National College of Ireland

Orchestration and CI/CD automation using MLOps for Cloud-native container deployments

MSc Research Project
Cloud Computing

Shraddha Bhosle
Student ID: x21177252

School of Computing
National College of Ireland

Supervisor: Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shraddha Bhosle
Student ID:	x21177252
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Rejwanul Haque
Submission Due Date:	14/12/2023
Project Title:	Orchestration and CI/CD automation using MLOps for Cloud-native container deployments
Word Count:	6064
Page Count:	18

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:	Shraddha Bhosle
Date:	31st January 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):	

Orchestration and CI/CD automation using MLOps for Cloud-native container deployments

Shraddha Bhosle
x21177252

Abstract

With the rise of cloud-native deployments and machine learning, there is a growing demand for efficient orchestration and automation in creating and managing machine leraning models. Machine learning Operations (MLOps) blends machine learning and DevOps, streamlining the process from model creation to deployment in containerized systems. This results in a more effective development lifecycle, increased efficiency, and less manual intervention. The ultimate objective of any industrial machine learning work is to create ML products and deliver into production quickly. However, it is extremely difficult to automate and operationalize ML solutions for large datasets, therefore many ML endeavors fail to meet their goals. This is addressed by the MLOps concept. MLOps is a relatively new concept that inherits its main features from DevOps and applies them to Machine Learning to shorten the time it takes to implement ML model into production. This thesis aimed to analyze this new method and explore several tools for building an MLOps architecture. To validate the functionality of the pipeline, the developed ML application utilized Logistic Regression and Multinomial Naive Bayes models, implementing Natural Language Processing techniques. It is containerized using Docker and delivered via an Elastic Kubernetes Service (EKS) built using Kubeflow pipelines and used GitHub actions to automate workflows directly within GitHub repository. This paper offers an extensive analysis of entire implementation process, starting with model development to cloud deployment. By incorporating MLOps, the project successfully established an automated and efficient pipeline for creating, managing, and deploying machine learning models within containerized systems.

1 Introduction

Software development and deployment techniques have transformed due to the fast advancement of cloud-native container deployments. One of the fundamental areas of advancement in it is the integration of Machine Learning (ML) into cloud-native. This research explores the field of orchestration and continuous integration/continuous delivery (CI/CD) automation, utilizing the newly emerging field of MLOps (Machine Learning Operations) to tackle the difficulties related to cloud-native containerized environments. Because of the good flexibility and scalability of cloud infrastructure, DevOps can bridge the gap between development and operations teams by merging them into one team. Automation facilitates CICD in DevOps. Integrating newly developed code by developers regularly is known as continuous integration, or CI. The capacity to provide changes of

any kind such as new features, configuration adjustments, bug repairs, and experiments safely, swiftly, and sustainably into production or users' hands is known as continuous delivery, or CD. Poloskei and Istvan (2022)
Unlike traditional software development, ML models rely largely on data. Data is essential for training ML models and evaluating their performance to discover areas for development. Data scientists play an important role in solving business challenges by studying data and building models. Building a CI/CD pipeline for ML is more difficult, but the advantages are the same as in typical software projects, where code changes can be immediately deployed and feedback is immediate. ML models must be deployed, transferred, and retrained, which can take time, especially when working with huge datasets. With incorporation of the DevOps concepts into ML development, organizations can improve the overall quality of ML systems and also it can reduce release problems, which will ultimately benefit user experience Cepuc et al. (2020).
In the IT business, cloud computing has become one of the most popular concepts. Its success belongs to its on-demand services rather than the deployment of a full infrastructure, which would require extra expenses for things like recruiting employees, buying and maintaining equipment, and so on. In this research, AWS, a public cloud option is used because of its extensive range of cloud services. Software applications usually include a small amount of machine learning, however these systems have lagged behind the CI/CD trend. In the CI/CD pipelines, machine learning systems bring new challenges and complexity. MLOps, as the name implies, derives its fundamental concepts from DevOps. MLOps is the general term for extending the DevOps technique to incorporate machine learning system characteristics. Makinen et al. (2021)
MLOps incorporates additional elements including continuous training and monitoring, as well as tracking and versioning of the tests conducted to develop a model. These distinctions between standard software and ML models contribute to MLOps' unique qualities. Utilizing MLOps allows businesses to change their models rapidly and simply while maintaining flexible model management. The advantages of MLOps can help create models with greater accuracy and faster time to market by cutting down on the amount of time needed to generate the model.di Laure and Sessione (2021)
Sections have been created and organized further. Section 2 shows a overview of the prior research conducted by researchers in this field. Section 3 concentrates on the methods and technique used to complete the study. Section 4 include the design details. Section 5 shows the implementation of overall project. Whereas, Section 6 highlights the evaluation and discussion of research. Section 7 concludes the research paper and outlines the expectations for the future.

1.1 Research Question

How do orchestration and CI/CD automation streamline deployment using MLOps and contribute to improving ML model fine-tuning for enhanced flexibility in multi-cloud environments?

1.2 Research Objective

To answer the main research question, this paper outlines particular research goals that will enhance the knowledge and use of orchestration, CI/CD automation, and MLOps in the context of cloud-native container deployments and it has the below objectives.

