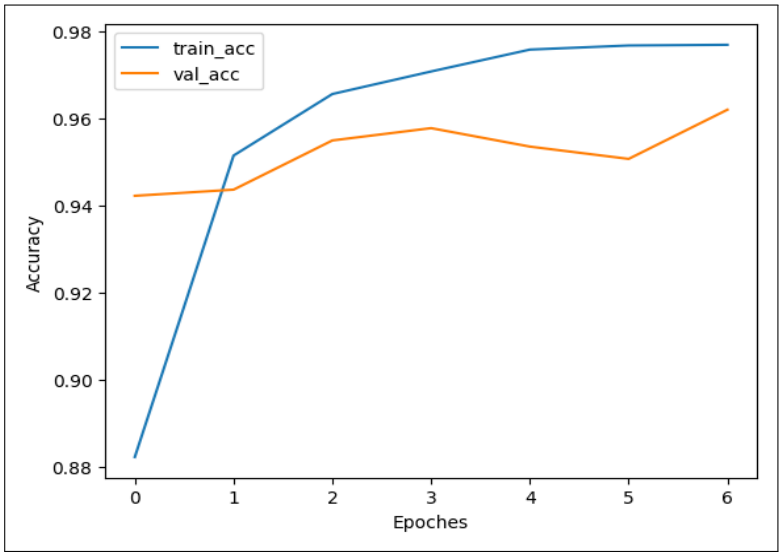


Figure 5.7: Accuray of LSTM model training in each epoch

5.2.3 Vulnerability Scaning Result

To evaluate the vulnerability of the Email Data Cleansing service, the number of vulnerability alerts from dependabot on GitHub is used. Figure 5.8 shows the number of vulnerability alerts from dependabot on GitHub in October 2023. The results show that the previous Email Data Cleansing service had 407 vulnerability alerts. Most of the vulnerability alerts were related to deprecated Python libraries such as Tensorflow and

Keras. After implementing the new Email Data Cleansing service and upgrading the deprecated Python libraries to newer versions, the number of vulnerability alerts from dependabot has been successfully reduced to zero.

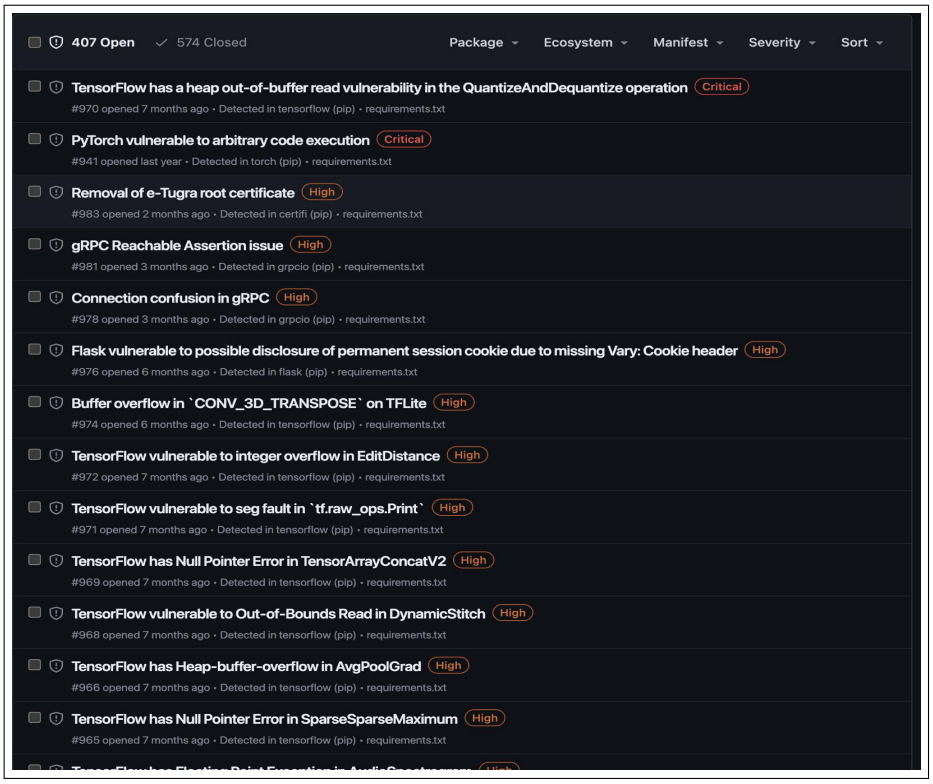


Figure 5.8: Vulnerability of previous Email Data Cleansing service

5.3 Monitoring Service of Application Resources

After implementing the monitoring service of the application resources, The plan was changed to implement another dashboard named overview dashboard. This dashboard will be used to view the overview of all applications in a single dashboard While the dashboard of SMT and CFM application resources and the dashboard of AOD and CPC application resources will be used to investigate the problem in more detail. The schedule report dashboard of the datadog feature had been used to send a visual summary of an overview dashboard to the developer and business analyst in the customer service team on a weekly basis. In terms of the usage of the dashboard, It has been observed that the solution offered is not considered a suitable developer within the team. This is because the developer team is not familiar with the use of a dashboard and in some cases, the dashboard does not show the details of the problem, So the developer uses log filtering to investigate the problem instead of using the dashboard. On the other hand, business analysts and QA in the customer service team are more often to use the dashboard to monitor the application resources. Specifically, the dashboard that visualize the number of messages that the service publishes or consumes from the Kafka topic is used to monitor whether the service is working correctly or not.

5.3.1 Overview Dashboard

5.3.1.1 Alerting System

Figure 5.9 shows the dashboard of integrated with the alerting system. It will shows a time series line graph of the nubmers of pod current running in the system. If the number of pods are less than the threshold. It will

send the alerting email to a corresponding team members. This dashboard is useful to use as an overviewing tool because it only shows the siginificance alerts of every applications.

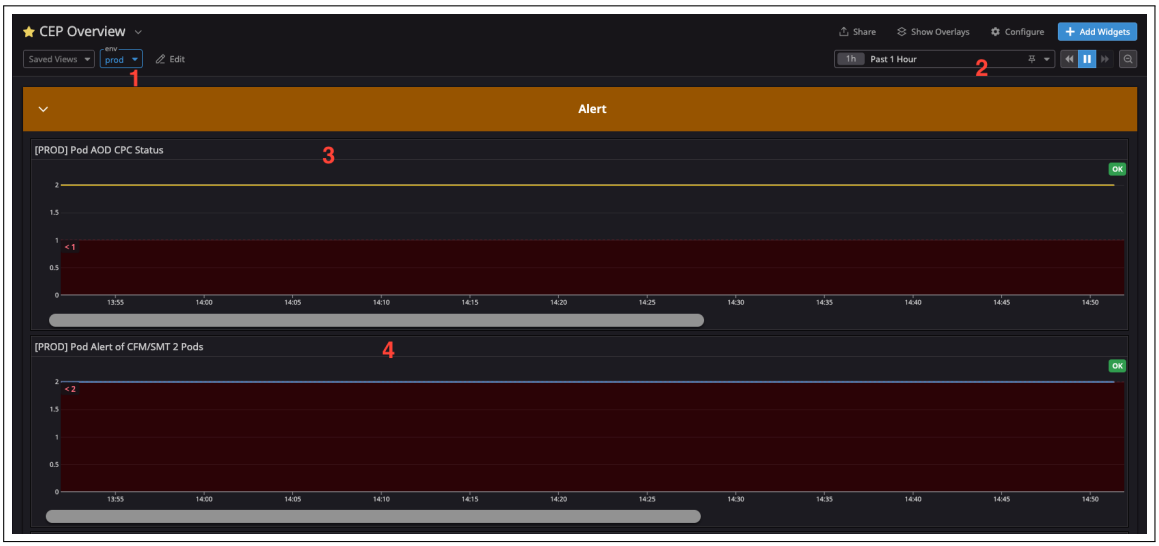


Figure 5.9: Dashboard shows the alerts in the applications

The overview dashboard alerting system consists of 4 parts as follows:

1. Environment dropdown

   The environment dropdown is used for select the environment of application. The environments are spilted into 4 environments which are development (DEV), quality assurance (QA) ,user acceptance testing (UAT) and production (PROD).

2. Time dropdown

   The time dropdown is used for select the time range of dashboard. The time range are spilted into different range such as last 15 minutes, last 30 minutes, last 1 hour, last 4 hours, and etc.

3. Pod Status of AOD application

   The time series line graph alert in case the number of pods running in AOD applications decrease below a certain threshold then, it will alert the team members by sending email as well as this time series line graph.

4. Pod Status of CFM and SMT application

   The time series line graph alert in case the number of pods running in CFM and SMT applications decrease below a certain threshold then, it will alert the team members by sending email as well as this time series line graph.

5.3.1.2 Request API Metrics

Figure 5.10, Figure 5.11 show the dashboard of API metrics and resources usage of the applications. The dashboard is spilt into 2 parts. The first part shows request hit and status code of API endpoint of the applications. The second part shows the resources usage of the application This dashboard is useful to use as an overviewing tool because it only shows the siginificance API metrics of every applications.

Figure 5.10: Dashboard shows the status code of api endpoint of the applications

The first part of overview dashboard request API metrics consists of 7 parts as follows:

1. Pie chart of https status code of API endpoint in every applications

   This pie chart shows the number of http status code of every applications in a certain time range. If the status code 5xx or 4xx is too high, there might be some problem occur in the application.

2. Pie chart of https status code of API endpoint in AOD backend for frontend application.

   This pie chart shows the number of http status code of AOD backend for frontend application in a certain time range. If the status code 5xx or 4xx is too high, there might be some problem occur in the application.

3. Pie chart of https status code of API endpoint in Core CPC application.

   This pie chart shows the number of http status code of core CPC application in a certain time range. If the status code 5xx or 4xx is too high, there might be some problem occur in the application.

4. Pie chart of https status code of API endpoint in SMT config management application.

   This pie chart shows the number of http status code of SMT config management in a certain time range. If the status code 5xx or 4xx is too high, there might be some problem occur in the application.

5. Pie chart of https status code of API endpoint in AOD backoffice application.

   This pie chart shows the number of http status code of AOD backoffice in a certain time range. If the status code 5xx or 4xx is too high, there might be some problem occur in the application.

6. Pie chart of https status code of API endpoint in AOD customer service level offer application.

   This pie chart shows the number of http status code of AOD customer service level offer application in a certain time range. If the status code 5xx or 4xx is too high, there might be some problem occur in the application.

7. Pie chart of https status code of API endpoint in Zendesk backend for frontend application.

   This pie chart shows the number of http status code of Zendesk backend for frontend application in a certain time range. If the status code 5xx or 4xx is too high, there might be some problem occur in the application.

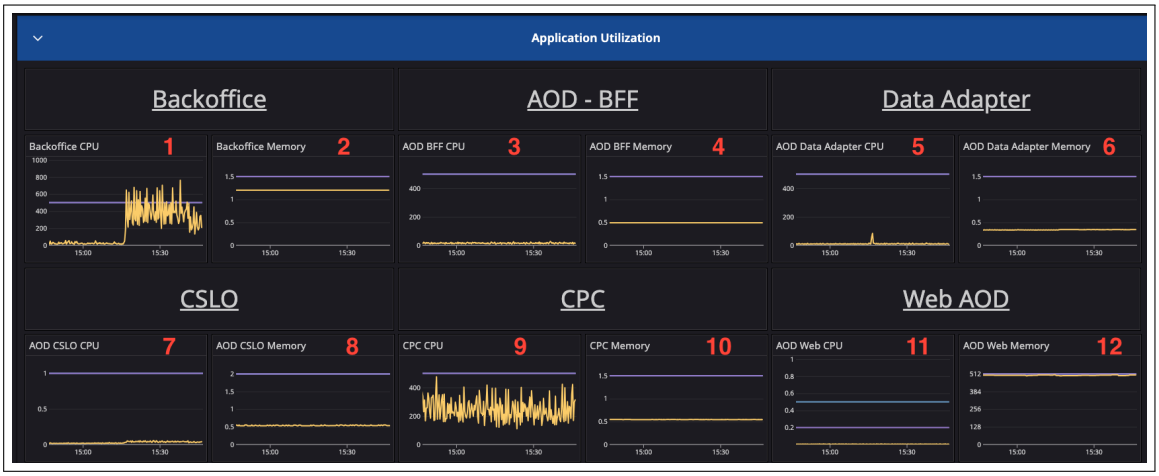


Figure 5.11: Dashboard shows the resources usage of every application

The second part of overview dashboard request API metrics consists of 12 parts as follows:

1. AOD backoffice CPU usage

   The time series line chart that shows the CPU usage in a certain period of time of AOD backoffice application.

2. AOD backoffice memory usage

   The time series line chart that shows the memory usage in a certain period of time of AOD backoffice application.

3. AOD backend for frontend CPU usage

   The time series line chart that shows the CPU usage in a certain period of time of AOD backend for frontend application.

4. AOD backoffice memory usage

   The time series line chart that shows the memory usage in a certain period of time of AOD backend for frontend application.

5. AOD data adapter CPU usage

   The time series line chart that shows the CPU usage in a certain period of time of AOD data adapter application.

6. AOD backoffice memory usage

   The time series line chart that shows the memory usage in a certain period of time of AOD data adapter application.

7. AOD customer service level offer CPU usage

   The time series line chart that shows the CPU usage in a certain period of time of AOD customer service level offer application.

8. AOD customer service level offer memory usage

   The time series line chart that shows the memory usage in a certain period of time of AOD customer service level offer application.

9. Core CPC CPU usage

   The time series line chart that shows the CPU usage in a certain period of time of core CPC application.

10. core CPC memory usage

    The time series line chart that shows the memory usage in a certain period of time of core CPC application.

11. AOD web CPU usage

    The time series line chart that shows the CPU usage in a certain period of time of AOD web application.

12. AOD backoffice memory usage

    The time series line chart that shows the memory usage in a certain period of time of AOD web application.

5.3.1.3 Databases Metrics

Figure 5.12, Figure 5.13 show the dashboard databases in every applications. The dashboard is spilt into 2 parts. The first part shows all the key important metrics of PostgreSQL database. The second part shows all the key important metrics of cosmosDB database. This dashboard is useful to use as an overviewing tool because it only shows the siginificance database metrics of every applications.

Figure 5.12: Dashboard shows metrics of PostgresSQL databases

The overview dashboard PostgreSQL metrics consists of 8 parts as follows:

1. AOD Backoffice storage usage

   The time series line graph that shows the storage usage in % AOD backoffice application.

2. AOD Backoffice memory usage

   The time series line graph that shows the memory usage in % AOD backoffice application.

3. core CPC storage usage

   The time series line graph that shows the storage usage in % core CPC application.

4. core CPC memory usage

   The time series line graph that shows the memory usage in % core CPC application.

5. CFM storage usage

   The time series line graph that shows the storage usage in % CFM application.

6. CFM memory usage

   The time series line graph that shows the memory usage in % CFM application.

7. SMT storage usage

   The time series line graph that shows the storage usage in % SMT application.

8. SMT memory usage

   The time series line graph that shows the memory usage in % SMT application.

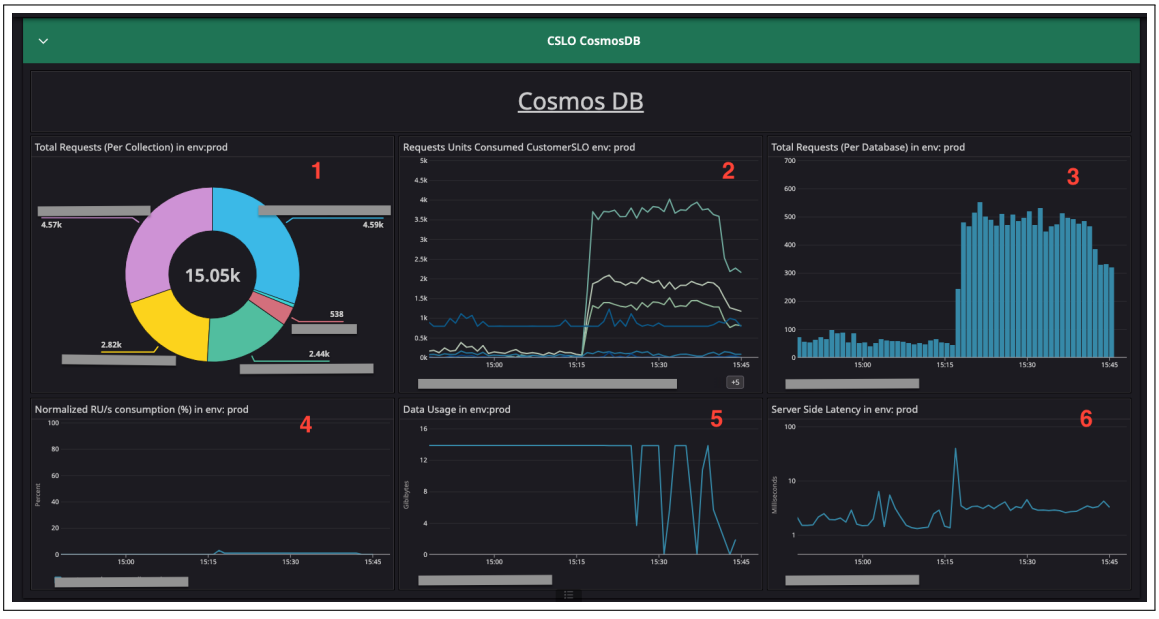


Figure 5.13: Dashboard shows metrics of cosmosDB databases

The overview dashboard CosmosDB metrics consists of 6 parts as follows:

1. Total Requests of Azure cosmosDB database in pie chart format

   This part shows the total request of Azure cosmosDB database in pie chart format. It helps to understand which collection of database has the most used.

2. Total Requests of Azure cosmosDB database in time series format

   This part shows the total request of Azure cosmosDB database in time series format. It helps to understand which collection of database has the most used.

3. Total Requests per database in time series format

   This part shows the total request base on the databases of Azure cosmosDB. It helps to understand which period of time that the database has the most load in a day.

4. Normalizes RU/s Consumption (%) Normalizes RU/s Consumption (%) is the percentage of the provisioned throughput that is consumed by the request.

5. Data Usage

   This part shows time series of total storage usage in gigabytes. It helps to identify the storage usage in order to make some adjustment for further optimization in case the usage is too high.

6. Server side latency

   This part shows server side latency in a times series format. So that it is easy to investigate when there is a problem related to the request time out of databases.

5.3.1.4 Kafka metrics

Figure 5.14, Figure 5.15 show the dashboard of Kafka of every applications. The dashboard is spilt into 2 parts. The first part shows Kafka lag in the topic of every applications. The second part shows custom metrics Kafka error metrics of every applications. This dashboard is useful to use as an overviewing tool because it only shows the siginificance Kafka metrics of every applications.

Figure 5.14: Dashboard shows the Kafka log of applications

The first part of overview dashboard Kafka metrics consists of 6 parts as follows:

1. Lag offset of every topics in every applications

   It shows the current lag offset of every topic in every applications. It also shows the consumer Id as well as the topic and the lag offset. It helps on the monitoring by containing every topic to a single table for a easier monitor.

2. Lag offset of CFM

   It shows the current lag offset of CFM Kafka topics. It also shows the partitions with the lags for a easier investigate an issue related to Kafka.

3. Lag offset of SMT

   It shows the current lag offset of SMT Kafka topics. It also shows the partitions with the lags for a easier investigate an issue related to Kafka.

4. Lag offset of AOD customer service level offer

   It shows the current lag offset of AOD customer service level offer Kafka topics. It also shows the partitions with the lags for a easier investigate an issue related to Kafka.

5. Lag offset of AOD data adapter

   It shows the current lag offset of AOD data adapter Kafka topics. It also shows the partitions with the lags for a easier investigate an issue related to Kafka.

6. Lag offset of core CPC

   It shows the current lag offset of core CPC Kafka topics. It also shows the partitions with the lags for a easier investigate an issue related to Kafka.

