National College of Ireland

Multi-Cloud Deployment and Performance Benchmarking of a Dockerized Fake News Detection Application

MSc Research Project
MSc Cloud Computing

Roshan Jawahar
Student ID: x23128356

School of Computing
National College of Ireland

Supervisor: Abubakr Siddiq

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Roshan Jawahar
Student ID:	x23128356
Programme:	MSc in Cloud Computing **Year:** 1
Module:	MSc Research Project
Supervisor:	*Abubakr Siddiq*
Submission Due Date:	12/12/2024
Project Title:	Multi-Cloud Deployment and Performance Benchmarking of a Dockerized Fake News Detection Application
Word Count:	6776 **Page Count:** 22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Roshan Jawahar

Date: 12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Multi-Cloud Deployment and Performance Benchmarking of a Dockerized Fake News Detection Application

Roshan Jawahar
x23128356

Abstract

This Research project aims to address a prevalent issue of identifying any fabricated information being shared online to curb the widespread of any false information being shared online. To achieve this, a robust fake news classification application is developed using flask framework with an underlying advanced Machine learning model such as Hybrid Ensemble Classifier to analyse the textual news data and distinguish them as genuine or fake. A Docker image is created for this application along with its dependencies such as Flask, packed in the required format for improving scalability and simplifying deployment across various platforms for ease of access. The Application is tested with high concurrency loads generated by Locust and to combat any performance issues for dynamically scaling the computational resources based on demand. For this reason, it is essential that the metrics are collected and studied in detail. This project utilizes AWS CloudWatch and Azure Monitor cloud tools to collect different metrics of the application under study to organize a performance evaluation between AWS and Microsoft Azure to assess the application's efficiency. This comparison is essential in aiding the decision that would have to be taken when choosing wither of these cloud platforms for a Dockerized application deployment of similar magnitude.

1 Introduction

Social media plays an important role in today's world in several aspects and it is and integral part of the modern age society, drastically changing the way people communicate and stay connected through sharing crucial information on a day-to-day basis. Social media platforms like X (formerly Twitter), Instagram and Facebook act as the mainstream media in multiple regions of the world where people share their opinion on public matters, paving way to global interactions and boosting their businesses. In some regions, essential services like Facebook Free Basics provide internet access to those who were not privileged enough to access it, which is crucial for sharing and offering information about healthcare, job opportunities and education. This proves that, for these people, social media is not just entertainment, but much more. As these platforms become an indispensable part of their daily lives, verifying the content shared on these platforms must be verified for its authenticity as the spread of misinformation poses a serious risk. Therefore, developing a trustworthy fake news detection system is of the utmost importance to safeguard social stability and public trust.

This constant threat underscores the necessity of trustworthy tools and detection systems to combat the spread of misinformation and to ensure the authenticity of the shared content. So, developing a application that can accurately classify fake news from the authentic news is the need of the hour, as it provides a efficient and practical solution to this problem. Its flexibility, ease of setup and enormous community support makes it the ideal framework to build scalable web applications. This application that has been created in this research has a Machine Learning model combined with in the form of Hybrid Ensemble Classifier, which is a combination of Decision Tree, random Forest and Gradient Boosting ML models. The ability to process large amounts of data in real time has led to this choice to ensure the integrity of content on social media.

To make sure that application is easy to deploy across multiple environments without any complications, the services of Docker is called upon here. The packaging of the application along with its dependencies into a lightweight container in the form on a Docker Image simplifies the deployment approach and ensures the performance is even across multiple cloud platforms. This has several advantages, from consistent and efficient scaling, which enables the ability to handle high user traffic while also maintaining high reliability. The successful deployment of the application on AWS and Azure cloud platforms majorly depends on tools like Docker.

1.1 Aim of the study

The primary objective of this whole research project is to efficiently evaluate the performance of a Dockerized web application that is deployed on AWS and Azure cloud platforms. The main focus of this study is in determining the better cloud platform in terms of reliability, scalability and cost efficiency when a application of this configuration is deployed on cloud. The examination of two different deployment strategies in the form of AWS's Elastic Beanstalk and Azure's Container Apps and their impact on the application's overall scalability, fault tolerance and the performance is looked upon in this report. The cloud services used to conduct the performance analysis on the operational health of the deployed applications are AWS CloudWatch and Azure Monitor. The applications have been subjected to high load generated using the load testing tool, "Locust" to simulate real world scenarios. This research provides valuable insights for the organization and its developers on what is the optimal cloud deployment strategy that brings the most performance out of their application.

Research Question: In what ways do cloud specific deployment strategies of a Dockerized application impact real-time performance under load?

1.2 Research Objectives

The objectives of this research are as follows:

1. To create a web application with a underlying Machine Learning model using the Flask framework.
2. To use Docker Desktop and dockerize the application using the Dockerfile created which is compatible with the application.
3. To deploy the Dockerized application on the Azure cloud environment successfully using the tools like Azure Container Registries and Azure Container Apps.
4. To utilize a CI/CD pipeline on the AWS deployment phase using AWS CodePipeline and AWS Elastic Beanstalk where the source of the codebase is the GitHub repository contain the application code and the Dockerfile.
5. To create load on the application using Locust to simulate real life scenarios to generate invaluable data for collecting insights using monitoring tools such as AWS CloudWatch and Azure Monitor.
6. To conduct performance analysis of the applications using the collected metrics to determine the better cloud platform for deploying a Dockerized application.

1.3 Structure of the report

Followed by the Introduction and research objective in Section 1, the Section 2 contains the related work of various authors discussing several parts of this project such as Docker, performance comparison and even about the complications of training the Machine Learning models. The approaches employed in the literature reviews are critically reviewed, evaluated and scrutinized before considering the outcomes of the literature study.

Section 3 explains the methodology, pertaining to training the Machine Learning models, starting from preprocessing the data to remove null values and drop unnecessary columns, model training and finally the use of different visualization techniques such as correlation matrix, bar graph to map the outcomes.

Section 4 discusses the design specifications along with the architecture and the workflow of the research project followed by Section 5, which talks about the detailed flow of end-to-end implementation showing the cloud deployment processes in both AWS and Microsoft Azure.

In Section 6, the application performance testing is conducted with Locust and metrics obtained from AWS CloudWatch and Azure Monitor are collected to evaluate the end results.

Finally, Section 7 is about the key conclusion derived from the experiments along with discussing the limitations and future works that this research paves the way for.

2 Related Work

2.1 Docker Deployment Advantages

One of the leading technologies in the world of containerization is the tool, Docker, which has redefined the process of deployment and application management on cloud environments as it has allowed the aspect of virtualization of OS into a lightweight containers.IT has been adopted in various domains and it is due to its advantages over other techniques in aspects such as scalability, efficiency and rather simplified application management. Docker's role in managing the resources for large-scale systems is studied in this research paper where the need for effective monitoring is emphasized to enable optimal deployment strategies, enable auto scaling and to evaluate system performance. This research focusses on the metrics such as CPU Utilization and Network I/O overhead that is caused by Docker containers and images and offers detailed insights into the trade-offs between efficiency and performance when places under different loads (Emiliano Casalicchio, 2017).

Another similar study dives deep into the deployment of an astronomy application using Docker to present its advantages such as portability, ease of use and a vast open-source community support. The research is split into two parts, where the first part is about a demonstration of the deployment and next one pertaining to the detailed set up to the tools, services and the parameters for the purpose of data processing of these Docker containers. This displays the strengths of the Docker deployments in a way where it simplifies software deployment and administration of the systems through its capabilities pertaining to container management, and the paper is concluded stating that these are the most important aspects of deploying a Docker image on Cloud (D. Morris, 2017).

The concept of virtualization is very much prevalent in today's world as it allows multiple services run on cloud platforms seamlessly. This technology provides support to create several virtual machines on a single physical machine by utilizing tools and techniques such as hypervisors and containers. Here is where Docker comes in as a open-source containerization platform, where it provides and enables lightweight virtualization while producing practically no or very little overhead. This makes it ideal for deploying or hosting applications that are based on microservice architecture. A study relating to this topic was conducted where the performance of Docker was compared against conventional methods of creating virtual machines utilizing benchmarking tools such as Phoronix, Sysbench and Apache benchmark, it revealed valuable insights about the CPU, storage, memory and performance related to load-testing, in turn highlighting Docker deployment's efficiency for cloud platforms (Amit M Potdar, 2020).

A similar paper has been studied which deals with containerization and the benefits of integrating it with deep learning model deployments. This technology simplifies the frequent management of software frameworks that needs updating. This study benchmarks the impact of Docker in machine learning application by comparing it with the essential metrics such as Memory, CPU and I/O performance inside a container environment and outside it. The findings signify that the metrics displayed inside the container environment have minimal overhead for processing computationally intensive tasks and states that Docker can perform great for machine learning applications with minimal performance trade-offs (P. Xu, 2017).