- Create and design an effective workflow for ML operations, concentrating on model training, deployment, and monitoring,
- Implementing automation in the software delivery process aims to reduce human error and maintain consistency and rapid deployment of new features upon completion of development,
- Automation and orchestration enable the use of cloud-native containerization. With the use of this technology, organizations can switch between cloud providers or implement a hybrid approach without experiencing significant disruptions, since it can resolve issues with OS dependencies and vendor lock-in,
- This can help with the current technique's cross-platform compatibility problems. MLOps methods, such as fine-tuning and deploying containerized models to the cloud can be used to manage bigger datasets and more demanding workloads efficiently,
- For partially automated testing environments, when code is integrated into the main branch after reviewers have accepted the alterations, clients have to invest time and effort in trial and error. Through the use of trained MLOps models, it will automatically fix the problem with the intended outcome.

2 Related Work

2.1 MLOps: Principles and Practices

The purpose of Mäkinen, Skogström, Laaksonen and Mikkonen (2021) research, "Who needs MLOps: What data scientists seek to accomplish and how can MLOps help?" was to investigate the usefulness of MLOps within the community of data science. For the purpose of the study, 331 experts from 63 different countries were questioned to know more about their goals and how MLOps may support them. The challenges and work put into the paper are summed up as: Understanding the goals and challenges faced by data scientists working in the field of ML. Acknowledging MLOps' importance in ML model development and application. Analyzing MLOps' potential to improve the accuracy and efficiency of ML models.

Zhou et al. (2020) work "Towards MLOps: A Case Study of ML Pipeline Platform" provides a case study that explores the construction and deployment of ML pipeline platforms. The study's major subjects include the MLOps concept and its importance in integration of ML models in production. The study emphasizes the need of continuing training for increasing the accuracy and efficiency of ML models and provides insight into the challenges faced during the establishment of a ML pipeline platform. The obstacles encountered in constructing a platform for a ML pipeline are highlighted in the paper. Some of these challenges include data manipulation, model training, model deployment, and continuing training. Selecting the best tool for each work in the ML pipeline.

The article "ML Operations (MLOps): Overview, Definition, and Architecture" by Kreuzberger et al. (2023) tackles the difficulties of automating and operationalizing ML (ML) products in production settings and offers a thorough introduction to the MLOps

concept. In order to offer an in-depth knowledge of the essential concepts, components, roles, architecture, and processes, the authors executed mixed-method research that includes expert interviews, a study of the literature, and an evaluation of tools. Additionally, author defined MLOps and list the problems that remain unresolved in the area. The principles, elements, roles, architecture, and processes required for putting MLOps into reality were covered by the writers.

Through careful examination of the amount of existing literature by Ratilainen (2023) in "Adopting ML Pipeline in Existing Environment", presents a full review of the condition of the MLOps discipline today in Adopting ML Pipeline in Existing Environment. Gaining knowledge of the most recent developments, new trends, and important issues facing the MLOps sector is the goal. They give a general summary of the Bank of Finland and the difficulties the company has faced in its current MLOps setting. A simplified end-to-end ML pipeline is presented, demonstrating the use of the selected technologies. The pipeline functions as a useful example of their functioning and integration.
All the above papers helped in finding answers about the actual concept of MLOPS.

2.2 CI/CD in Cloud-native Environments

The paper entitled by Sinde et al. (2022) "Continuous integration and deployment automation in AWS cloud infrastructure" describes the implementation of CI and CD in the context of AWS. The authors discuss the advantages of utilizing CI/CD, such as relieving developers from manual duties, decreasing mistakes, and promoting regular testing. The article also examines AWS capabilities such as AWS CodePipeline and AWS CodeBuild that may be used to implement CI/CD. Overall, the study highlights the significance of CI/CD as a best practice and an essential component of a DevOps project.

Implementing CI/CD in the context of ML Operations (MLOps) is covered in the article "On Continuous Integration / Continuous Delivery for Automated Deployment of ML Models using MLOps" by Garg et al. (2021). The authors outline the advantages of CI/CD in MLOps, including the ability to automate ML model development, testing, and deployment. The article also covers a number of technologies, including Jenkins and GitLab, that may be used to integrate CI/CD in MLOps. In order to guarantee that ML models are created, implemented, and maintained efficiently and offer value to enterprises and end users, the authors highlight the need of utilizing CI/CD in MLOps.

The subject of Cepuc et al. (2020) paper, "Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services using Jenkins, Ansible, and Kubernetes," is how to implement a continuous integration and deployment pipeline for containerized applications using such tools. A wide range of containerized applications can benefit from the CI/CD pipeline solution since it is designed to be more adaptable and scalable to meet their demands. Setting up a CI/CD for containerized apps on AWS may be challenging overall owing to the complexity of the technologies and the need to ensure compatibility and security. These concerns are addressed in the report, along with recommendations for the benefits and best practices for implementing such a pipeline.
Above papers helped in knowing practical implementation of CI/CD in cloudnative env.