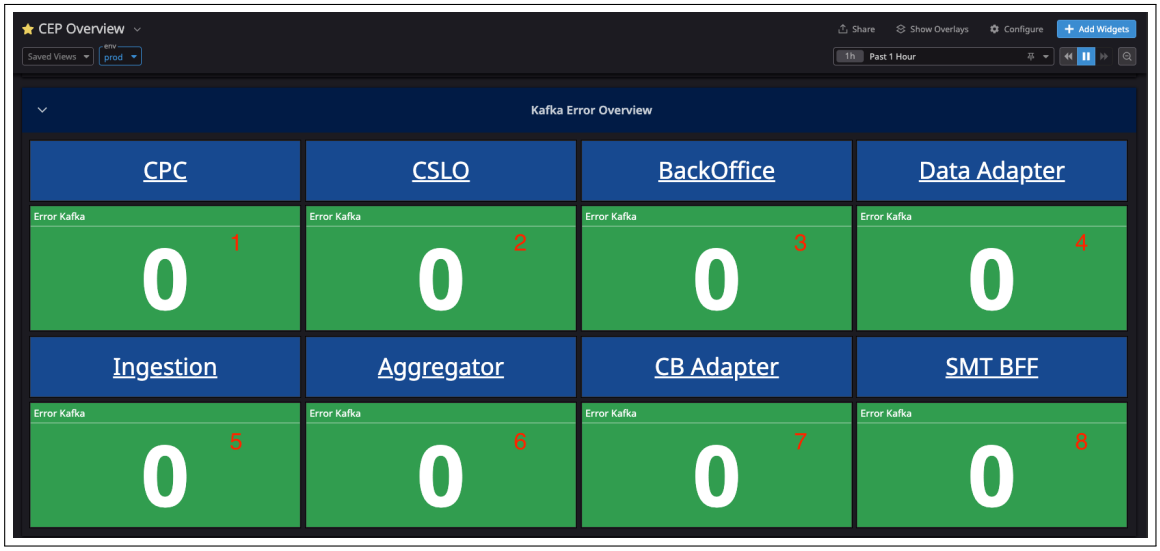


Figure 5.15: Dashboard shows the Kafka log of applications

The second part of overview dashboard Kafka metrics consists of 8 parts as follows:

1. Kafka error of core CPC

   It shows the custom metrics counter of the Kafka error in core CPC application. It shows the total errors of core CPC application in the current time frame.

2. Kafka error of customer service lever offer

   It shows the custom metrics counter of the Kafka error in customer service lever offer application. It shows the total errors of customer service lever offer application in the current time frame.

3. Kafka error of AOD backoffice

   It shows the custom metrics counter of the Kafka error in AOD backoffice application. It shows the total errors of AOD backoffice application in the current time frame.

4. Kafka error of AOD data adapter

   It shows the custom metrics counter of the Kafka error in AOD data adapter application. It shows the total errors of AOD data adapter application in the current time frame.

5. Kafka error of Zendesk ingestion

   It shows the custom metrics counter of the Kafka error in Zendesk ingestion application. It shows the total errors of Zendesk ingestion application in the current time frame.

6. Kafka error of CFM aggregator

   It shows the custom metrics counter of the Kafka error in CFM aggregator application. It shows the total errors of CFM aggregator application in the current time frame.

7. Kafka error of CFM CB adapter

   It shows the custom metrics counter of the Kafka error in CFM CB adapter application. It shows the total errors of CFM CB adapter application in the current time frame.

8. Kafka error of SMT backend for frontend

   It shows the custom metrics counter of the Kafka error in SMT backend for frontend application. It shows the total errors of SMT backend for frontend application in the current time frame.

5.3.2 Dashboard of SMT and CFM Application Resources

5.3.2.1 Request API Metrics

Figure 5.16, Figure 5.17 and Figure 5.18 show the dashboard of API metrics and logs of SMT and CFM application. The dashboard is spilt into 3 parts. The first part shows the latency, request hit, error rate of API endpoint and error log of applications. The second part shows logs of the service and the third part shows the https status code of API endpoint at the given time.

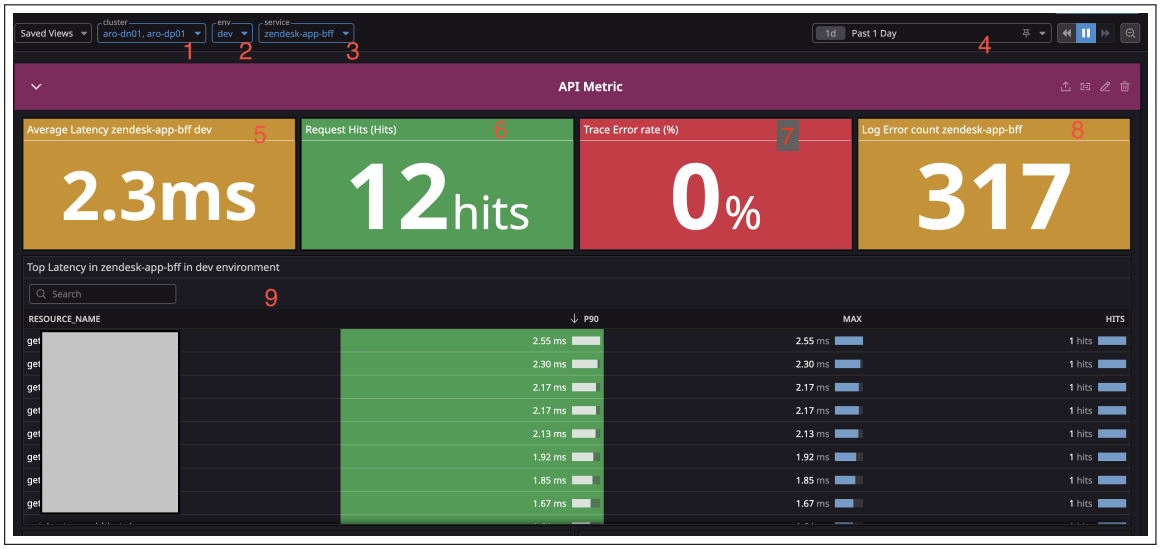


Figure 5.16: Dashboard shows the latency, request hit, error rate of API endpoint and error log of applications

The first part of request API metrics dashboard consists 9 parts as follows:

1. Cluster dropdown

   The cluster dropdown is used for select the cluster of application. The cluster is spilted into 2 clusters which are production (Prod) cluster and non-producttion (Non-prod) cluster.

2. Environment dropdown

   The environment dropdown is used for select the environment of application. The environments are spilted into 3 environments which are development (DEV), user acceptance testing (UAT) and production (PROD).

3. Service dropdown

   The service dropdown is used for select service of application. The service are spilted into 6 services which are Backend for Frontend (BFF), Config Management, Ingestion, Aggregator, CB-Adapter, Email-Cleansing.

4. Time dropdown

   The time dropdown is used for select the time range of dashboard. The time range are spilted into different range such as last 15 minutes, last 30 minutes, last 1 hour, last 4 hours, and etc.

5. Average latency of API endpoints

   This query value show the average latency of all API endpoints of service application in the given environment and time range.

6. Request hit of API endpoints

   This query value show the number of request of all API endpoints have been hit in the given environment and time range.

7. Error rate of applications

   This query value show the error rate of all API endpoints of service application in the given environment and time range. This metric indicate the number of error in application. If the error rate is too high, there might be some problem in application.

8. Error log of applications

   This widget show the count all errors log of the sercvice.

9. Top latency of API endpoints

   This table show the top p90 and p100 latency of API endpoints of service application in the given environment and time range. This table help the team to visual the latency of API endpoints. If the latency of API endpoints is too high, the team can investigate the problem of API endpoints or optimize the latency of API endpoints.

Figure 5.17: Dashboard shows the Log Pattern an Error log of the application

The second part of the request API Metrics dashboard consists of 2 parts as follows:

1. Log Errors

   This widget show all of error log of application in the given environment and time range.

2. Log pattern

   This widget show all the log pattern of the given environment.

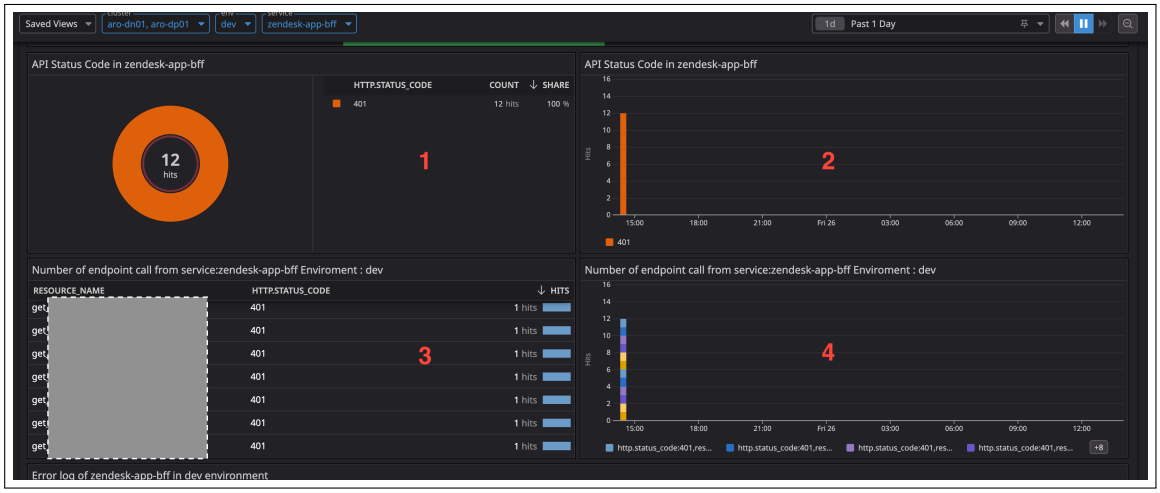


Figure 5.18: Dashboard shows the http status code of API endpoint in applications

The third part of the request API Metrics dashboard consists of 4 parts as follows:

1. Pie chart of http status code of API endpoints

   This pie chart shows the number of http status code of application in some time range. So If the number of 5xx http staus code or 4xx http status code is too high, There might be some problem occur in application.

2. Time series show number of http status code

   This time series show the number of http status code of application as a time series. This time series to identify the right time that the problem occur in application.

3. Table show number of http status code of each API endpoints

   This table show number of http status code with name of the endpoints. This table help team to identify which endpoint has http status code problem. This information can further use to investigate the problem of endpoint.

4. Time series of http status code of API endpoints

   This time series show the number of http status code with the name of endpoints as a time series.

5.3.2.2 Pod Utilization Metrics

Figure 5.19 shows the dashboard of Pod Utilization metrics of SMT and CFM application. User can select the cluster, environment, service and time range of dashboard. This dashboard help the team to monitor the resource usage of pod. This dashboard will be focus monitor the CPU and memory usage of pod.

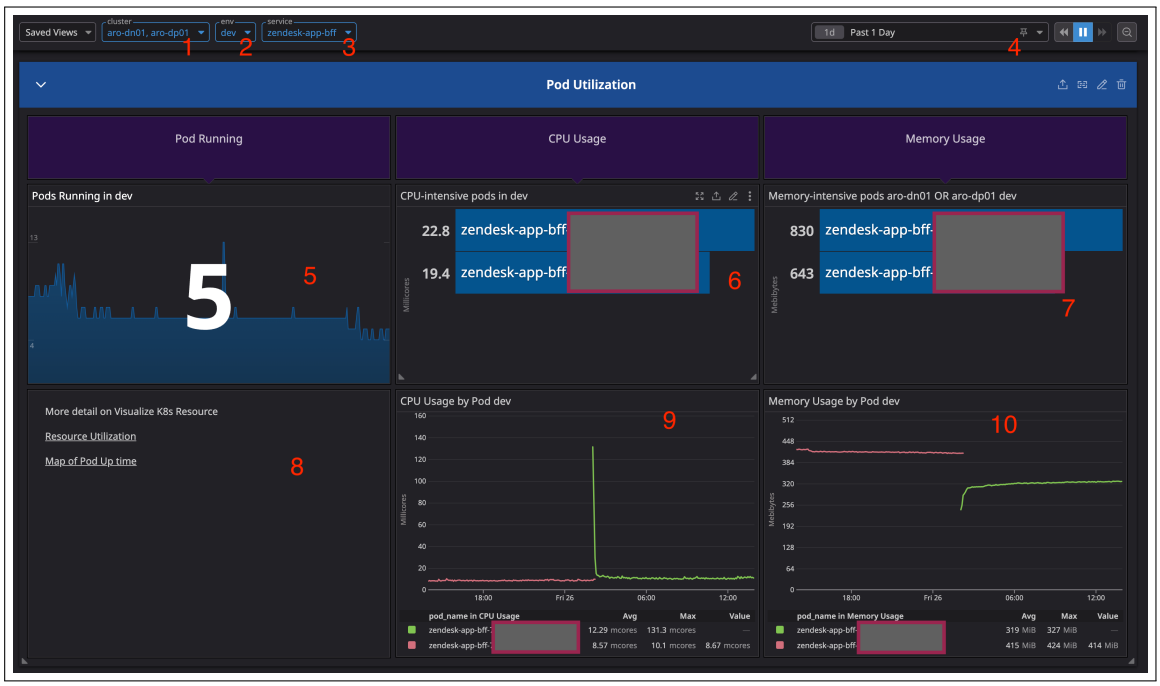


Figure 5.19: Dashboard shows Pod Utilization of SMT and CFM application

The dashboard of Pod Utilization metrics consists 10 parts as follows:

1. Cluster dropdown

   The cluster dropdown is used for select the cluster of application. The cluster is spilted into 2 clusters which are production (Prod) cluster and non-producttion (Non-prod) cluster.

2. Environment dropdown

   The environment dropdown is used for select the environment of application. The environments are spilted into 3 environments which are development (DEV), user acceptance testing (UAT) and production (PROD).

3. Service dropdown

   The service dropdown is used for select service of application. The service are spilted into 6 services which are Backend for Frontend (BFF), Config Management, Ingestion, Aggregator, CB-Adapter, Email-Cleansing.

4. Time dropdown

   The time dropdown is used for select the time range of dashboard. The time range are spilted into different range such as last 15 minutes, last 30 minutes, last 1 hour, last 4 hours, and etc.

5. Pod Running

   The current total pod running the namespace of the given environment.

6. CPU Intensive Pod

   The current total CPU usage of the service of the given environment.

7. Memory Intensive Pod

   The current total Memory usage of the service of the given environment.

8. Link to other Resource

   The clickable link which will redirect to the external dashboard that contains a useful information about the POD.

9. CPU Usage Pod

   The time series CPU usage of the service of the given environment.

10. Memory Usage Pod

    The time series Memory usage of the service of the given environment.

5.3.2.3 PostgreSQL Database Metrics

Figure 5.20 and Figure 5.21 shows the dashboard of Postgres database metrics of SMT and CFM application. The dashboard is spilt into 2 parts. The first part shows the resource usage of PostgreSQL database server and the second part shows the storage usage and performance of PostgreSQL database server.

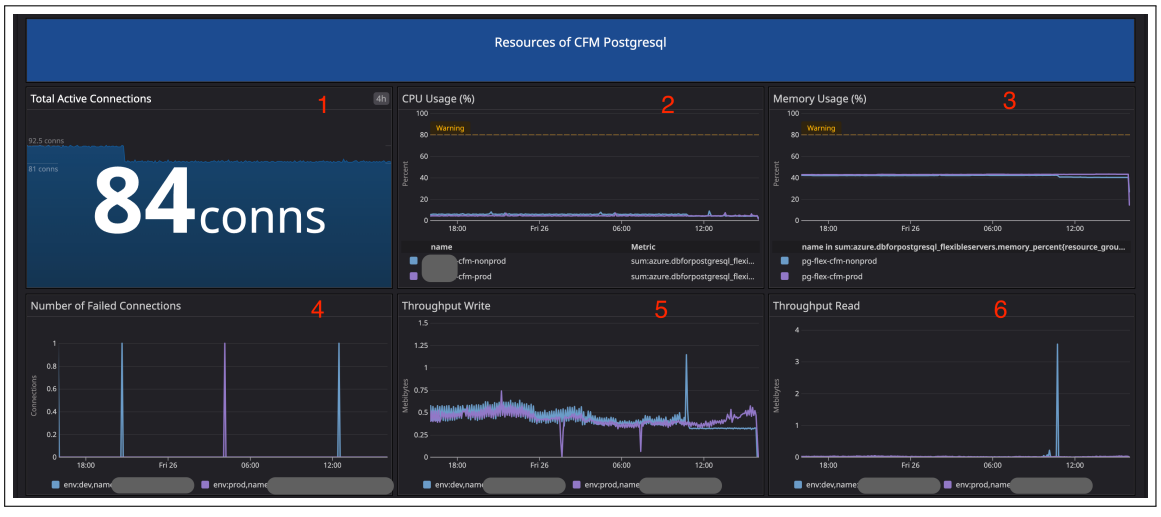


Figure 5.20: Dashboard shows resource usage of CFM PostgreSQL server

The first part of Postgres database metrics dashboard consists 6 parts as follows:

1. Number of Active Connections

   This metric represents the number of active connections to the PostgreSQL database server. It helps to identify the server's current connectivity.

2. CPU usage (%) of PostgreSQL database server

   This metric indicates the percentage of CPU resources utilized by the PostgreSQL database server. It provides insights into the server's processing load.

3. Memory usage (%) of PostgreSQL database server

   This metric shows the percentage of memory resources used by the PostgreSQL database server. It helps monitor the server's memory usage.

4. Number of Failed Connections

   This metric represents the number of failed connections to the PostgreSQL database server. It helps identify any issues with the database server's connectivity.

5. Throughput write PostgreSQL database server

   This metric represent the number of write operations that the database can perform in specific time frame. The higher the throughput write, the better the performance of the database.

6. Throughput read PostgreSQL database server

   This metric represent the number of read operations that the database can perform in specific time frame. The higher the throughput read, the better the performance of the database.

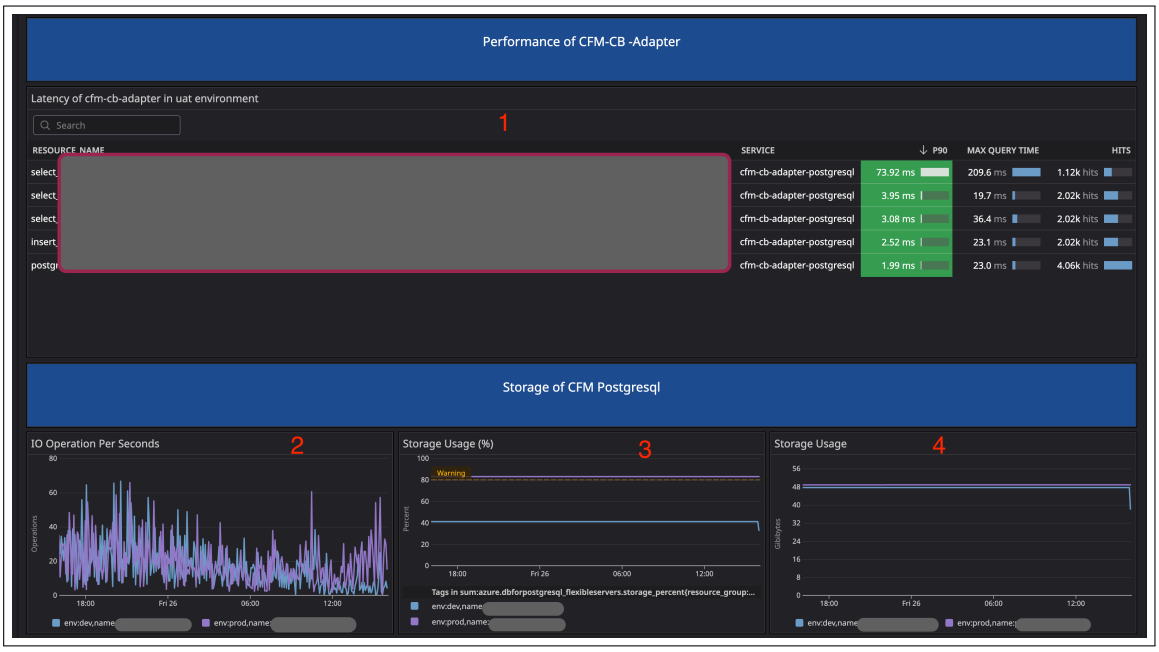


Figure 5.21: Dashboard shows storage usage and performance of CFM PostgreSQL server

The second part of Postgres database metrics dashboard consists 4 parts as follows:

1. Top latency of PostgreSQL database server queries

   This metric identifies the queries with the top highest latency in the PostgreSQL database server, helping to identify performance bottlenecks. Team can use this information to optimize the SQL query or database schema

2. IO Operation of PostgreSQL database server

   This metric provides insights into the level of activity in the PostgreSQL database, specifically in terms of read and write operations. It helps measure the workload and busyness of the database when IOPs is come close to the limits, It might make the database slow down.