Docker images and containers enhance the performance of the deployment process of various models and applications as it provides lightweight and scalable solutions unlike hypervisor-based virtualization solutions, which makes it suitable for microservice architectures. An application deployment model for the purpose of optimizing the placement of containers and task assignments is formulated in a study to minimize the costs of deployment and simultaneously meet the requirements related to service delays. The display of flexibility and cost-efficiency was displayed by the container framework unlike the existing conventional strategies that make use of tools like Docker Swarm and Google Cluster Traces (Xili Wan, 2018).

On a similar note, an investigation on the utilization of Docker technology in deploying Machine learning projects and developments has been undertaken, this investigation identified 6 project categories which leveraged Docker to avail services such as platform portability and runtime compatibility. The conclusion was that, in order to use Docker for deployments of these projects, the fine-tuning of several parameters pertaining to the deployment process had to be done as some Docker images might require more resources to run which poses a serious issue when the deployment is done in a large-scale capacity (Moses Openja, 2022).

2.2 Benefits of Cloud Computing

There are many advantages of cloud computing which makes it favorable to those who want to make the best of their resources. It offers flexibility in that the user does not have to invest on hardware and services; he or she only pays for services consumed. Cloud services are elastic since resource can easily be increased or decreased depending with the traffic hence guaranteeing flexibility (Saif, 2021). It also promotes remote working since users can work on data and applications from anywhere, they have internet connection. Integration is enhanced since users are able to edit files and applications within the same or different rooms. In addition, cloud platforms are reliable because they update themselves and have automatic backups in an event of a failure.

2.3 Fake News Classification using Machine Learning

The classification of fake news has received a lot of attention to looking for ways to increase the accuracy of the machine learning models that are used in the detection on the fake news. Some of the used models are Naive Bayes, Decision Trees (J48), Random Forest, Extreme Gradient Boosting (XGBoost), Gradient Boosting, and multi-source framework. Naïve Bayes, together with Count Vectorizer has fair accuracy of detection of fake news with the precision rate in the range of 84 % – 94%.

In the work mentioned given by (Adiba, 2020) corresponds to the increased problem in the spread of fake news on online platforms on the part of persons who have ill intentions or a wrong approach. Natural Language Processing and machine learning techniques are applied to this study, with the Naive Bayes Classifier – a Bayesian-based classification algorithm – applied for sorting the fake news. The proposed approach is the improvement of the Naive Bayes algorithm based on the enriched corpora concept, which uses a higher number of

corpora for the training of the algorithm model. One of the main issues encountered during the work on the project was how to sort out and differentiate between the reliable and fake news included into the numerous online articles. First, the NB method attained 87% classification rate, and after enhancing the corpora, it was increased to 92%, a higher rate than previous studies on fake news detection.

The problem of fake news spread, including through the social media is discussed in (Yuslee, 2021) and it is done by reviewing the application of the Naive Bayes algorithms for detecting fake news. Issues affecting the audience are shown where as much as advanced technology brings the news and fast information flow it also contributes to the influx of fake news. The outlined plan of analyzing the data is that the data is preprocessed through means of regular expression, stopwords elimination and lemmatization, then it is transformed into N-gram through Term Frequency-Inverse Document Frequency (TF-IDF) and Count Vectorizer. One of the most important problems in this research was to make the fake news detection procedure more accurate and robust. The results suggest that Naive Bayes with N-grams slightly increases the accuracy and on the other hand, TF-IDF Vectoriser takes more efficiency in identifying fake news with precision of 94%. On the other hand, Count Vectorizer gives a better differentiation between fake and real news.

Another study given by (Jehad, 2020), the detection of fake news, specifically political fake news is discussed by identifying the key issues of utilizing machine learning algorithms in handling the lack of benchmark datasets for massive, rapidly published news. The research focuses on the employment of two classifiers, namely Random Forest and on Decision Tree (J48), and then classify fake news. The data set used includes twenty thousand seven hundred and sixty-one sample size with a testing sample size of four thousand three hundred and forty-five. Here some of the preprocessing steps used in the current study include; Data cleaning where non-significant characters, numbers, English letters and white spaces are deleted, stopword removal. TF-IDF is then used to feed the feature extraction aspect of the classification process. The findings indicate that among the models, the Decision Tree model performed the best giving an accuracy of 89.11%, followed by the Random Forest model with an accuracy of 84.97.

The problem of fake news dissemination on online social media is also discussed in another study provided by (Birunda, 2021), where a new Score-based Multi-Source Fake News Detection framework is introduced. The framework is expected to identify fake news based on features derived from the text elements of real and fake news articles using Term Frequency-Inverted Document Frequency (TF-IDF). It also computes the credibility of news sources based on component variables such as site URL and Top-Level Domain (TLD). When the text-based features are combined with the source credibility score, the framework measures the procedural honesty of the news. The first and probably the most significant limitation of this study is that it was quiet challenging to quantify the amount of genuineness in news. To test the proposed framework, different classifiers of machine learning are used, and the highest classifier’s accuracy of 99.5% is achieved when the Gradient Boosting algorithm is applied.

Finally, (Haumahu, 2021), the research focuses on fake news specifically, hoaxes that are negative consequences in social communities in regard to hatred and division. The study recommends classifying the news as hoax and valid news using the Extreme Gradient Boosting (XGBoost) technique and analyzing the Indonesian news during the period December 2015 to early 2020. There are 500 articles in the dataset, of which 250 are valid news, and 250 are hoax news, and there are 80% training data and 20% testing data. One of the complexities with this study lies in correctly identifying fake news from actual news within this environment defined by political polarisation. The results prove that the presented XGBoost model successfully provides the accuracy of 89%, the precision of 90 %, and the recall of 80 %, thus, it can be supposed that the model effectively detects fake news.

2.4 Performance Monitoring on Cloud

A performance evaluation of the cloud-based web applications has been conducted under varied circumstances, where the examination of how altering the type of AWS Elastic Cloud Compute (EC2) instances based on its parameters and configuration metrics affect the performance of the application hosted on it, such as response times and latency, where the values fluctuated more 10 times the normal limits due to the change in the factors like EC2 usage timings and the type of instance (J. Mukherjee, 2014). Another paper focusses on the performance testing tools and talks about the importance of these tools in properly testing e-commerce websites like Amazon, the testing tools under the spotlight here are Blazemeter and Load impact. The emphasis is placed on the necessity of cloud-based performance analysis and assessment tools to ensure the web applications' operations are seamless and the customer satisfaction is always high (Dr. P. Arul, 2014).

The performance evaluation of cloud services has been conducted where a simulation and benchmarking methodology for the purpose of predicting cloud-based applications performance, especially using the MOSIAC framework. This technique exhibits accurate prediction metrics in terms of performance like throughput and message queuing length vary under various workloads (Antonio Cuomo, 2015). The focus on IaaS platform and models have been emphasized the topics of virtualization and the issue of multitenancy during the process of workload execution. The need for conduction a empirical evaluation of different policies like provisioning and allocation in order to optimize the performance and lower the incurred cost in a cloud based infrastructure. The importance of performance benchmarking and prediction of resource management in cloud environments is necessary (Antoniou, 2012).

After reviewing the limited literature available for the topic of Performance bench marking of a Dockerized application, the research gap identified here is the insufficient amount of research in the department of performance evaluation of docker deployments. This research project addresses exactly that and hopes to add to the existing research in a positive way.

3 Research Methodology

The research methodology in commenced with the identification of the ideal dataset and the choosing of the apt Machine Learning Models for conducting research of this magnitude. The dataset that was chosen was picked from Kaggle which contained the real and fake news published on social media platforms such as Facebook, Instagram and X. This dataset was further preprocessed and prepared to help with ML model training.

Dataset Description

The dataset for this fake news classification project comes from Kaggle and includes two separate CSV files: Fake.csv –with 23,502 fake news articles and True.csv – with 21,417 true news articles. Each file has the following columns: In this study, there are four variables, namely Title, Text, Subject and Date whereby Title is the title of the news article and Text constitutes the remaining body of the article. To achieve maximum throughput and since there are constraints in the resources used during processing, we decided to work with a reduced data set, namely 10,000 records from each of the files. This makes it easier to train and to subsequently test the classifiers on a balanced dataset. The given dataset is well suited for machine learning tasks connected with text analysis and allows developing models based on both lexical and semantic information from headlines and the text itself, which are short as well as extended, and represent fake news. Due to differences in articles' length, topics, and time of publication, this set of data is suitable to build models to detect patterns tied with misinformation, thus it is useful for analyzing possible classification task of Machine learning algorithms in distinguishing news content.

Data Preprocessing – Google Colab

Initially, several libraries are imported onto the Google Colab environment where the Data preprocessing and model training takes place, those libraries include, numpy and pandas for tasks such as data manipulation and conducting mathematical computations. Text processing and Natural processing Libraries such as re and string for regular expressions and string operations, Natural Language ToolKit (nltk) for NLP, where it comes along with nltk.corpus for cleaning stopwords, finally wordninja for splitting concatenated words. Visualization libraries such as matplotlib, seaborn and wordcloud. For ML, all the libraries are from SciKitLearn (SKL), from which all the ML models and the metrics to calculate the accuracies are imported.