2.3 Containerization in ML Deployment

The paper Poloskei and Istvan (2022) from 2022, "MLOps approach in the cloud-native data pipeline design," looks at the potential applications of MLOps techniques and concepts in the development of cloud-native data pipelines. The goal of this study is to talk about the challenges that come with using data models because of personnel and technology shortfalls. According to various measurements, the GPU usage during the ML task pipeline executions is sometimes high, which may cause performance bottlenecks when training many models at simultaneously, even when the models created varied greatly in the number of layers and parameters. By referring this, thorough and planned strategy has been follwed in project to ensure that the best strategies for feature engineering and hyperparameter tuning are employed to obtain the highest degree of accuracy.

The research "Designing an open-source cloud-native MLOps pipeline," proposed by Makinen et al. (2021), examined the requirements for a cutting-edge ML pipeline that provides automation and dependability at the majority of the ML process' phases. In order to communicate with most ML projects and teams wishing to extend and automate their ML process, an open-source, cloud-native MLOps pipeline was developed. As per paper, In the future, a custom operator will need to be constructed in order to manage retraining lifecycles instead of just a simple webhook triggering system. It is necessary to make the pipeline lighter. Currently, a powerful computer is needed to execute the pipeline locally. If the pipeline were lighter, creating and experimenting on local devices would be easier.

The Openja et al. (2022) work "Studying the Practices of Deploying ML Projects on Docker" was included in the Proceedings of the 26th International Conference. In order to understand the evolving trends in this field, the study examines the usage of Docker for deploying ML (ML) applications. This research looks at what kinds of ML projects utilize Docker, why and how they use it, and what kind of Docker images are produced. The study discovered that Docker is used for the deployment of six types of ML-based projects: ML Applications, MLOps/AIOps, Toolkits. This article offers insightful information on the new methods for deploying ML software projects using containers. It also emphasizes the advantages of utilizing Docker for ML projects, including resource management, scalability.
These articles were helpful for realizing role of Docker in deploying ML projects for resource management and scalability.

2.4 Challenges and Solutions in MLOps Deployment

22nd International Symposium on Symbolic and Numerical Algorithms for Scientific Computing (SYNASC) in 2020 included a paper titled "Sustainable MLOps: Trends and Challenges" by Tamburri (2020). The necessity for reproducibility, and fairness in ML models is one of the issues, along with other concepts, that are covered in the article on sustainable MLOps. It also emphasizes how crucial it is to take into account how MLOps could impact the environment, particularly in light of the energy requirements of the inference and training processes. The study suggests a number of approaches to deal with these issues, such as the application of energy-efficient hardware, model compression, and explainable AI. The study offers insightful information on the situation of MLOps today and the issues that must be resolved to guarantee ethical and sustainable AI/ML systems.

The paper by Testi et al. (2022) "MLOps: A Taxonomy and a Methodology" offers a framework for putting ML Operations (MLOps) into practice. After a thorough examination of the body of scientific literature, the writers pinpoint the main ideas and subjects of MLOps research. The significance of MLOps in the context of AI and data science is also covered. This involves automating the development, testing, and deployment of ML models in addition to keeping an eye on and sustaining these models' performance in real-world settings.The goal of future work is to apply our suggested technique to use cases like biological imaging and finance. The study tries to describe a high-level strategy for handling MLOps initiatives.
These papers hepled to explore the difficulties and solutions surrounding the implementation of MLOps.

2.5 Case Studies and Real-world Applications of MLOps

The utilization of ML Operations (MLOps) and Artificial Intelligence Operations (AIOps) frameworks for the orchestration of ML/AI models by Kumar (2022) is covered in the study titled "Orchestration of ML/AI models using MLOps/AIOps frameworks". In order to increase team collaboration, scalability, and security of ML/AI systems, the article highlights the necessity of automating and simplifying the process as well as the usage of a common architecture and development techniques.The study also emphasizes the significance of standardization and best practices in MLOps/AIOps. This paper advocates for the adoption of advanced solutions such as TFX, Kubeflow Pipelines, and Hybrid Cloud Build in the context of AIOps/MLOps architecture". Furthermore, the study underlines the need of Kubeflow, a Kubernetes framework, for developing and deploying portable ML workloads.

According to Kohler and Anders (2022) paper from 2022, "Evaluation of MLOps Tools for Kubernetes: A Rudimentary Comparison Between Open Source Kubeflow, Pachyderm, and Polyaxon," three open-source MLOps solutions for Kubernetes Kubaflow, Pachyderm, and Polyaxon are compared. The article's objective is to evaluate the tools' features and capabilities and provide details on how useful they are in different scenarios. According to the study, Pachyderm, Polyaxon, and Kubeflow are all capable MLOps solutions for Kubernetes, each with particular benefits and drawbacks. The importance of selecting the right tool for the job is emphasized by the author, taking into account factors like data volume, model complexity, and level of automation needed.

The article by Krishna and Gawre (2023) "MLOps for Enhancing the Accuracy of ML Models using DevOps, Continuous Integration, and Continuous Deployment" only covers the incremental steps of precision processes; however, the pipeline could be enhanced with feature engineering and feed forwarding techniques to become an enterprise-ready, fully automated, end-to-end MLOps pipeline. In order to free up time for data scientists and ML engineers to conduct creative thinking and research while continuously improving their ML models, this project intends to save customers time. The primary goal of this work is to demonstrate how hyperparameters can be dynamically changed to obtain higher accuracy without requiring human intervention. How MLOps have been applied in different industries helped in providing valuable insight into what works and does not works.