3. Storage usage of PostgreSQL database server (Time Series)

   This metric provides the storage usage of the PostgreSQL database server over a period of time, allowing for trend analysis.

4. Storage usage (Bytes) of PostgreSQL database server

   This metric provides the storage usage of the PostgreSQL database server in bytes.

5.3.2.4 Redis Cache Metrics

Figure 5.22 shows the dashboard of Redis metrics of SMT and CFM application.

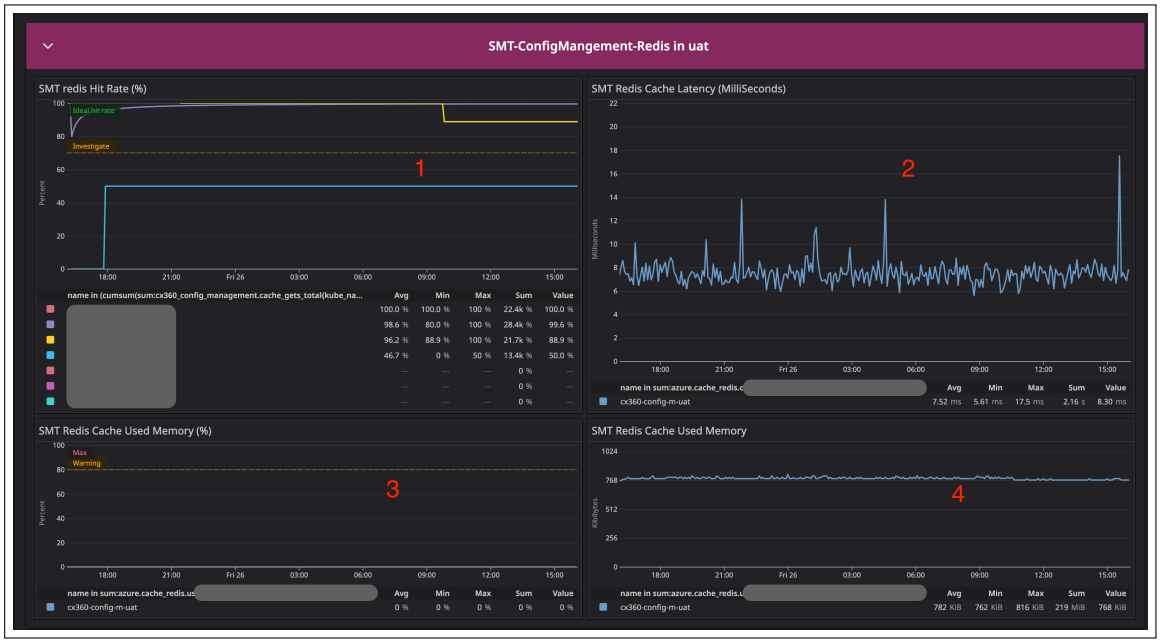


Figure 5.22: Dashboard shows Redis metrics of SMT and CFM applications

The Redis metrics dashboard consists 4 parts as follows:

1. Redis Hit Rate (%)

   This part shows Redis hit rate in percentage (%). The Redis hit rate is calculated by dividing number of cache hits by the total number of cache accesses. User can adjust the time range and environment on dropdown.

2. Redis Cache Latency

This part shows Redis cache latency in milli-seconds. User can adjust the time range and environment on dropdown.

3. Redis Cache Memory Usage (%)

   This metric shows the memory used in Redis as (%). User can adjust the time range and environment on dropdown.

4. Redis Cache Memory Usage (Bytes)

   This metric shows the memory used in Redis as Bytes. User can adjust the time range and environment on dropdown.

5.3.2.5 Kafka Metrics

Figure 5.23 and Figure 5.24 shows the dashboard of Postgres database metrics of SMT and CFM application. The dashboard is spilt into 2 parts. The first part show the Total lag message of the Kafka. The second part show the Total Success and Fail of Publish and Consume message.

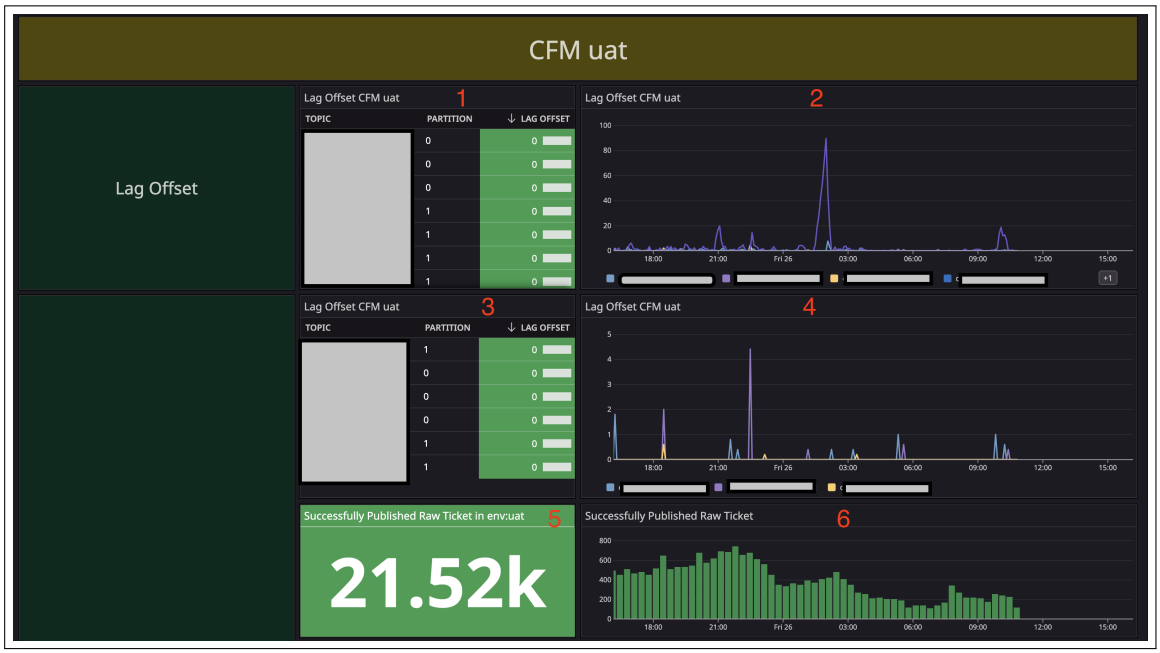


Figure 5.23: Dashboard shows the total Lag of Kafka topics

The first part of Kafka metrics dashboard consists 6 parts as follows:

1. Current Lag Offset of Every topic in the consumer group
2. Time series Lag Offset of Every topic in the consumer group
3. Current Lag Offset of the topic in the consumer group
4. Time series Lag Offset of the topic in the consumer group
5. Count of total Success Publish Ticket to the Kafka Topic from Ingestion
6. Time Series of Success Publish Ticket to the Kafka Topic from Ingestion

Figure 5.24: Dashboard shows the Success and Fail publish and consumer Kafka message

The second part of Kafka metrics dashboard consists 10 parts as follows:

1. Count of total Failed Publish Ticket to the Kafka Topic from Ingestion
2. Time Series of Failed Publish Ticket to the Kafka Topic from Ingestion
3. Count of total Success Publish Message to the Kafka Topic from Ingestion
4. Time Series of Success Publish Message to the Kafka Topic from Ingestion
5. Count of total Failed Publish Message to the Kafka Topic from Ingestion
6. Time Series of Failed Publish Message to the Kafka Topic from Ingestion
7. Count of total Success Publish User to the Kafka Topic from Ingestion
8. Time Series of Success Publish User to the Kafka Topic from Ingestion
9. Count of total Failed Publish User to the Kafka Topic from Ingestion
10. Time Series of Failed Publish User to the Kafka Topic from Ingestion

5.3.2.6 Custom Metrics

For custom metrics, There are 2 important metrics that need to be monitored. The first metric is the number of success and fail messages that CFM aggregator service call API to email cleansing service. This is important because in some case that the email message is too long It make the email cleansing service process the message too long and cause a timeout error. The second metric is the number of success and fail messages cb-adapter service send to ClaraBridge via REST API. This setup is set because in some case ClaraBridge is down and cb-adapter service fail to send the message to ClaraBridge.

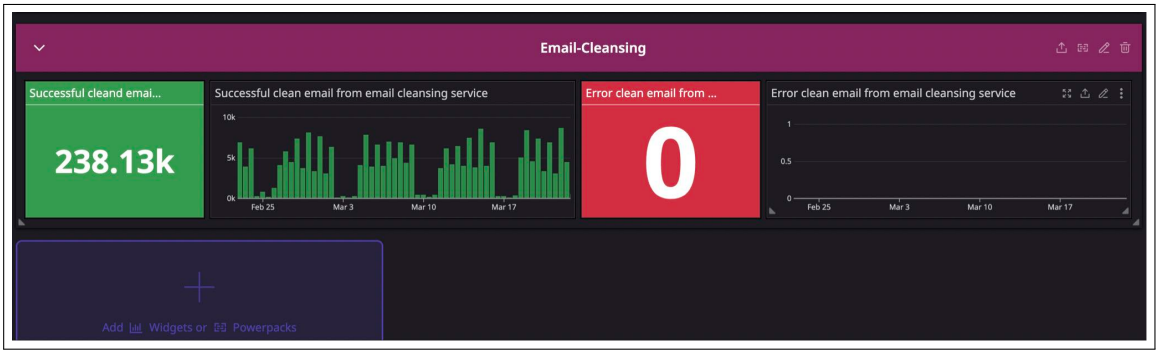


Figure 5.25: Dashboard shows number of success and fail message process by email cleansing service

Figure 5.25 shows the dashboard of number of success and fail messages process from by cleansing service. User can adjust the time range and environment on dropdown.

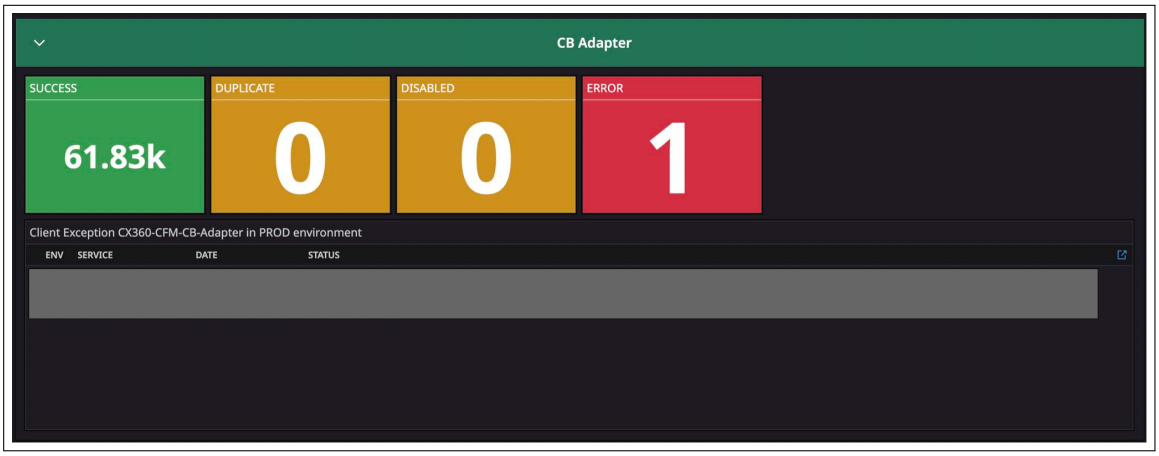


Figure 5.26: Dashboard shows number of success and fail message cb-adapter service send to ClaraBridge

Figure 5.26 shows the dashboard of number of success and fail messages cb-adapter service send to ClaraBridge. There are 4 status of response that cb-adapter service send to Clarabridge which are success, duplicate, disable, and error.

5.3.3 Dashboard of AOD and CPC Application Resources

5.3.3.1 Request API Metrics

Figure 5.27 and Figure 5.28 shows the dashboard of API metrics of AOD and CPC application. The dashboard is spilt into 2 parts. The first part shows the latency, request hit, error rate of API endpoint and error log of applications. The second part shows the https status code of API endpoint at the given time.

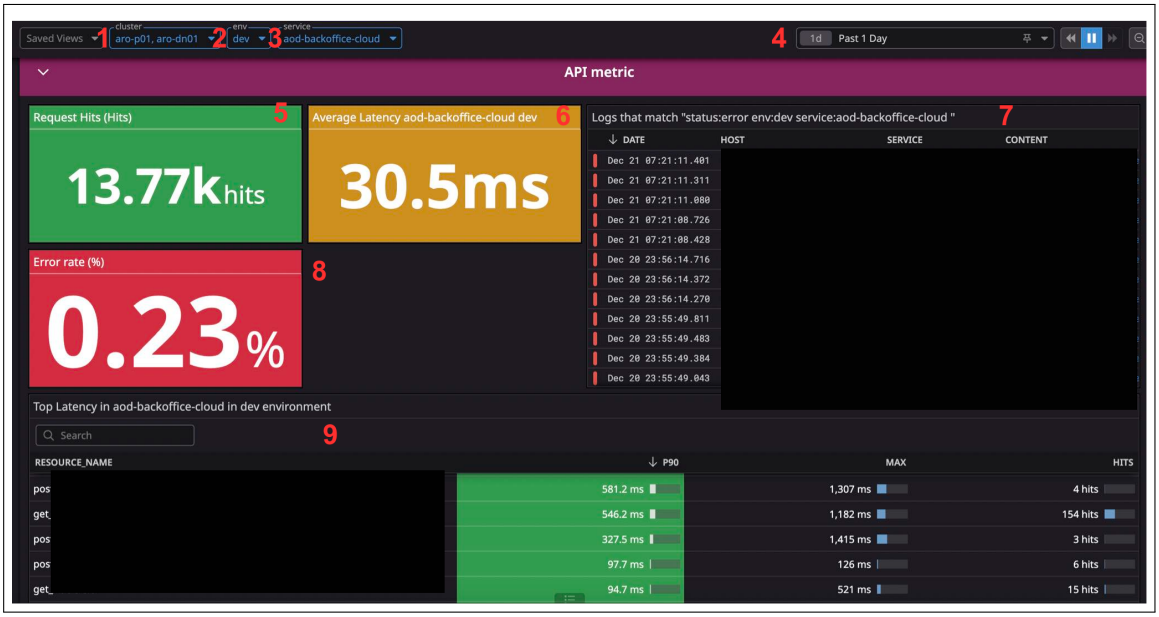


Figure 5.27: Dashboard shows the latency, request hit, error rate of API endpoint and error log of applications

The first part of request API metrics dashboard contains 9 parts as follows:

1. Cluster dropdown

   The cluster dropdown is used for select the cluster of application. The cluster is spilted into 2 clusters which are production (Prod) cluster and non-producttion (Non-prod) cluster.

2. Environment dropdown

   The environment dropdown is used for select the environment of application. The environments are spilted into 4 environments which are development (DEV), quality assurance (QA), user acceptance testing (UAT) and production (PROD).

3. Service dropdown

   The service dropdown is used for select service of application. The service are spilted into 6 services which are Backoffice, CustomerSLO, Data Adapter, CPC, Backend for Frontend (BFF) and Web AOD.

4. Time dropdown

   The time dropdown is used for select the time range of dashboard. The time range are spilted into different range such as last 15 minutes, last 30 minutes, last 1 hour, last 4 hours, and etc.

5. Request hit of API endpoints

   This query value show the number of request of all API endpoints have been hit in the given environment and time range.

6. Average latency of API endpoints

   This query value show the average latency of all API endpoints of service application in the given environment and time range.

7. Error log of applications

   This widget show all of error log of service application in the given environment and time range.

8. Error rate of applications

   This query value show the error rate of all API endpoints of service application in the given environment and time range. This metric indicate the number of error in application. If the error rate is too high, there might be some problem in application.

9. Top latency of API endpoints

   This table show the top p90 and p100 latency of API endpoints of service application in the given environment and time range. This table help the team to visual the latency of API endpoints. If the latency of API endpoints is too high, the team can investigate the problem of API endpoints or optimize the latency of API endpoints.

Figure 5.28: Dashboard show the http status code of API endpoint in applications

The second part of request API metrics dashboard contains 4 parts as follows:

1. Pie chart of http status code of API endpoints

   This pie chart show the number of http status code of application in some time range. So If the number of 5xx http staus code or 4xx http status code is too high, There might be some problem occur in application.

2. Time series show number of http status code

   This time series show the number of http status code of application as a time series. This time series to identify the right time that the problem occur in application.

3. Table show number of http status code of each API endpoints

   This table show number of http status code with name of the endpoints. This table help team to identify which endpoint has http status code problem. This information can further use to investigate the problem of endpoint.

4. Time series of http status code of API endpoints

   This time series show the number of http status code with the name of endpoints as a time series.

5.3.3.2 Pod Utilization Metrics

Figure 5.29 shows the dashboard of Pod Utilization metrics of AOD and CPC application. User can select the cluster, environment, service and time range of dashboard. This dashboard help the team to monitor the resource usage of pod. This dashboard will be focus monitor the CPU and memory usage of pod.

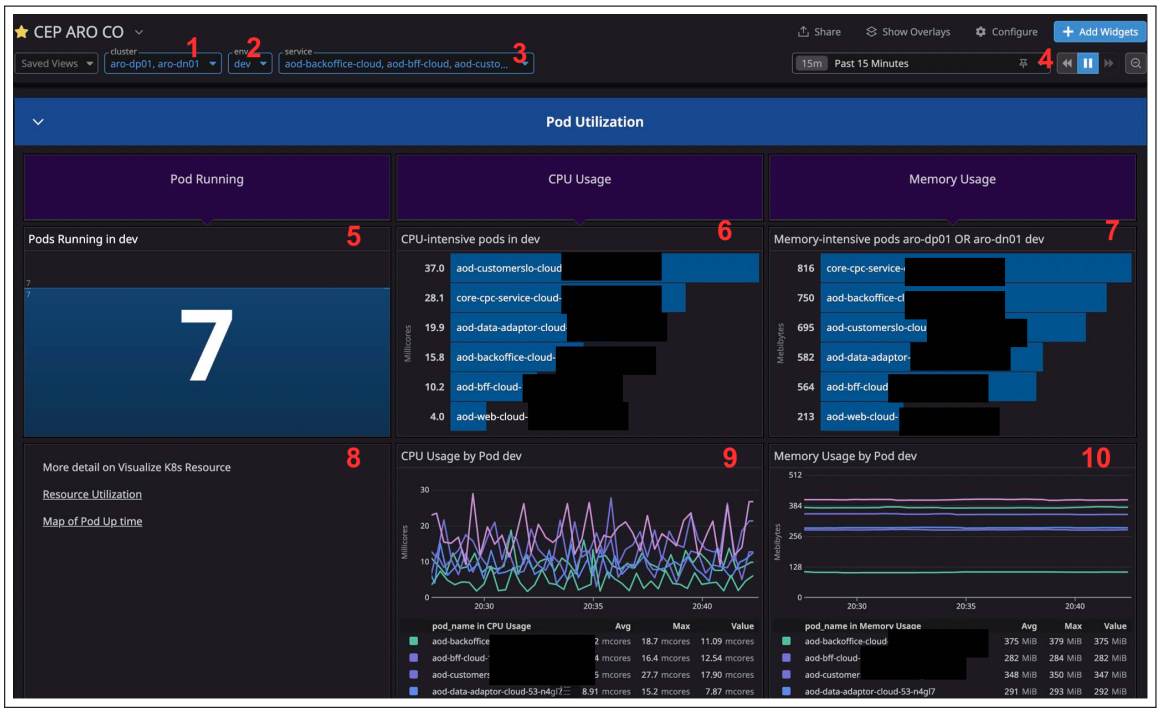


Figure 5.29: Dashboard shows Pod Utilization of AOD and CPC Application

The dashboard of Pod Utilization metrics contains 10 parts as follows:

1. Cluster dropdown

   The cluster dropdown is used for select the cluster of application. The cluster is spilted into 2 clusters which are production (Prod) cluster and non-producttion (Non-prod) cluster.