After importing the necessary libraries, they are utilized in the subsequent data preprocessing steps where the clean_text column of the final dataset is processed by removing the HTML references and the presence of URLs, expanding shortened words (eg. isn't, didn't), filtering out the punctuation and characters that are non-printable, separating the alpha numeric words, dealing with words that are consecutive and repetitive, separating attached words and removing the stop gap words. The primary purpose of this is to remove

noise from the dataset and standardize the text, which will help greatly in processing the meaningful features for processing using NLP task of Fake news classification.

Data Visualization

Word Cloud is used initially to visualize the most recurring words in the whole dataset. It is done to provide a visually engaging way to know and identify the most prominent words or the theme of the articles present in the article. The size of the word is directly proportional to the number of times they occur in the articles in the dataset. This is particularly useful to determine what kind of words are used by fake news perpetrators and the kind of words that are most likely to be present in those articles.

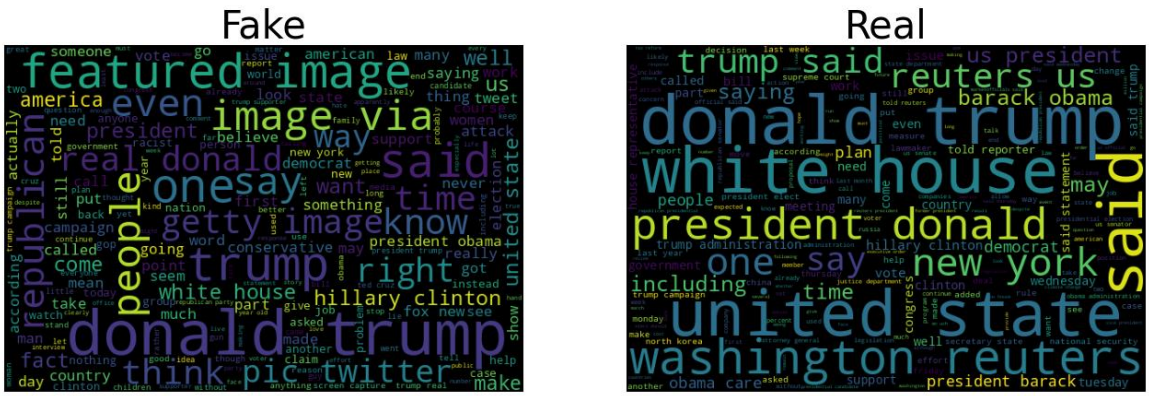

Figure 1: Word Cloud Comparison Between Fake and Real News Articles

The value count of the 20,000 articles present in the dataframe is visualized in this part using a bar plot, this is necessary in understanding the text data's characteristics as identifying the text lengths or the outliers in terms of number of words present in the article greatly aids in data processing and while feature engineering several automated tasks such as text classification. This is done in situations where the text might hold significant value in possessing a relevant feature.

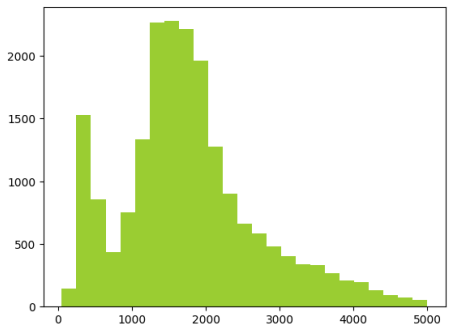

Figure 2: Histogram Distribution of Text Lengths in News Articles

3.3 Tools and Technologies Used

1. **Kaggle:** For collecting the relevant social media news dataset.
2. **Google Colab:** For the purpose of data Pre-processing and Machine Learning model training, such as Decision Tree, Random Forest, Gradient Boosting Classifiers and Hybrid Ensemble Classifier like stacking learning in this case.
3. **VS Code:** To create the Application for Fake news detection using **Flask** framework based on Python.
4. **Docker Desktop:** Containerization tool which was used to pack the application along with tis dependencies into a **Docker image** using a **Dockerfile**.
5. **Microsoft Azure:** Cloud Service Provider in which the containerized application is deployed using services like **Azure Container Registries (ACR)** to store the Docker image, **Azure Container Apps (ACA)** for deploying the application and **Azure Monitor** for collecting insights on the operational health of the application.
6. **Azure CLI:** Used to push the Docker image form Docker Desktop to the Azure Container Registries.
7. **GitHub:** Used to create a repository for maintaining the application codebase along with its Dockerfile which is later utilized by AWS for setting up a CI/CD pipeline.
8. **Amazon Cloud Services (AWS):** Another Cloud service provider where the application is deployed using the **CI/CD pipeline** with the help of tools such as **AWS CodePipeline** and **AWS Elastic Beanstalk** for deployment**. AWS CloudWatch** is also made use of here to gain valuable insights on the application's operational health.
9. **Terraform:** Open-source tool that provides IaaS for defining and creating the AWS Elastic Beanstalk and its infrastructure parameters and handling its resource allocation.
10. **Locust:** An open-source tool used to write scripts for load testing the deployed applications to create a real-world scenario and collect the operational metrics of the application pertaining **Latency**, **CPU Usage** and **Network I/O**

4 Design Specification

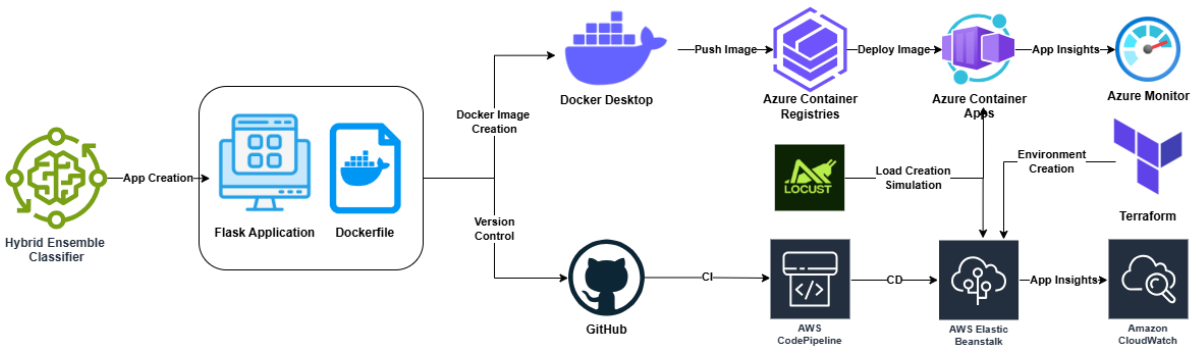

Figure 3: Architecture Diagram of the research project

The architecture flow of this research project starts with training the Hybrid Ensemble Classifier Machine Learning model with the Kaggle dataset containing the fake and real news articles from social media. The trained Model is then used as the base the create a Flask application. A Dockerfile is created that includes the instructions for creating a Docker image for the application. Hereon, the cloud deployment is executed in two phases;

The Azure deployment phase starts with the Docker image creation of the application code base and the associated Dockerfile, this step is successfully completed using Docker Desktop. Subsequently, a repository on the Azure Container Registries (ACR) is created to which the Docker image is pushed to. ACR acts as the source for the Dockerized application deployment on Azure Container Apps (ACA), where the Docker image is fetched from and deployed, producing a URL for users to access it. Finally, a Azure monitor dashboard is set up for the deployed application to gain valuable insights on the operational health while it is being subjected to heavy loads generated by a Locust script. Those metrics include Average Response time, CPU Usage Network I/O in bytes.

The deployment phase on AWS is initiated when the application codebase and the supporting Dockerfile is pushed on to a GitHub repository that was created beforehand. Then, a Continuous Integration / Continuous Deployment pipeline is set up in AWS with the usage of tools like AWS CodePipeline and AWS Elastic Beanstalk. The source for AWS CodePipeline is set as the GitHub repository that contains the Flask application and the Dockerfile. The deployment stage in the pipeline is satisfied by AWS Elastic Beanstalk, and the Beanstalk environment was created using a Terraform script which contained all the parameters set for the environment creation. The Elastic Beanstalk builds the Docker image from the obtained Dockerfile in the underlying Elastic Cloud Compute (EC2) instances that were created by an automated process by AWS Elastic Beanstalk. A metrics monitoring dashboard was set up for the deployed application using AWS CloudWatch for extracting valuable insights on the application and its functions when it is put under heavy loads by simulating large number of users accessing the site using a Locust script. The metrics that were collected are ApplicationLatencyP10, CPUUtilization, Network I/O.

Finally, a performance comparison was conducted by evaluating the similar metrics with each other to determine the better cloud service provider for the deployment of a Dockerized application of this specific configuration and of this scale.