3 Methodology

3.1 Overview of the Research Approach

The Phishing Link Detection ML software starts an effective procedure to identify phishing links in this organized flow. The procedure begins with gathering data from the large Phishtank Link dataset which is publicly available, which is then carefully pre-processed and features are extracted using natural language processing (NLP) techniques like tokenization and stemming, improving the dataset for further ML tasks. Based on performance measures, models such as logistic regression and multinomial Naive Bayes are chosen and fine-tuned to yield the best possible outcomes. The deployment process is streamlined by integration with a CI/CD pipeline, which facilitates shifting to a cloud environment. For consistency, Docker containerization is used in the deployment process, and GitHub Actions automated the deployment and continuous integration stages. Assuring effective monitoring and scaling capabilities, the managed flow also includes Kubernetes orchestration and Kubeflow pipelines. The diagram below provides a complete flow chart describing project's whole path.

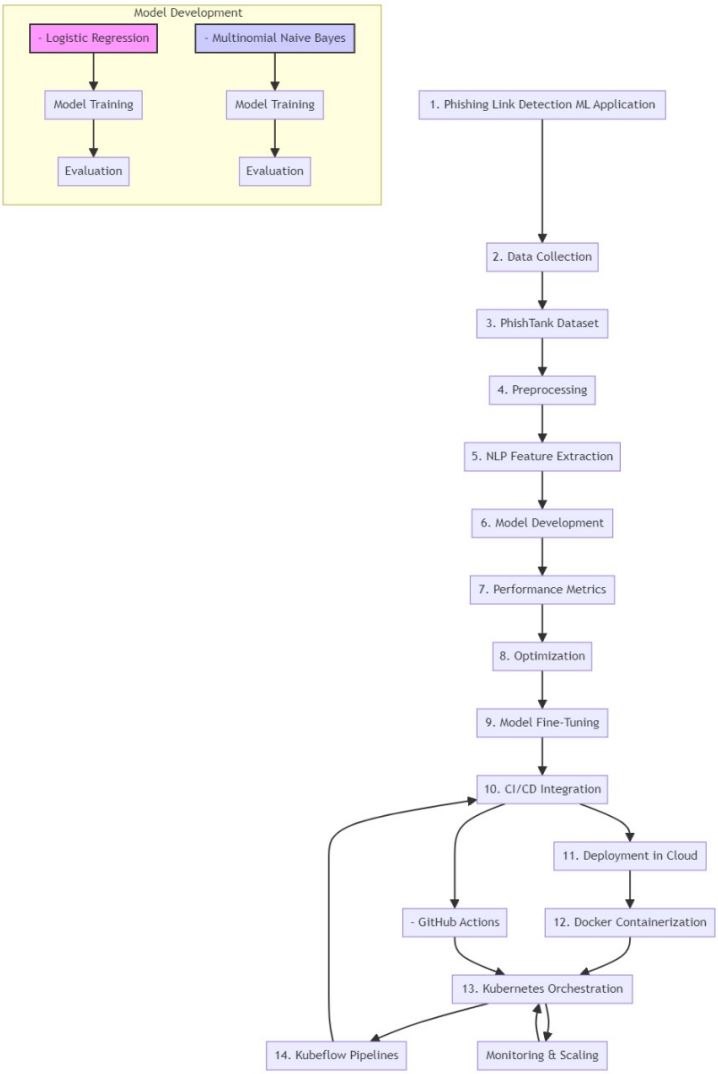

Figure 1: End to end Flow Chart

3.2 ML Model Deployment

Data collection is an important phase in ML since it involves gathering the data that is needed to train and evaluate a model. Gathering data from PhishTank, a broad database of affirmed phishing URLs, was the most important phase in fostering the ML model for recognizing phishing joins. The database used here is public database which contains 549346 URLS. The model's training and testing were both grounded in this dataset. Utilizing Normal Language Handling (NLP) strategies was a significant first move toward quite a while preprocessing stage. To set up the URLs for ML calculations, this method standardized the language, tokenized it, and separated key highlights from it. Selecting a model was the next step. The two models chosen were multinomial Naive Bayes and logistic regression since they were both effective in handling text data. Logistic regression is used to separate phishing from non-phishing joins. This process is known for its simplicity and usefulness in handling two requests at once. The Multinomial Naive Bayes estimation well-known for its proficiency with text data was used to request the features retrieved from the URLs. Using the handled dataset, the two models were trained completely.

3.3 Model Training and Evaluation

It was difficult to train the algorithms to detect phishing links, involving feature selection and hyperparameter modifications for optimal performance. Crucial estimates from the URL data were extracted, such as URL length, secure protocol utilization, and suspicious token existence, to help in feature selection. These characteristics were chosen because they can be helpful in spotting fake links and are frequently observed in phishing attempts. To address the models' correctness, hyperparameter filtering was completed. For the logistic regression model, two limitations were aligned: the regularization strength and the solver type. A smooth adjustment was made to the MNB model to supervise characteristics that were not present in the training sets. Grid search and cross-validation were two of the strategies used to coordinate an array of basics to determine the most desired attributes for each model's limits.
F1 score, a consonant mean of review and accuracy, was particularly helpful in evaluating the balance between the two metrics. These evaluation measurements confirmed that the models would be reliable and robust in the real world.