2. Environment dropdown

   The environment dropdown is used for select the environment of application. The environments are spilted into 4 environments which are development (DEV), quality assurance (QA), user acceptance testing (UAT) and production (PROD).

3. Service dropdown

   The service dropdown is used for select service of application. The service are spilted into 6 services which are Backoffice, CustomerSLO, Data Adapter, CPC, Backend for Frontend (BFF) and Web AOD.

4. Time dropdown

   The time dropdown is used for select the time range of dashboard. The time range are spilted into different range such as last 15 minutes, last 30 minutes, last 1 hour, last 4 hours, and etc.

5.3.3.3 PostgreSQL Database Metrics

Figure5.30 and Figure5.31 shows the dashboard of Postgre database metrics of AOD and CPC application. The dashboard is spilt into 2 parts. The first part shows the resource usage of PostgreSQL database server and the second part shows the storage usage and performance of PostgreSQL database server.

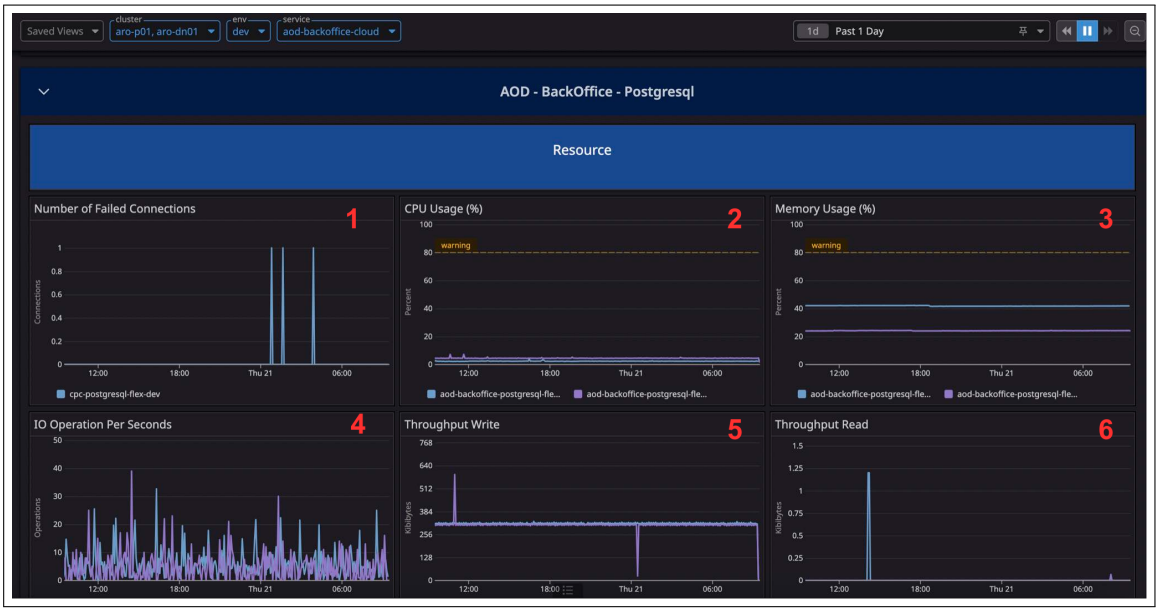


Figure 5.30: Dashboard shows resource usage of Backoffice PostgreSQL server

The first part of Postgre database metrics dashboard contains 6 parts as follows:

1. Number of failed connections of PostgreSQL database server

   This metric represents the number of failed connections to the PostgreSQL database server. It helps identify any issues with the database server's connectivity.

2. CPU usage (%) of PostgreSQL database server

   This metric indicates the percentage of CPU resources utilized by the PostgreSQL database server. It provides insights into the server's processing load.

3. Memory usage (%) of PostgreSQL database server

   This metric shows the percentage of memory resources used by the PostgreSQL database server. It helps monitor the server's memory consumption.

4. IOPs (Input/Output Operations per Second) of PostgreSQL database server

   This metric provides insights into the level of activity in the PostgreSQL database, specifically in terms of read and write operations. It helps measure the workload and busyness of the database when IOPs is come close to the limits, It might make the database slow down.

5. Throughput read PostgreSQL database server

   This metric represent the number of read operations that the database can perform in specific time frame. The higher the throughput read, the better the performance of the database.

6. Throughput write PostgreSQL database server

This metric represent the number of write operations that the database can perform in specific time frame. The higher the throughput write, the better the performance of the database.

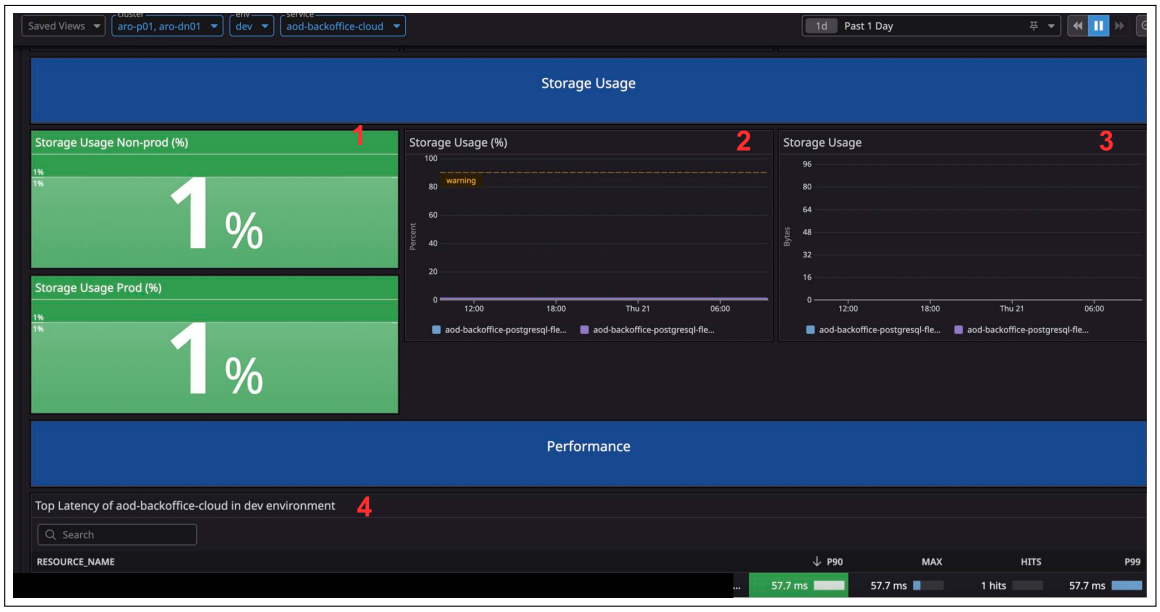


Figure 5.31: Dashboard shows storage usage and performance of Backoffice PostgreSQL server

The second part of Postgre database metrics dashboard contains 4 parts as follows:

1. Storage usage of PostgreSQL database server (Query Value)

   This metric provides the current storage usage of the PostgreSQL database server as a single value.

2. Storage usage of PostgreSQL database server (Time Series)

   This metric provides the storage usage of the PostgreSQL database server over time, allowing for trend analysis.

3. Storage usage (Bytes) of PostgreSQL database server

   This metric provides the storage usage of the PostgreSQL database server in bytes.

4. Top latency of PostgreSQL database server queries

   This metric identifies the queries with the top highest latency in the PostgreSQL database server, helping to identify performance bottlenecks. Team can use this information to optimize the SQL query or database schema

5.3.3.4 Azure comosDB Database Metrics

Figure5.32 shows the dashboard of Azure comosDB database metrics of CSLO service.

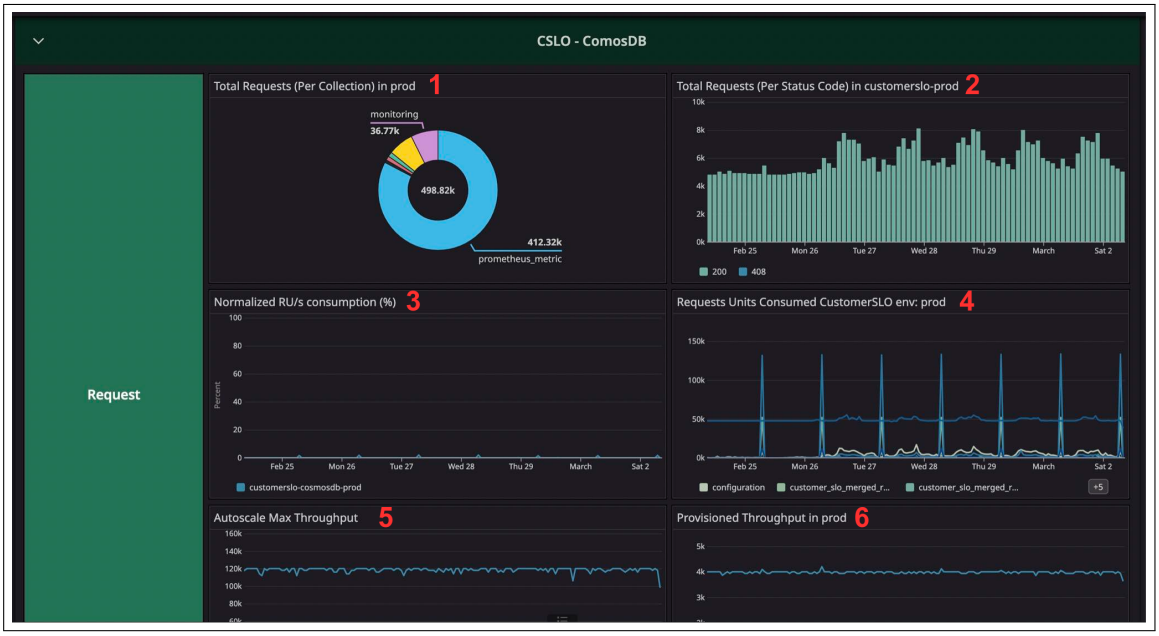


Figure 5.32: Dashboard shows metric related to Azure comosDB Database in CSLO service

1. Total Requests of Azure comosDB database in pie chart format

   This part shows the total request of Azure comosDB database in pie chart format. It helps to understand which collection of database has the most used.

2. Total Requests base on the status code in time series format

   This part shows the total request base on the status code of Azure comosDB database. This part give the information about the operation of database. whether the operation is success or fail at the given time.

3. Normalizes RU/s Consumption (%)

   Normalizes RU/s Consumption (%) is the percentage of the provisioned throughput that is consumed by the request.

4. Request units (RUs) CSLO Consumed in time series format

   This part shows the request units (RUs) CSLO consumed in time series format. This part give the information about the request units (RUs) CSLO consumed at the given time. The higher the request units (RUs) CSLO consumed, This might be the sign that the database is saturated.

5. Autoscale max throughput

   This part shows the autoscale max throughput of Azure comosDB database. This part give the information about the maximum throughput of the database.

6. Provision throughput

   This part shows the provision throughput of Azure comosDB database. This part give the information about the provision throughput of the database at the given time.

5.3.3.5 Redis Cache Metrics

Figure5.33 shows the dashboard related to redis metrics in aod backoffice applications. The main focus of Redis cahe metrics will be on redis cache hit rate. So If the cahce hit rate is less, It can be assume that the redis performance badly and might consider to be remove from application

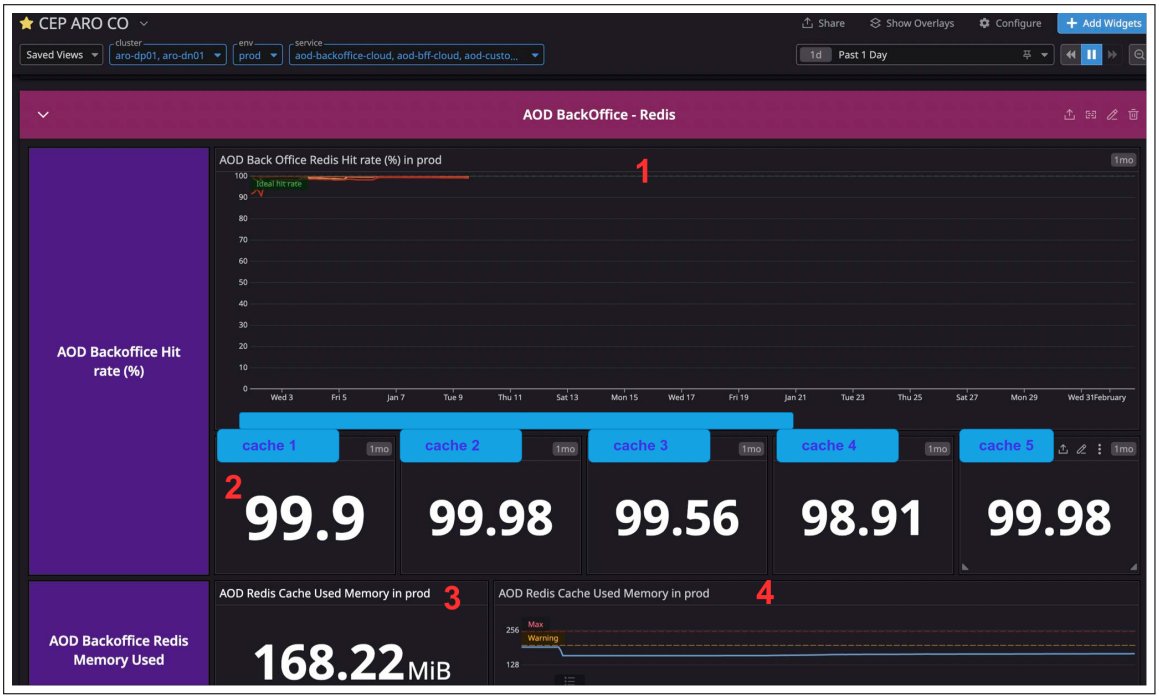


Figure 5.33: Dashboard shows metric related to redis

The Redis metrics dashboard consists 4 parts as follows:

1. Redis cache hit rate in time series format (%)

   This part shows redis hit rate in percentage (%) the redis hit rate is calculate by dividing number of cache hits by the total number of cache accesses. User can adjust the time range and environment on dropdown.

2. Redis cache hit rate in query value format

   This query value shows the Redis cache hit rate at the range time. So if user select range 1 day. datadog will gather redis cache accesses in 1 day to calculate redis cache hits.

3. Redis cache memory used in query value format

   This metric shows the memory used in redis cache represent in query value format at the given time. User can adjust the environment of redis cache using dropdown env

4. Redis cache memory used in time series format

   This metric shows the memory used in redis cache. this metrics can be use to view the saturation of redis cache.

5.3.3.6 Kafka Metrics

Figure 5.34 shows the dashboard related to Kafka consumer lag metrics in CPC application.
The Kafka consumer lag metrics in CPC application metrics dashboard consists 2 parts as follows:

1. Kafka consumer lag metrics represent in time series format

   This part shows Kafka consumer lag metric in time series format x-axis represent time and y-axis represent Kafka lag metric.

2. kafka consumer lag metrics in table

   This part shows Kafka consumer lag metric in table format. This table also provides detail Kafka lag into specific partition of Kafka topic. The color logic has been set that If the Kafka lag is more than 100,000, the color of the Kafka lag will be red. If the Kafka lag is equal to zero, then the color will be green.

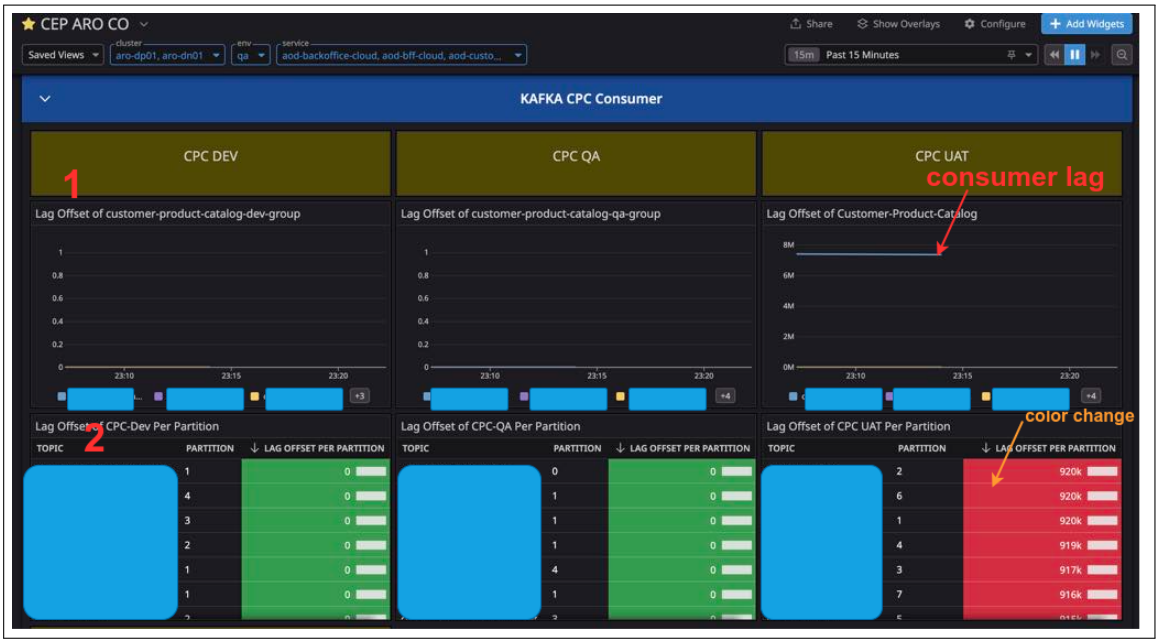


Figure 5.34: Dashboard shows metric related to kafka

Figure 5.35 and figure 5.36 shows dashboard metric related to Kafka total message consume into CSLO application. For CSLO service, there are 4 Kafka topics that are used to consume the message. User can used the dropdown filter the environment and select period of time to view the kafka metrics. This two dashboard are used to monitor number of message consume into CSLO application and number of fail receiving message from kafka topic. Note that, sometimes external application send message that CSLO application can not extract the message from Kafka topic. This might happen due to payload external application publish to kafka topic is not expected. Monitor the number of fail receiving message from kafka topic can help the team to investigate the problem and fix the problem in time.