5 Implementation

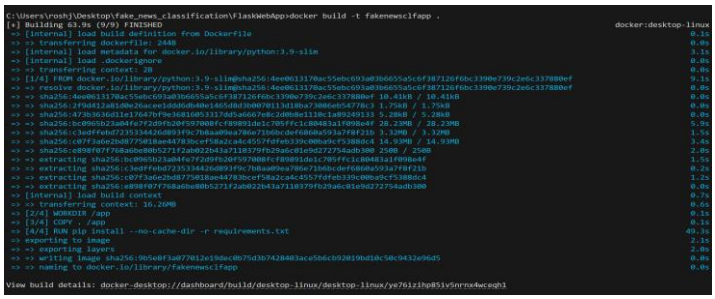

Figure 4: Command for building Docker image

A docker image was created for the Application using Docker Desktop. It was done by creating a Dockerfile that contained all the instructions for building the said image. The Image creation was initiated using the command “docker build -t fakenewsclfapp” on cmd and the image is stored by Docker Desktop. This Docker image is pushed to Azure Container Registries’ repository using a series of commands executed on the PowerShell terminal on the local device which will be looked at in detail in the configuration manual.

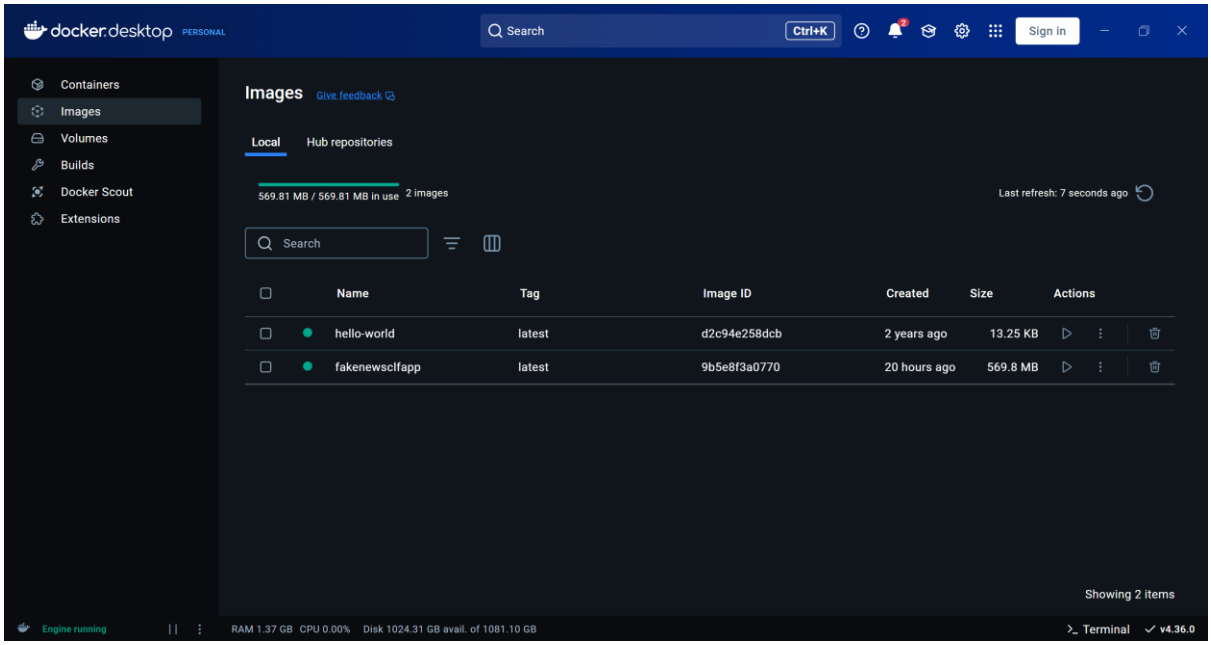

Figure 5: Presence of the Docker Image in Docker Desktop

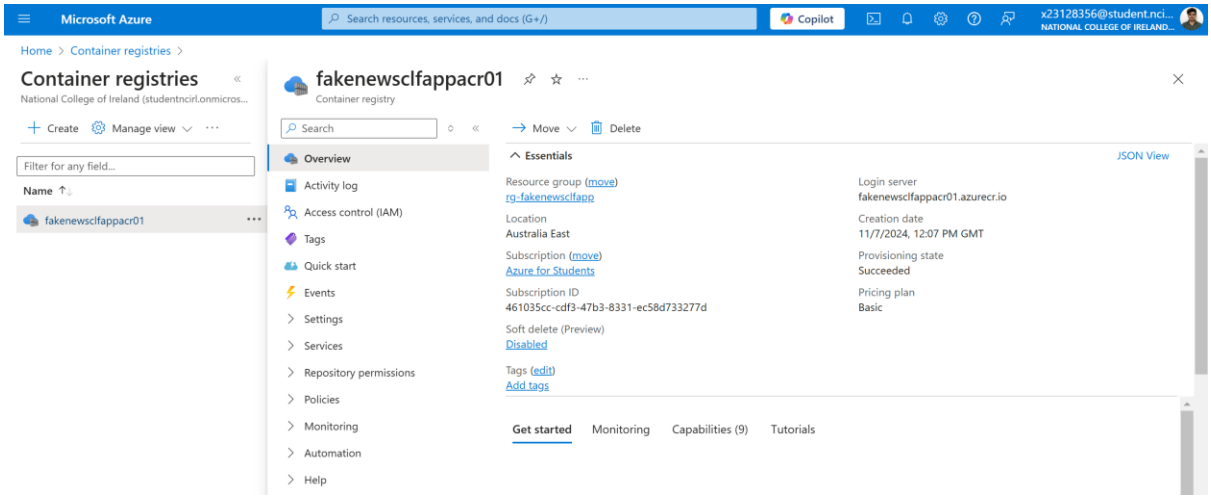

Figure 6: Azure Container Registry repository containing the Docker Image

The Docker Image of the Flask application is pushed onto the Repository that was created on Azure Container Registries. The application was initially Dockerized using Docker Desktop, where the app along with its dependencies like the configuration files and requirements were formed into a Docker Image. The Azure Command-Line Interface was then installed and configured in order to push the Docker image onto the ACR repository.

The repository was assigned a tag to easily identify it in the latter stages of the deployment process The usage of ACR has enabled the integration of other Microsoft Azure services, especially while integrating other services like Azure Container Apps and Azure Monitor for collecting valuable insights on the performance of the said tools.

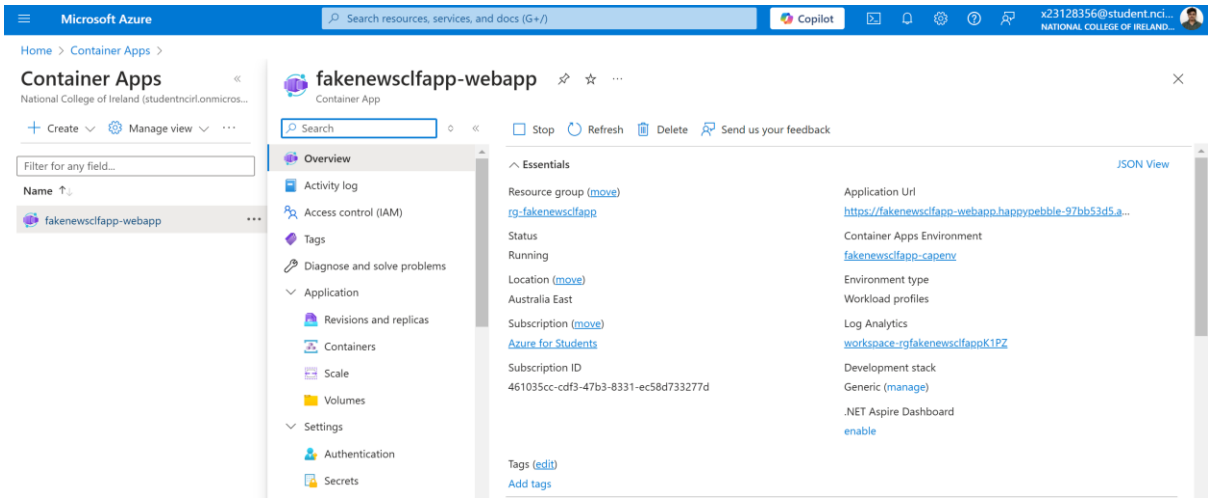

Figure 7: Azure Container Apps where the Dockerized application is deployed

The Deployment of the Docker image present in the ACR repository is done in this stage, on Azure Container Apps. The Application is created and the Repository present in ACR is set as the source for the deployment, where the Docker image is deployed in an automated process. The Image is identified using the tag given to it in the previous stage. The application's various configuration settings were confirmed in this stage such as the required allocation of storage space and memory for performing the computational tasks. This whole process is situated inside the Azure ecosystem and this enables for smooth integration, deployment and the surveillance processes using Azure Monitor.

The deployment process on AWS different than the Azure deployment, where the Application codebase along with the Dockerfile created was pushed onto a GitHub repository that was created beforehand. This can be described as the initial point to trigger the cloud deployment on AWS.