3.4 Containerization of the ML Application

One advantage of containerization is the ability to split an application's dependencies and create self-managed, remotely manageable packages. Containers make development cycles even more agile. In cloud computing, Docker is the most widely used container. The phishing link recognition ML application was containerized using a well-known platform that bundles apps into containers to ensure consistency across various PC configurations. The tool that was used for this project was Docker.

3.5 Cloud Infrastructure Setup

The ML application was launched using cloud engineering-based on Amazon Web Services (AWS), with EKS acting as the orchestrator. The initial task, to take care of it was setting up the EKS cluster. This occurrence's setup satisfied the ML application's

needs for memory and processing power. At that moment, EKS was built up to control and grow the Kubernetes clusters. By offering a supervised environment, EKS makes Kubernetes deployment and operation on AWS easier. Then built up Virtual Private Clouds (VPCs) as a component of this setup to make sure the Kubernetes clusters had their own dedicated network space.
Moreover, AWS CloudFormation was used to automatically provision the cloud assets. Using CloudFormation layouts, computing events, networking limits, and EKS configurations were all fully described. This automation not only accelerated the development cycle by adhering to established protocols for cloud-local conditions and MLOps techniques, but it also guaranteed consistency and durability in the cloud foundation's deployment.

3.6 CI/CD Pipeline Integration

The implementation of a robust CI/CD workflow for ML model deployment using GitHub Actions has been conducted effortlessly within the repository. The entire procedure is orchestrated by a dedicated workflow file located in the github/workflows directory. This automated process is triggered on every push to the main branch. It is made up of different jobs that cover important tasks such as code retrieval, environment configuration, build tag and push the Docker image, connecting to kubeflow pipeline. Conveying the redesigned application to the AWS EKS cluster, were started by GitHub Activities with each code commit. Security is ensured by the use of secrets to handle private information. This extensive automation ensures that any changes to the ML model are automatically published to production, optimizing the development and deployment lifecycle using the capabilities built into GitHub Actions. Scaling of application instances is done by defining minimum and maximum replicas and threshold values.

3.7 Deployment and Orchestration with Kubeflow

Kubeflow, a Kubernetes ML system, made it more straightforward to convey the ML application into AWS Elastic Container Service (ECS) John et al. (2021). Kubeflow was picked up because it makes running ML work processes in a Kubernetes climate simpler inside and out: deployment, versatility, and management. Preprocessing data, training models, assessing them, and at long last conveying them were totally depicted by these pipelines. Kubeflow empowered the orchestration of different work processes. By integrating Kubeflow into the AWS Elastic Container Service (ECS) climate, enable the option to send and oversee ML models productively in a cloud-local climate, demonstrating the pragmatic use of MLOps standards.

4 Design Specification

With this architecture, developers can create and test code locally inside of containers. When they are done, the code is published to a Git repository hosted on websites such as GitHub. The automated creation of a new Docker image-based on the modified code is then initiated via GitHub Actions. Elastic Container Registry (ECR) is where the final Docker image is kept. The deployment procedure is managed via a Kubeflow pipeline that is coordinated via the Kubeflow dashboard. Through a LoadBalancer Ingress, the application is made accessible to the outside world, making the deployed FastAPI application easier to access. The final deployment happens on the cloud, and the entire

deployment is coordinated on an EKS cluster version 1.25, with AWS Cluster and CLI managing the primary instance. The below diagram shows overall architecture of the workflow.