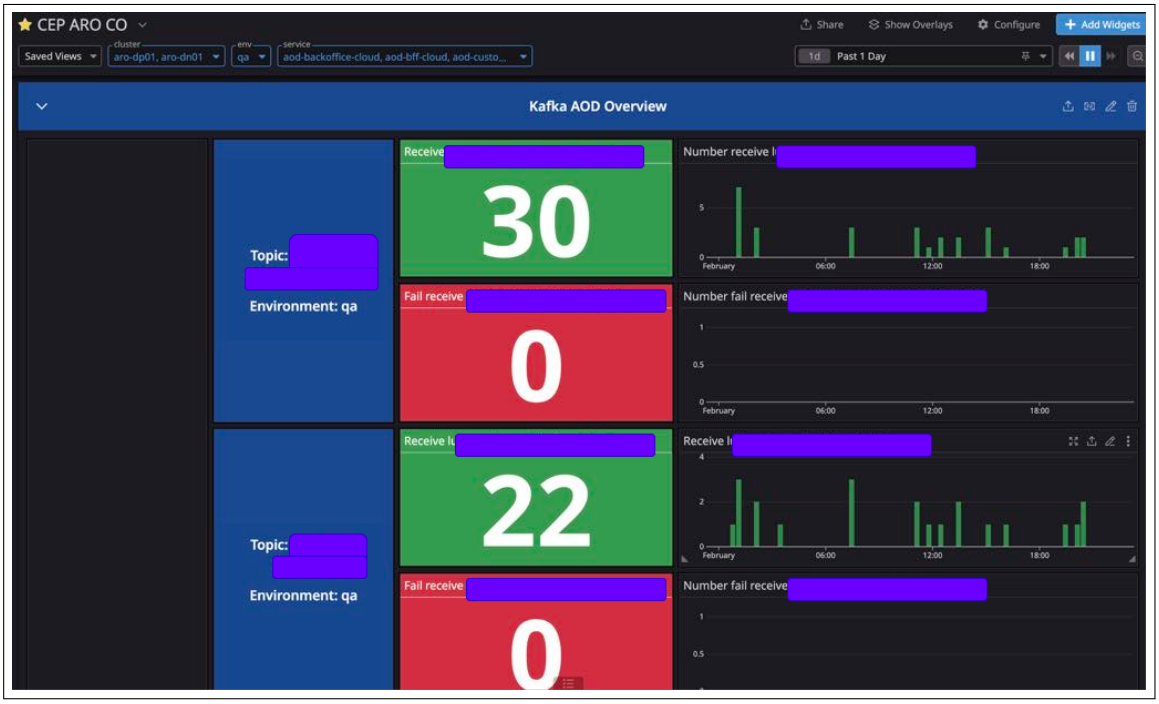


Figure 5.35: Dashboard shows metric related to Kafka total message consume into CSLO application part 1

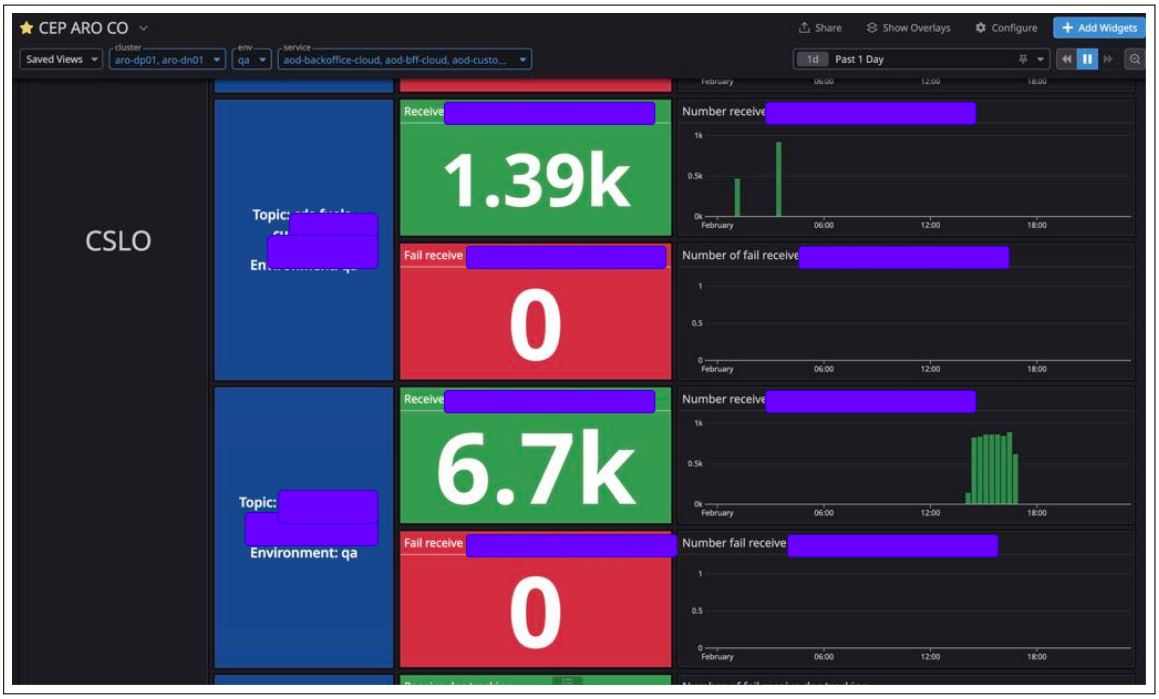


Figure 5.36: Dashboard shows metric related to Kafka total message consume into cslo application part 2

Figure 5.37 shows dashboard monitor number of message backoffice publish messages to data adapter and external application. For this dashboard, users can use the dropdown filter to select the environment and period of time to view the Kafka metrics. Also, this dashboard monitors the number of failed published messages to the data adapter and external application. The failed published messages to the data adapter and external

application can be caused by situations such as the payload of the message being too large and timing out when publishing to the data adapter and external application, etc.

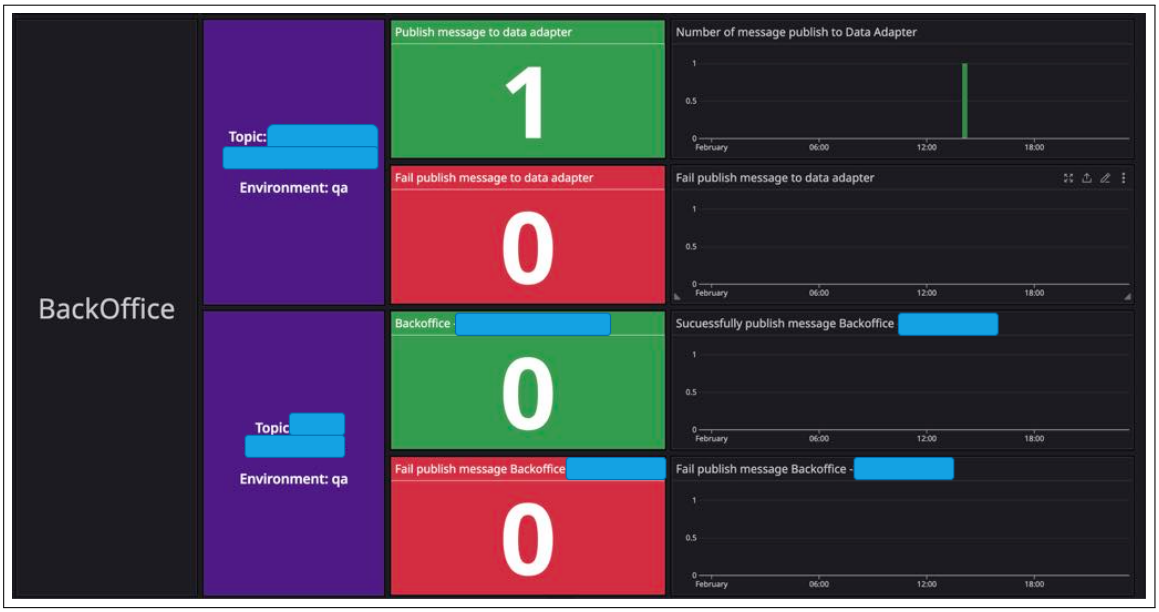


Figure 5.37: Dashboard shows metric related to Kafka total message backoffice publish to data adapter and external application

Figure 5.38 shows dashboard monitor number of message successfully / fail consume and publish from data adapter. For this dashboard, can used the dropdown filter the environment and select period of time to view the Kafka metrics

Figure 5.38: Dashboard shows metric related to Kafka total message consume/publish from data adapter

5.3.4 Datadog Real User Monitoring

Figure 5.39, Figure 5.40 and Figure 5.41 shows the metrics and other session replay of Datadog Real User Monitoring (RUM). The Datadog Real User monitoring is used to track and analyze how users interact with

the frontend part of the website. It does not only store the key performance of the website but it is also useful in error tracking in case anything fails. Datadog Real User Monitoring consists of the Performance Monitoring which contains the metrics and Session Replay which stores the user session replay in video. For the scope of this project, the team will focus on the performance monitoring of the AOD Frontend part only.

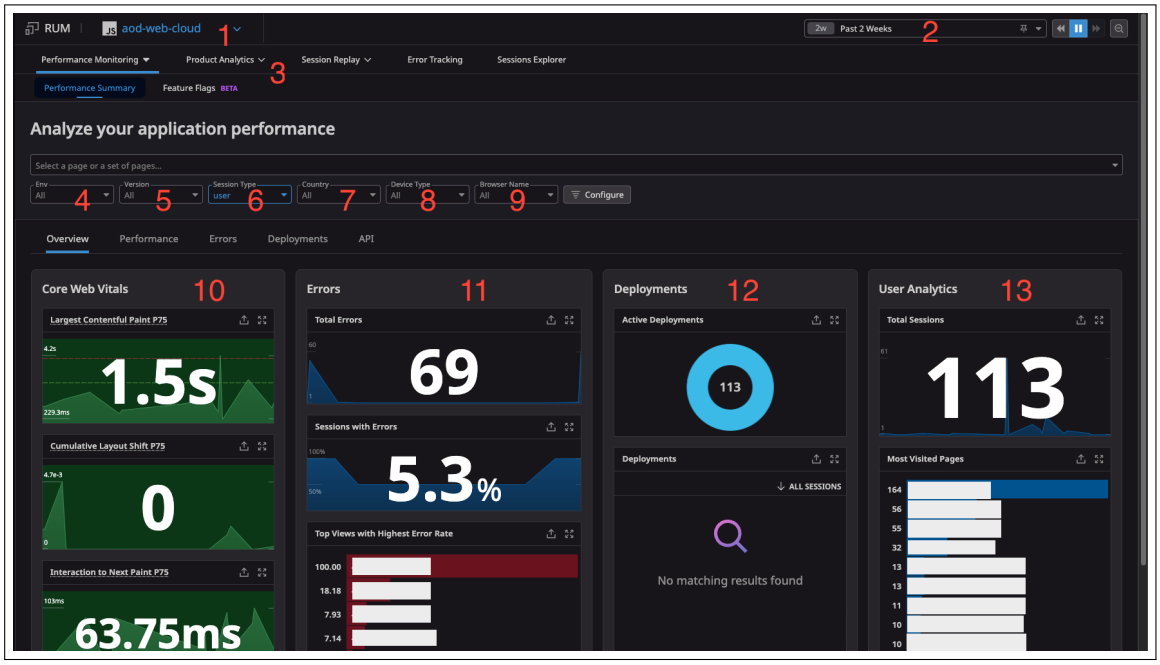


Figure 5.39: Datadog Real User Monitoring performance page

The datadog real user monitoring performance page consists of 13 parts as follows:

1. Service dropdown

   The service dropdown is used to select a service of Datadog Real User Monitoring.

2. Time dropdown

   The time dropdown is used for select the time range of dashboard. The time range are spilted into different range such as last 15 minutes, last 30 minutes, last 1 hour, last 4 hours, and etc.

3. Real User Monitoring Tabs

   The tabs that contain all crucial part of Datadog Real User Monitoring.

4. Environment dropdown

   The environment dropdown is used to select an environment of the service in Real User Monitoring.

5. Version dropdown

   The version dropdown is used to select application version for the Datadog Real User Monitoring.

6. Session Type dropdown

   The session type dropdown is used to filter the session type of user in Datadog Real User Monitoring.

7. Country Dropdown

   The country dropdown is used to filter the country of user in Datadog Real User Monitoring.

8. Device Type

   The device type is used to filter the device of user in Datadog Real User Monitoring.

9. Browser Name

   The browser name is used to filter the browser of user in Datadog Real User Monitoring.

10. Core Web Vitals

    It shows the core web vitals metrics of the frontend side from the loading time down to the website respond after the user has interacted with in a certain period of time. For core web vitals there are 3 metrics as follows

    (a) Largest Contentful Paint p75 (LCP) It is the 75th percentile value of the time it takes for the largest content element in the viewport (like an image or a block of text) to become visible to the user. It indicates perceived load speed and helps determine when the main content of the page has finished rendering.

    (b) Cumulative Layout Shift p75 (CLS) It is the 75th percentile value of the visual stability of a page as it loads. It quantifies how much the content of a page moves around during the loading process. Unexpected layout shifts can be frustrating for users, especially when they cause them to accidentally click on the wrong element.

    (c) Interaction to Next Paint (I2NP75) It is the 75th percentile value of the time it takes for the next page to start rendering after a user interacts with a link or button. It measures the speed of transitioning between pages.

11. Errors

    It shows the total errors in the frontend web and shows which session has the most errors as well as which page has the most error in a certain period of time. For errors, there are 3 metrics which are Total Errors, Session with error and Top views with highest error rate.

    (a) Total Errors This metric tracks the overall count of errors that occur within the application. It provides an overview of the total number of errors encountered by users.

    (b) Session with error Session errors measure the number of error occurrences within a single user session. It helps to identify how errors impact individual user experiences and sessions.

    (c) Top views with highest error rate This metric identifies the specific page or view within the application that has the highest error rate, It helps to improve the optimization efforts and address issues affecting user interactions on that particular page.

12. Deployment

    It shows the total distinct deployment of the service in a certain period of time.

13. User Analytics

    It shows the total distinct session as well as the count of the most visited pages in certain period of time. For user analytics, there are 2 metrics which are Total Sessions and Most Visited Pages.

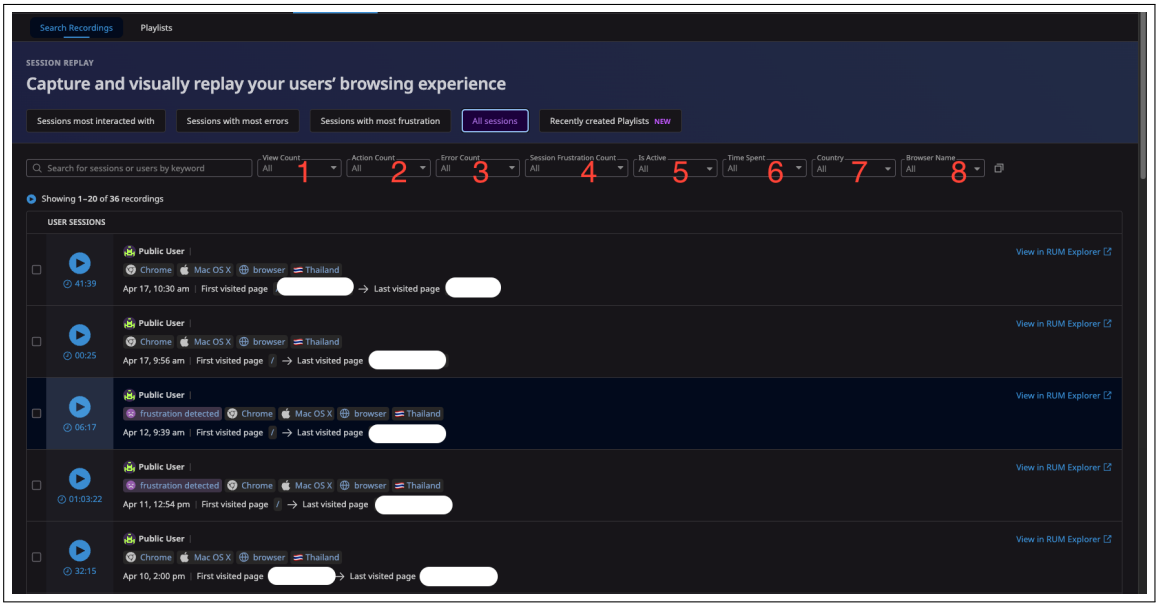


Figure 5.40: Datadog Real User Monitoring Session Replay page

Figure 5.40 shows the Datadog Real User Monitoring session page. Clicking on the video icons on the left side will trigger a video popup resembling the image shown in 5.41. The Datadog Real User Monitoring session page consists of 8 parts as follow:

1. View Count

   It shows the total view count of the user session in Datadog Real User Monitoring.

2. Action Count

   It shows the total action count of the user session in Datadog Real User Monitoring.

3. Error Count

   It shows the total error count of the user seesion in Datadog Real User Monitoring.

4. Session Frustration Count

   It shows the total frustration detected of the user session in Datadog Real User Monitoring.

5. Is Active

   It shows whether this user session is still active or not.

6. Time Spent

   It shows the total time spent in this user session from the first visited page until the last known action before connection lost.

7. Country

   It shows the country that the user is using for this session in Datadog Real User Monitoring.

8. Browser Name

   It shows the browser that is used for the user session in Datadog Real User Monitoring.

Figure 5.41 shows the video of the user session in Datadog Real User Monitoring. It captures cursor movements, clicks, scrolls, and other actions users take while interacting with the frontend web application. However, it's designed to prioritize user privacy by blinding sensitive information such as keystrokes or form input. This way, it can be used to observe how users navigate and interact with the website without compromising their privacy or security.

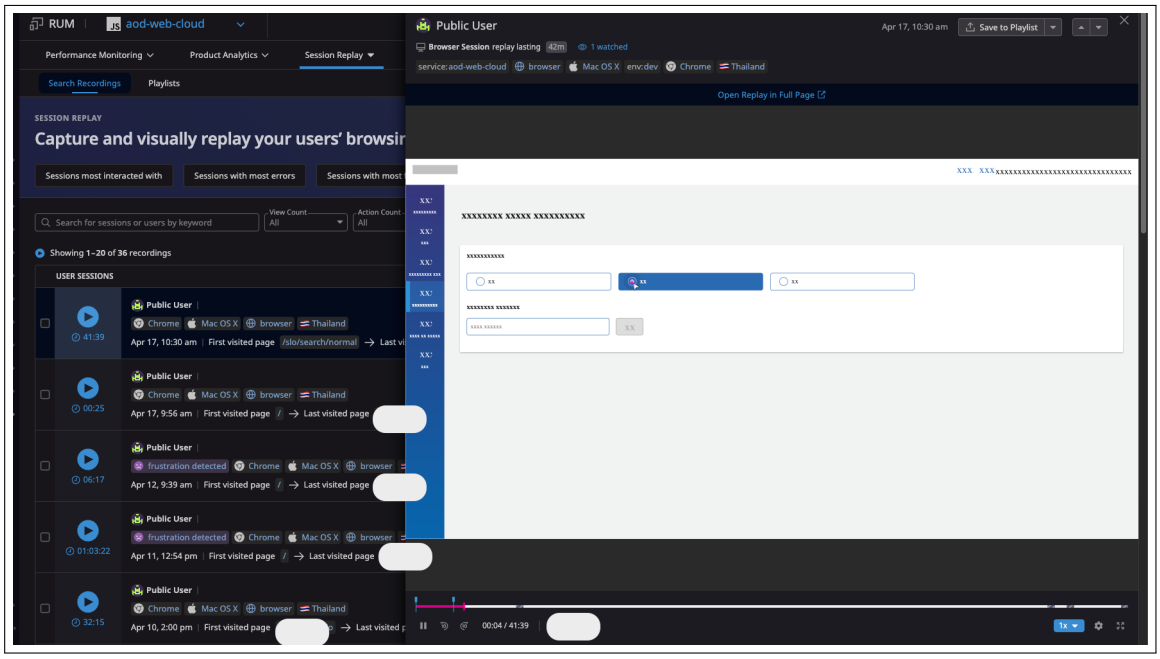


Figure 5.41: Shows Datadog Real User Monitoring Session Replay video with

5.4 Implement Result of Alert System

After implementing the alert system, team members were asked for feedback and to review the alert system. The results showed that some alerts were sending too many emails to the team each day. As a result, adjustments were made to set the threshold of the alert system in order to reduce the number of alert emails. Some alerts were being triggered, but the team did not consider them important. For example, the alert for failed requests to cb-adapter. This alert is triggered when the cb-adapter service fails to send a message to ClaraBridge. However, the team does not consider this alert significant because the cb-adapter service has a retry mechanism. When it fails to send a message, it saves the data to the database and retries sending the message again. Therefore, a decision was made to set the threshold for this alert. When there is a high number of failed requests to cb-adapter in a short period of time, the alert will be triggered.

5.4.1 Databases Alert

5.4.1.1 Result of Databases Alerting System

Figure 5.42 shows the alert email message that occurs from CosmosDB. The email alert will be sent to the team members with the data shown in the figure. The alert shows that the request unit consumption exceeds the limit. It also shows the environment where the errors occurred as well as the service name that will be used to track the issue. The alert also shows the time series graph of the Request Unit that exceeds the threshold. The email is also attached with an view dashboard link which when clicked will redirect to the dashboard page for further investigation.