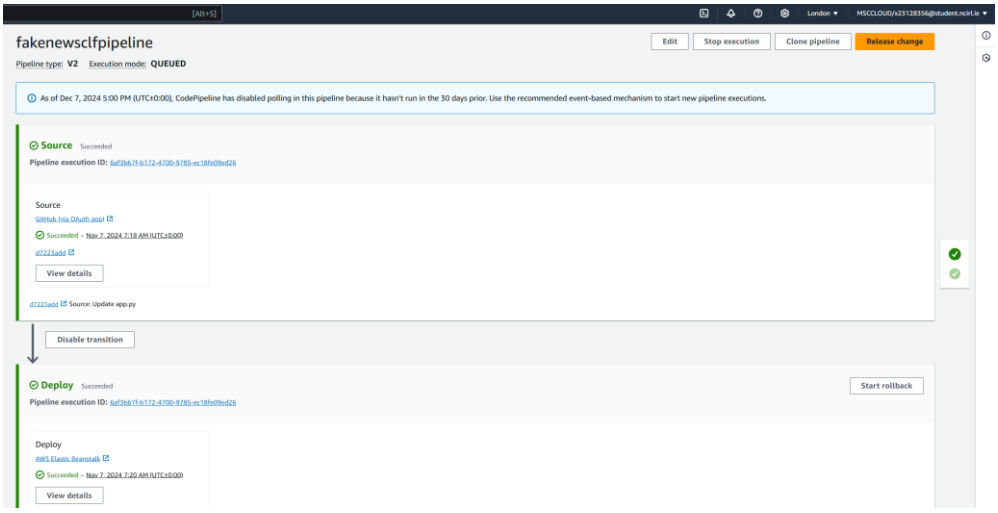

Figure 8: AWS CodePipeline showing the CI/CD piepline

The Deployment process on Amazon Cloud Services was successfully orchestrated by setting up a Continuous Integration / Continuous Deployment (CI/CD) pipeline. The first half of the process was enabled be the setting of a pipeline using the service AWS CodePipeline. The source for the pipeline is set as the GitHub repository where the code and the Dockerfile resides. The usage of CodePipeline is attributed to the fact that the pipeline is always triggered automatically whenever the updated code is pushed onto the GitHub repository. The next stage of automated deployment is executed using AWS Elastic Beanstalk.

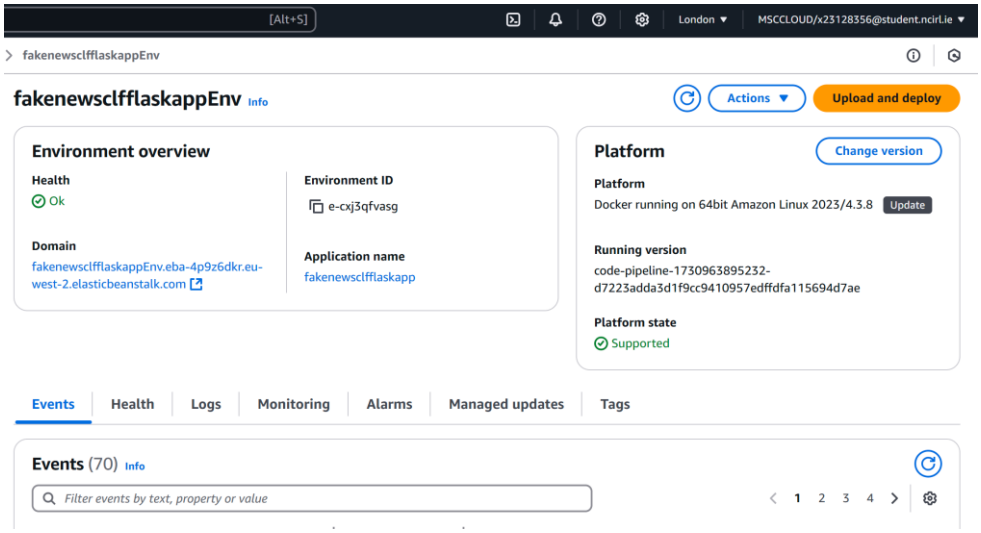

Figure 9: AWS Elastic Beanstalk environment where the application is deployed

An environment is created on the AWS Elastic Beanstalk service to deploy the application and the Dockerfile, this environment is connected with the AWS CodePipeline, from where the codebase and the required files are fetched. The platform on which the application would be run is chosen as Docker which in turn automates the Docker image creation and deployment of the same on the underlying EC2 instances created by the Elastic Beanstalk environment. This pipeline is fully automated where it does not need any human intervention while pushing an application update or for the purpose of upkeeping the whole CI/CD ecosystem.

Figure 10: AWS CloudShell where the terraform (.tf) script has been uploaded

The above AWS Elastic beanstalk environment is created using a Terraform script that was run on AWS CloudShell for the purpose of automating environment creation, therefore eliminating the need of repetitive configuration of environment and the script can also be used for future deployments with the same configurations accurately. The parameters

such as the deployment region, application platform are hardcoded into the script to ensure consistency while using this script for multiple future deployments.

6 Evaluation

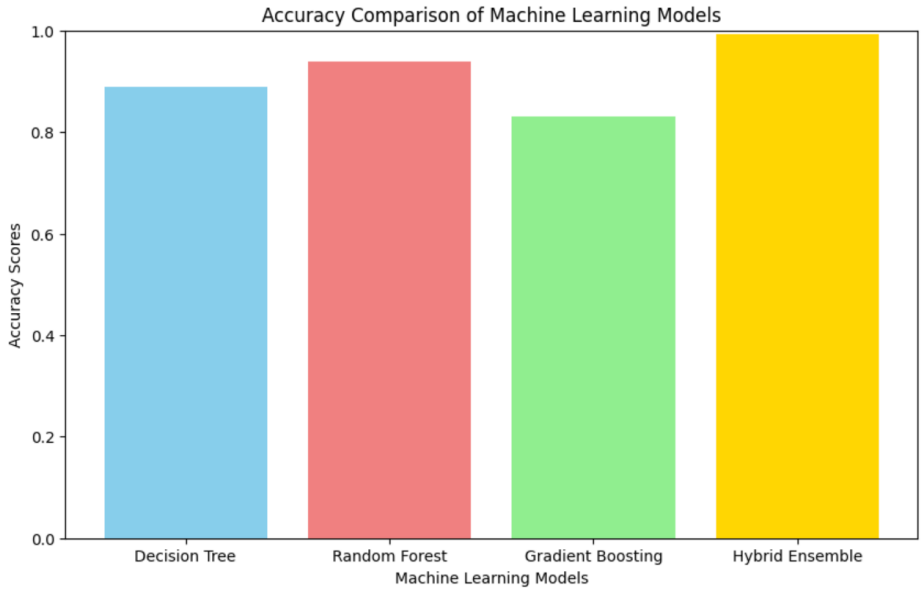

Figure 11: Accuracy scores of the Machine Learning models

The Machine Learning models that were trained with the Fake and Real news datasets were all classification models such as Decision Tree, Random Forest, Gradient Boosting Classifiers. The accuracies that were obtained by these models were respectable with all of them performing well enough to have an accuracy score of more than 80% each. The final model that as utilized was the Hybrid Ensemble Model which is a Stacking model, where all the other classifier models are combined together and their cumulative performance has yielded an accuracy of 99.35%. This model was chosen as the base for the development of the web application based on the Flask framework.