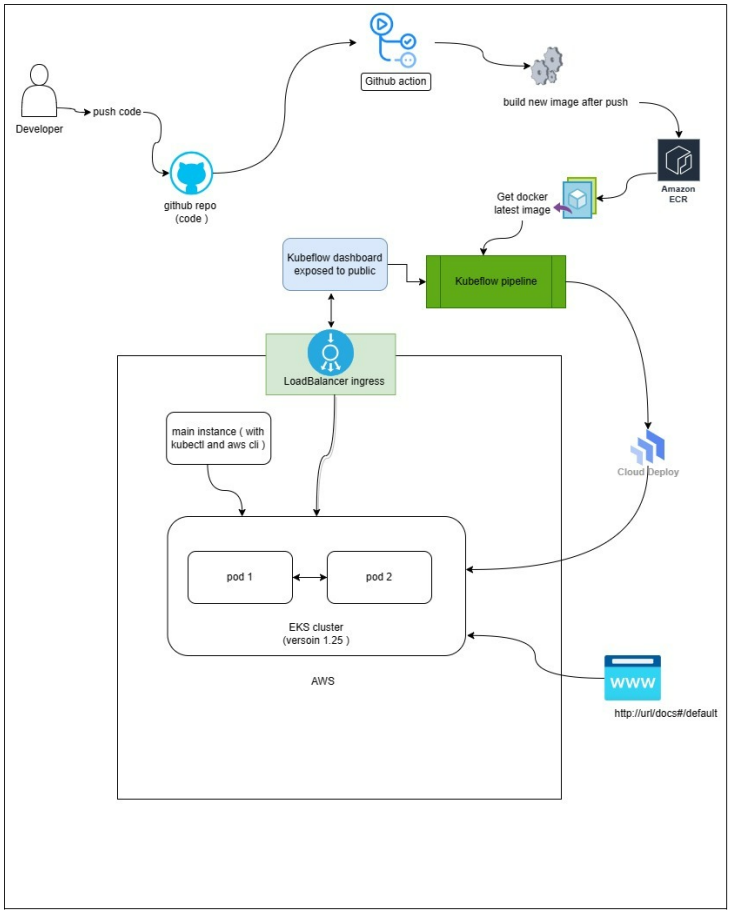

Figure 2: Architecture Diagram

4.1 Security measures

To improves security and access control in cloud environments, implemented OIDC (OpenID Connect) with limited role access in AWS. OIDC integration centralizes authentication and authorization processes. Users only get permissions that are required for their responsibilities, due to limited role access. AWS certificate manager service has been used to issue a certificate for the domain to access Kubeflow securely on the domain.

5 Implementation

5.1 Tools and AWS services Overview

Below tools have been used for orchestartion and automation purpose.
1. Kubeflow- Open source ML platform for Kubernetes, which facilitates end-to-end ML

scenarios.
2. Github Actions - Used for automating developer workflows and CICD is one of the workflows which can be automated by this.
3. Docker - Facilitates lightweight, portable application packaging through containerization.
4. Kubernetes - In a cluster environment, Kubernetes orchestrates the deployment, scaling, and administration of containerized applications.
AWS services used for implementation:

- Route 53 - It is used to host domain and access Kubeflow on a domain,
- AWS Certificate Manager - Secure Sockets Layer/Transport Layer Security certificates are provisioned, managed, and deployed by this for usage with AWS services and internal linked resources,
- Elastic Load Balancer - Over several targets, including IP addresses, EC2 instances, and containers, AWS Elastic Load Balancer effectively divides incoming application traffic,
- Ec2 instance - Within the AWS Cloud, scalable computing capability is offered via Elastic Compute Cloud (EC2),
- Elastic Kubernetes Service - Containerized applications utilizing Kubernetes can be deployed, maintained, and scaled more easily using EKS, an entirely controlled Kubernetes service,
- Elastic container registry- Developers can easily store, manage, and deploy docker container images with the help of ECR,
- Virtual Private Cloud- It enables user to launch AWS resources, such as EC2 instances, into a network which is virtual and defined,
- AWS CloudFormation - Customers can provision and manage AWS resources in a scalable, predictable, and safe manner by using a template file.

5.2 Setting up the ML Learning Environment

Configuring system with the essential tools and libraries for data analysis, model training, and evaluation is part of setting up a ML environment. This configuration is now essential to ensure the phishing link detection version is developed and implemented smoothly. Key software components included Python, a popular programming language for system understanding, a Jupyter notebook, environments for interactive computational work. Integrating necessary libraries to analyze information and learn about objects has become the ultimate benchmark. These packages include Pandas for data processing, NumPy for numerical calculations. Seaborn, scikit-learn, and NLTK are installed for data visualization, ML, and natural language processing, respectively. NetworkX is added for network analysis capabilities, while Selenium is installed for web automation. To handle object serialization and deserialization, Python's pickle module is used in project.

5.3 Data Collection and Pre-processing

The first step in the procedure is to compile data from the reliable PhishTank public dataset, which is a vast collection of phishing URLs. First, the data is preprocessed to eliminate extraneous metadata. Next comes the standardization of textual material, which guarantees uniform URL forms and gets rid of anomalies. Through URL tokenization, every URL is broken down into keywords for comprehensive examination. Stemming and stopword removal cut down on noise and preserve coherence. Feature extraction, which stands for distinct components for the training and assessment of the ML algorithm, is the last step. By using an organized method, the data is ready to be used in the construction of machine learning models that identify and distinguish phishing URLs.

5.4 Model Training and Optimization

During the primary research phase, the preprocessed dataset was utilized to instruct the system in learning two models: multinomial Naive Bayes and logistic regression. Multinomial Naive Bayes is well suited to textual content kind because of its strength with discrete facts, and logistic regression is a linear variation that performs effectively with binary class problems. The hyperparameter tuning method allowed to improve their overall performance. Logistic regression has highly-tuned parameters such as the algorithm type and regularization power. The alpha parameter, which regulates the data smoothing, changed into the middle of interest while discussing multinomial Naive Bayes. The maximum hit setting was found by carefully experimenting with different parameter combinations while utilizing techniques like grid search and go-validation. Through this rigorous optimization procedure, the models had been suitably tweaked to achieve optimum accuracy and portability in phishing link identification.