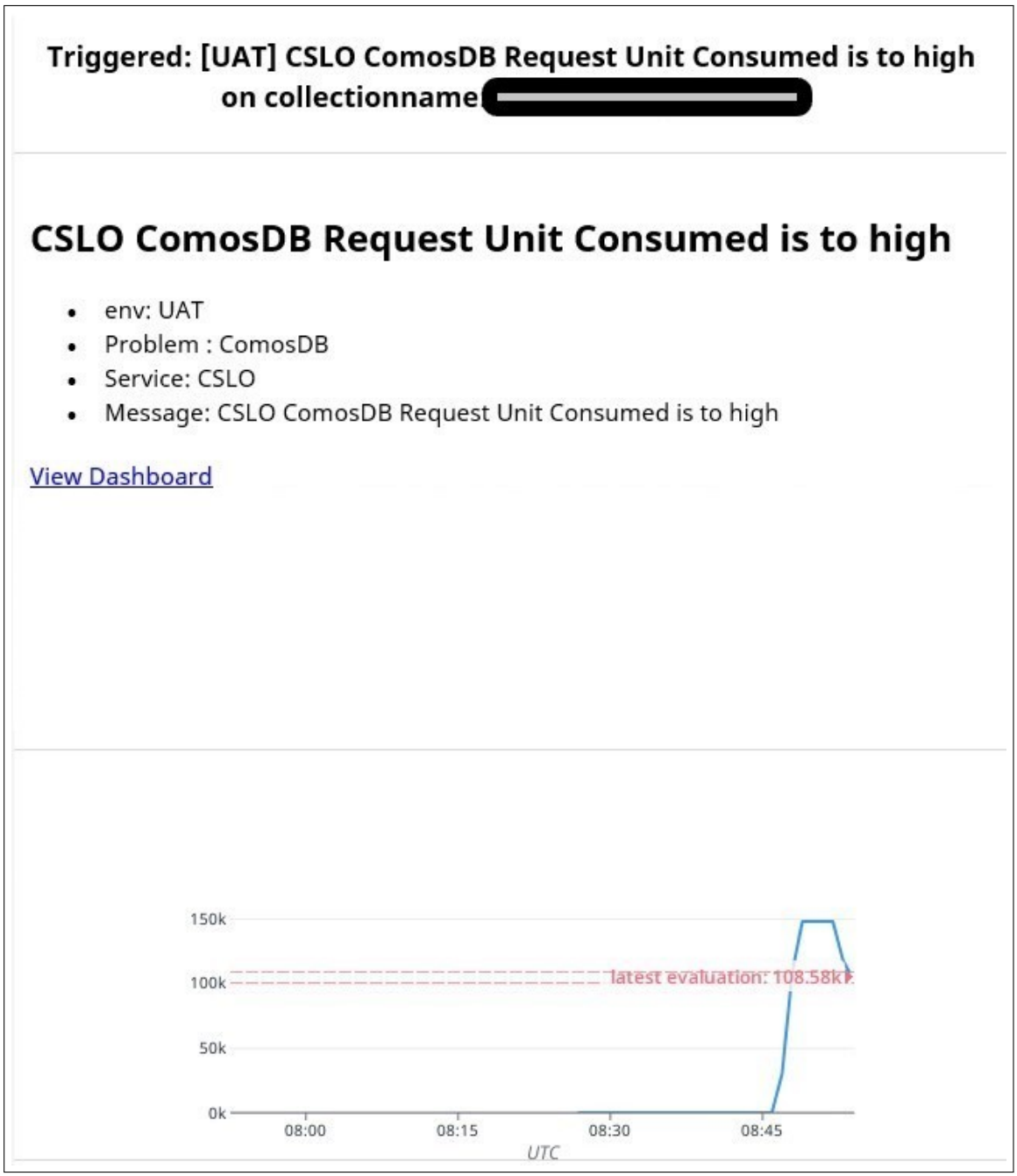


Figure 5.42: The Email Alert of CosmosDB Request Unit Exceeds Threshold

5.4.2 Services Alert

5.4.2.1 Result of Pod Alerting System

Figure 5.43 shows that the pod of a service is down.

Team members will receive this email alert in the case of a problem related to the pod status. The service name and the environment where the error has occurred are attached in the email alerts. The email alert also sends out a time series graph that can be used to investigate the issue. Lastly, it also has the link to the dashboard for easier issue tracing.

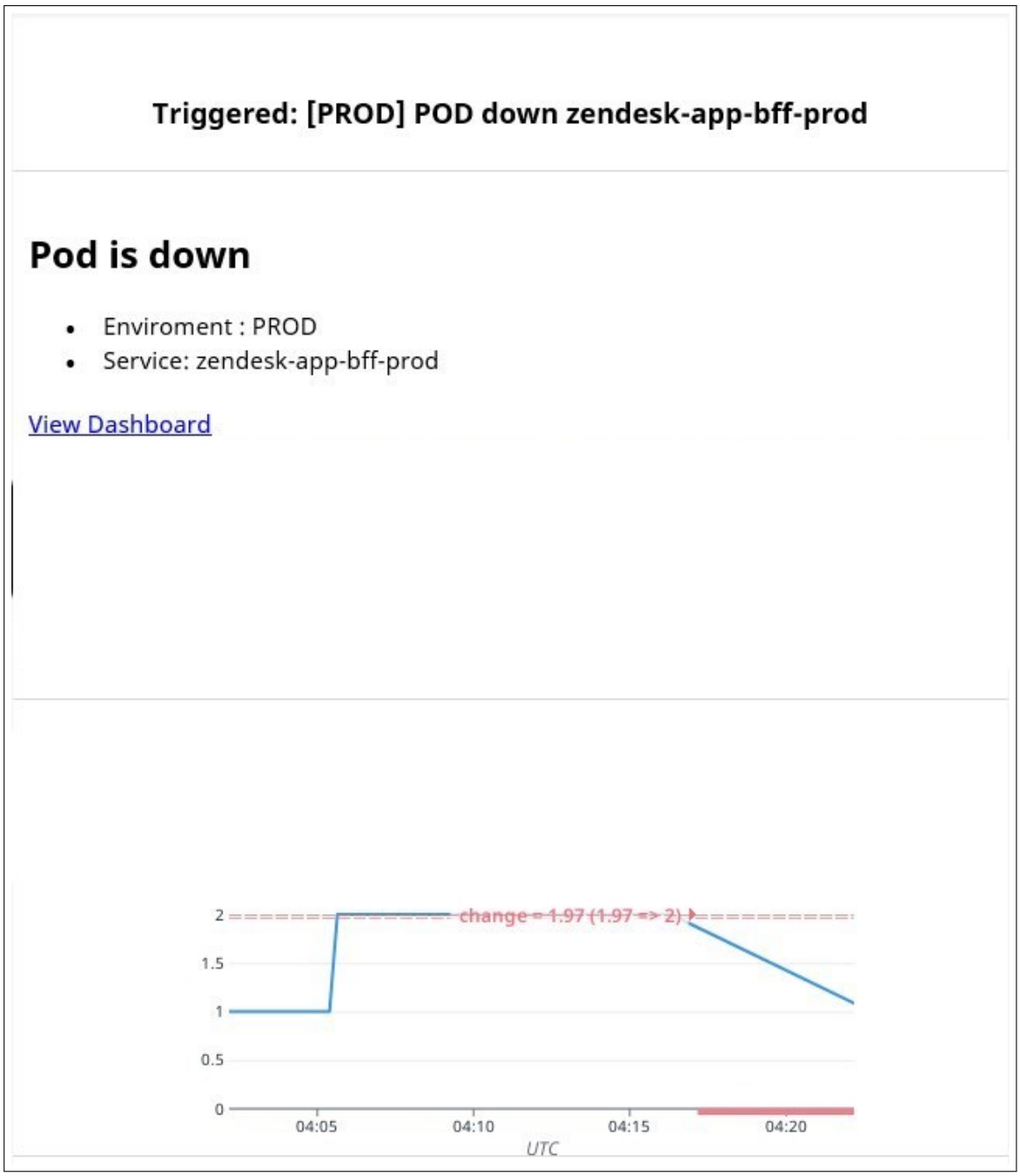


Figure 5.43: The Email Alert of Pod Status

5.4.3 Kafka Alert

5.4.3.1 Result of Kafka Alerting System (Fail to Convert)

Figure 5.44 shows the alert email message related to Kafka. The alert will send this email message to the team members which are blinded in this figure. The alert shows that the Kafka is failed to consume the message from the topic. It also sends out the environment and the service name that might be useful for tracking the problem. The alerts also sends out the time series of a Kafka error count in the services that go beyond the threshold. It also has a View Dashboard link when clicked will redirected to the dashboard page of the Kafka.

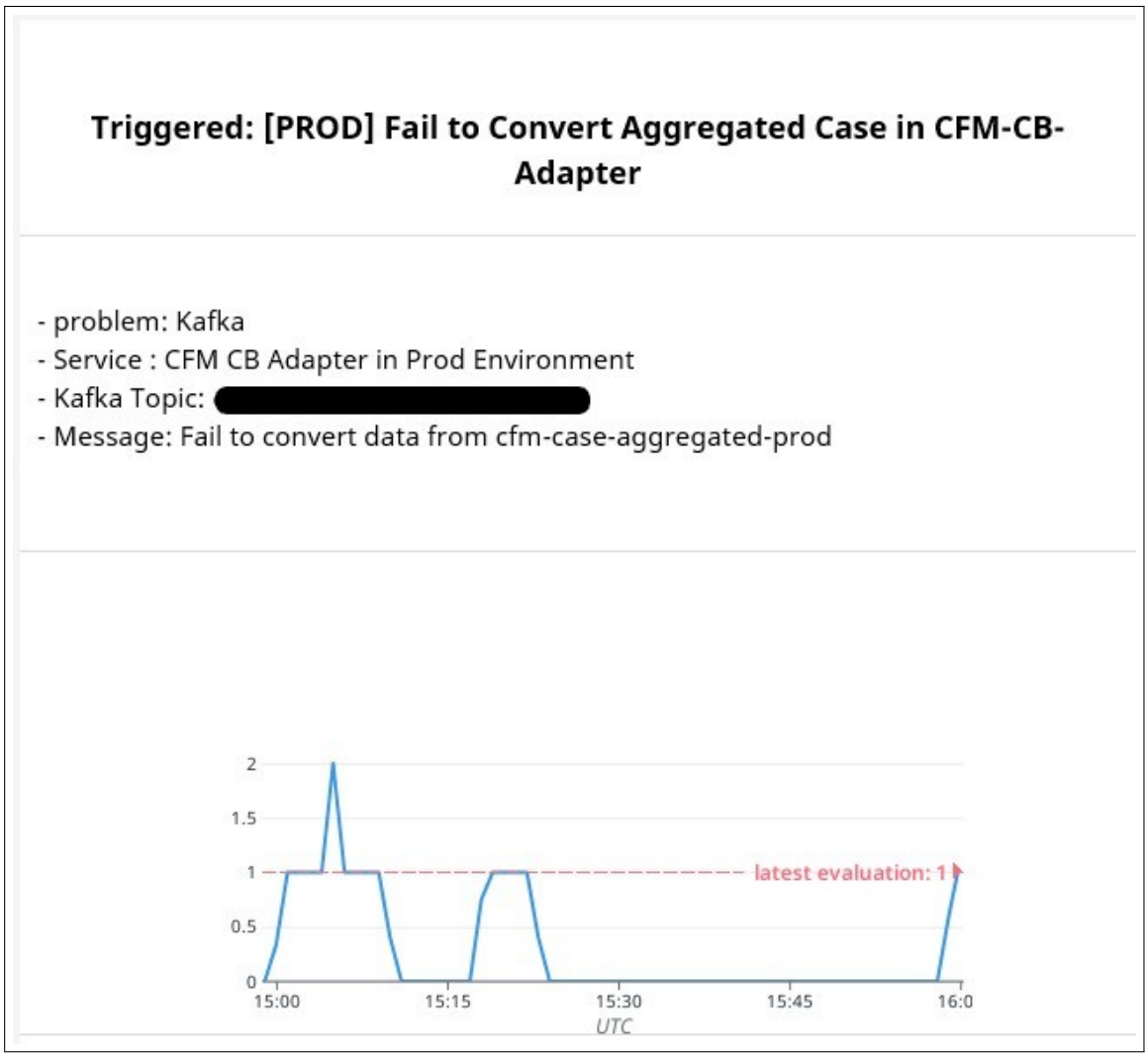


Figure 5.44: The Email Alert of Kafka Error Converting Data

5.4.3.2 Result of Kafka Alerting System (`Lag Exceeds Theshold`)

Figure 5.45 shows the email alert related to the Kafka lag. Team members will receive this email after the incident has occurred. The message in the alert shows that the lag in the Kafka topic has exceeded 1 million. The service and environment are also attached in the email alert. Time series graph of the Kafka Lag is attached in the email alert as it is useful for investigating the problem. Further investigation can be done easily by following the View Dashboard link in the footer of the email.

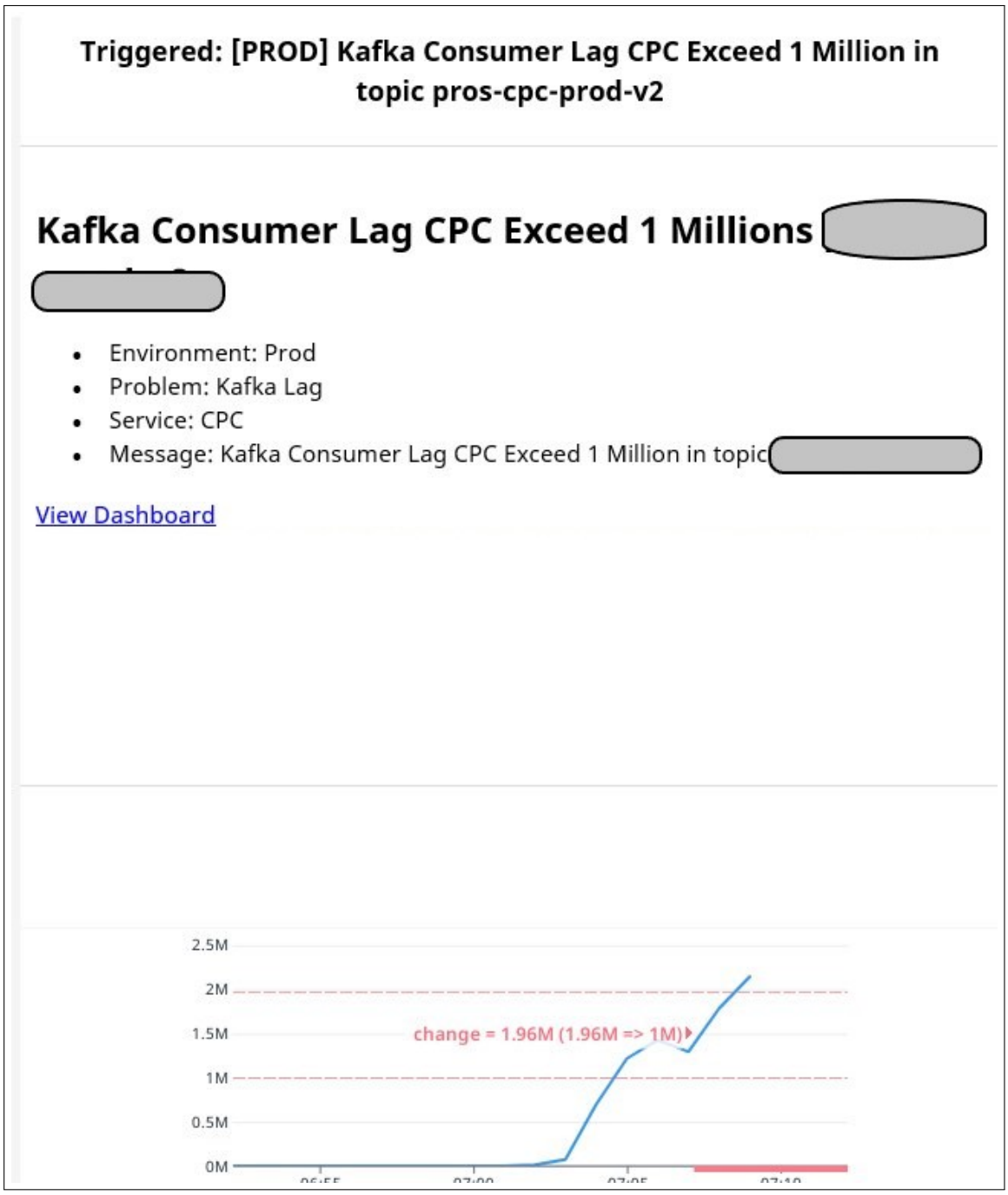


Figure 5.45: The Email Alert of Kafka Lag Exceeds 1 Million

5.5 Implement Result of Auto Pod Scaling

5.5.1 Proof of Concept on Implement KEDA in Testing Cluster

After implement auto pod scaling feature, there was a problem in installing the KEDA to the cluster. It has been identified that installing KEDA onto the cluster necessitates having cluster admin permissions. To achieve this, collaboration was needed with the related team that responsible to manage the ARO cluster to install KEDA to the cluster. Due to lack of experience in implementing KEDA, the team decided to let us do a proof of concept (POC) first by implementing KEDA into the testing cluster first. So if there are some problems occur, it will not affect other teams deploying applications in the same cluster. For the POC task, it started by selecting the application that will be implemented with the auto pod scaling feature in the testing

cluster. After consideration, the cb-adapter application was selected to do the POC because the cb-adapter application is less complex than other applications and easy to create a consumer lag in the application. After setting up all of the KEDA components in the testing cluster and setting up the testing namespace, the helm command line has been used to deploy the cb-adapter application to the testing namespace and then apply the scaledobject and triggerauthentication to the testing namespace as shown in figure 5.46.

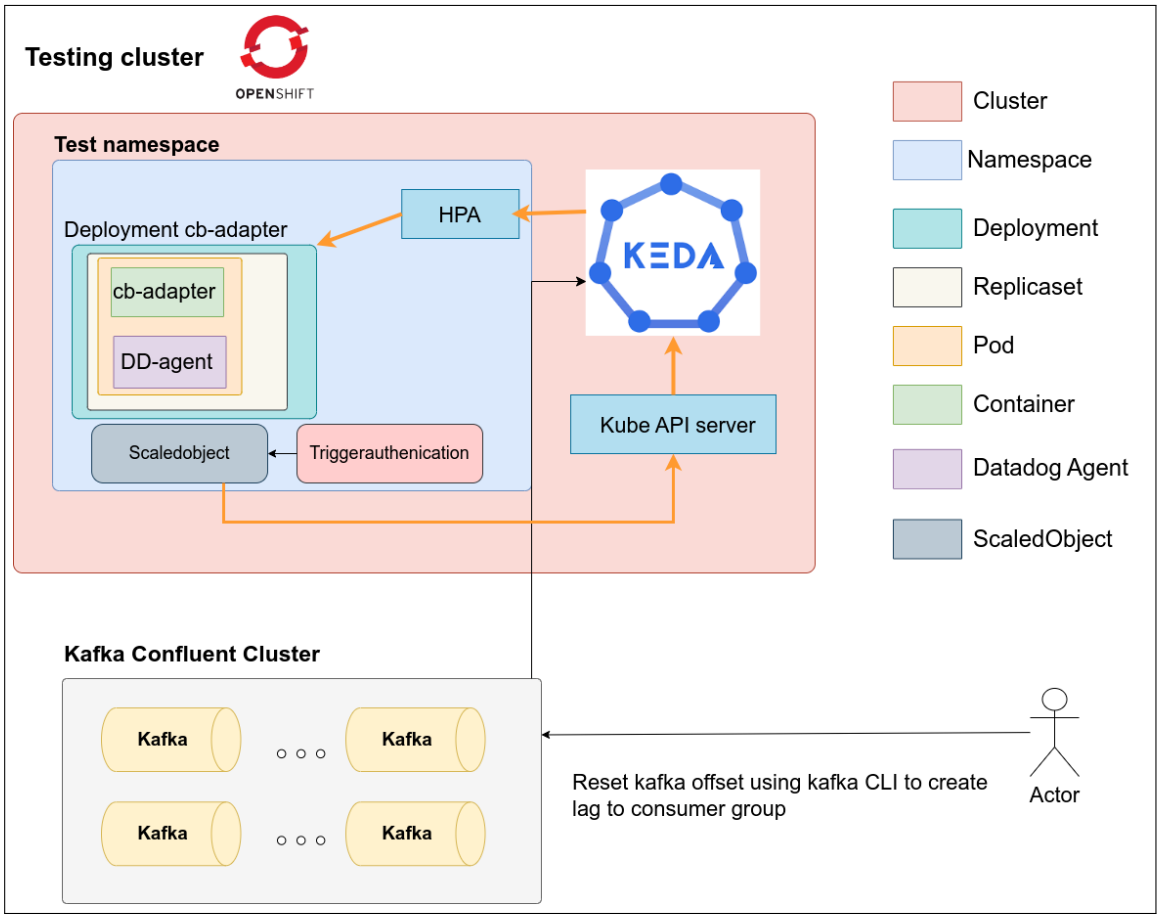


Figure 5.46: Proof of concept on implement KEDA in testing cluster with cb-adapter application

After deploy scaledobject and triggerauthentication, then consumer lags were created to the cb-adapter application by reset the offset of kafka consumer group by using kafka cli with the follow command.

```
1  kafka-consumer-groups.sh --bootstrap-server kafka-bootstrap-server-name:9092
2  --group consumer_group_name --reset-offsets --shift-by -10
3 --topic topic_name --execute
```

After resetting the offset of the Kafka consumer group, the cb-adapter application has autoscale up to 2 pods as shown in figure 5.47. A few minutes later, the cb-adapter application has autoscale down to 1 pod as the application consumes the message from the Kafka topic and the lag of the Kafka consumer group decreases to zero. Figure 5.47 shows the pod trying to auto-scale from 1 to 2 pods when the lag of the Kafka consumer group increases. This Figure is captured from the deployment of the cb-adapter page in the test cluster ARO. Figure 5.48 shows the event of the HPA starting to scale cb-adapter from 1 pod to 2 pods. The event started when the HPA name 'keda-hpa-cfm-adapter-kafka-scaled-object' started to notice the Kafka consumer lag and started to generate a new size of the replicaset as shown in number 1 in the figure. Then HPA triggers the deployment of the cb-adapter to scale the replica set from 1 to 2 as shown in number 2 in the figure. Then the

replica set starts to create pods from this result, it can be concluded that the KEDA works correctly and the cb-adapter application has been successfully autoscaled from 1 to 2 pods based on Kafka consumer lag.