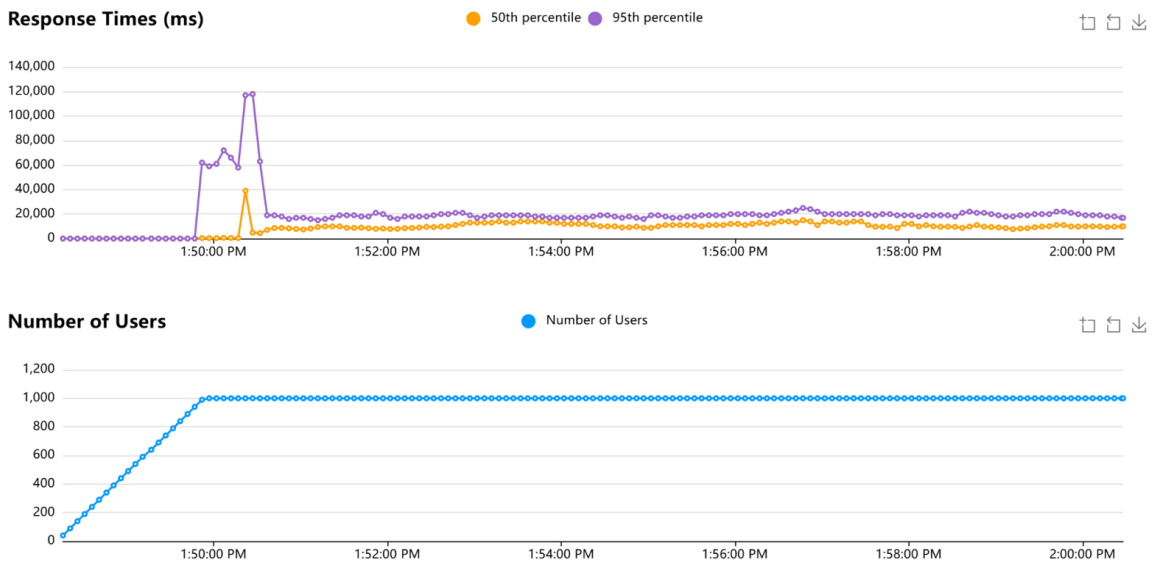

Figure 12: Locust Load testing results of Azure application

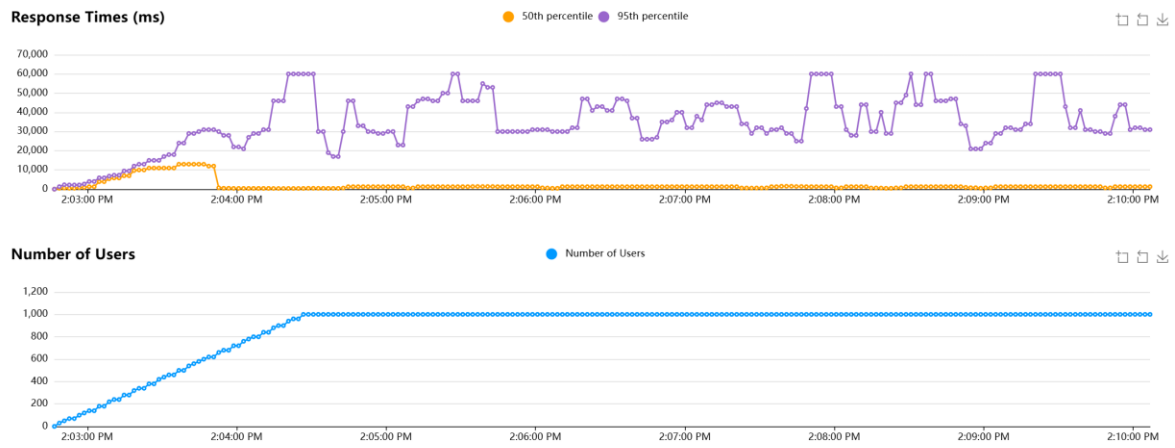

Figure 13: Locust load testing results of AWS application

A Locust Script was created in order to generate load on the deployed applications URLs to simulate real-time users. The parameters were 1000 users accessing the site at a frequency of 10 users joining in every second. Figures are the outputs of the testing phases on AWS and Azure deployed applications. It is evident that Azure has performed marginally better than AWS as the latency experienced by the majority of users (95th percentile) are higher in AWS, with an average latency period of 40,000 ms, which is the double of Azure which exhibited latency values of around 20,000 ms for the users.

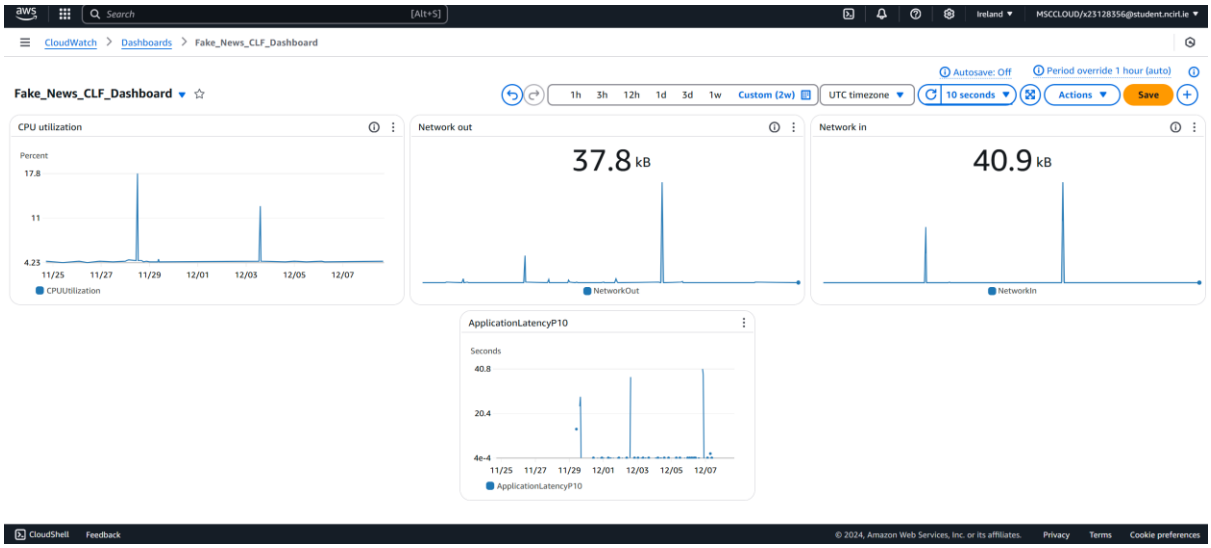

Figure 14: AWS CloudWatch dashboard containing the necessary metrics

A dashboard on AWS was set up using the service AWS CloudWatch for obtaining valuable metrics from the deployed application, the metrics that were chosen as critical for understanding the operational capacities of the application are ApplicationLatencyP10, which records the average latency experienced by the simulated users who are below the 90th percentile. CPUUtilization, which gives valuable insights on how much of the CPU is needed to handle a specific amount of load. NetworkIn, that records the amount of data ingested by the application, indicating the ability to process large amounts of incoming users. finally, NetworkOut, which can be interpreted as the application's ability to send the users the requested data.

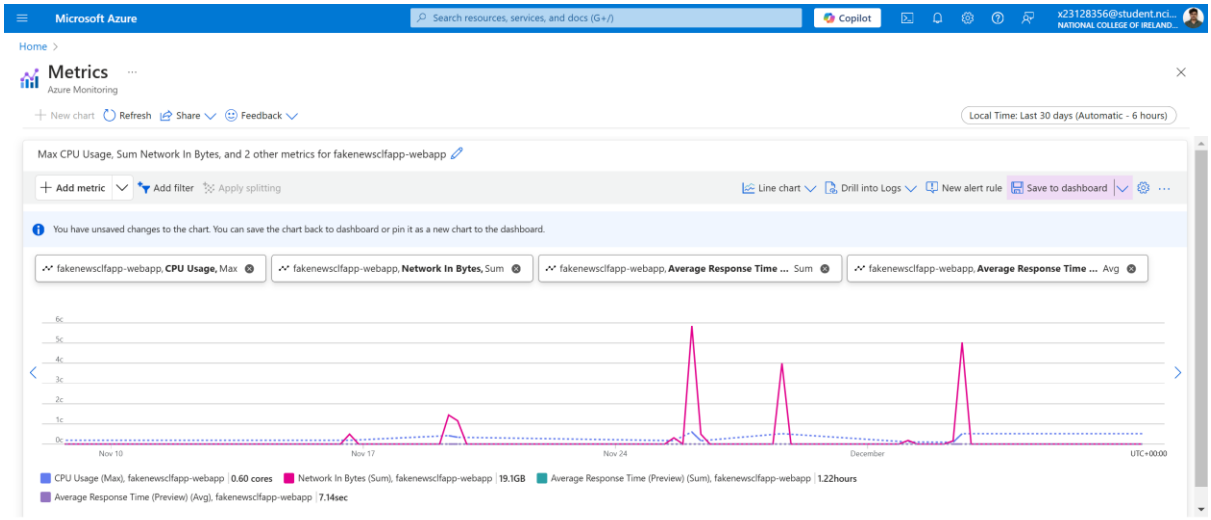

Figure 15: Azure monitor Dashboard containing the necessary metrics

Another Dashboard was created on Microsoft Azure using the service Azure Monitor, where invaluable insights of the operational health of the deployed application on Azure has been recorded. The metrics that were chosen are similar to the ones that are on the AWS CloudWatch dashboard as there is a need to conduct a Performance comparison to determine the better Cloud Service Provider for the deployment of a Dockerized application. Ther metrics were, Average Response Time (Latency), CPU Usage, Network In Bytes and Network Out Bytes.