5.5 Containerization with Docker

The first thing to do was to write a Dockerfile, which is a script that contains instructions for creating the Docker image. The Dockerfile defines the operating system, libraries, and dependencies that might be needed, along with the process for deploying the application code into the container. The purpose of this Dockerfile is to build an image for a Python application with dependencies for utilizing Kubernetes and Kubeflow Pipelines. The application is a web application-based on FastAPI. Following the selection of the base image which is often a lightweight operating system such as ubuntu in the Dockerfile, the critical requirements were set up, the application code was copied, and the command to send out the application was defined. After preparing the Dockerfile, the docker build command was used to create a Docker image. This approach greatly reduced the complexity of usage deployment, scaling, and administration.

5.6 Deployment on AWS EKS and Kubeflow Integration

An important step forward in the deployment process was the deployment of a ML application that was Dockerized and hosted on AWS Elastic Kubernetes Service (EKS). By utilizing AWS EKS, a managed Kubernetes provider, for scaling and reliability, an effective infrastructure for ML workloads was guaranteed. In order to provide a robust and expandable environment for container deployment, an EKS cluster customized to the

application's unique requirements was first configured as part of the deployment process. Then, Kubeflow an open-source platform for Kubernetes ML learning was smoothly included into the EKS setup. End-to-end ML learning procedures, training of model, assessment, and deployment, were made easier by this connection. A strong basis for deploying and administering the ML learning application was made possible by the combination of Kubeflow and AWS Elastic Container Service. The final outcome was a seamlessly automated and organized ML learning system that used the benefits of Kubeflow with AWS EKS to improve deployment efficiency.

5.7 CI/CD Pipeline Setup and Automation

This project used GitHub Actions, a workflow automation tool, to build a strong CI/CD pipeline. The pipeline was set up to automatically run tests on the code whenever changes were made in order to maintain code quality. This automated testing made sure that changes didn't result in regressions or mistakes. The pipeline automatically produced a Docker image and uploaded it to a container registry when the tests were successful. Following this, the updated Docker image was promptly deployed to the AWS EKS environment, eliminating the need for manual intervention. This led to the creation of a continuous delivery and integration pipeline, which was crucial in accelerating the software development lifecycle while upholding high standards of quality. Overall, the project's software development benefited greatly from this CI/CD pipeline, which demonstrated its importance by guaranteeing the most recent software version. Using GitHub Actions, the CI/CD pipeline was built up to automate testing, Docker image creation, and AWS EKS deployment.

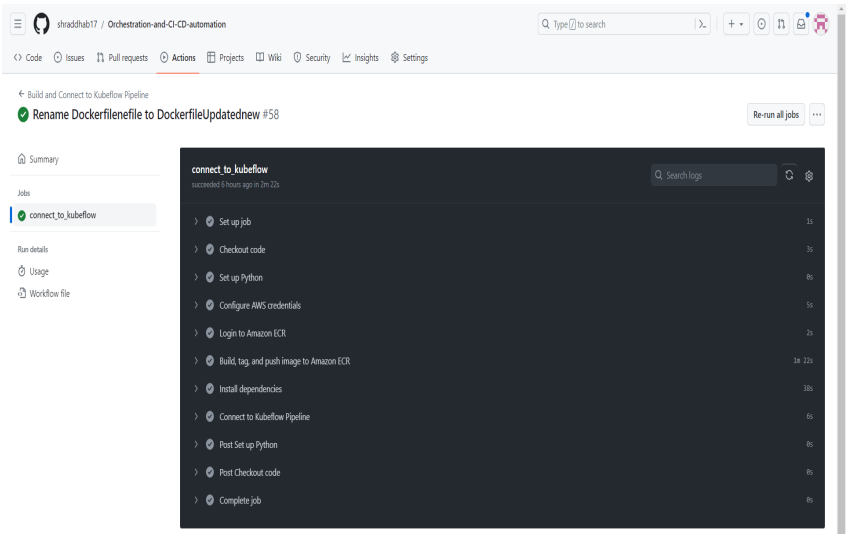

Figure 3: Steps in Github Actions workflow

6 Evaluation

6.1 Model Performance and Analysis

The logistic regression and Multinomial Naive Bayes models were extensively evaluated to determine how well they performed using a dataset that was divided into training and testing sets. The logistic regression version's training and testing accuracy were 97.84% and 96.41%, respectively. A F1-rating of 93% was the result of its 91% accuracy in figuring out phishing hyperlinks and its 97% consider non-phishing hyperlinks finished an excellent 99% precision, 96% bear in mind, and 98% F1-rating. Especially, the metrics for correctly identifying phishing hyperlinks highlight the version's robustness. Training