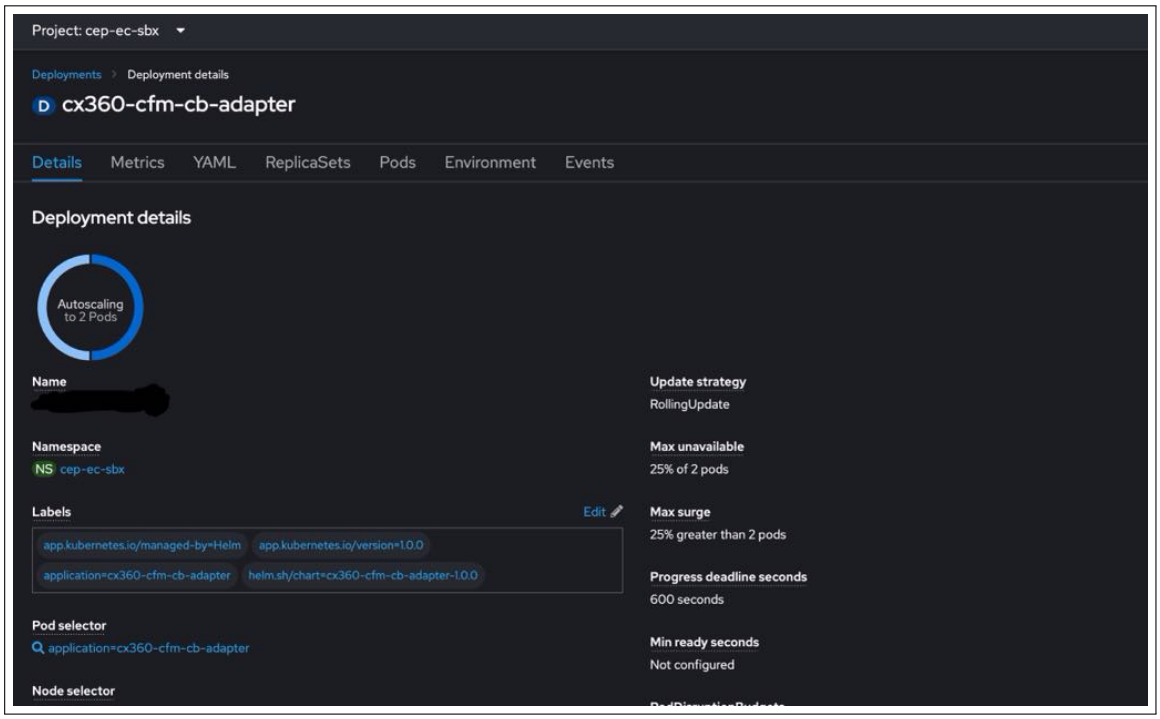


Figure 5.47: ARO Deployment of cb-adapter run on testing namespace, cluster shows the pod autoscaling from 1 to 2 pod base on kafka consumer lag

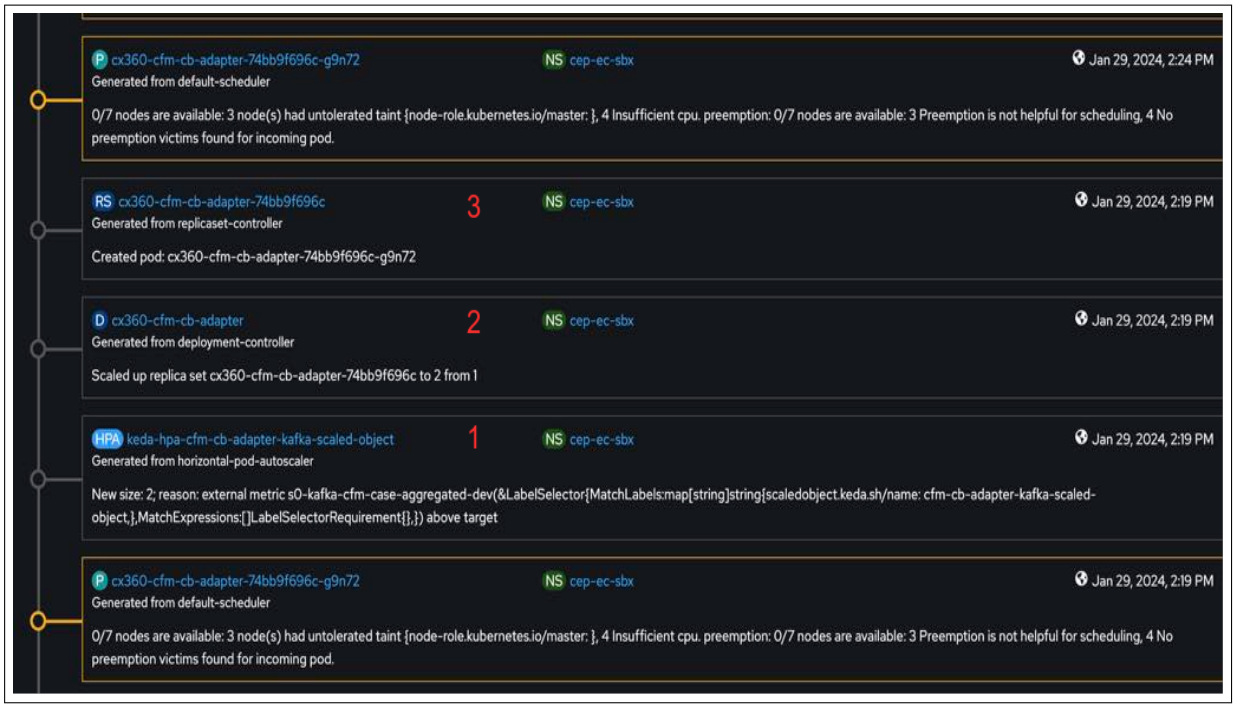


Figure 5.48: Event hpa start to scaling cb-adapter from 1 pod to 2 pod

5.5.2 Implement Auto Pod Scaling in Non-Production and Production Cluster

After testing KEDA on the testing cluster and found that KEDA works correctly. The team has been contacting the related team to install KEDA in the non-production and production clusters. The plan to implement auto pod scaling has changed to not implement this feature in CSLO service because auto-scaling pod might cause request unit Azure comosDB to increase and exceed the limit. When this happens, the team has to pay more money to increase the throughput of Azure comosDB. To tackle this problem, the Team decided to change the database from Azure comosDB to Postgresql database in the future. For the CPC application, The scaled object and trigger authentication have been created and applied to the QA, UAT, and Prod environments successfully. After receiving the requirements from the team, it was determined that the CPC application should consume messages from the Kafka topic within a maximum timeframe of 5 hours. With this information, The parameter of the scaled object `lagThreshold` has been explored to find the best value for the CPC application. After careful observation of the Kafka consumer lag of the CPC application, it has been determined that the application is capable of consuming messages from the Kafka topic at a rate of 6-12 messages/second per pod or 21,600-43,200 messages/hour per pod when there is a consumer lag. Based on this analysis, In 5 hours single pod can consume 108,000-216,000 messages. So the `lagThreshold` has been set to 100,000. With these parameter settings, the CPC application will initiate scaling when the lag of the Kafka consumer group exceeds 200,000 messages. For the CFM aggregator service and email cleansing service, there are a few scenarios in the service that might have consumer lag. This is because the aggregator service and email cleansing service process the message faster than the message coming to the Kafka topic. The situation was simulated to induce Kafka consumer lag at the CFM aggregator, aiming to test and determine the most suitable `lagThreshold`. Ultimately, the `lagThreshold` was set to 300 after thorough evaluation. So the CFM aggregator service and email cleansing service will initiate scaling when the lag of the Kafka consumer group exceeds 300 messages. In more detail, the configurations of the scaled object of the CPC application, CFM aggregator service, and email cleansing service are shown in Appendix 6.2.2. To validate the auto pod scaling feature in the non-production and production clusters, tests were conducted by inducing consumer lag in the CPC application in the development (DEV) and quality assurance (QA) environments at Kafka Topic A. The speed comparison of the message consumption between the CPC application in the DEV environment, which does not have auto-scaling, and the CPC application in the QA environment, which has the scaled object and trigger authentication for auto-scaling. It is important to note that both the CPC DEV and CPC QA environments have the same CPU and memory quota configuration.

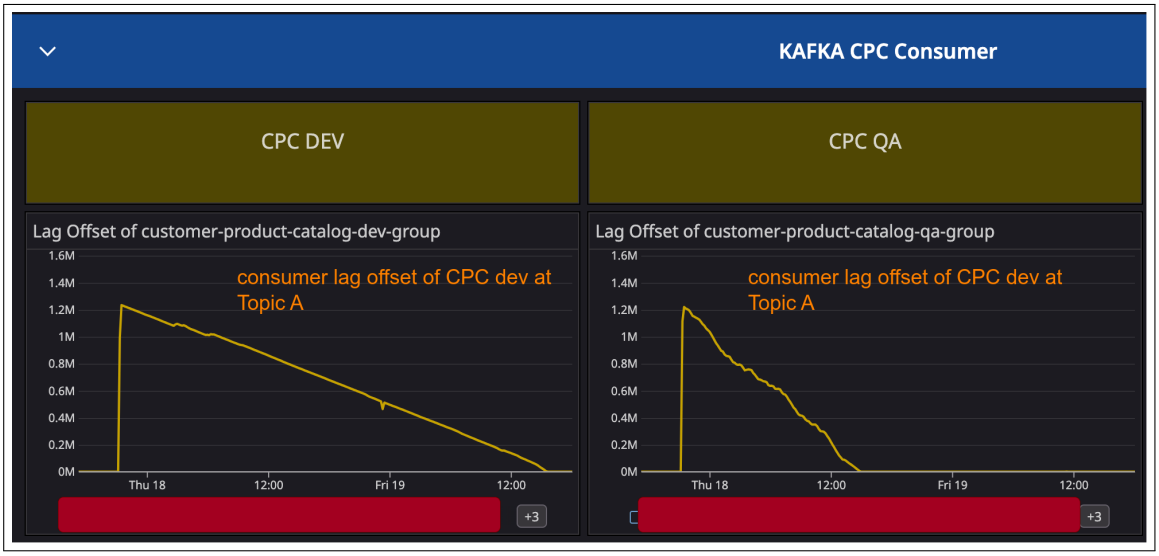


Figure 5.49: Time series show comparison of Kafka consumer lag offset of CPC in DEV and QA environment at topic A

After inducing consumer lag in the CPC application in the development (DEV) and quality assurance (QA) environments, the Kafka consumer lag offset of the CPC application in the DEV and QA environments at Kafka Topic A was compared in the time series graph, as shown in Figure 5.49. The graph illustrates that the Kafka consumer lag offset in the CPC QA environment is able to consume messages from Topic A at a faster rate compared to the CPC DEV environment. This is attributed to the autoscaling feature in the CPC QA environment, which scales the number of pods from 1 to 2 when the lag of the Kafka consumer group exceeds the threshold. Figure 5.50 shows the comparison of the consume rate of the CPC DEV and QA environments at Kafka Topic A. This illustrates that the CPC QA environment can consume message from Topic A at approximate 20 messages/second while the CPC DEV environment can consume message at approximate 8.5 messages/second.

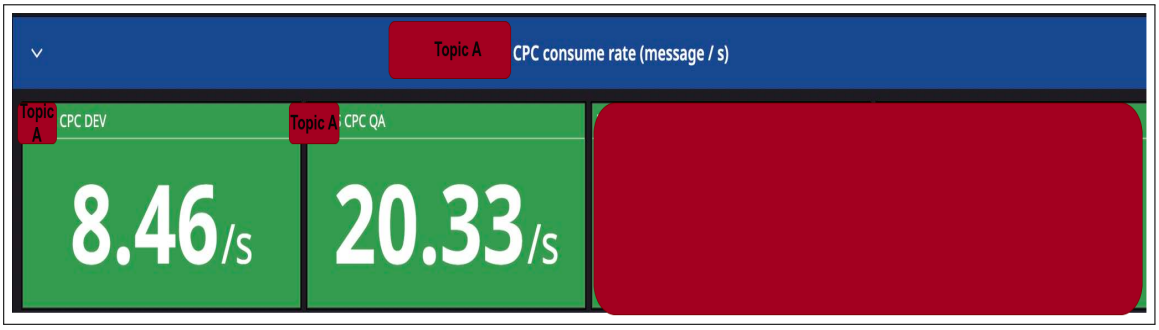


Figure 5.50: The Comparison of consume rate of CPC DEV and QA environment at topic A

5.6 Implement Result of API Load Test with K6

The API load test has been done by using K6 and run test on biweekly basis using github action workflow with scheduler trigger. Figure 5.51 shows the github action workflow of K6 API load test.

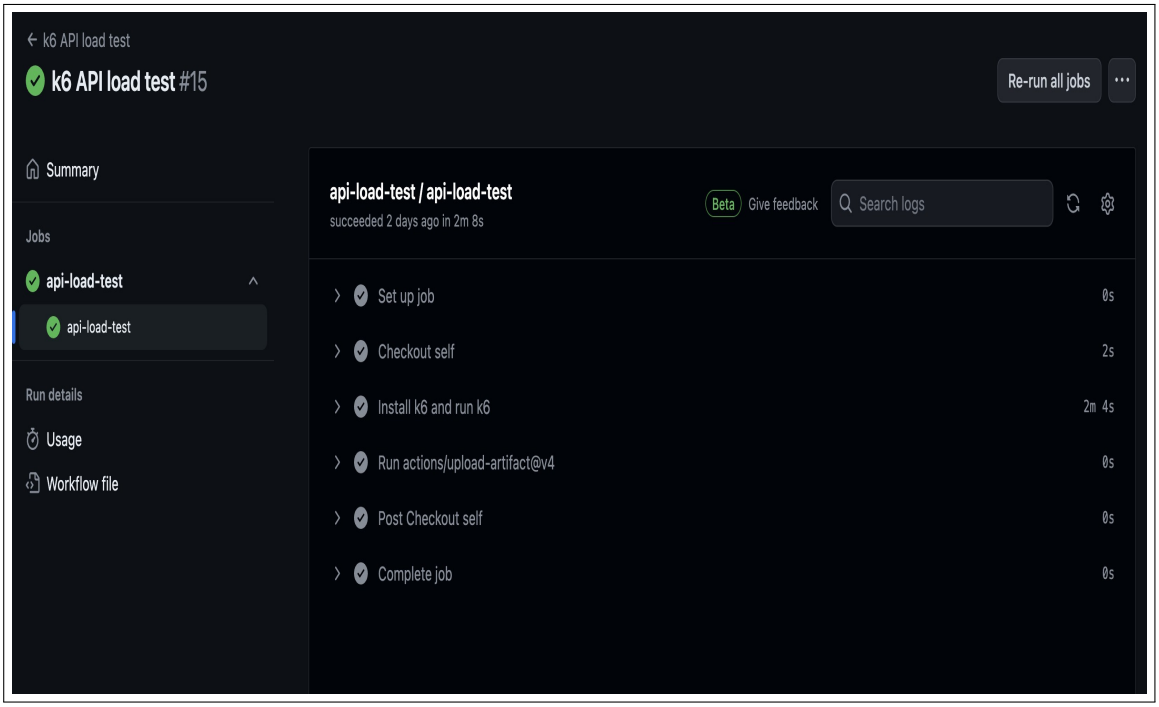


Figure 5.51: K6 api load test workflow

After github action pipeline run the K6 API load test finish, It upload the load test result as a html file to the github page. Figure 5.52 shows load test result overview of K6 API load test which generate from K6 api load test workflow. The overview of K6 API load test consists 3 parts as follows:

1. HTTP Performance Overview

   This part shows the performance of the HTTP request. The green line represent the request rate in request per second (rps). The blue line represent the response duration at p95 in milli-seconds. The purple line represent the request fail

2. Virtual users (VUs)

   This part shows the number of virtual users that simulate the load test. The green line represent the number of virtual users. The blue line represent the number of http request.

3. Transfer rate

   This part shows the transfer rate of the http request. The green line represent the data received in bytes per second. The blue line represent the data sent in bytes per second.

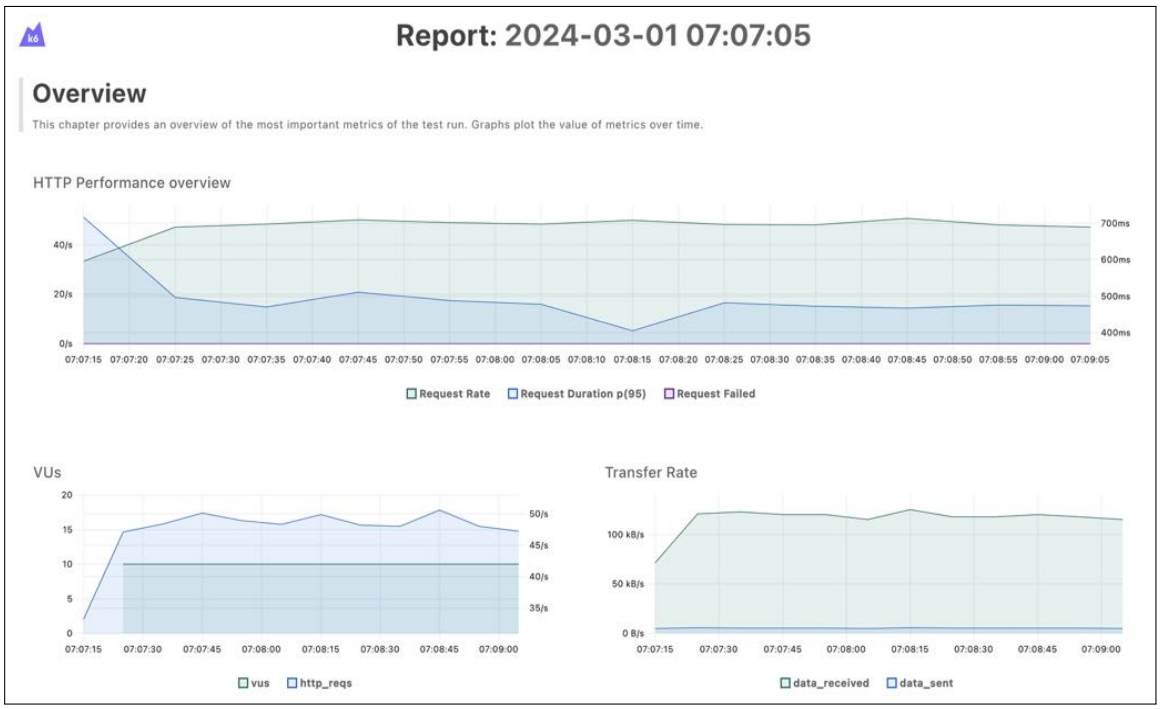


Figure 5.52: Example result of K6 API load test overview

Figure 5.53 shows the summary of K6 API load test. This include all http built-in metrics from k6. There are 4 important metrics that are needed to focus on, including:

1. `http_reqs`

   This metric shows the number of HTTP requests that k6 has sent to the application.