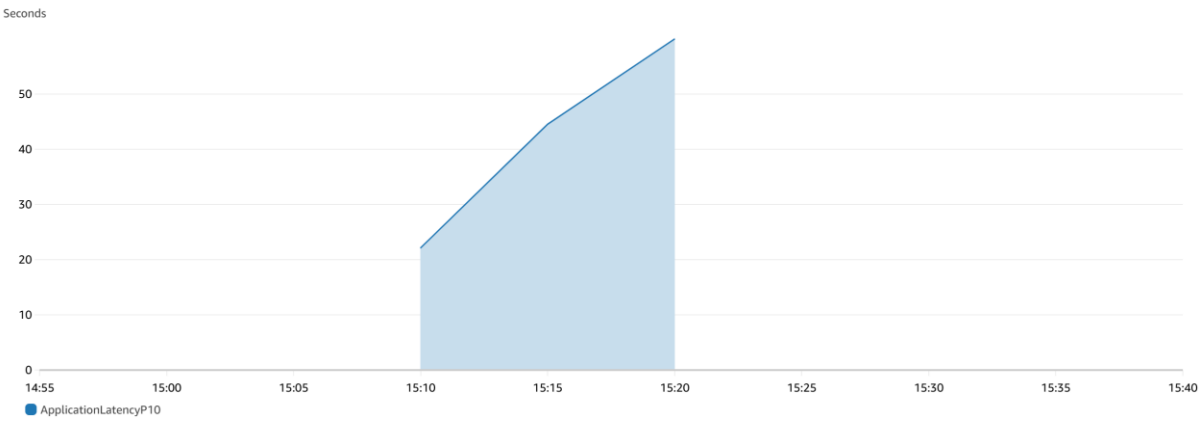

Figure 16: Latency on AWS

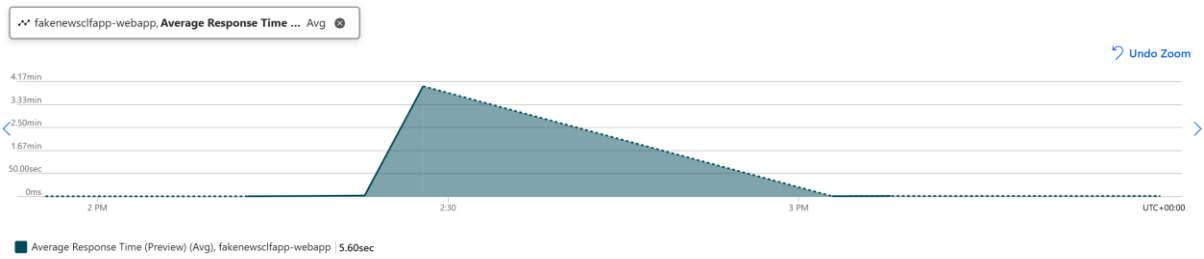

Figure 17: Latency on Azure

The recorded measurements for the latency values while conducting the load testing experiments using the Locust script revealed that there were significant differences between

the two cloud platforms as the latency exhibited by Azure was 6 seconds on average and AWS's latency was around 30 seconds on average. Another observation was that the values of Azure was stable throughout the testing phase while AWS graph was all over the place indicating a lot of fluctuations in latency. This leads to conclude that Azure performs relatively well and consistent even under high loads and its ability to maintain it for long periods proves it can be relied upon for applications that are latency sensitive like gaming apps, which requires low latency and real-time processing.

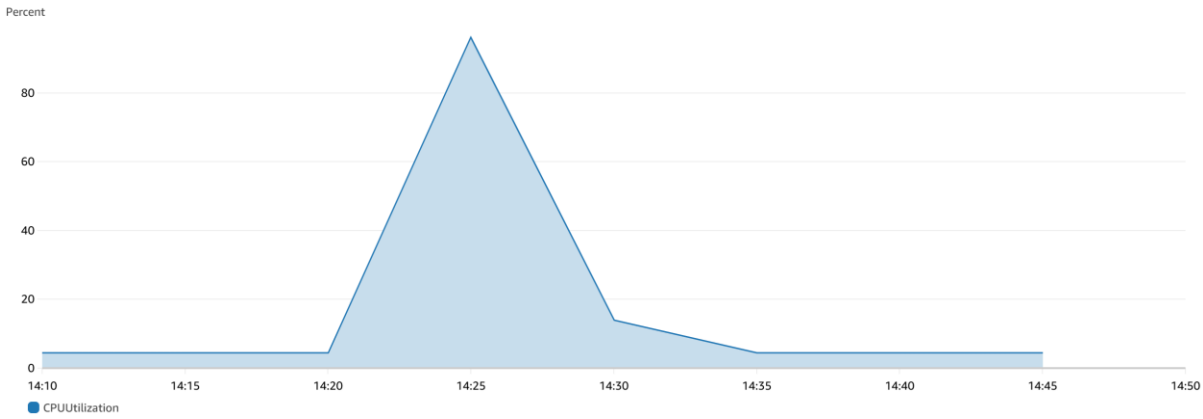

Figure 18: CPU Utilization on AWS

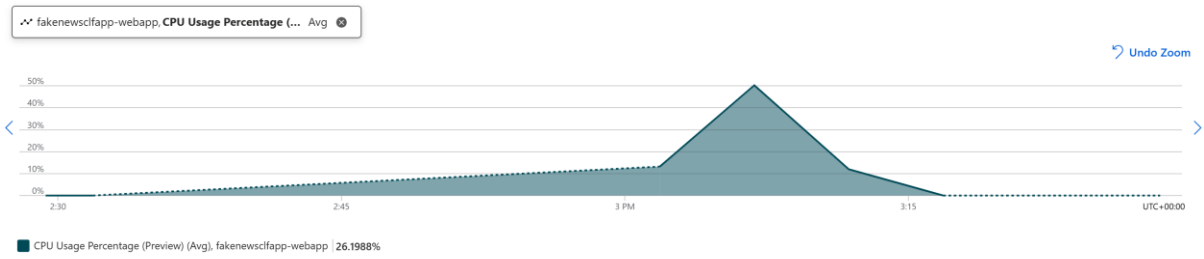

Figure 19: CPU usage on Azure

During the bench marking of the CPU Usage metric, Azure displayed impressive ability of running and processing the load with an average of just 26% of CPU capacity utilized and around 50% maximum utilization of CPU limits. In contrast, AWS had nearly full utilization of its capacity at one point and had an average usage of 35% throughout the testing period. This leads to the conclusion that AWS was pushed to its limits unlike Azure to run the same amount of load. This is led to believe that Azure has superior scalability and workload distribution which has allowed it to efficiently handle the incoming users and requests with ease and not reach its resource thresholds. As far as AWS is concerned, poor optimization of its resources may be a cause of performance bottlenecks present when under heavy loads.

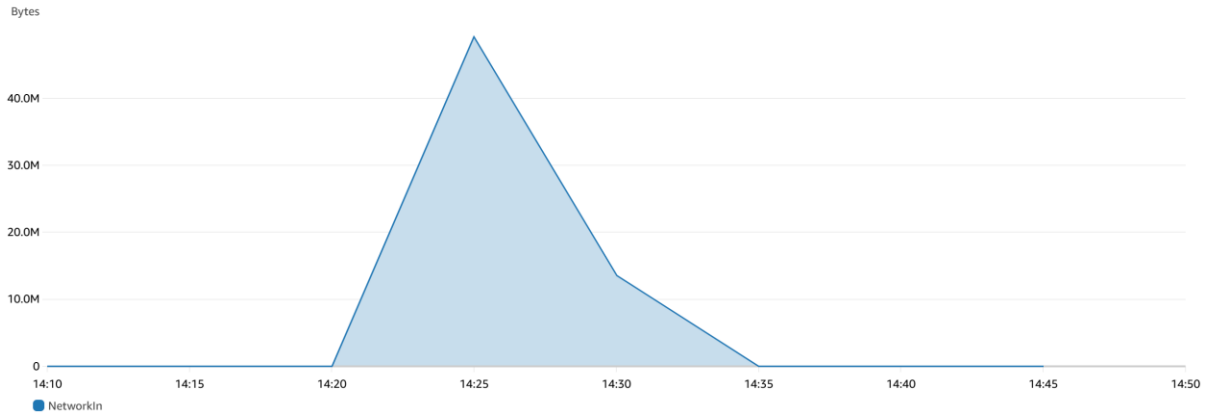

Figure 20: Network In (bytes) in AWS

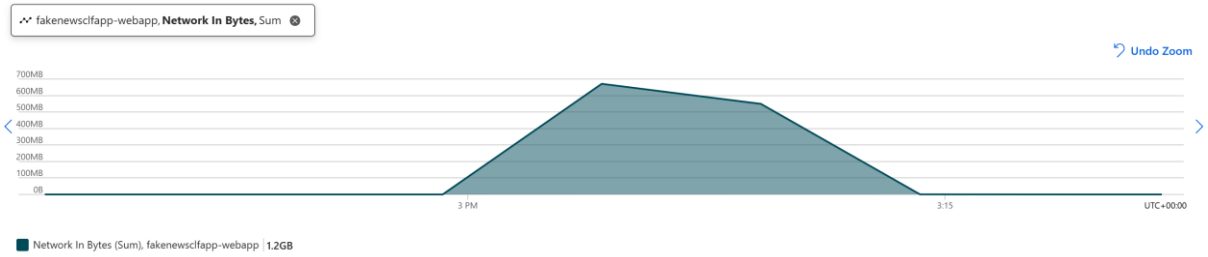

Figure 21: Network In (bytes) in Azure

There was significant different in the Network IN metric that was recorded during the load testing process which read that the Azure application was able to ingest more than 700 million bytes of data while AWS was able to process 50 million bytes of data under identical testing conditions. This revealed that Azure is displaying a significant performance edge over AWS as it is able to process larger volumes of data in a similar timeframe. This could been an indication of Azure's robust anad efficient network infrastructure that is apt for application that need complex and in-depth data processing and ingestion, as well as requiring a considerable amount of network throughput. Azure is able to support and process data-intensive workloads while also displaying enormous reliability unlike AWS.

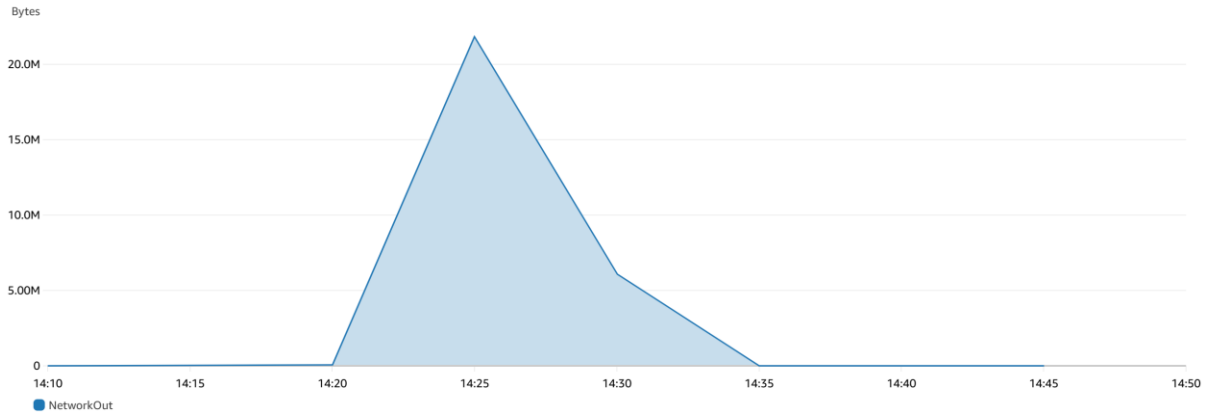

Figure 22: Network Out (bytes) in AWS

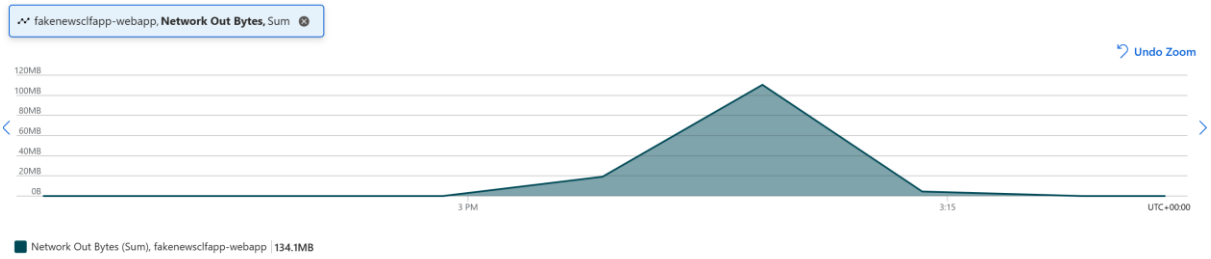

Figure 23: Network Out (bytes) in Azure

By transferring out nearly 120 million Bytes of data, Azure has far succeeded AWS in this aspect which has the capacity to only transmit 22 million Bytes of data in the same time period and under the same load conditions. The primary reason for this difference is the presence of a strong system that is able to handle larger amounts of outbound data more efficiently than AWS. Azure's ability to handle and support data driven applications that require high data transfer capabilities to run has ensured consistent and a service delivery system that reliable all the way.

GitHub

AWS Application

Azure Application

7 Conclusion and Future Work

This research project was fully focused on developing and deploying a Fake news detection Flask application that was powered by Hybrid Ensemble Machine Learning Model, which was then Dockerized to be deployed on Microsoft Azure and Amazon Web Services cloud platforms. Two distinct workflows were chosen for the deployment processes on the cloud service providers. On AWS The whole application codebase along with the Dockerfile was hosted onto the GitHub repository, which was then utilized to trigger the pipeline set up using AWS CodePipeline for a CI/CD deployment, where the latter part was done using AWS Elastic Beanstalk. On Azure, the Docker image created using Docker Desktop for the application using the Dockerfile was stored on Azure Container Registrires' repository and deployed on Azure Container Apps. To conduct the performance comparison, monitoring toos such as AWS CloudWatch and Azure Monitor were chosen for the respective cloud platforms. The metrics that were chosen to be recorded were CPU Usage/Utilization, Network I/O, and ApplicationLatency/Average Response Time. A simulation script was created using Locust for the purpose of Load testing the web applications to collect data and to be aware of the operational health of the applications.

All the findings led to the fact that Microsoft Azure was better at Dockerized application deployments than AWS, as it excelled in several areas such as data throughput and processing the incoming network. Both these platforms possessed a robust and seamless deployment pipeline. The importance of conducting a evaluation on the cloud service providers were demonstrated in detail in this research, which will aid in users choosing the better platform for their requirements based on the performance exhibited by the respective cloud platforms.

Future Work:

The potential avenues for conducting future research which takes this research forward are immense. Some of them are the integration of Serverless deployment services such as AWS Lambda and Azure Functions, where the applications can be run without need of a managing server infrastructure. An event-driven architecture can be set up to ensure only the necessary resources are consumed, reducing the operational costs. Another benefit of a serverless architecture is that the scaling and load balancing aspects of an applications is completely automated.

On that note, dynamic scaling policies can be enabled in the future builds of the application that will be able to scale in accordance to the real-time performance of the application. It can be defined by keeping CPU and Memory usage, latency and request rates in mind. It can be configured to scale horizontally or vertically based on the needs of the application. More advanced parameters and settings can be enabled by integrating cloud services such as AWS Auto Scaling or Azure Autoscale.

8 References

Adiba, F. I. T. K. M. M. M. a. R. M., 2020. *Effect of corpora on classification of fake news using naive Bayes classifier.* [Online]
Available at: https://researchlakejournals.com/index.php/AAIML/article/view/45
Amit M Potdar, N. D. G. S. K. M. M. M., 2020. *Performance Evaluation of Docker Container and Virtual Machine.* [Online]
Available at: https://www.sciencedirect.com/science/article/pii/S1877050920311315
Antonio Cuomo, M. R. a. U. V., 2015. *Performance prediction of cloud applications through benchmarking and simulation.* [Online]
Available at: https://www.inderscienceonline.com/doi/abs/10.1504/IJCSE.2015.071362
Antoniou, A., 2012. *Performance Evaluation of Cloud Infrastructure using Complex Workloads.* [Online]
Available at:
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=13641f64d663c7b5ed4ed71ea4fe14d7b7d372c0
Birunda, S. a. D. R., 2021. *A novel score-based multi-source fake news detection using gradient boosting algorithm.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/9395896
D. Morris, S. V. N. H. R. M., 2017. *Use of Docker for deployment and testing of astronomy software.* [Online]
Available at: https://www.sciencedirect.com/science/article/pii/S2213133717300161
Dr. P. Arul, M. A., 2014. *LOAD TESTING FOR JQUERY BASED E-COMMERCE WEB APPLICATIONS WITH CLOUD PERFORMANCE TESTING TOOLS.* [Online]
Available at: https://d1wqtxts1xzle7.cloudfront.net/68709915
Emiliano Casalicchio, V. P., 2017. *Measuring Docker Performance: What a Mess!!!.* [Online]
Available at: https://dl.acm.org/doi/abs/10.1145/3053600.3053605

Haumahu, J. P. S. a. Y. Y., 2021. *Fake news classification for Indonesian news using Extreme Gradient Boosting (XGBoost).* [Online]
Available at: https://iopscience.iop.org/article/10.1088/1757-899X/1098/5/052081/meta
J. Mukherjee, M. W. a. D. K., 2014. *Performance Testing Web Applications on the Cloud.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/6825689
Jehad, R. a. Y. S., 2020. *Fake news classification using random forest and decision tree.* [Online]
Available at: https://www.iasj.net/iasj/download/250deb09301a4b5a
Moses Openja, F. M. F. K. B. C. a. H. L., 2022. *Studying the Practices of Deploying Machine Learning Projects on Docker.* [Online]
Available at: https://dl.acm.org/doi/abs/10.1145/3530019.3530039
P. Xu, S. S. a. X. C., 2017. *Performance Evaluation of Deep Learning Tools in Docker Containers.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/8113094
Saif, M. N. S. a. A.-A. H., 2021. *Efficient autonomic and elastic resource management techniques in cloud environment: taxonomy and analysis.* [Online]
Available at: https://link.springer.com/article/10.1007/s11276-021-02614-1
Xili Wan, X. G. T. W. G. B. B.-Y. C., 2018. *Application deployment using Microservice and Docker containers: Framework and optimization.* [Online]
Available at: https://www.sciencedirect.com/science/article/pii/S1084804518302273
Yuslee, N. a. A. N., 2021. *Fake news detection using naive bayes.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/9612540