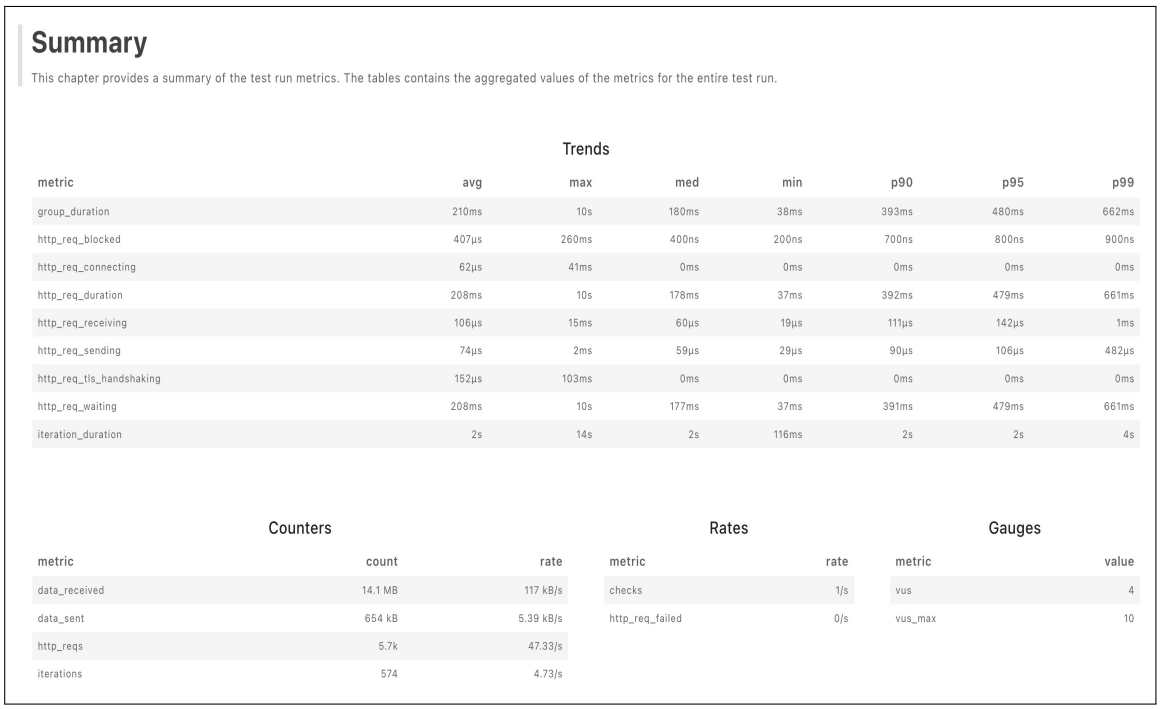
2. `http_req_failed`

   This metric shows the number of HTTP requests that k6 has failed to send to the application.

3. `http_req_duration`

   This metric shows the time it takes to complete the process from sending the request to receiving the response (without the initial DNS lookup and connection).

After implement the K6 API load test via run the test on biweekly basis using github action workflow with scheduler trigger, the team able to monitor the load performance of the application when there is a new feature release or when there is a new deployment of the application.



Summary

This chapter provides a summary of the test run metrics. The tables contains the aggregated values of the metrics for the entire test run.

Trends

metric	avg	max	med	min	p90	p95	p99
group_duration	210ms	10s	180ms	38ms	393ms	480ms	662ms
http_req_blocked	407µs	260ms	400ns	200ns	700ns	800ns	900ns
http_req_connecting	62µs	41ms	0ms	0ms	0ms	0ms	0ms
http_req_duration	208ms	10s	178ms	37ms	392ms	479ms	661ms
http_req_receiving	106µs	15ms	60µs	19µs	111µs	142µs	1ms
http_req_sending	74µs	2ms	59µs	29µs	90µs	106µs	482µs
http_req_tls_handshaking	152µs	103ms	0ms	0ms	0ms	0ms	0ms
http_req_waiting	208ms	10s	177ms	37ms	391ms	479ms	661ms
iteration_duration	2s	14s	2s	116ms	2s	2s	4s

Counters

metric	count	rate
data_received	14.1 MB	117 kB/s
data_sent	654 kB	5.39 kB/s
http_reqs	5.7k	47.33/s
iterations	574	4.73/s

Rates

metric	rate
checks	1/s
http_req_failed	0/s

Gauges

metric	value
vus	4
vus_max	10

Figure 5.53: Example result of K6 API load test summary

CHAPTER 6 CONCLUSIONS

6.1 Conclusions

6.1.1 Enhance the Performance of Email Data Cleansing Algorithm

From the results, It can be concluded that the new approach of email data cleansing algorithm has better performance than previous email data cleansing algorithm. This improvement was achieved through the utilization of a larger dataset for retraining the Long Short Term Memory (LSTM) model, the adoption of a different approach for non-sentence removal using the Spacy library, and the substitution of the stanza library with the presidio library for masking Personal Identifiable Information (PII) data. The outcomes demonstrate that the enhanced email data cleansing algorithm enhanced accuracy, faster processing speed, and increased resilience against potential vulnerabilities in the email data cleansing service.

6.1.2 Monitoring Service of Application Resources

After implementing the monitoring service of application resources using Datadog to monitor the resource usage of the application, It can be concluded that the monitoring service of application resources is working correctly and helps the team to monitor the resource usage of the application. The alert system has been set up to alert when there are issues that occur in the application. For the auto pod scaling feature, Kubernetes Event Driven Autoscaling (KEDA) has been used to implement auto pod scaling by trigger based on Kafka consumer lag. The proof of concept has been done by implementing KEDA in the testing cluster and deploying the cb-adapter application to the testing namespace. The result shows that the cb-adapter application has been successfully autoscale from 1 to 2 pods based on Kafka consumer lag as expected. After testing KEDA on the testing cluster and finding that KEDA is working correctly, the decision was made to implement the auto pod scaling feature in non-production and production clusters including Customer Feedback Management (CFM) and Customer Product Catalog (CPC) application. The result shows that the application that has an autoscale feature can consume messages from the kafka topic faster than the application that does not have an autoscale feature.

6.2 Future Works

6.2.1 Enhance the Performance of Email Data Cleansing Algorithm

1. Compare the performance of LSTM model with another machine learning model such as transformer model like BERT model, GPT model and etc.

6.2.2 Monitoring Service of Application Resources

1. Compare the performance of Datadog with another monitoring service such as Grafana and etc.
2. Improve the metrics use for alert system to reduce the number of alert email that not important.
3. Implement auto pod scaling base on http traffic request come to applications by using KEDA.

REFERENCES

1. Ian Goodfellow, Yoshua Bengio, and Aaron Courville, 2016, **Deep Learning**, MIT Press, http://www.deeplearningbook.org.

2. Shekhar, 2020, "Understanding Recurrent Neural Network (RNN) and Long Short Term Memory(LSTM)," Available at https://medium.com/analytics-vidhya/undestanding-recurrent-neural-network-rnn-and-long-short-term-memory-lstm-30bc1221e80d, [Online; accessed 07-Septemer-23].

3. Ottavio Calzone, 2022, "An Intuitive Explanation of LSTM," Available at https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c, [Online; accessed 12-Septemer-23].

4. Manu Rastogi, 2023, "Tutorial on LSTMs: A Computational Perspective," Available at https://towardsdatascience.com/tutorial-on-lstm-a-computational-perspective-f3417442c2cd, [Online; accessed 12-Septemer-23].

5. Tom Staite, 2020, "Everything you need to know about Regular Expressions (RegEx)," Available at https://medium.com/@tomstaite1/everything-you-need-to-know-about-regular-expressions-regex-3cbc5b95146, [Online; accessed 07-Septemer-23].

6. Christopher Marshall, 2019, "What is named entity recognition (NER) and how can I use it?," Available at https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d, [Online; accessed 30-October-23].

7. Martín Abadi, Paul Barham, and Team TensorFlow, 2016, "TensorFlow: A system for large-scale machine learning," **CoRR**, vol. abs/1605.08695, 2016.

8. S. Yegulalp, 2019, "What is TensorFlow? The machine learning library explained," Available at https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html, [Online; accessed 07-Septemer-23].

9. F. Arias, 2019, "Fuzzy String Matching in Python," Available at https://www.datacamp.com/tutorial/fuzzy-string-python, [Online; accessed 07-Septemer-23].

10. C. Gitau, 2018, "Fuzzy String Matching in Python," Available at https://towardsdatascience.com/fuzzy-string-matching-in-python-68f240d910fe, [Online; accessed 07-Septemer-23].

11. Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning, 2020, "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages," **CoRR**, vol. abs/2003.07082, 2020.

12. Sayali Moghe and Sanjan Das, 2022, "SpaCy for Natural Language Processing," Available at https://medium.com/@sayali.moghe.1008/spacy-for-natural-language-processing-fe7963e5fc57, [Online; accessed 29-Septemer-23].

13. Aradhita Bhandari, 2020, "Using SpaCy for Natural Language Processing," Available at https://medium.com/swlh/using-spacy-for-natural-language-processing-3d127d5ca5ba, [Online; accessed 30-October-23].

14. Spacy, 2022, "SpaCy Comparison," Available at https://spacy.io/usage/facts-figures#benchmarks, [Online; accessed 26-October-23].

15. Abderrahmane M., 2023, "Anonymizing Text with NLP: Strategies and Techniques for Ensuring Data Privacy," Available at https://medium.com/@manou.mohammedi/anonymizing-text-with-nlp-strategies-and-techniques-for-ensuring-data-privacy-598a99598b3a, [Online; accessed 29-Septemer-23].

16. Microsoft Presidio Team, 2023, "Presidio: Data Protection and De-identification SDK," Available at https://microsoft.github.io/presidio, [Online; accessed 26-November-23].

17. Alexander S. Gillis, 2023, "MongoDB," Available at https://www.techtarget.com/searchdatamanagement/definition/MongoDB, [Online; accessed 18-Septemer-23].

18. BDCC Global, 2023, "Azure Cosmos DB: Understanding the NoSQL Database Service Provided by Azure," Available at https://medium.com/@bdccglobal/azure-cosmos-db-understanding-the-nosql-database-service-provided-by-azure-1af8a02c3a51, [Online; accessed 27-Jan-24].

19. Kinsta, 2023, "What Is PostgreSQL?," Available at https://kinsta.com/knowledgebase/what-is-postgresql/, [Online; accessed 18-Septemer-23].

20. Redis, 2023, "Redis," Available at https://redis.io/, [Online; accessed 07-Septemer-23].

21. Apache Kafka, 2023, "Apache Kafka," Available at https://docs.confluent.io/home/overview.html, [Online; accessed 07-Septemer-23].

22. Angular, 2023, "Angular," Available at https://angular.io/, [Online; accessed 07-Septemer-23].

23. Angular, 2018, "Angular Overview," Available at https://medium.com/@mlbors/an-overview-of-angular-3ccd2950648e, [Online; accessed 15-Septemer-23].

24. Spring Boot, 2023, "Spring Boot," Available at https://spring.io/projects/spring-boot, [Online; accessed 07-Septemer-23].

25. R. Singh, 2023, "What is Java Spring Boot?," Available at https://www.ibm.com/topics/java-spring-boot#:~:text=starters%20are%20available.-,Standalone%20applications,app%20during%20the%20initialization%20process., [Online; accessed 07-Septemer-23].

26. R. Fadatare, 2023, "Which IDE is Best for Java Spring Boot Development?," Available at https://www.javaguides.net/2023/02/which-ide-is-best-for-java-spring-boot.html#:~:text=Conclusion,individual%20preferences%20and%20project%20requirements., [Online; accessed 07-Septemer-23].

27. Flask, 2023, "Flask," Available at https://flask.palletsprojects.com/en/2.0.x/, [Online; accessed 07-Septemer-23].

28. DataDog, 2023, "DataDog," Available at https://docs.datadoghq.com/, [Online; accessed 07-Septemer-23].

29. Kubernetes, 2023, "Kubernetes Documentation," Available at https://kubernetes.io/docs/home/, [Online; accessed 07-Septemer-23].

30. Bibin Wilson, 2023, "Kubernetes Architecture – Detailed Explanation," Available at https://devopscube.com/kubernetes-architecture-explained/, [Online; accessed 22-Septemer-23].

31. FoxuTech, 2023, "Horizontal Pod Autoscaler(hpa) — Know Everything About it," Available at https://foxutech.medium.com/horizontal-pod-autoscaler-hpa-know-everything-about-it-5637c7d2438a, [Online; accessed 21-Jaunary-24].

32. Cloud Native Computing Foundation, 2023, "Cloud Native Computing Foundation Announces Graduation of Kubernetes Autoscaler KEDA," Available at https://www.cncf.io/announcements/2023/08/22/cloud-native-computing-foundation-announces-graduation-of-kubernetes-autoscaler-keda/, [Online; accessed 19-Feb-24].

33. KEDA, 2024, "Kubernetes Event-driven Autoscaling," Available at https://keda.sh/, [Online; accessed 19-Jan-24].

34. Microsoft, 2023, "Azure Red Hat OpenShift Documentation," Available at https://learn.microsoft.com/en-us/azure/openshift/, [Online; accessed 07-Septemer-23].

35. Helm, 2023, "Helm Docs," Available at https://helm.sh/docs/, [Online; accessed 07-Septemer-23].

36. Grafana Labs, 2023, "K6 Documentation," Available at https://k6.io/docs/, [Online; accessed 07-Septemer-23].

37. GitHub Docs, 2018, "GitHub Actions Documentation," Available at https://docs.github.com/en/actions, [Online; accessed 07-Septemer-23].

38. Micrometer, 2023, "Micrometer," Available at https://micrometer.io/, [Online; accessed 07-Septemer-23].

39. Hiten Pratap Singh, 2023, "Unlocking Precision Metrics in Spring Boot with Micrometer: A Comprehensive Guide," Available at https://medium.com/javarevisited/unlocking-precision-metrics-in-spring-boot-with-micrometer-a-comprehensive-guide-6d72d6eaaf00, [Online; accessed 13-Mar-24].

40. Team Quickwork, 2021, "What is Zendesk? Learn about customer support automation," Available at https://medium.com/geekculture/what-is-zendesk-learn-about-customer-support-automation-4a014a829818, [Online; accessed 07-Septemer-23].

41. ClaraBridge, 2023, "ClaraBridge," Available at https://www.qualtrics.com/clarabridge/, [Online; accessed 07-Septemer-23].

42. Intellij, 2023, "Intellij," Available at https://www.jetbrains.com/idea/, [Online; accessed 07-Septemer-23].

43. S. Chand, 2023, "What is Java? A Beginner's Guide to Java and its Evolution," Available at https://medium.com/edureka/what-is-java-36be53c03cf1, [Online; accessed 07-Septemer-23].

44. N. Wijesiri, 2020, "What is python?," Available at https://medium.com/analytics-vidhya/what-is-python-d64059c65231, [Online; accessed 07-Septemer-23].

45. Diego Esteban Cortés, 2023, "Introduction to Typescript," Available at https://medium.com/@diego.coder/introduction-to-typescript-6b328184dba1#:~:text=Typescript%20is%20a%20superset%20that,such%20as%20Java%20or%20C%23., [Online; accessed 07-Septemer-23].

APPENDIX A

SCALEDOBJECT CONFIGURATION

Configuration of ScaledObject of CPC application

To deploy scaledobject into each namespace environment, Helm chart is used for help team manage scaledobject configuration. The following code shows scaledobject templates of helm chart.

```
{{- if .Values.keda.enabled }}
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: {{ include "project.fullname" }}-kafka-scaledobject-{{ .Values.environment }}
  namespace: {{ .Values.namespace }}
spec:
  scaleTargetRef:
    deploymentName: {{ include "project.fullname" }}
  pollingInterval: 15
  cooldownPeriod: 30
  minReplicaCount: {{ .Values.keda.spec.minReplicaCount }}
  maxReplicaCount: {{ .Values.keda.spec.maxReplicaCount }}
  fallback:
    failureThreshold: 10
    replicas: {{ .Values.keda.spec.minReplicaCount }}
  advanced:
    restoreToOriginalReplicaCount: true
  triggers:
  - type: kafka
    metadata:
      topic: {{ .Values.keda.trigger.topic }}
      bootstrapServers: {{ .Values.keda.trigger.bootstrapServers }}
      consumerGroup: {{ .Values.keda.trigger.consumerGroup }}
      lagThreshold: {{ .Values.keda.trigger.lagThreshold | quote  }}
      activationThreshold: {{ .Values.keda.trigger.activationThreshold | quote }}
      sasl: plaintext
      tls: enable
      authenticationRef:
        name: {{ include "project.fullname" }}-kafka-triggerauthentication-{{ .Values.
  environment }}
{{- end }}
```

Listing A.1: Helm ScaledObject Template

Table A.1 and Table A.2 shows the configuration of the ScaledObject of the CPC application in each environment. This include DEV, QA, UAT, and PROD environment. This configuration is used to describe the behavior of KEDA to scale the CPC application based on the Kafka consumer lag.

Table A.1 CPC ScaledObject Spec Configuration in each environment

ScaledObject spec name	Value QA	Value UAT	Value PROD
poolingInterval	30	30	30
minReplicaCount	1	1	2
maxReplicaCount	2	2	8
failureThreshold	20	20	20
fallback.replicas	1	1	2
restoreToOriginalReplicaCount	true	true	true

Table A.2 CPC ScaledObject triggers Configuration

ScaledObject trigger properties name	Value QA	Value UAT	Value PROD
lagThreshold	100,000	100,000	100,000
activationThreshold	0	0	0
offsetResetPolicy	latest	latest	latest
scaleToZeroOnInvaildOffset	false	false	false
limitToPartitionWithLag	true	true	true
sasl	plaintext	plaintext	plaintext
tls	enable	enable	enable

Configuration of ScaledObject of CFM Aggregator and Email Cleansing service

Table A.3 and Table A.4 shows the configuration of the ScaledObject of the CFM aggregator. Table A.5 and Table A.6 shows the configuration of the ScaledObject of the Email Cleansing service. This configuration is used to describe the behavior of KEDA that scales the CFM aggregator and Email Cleansing service based on the Kafka consumer lag.

Table A.3 CFM Aggregator KEDA ScaledObject Spec Configuration

ScaledObject spec name	Value UAT	Value PROD
poolingInterval	30	30
minReplicaCount	2	2
maxReplicaCount	3	3
failureThreshold	15	20
fallback.replicas	2	2
restoreToOriginalReplicaCount	true	true

Table A.4 CFM Aggregator KEDA Trigger Authentication Spec Configuration

ScaledObject trigger properties name	Value UAT	Value PROD
lagThreshold	300	300
activationThreshold	0	0
offsetResetPolicy	latest	latest
scaleToZeroOnInvaildOffset	false	false
limitToPartitionWithLag	true	true
sasl	plaintext	plaintext
tls	enable	enable

Table A.5 Email Cleansing KEDA ScaledObject Spec Configuration

ScaledObject spec name	Value UAT	Value PROD
poolingInterval	30	30
minReplicaCount	2	2
maxReplicaCount	3	3
failureThreshold	15	20
fallback.replicas	2	2
restoreToOriginalReplicaCount	true	true

Table A.6 Email Cleansing KEDA Trigger Authentication Configuration

ScaledObject trigger properties name	**Value UAT**	**Value PROD**
`lagThreshold`	300	300
`activationThreshold`	0	0
`offsetResetPolicy`	latest	latest
`scaleToZeroOnInvaildOffset`	false	false
`limitToPartitionWithLag`	true	true
`sasl`	plaintext	plaintext
`tls`	enable	enable