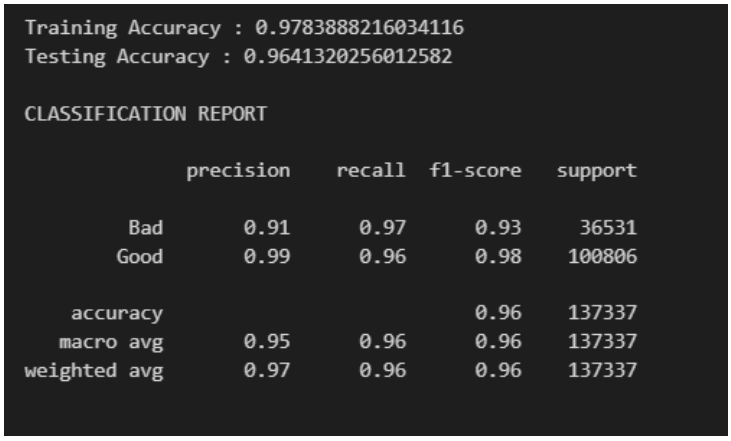

Figure 4: Training and testing accuracy of Logistic Regression

accuracy for the multinomial Naive Bayes model changed into 97.41% even as trying out accuracy was 95.73%. With a F1-score of 92%, it became capable of retrieving 93% of 'bad' hyperlinks with absolute precision. A precision and keep in mind of 97% were done for 'suitable' linkages, ensuing in a F1-score of 97%. The model's potential to properly categories hyperlinks is demonstrated by those effects. When it came to

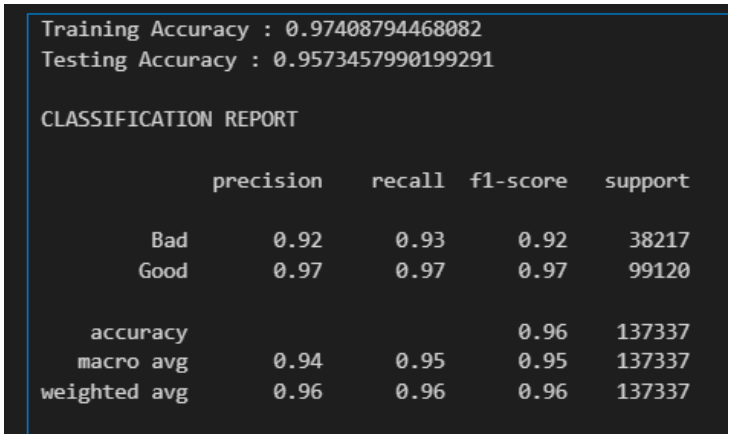

Figure 5: Training and testing accuracy of Multinomial Naive Bayes

identifying phishing URLs, each model performed well, with high accuracy, precision,

consider, and F1-ratings. The results demonstrate that the algorithms can distinguish between phishing and non-phishing URLs, implying that they can be employed in real-life scenarios to improve security.

6.2 Efficiency of the Containerization Process

The Docker containerization solution dramatically improved the ML application's deployment performance. Docker containers, which encapsulate a program and all of its dependencies, significantly reduced deployment times, allowing for consistent and quick releases. Because the container only required its necessary dependencies, resource use became streamlined by reducing the overhead often associated with digital devices. Containerization, which ensured consistency across many settings, also eliminated the "it really works on my system" problem. Whether on-premises development ML or in the cloud, this consistency was crucial to preserving the application's reliability and overall performance.

6.3 Effectiveness of Cloud Deployment

The application's implementation on AWS Elastic Kubernetes service (EKS) became top-notch in terms of scalability, load management, and uptime. The application may want to scale seamlessly with AWS Elastic Container Service (ECS), handling irregular workloads quickly. The application should be able to control remarkable masses without seeing a significant decrease in performance due to its scalability. Great uptime data, indicating exceptional dependability and continuous application availability, has been made possible by AWS EKS's dependable infrastructure. The application's performance became optimal because of the stable and robust environment provided by the cloud infrastructure, which dynamically adapted to the software's operating demands.

6.4 Impact of CI/CD Pipeline Automation

After GitHub actions were used to enforce the CI/CD pipeline, there was a significant improvement in deployment frequency, error reduction, and workflow efficiency.

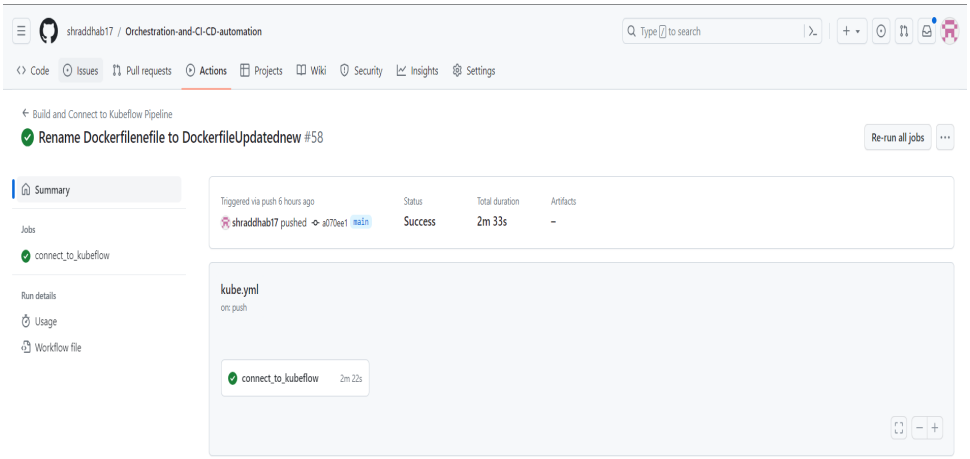

Figure 6: Github Actions workflow pipeline

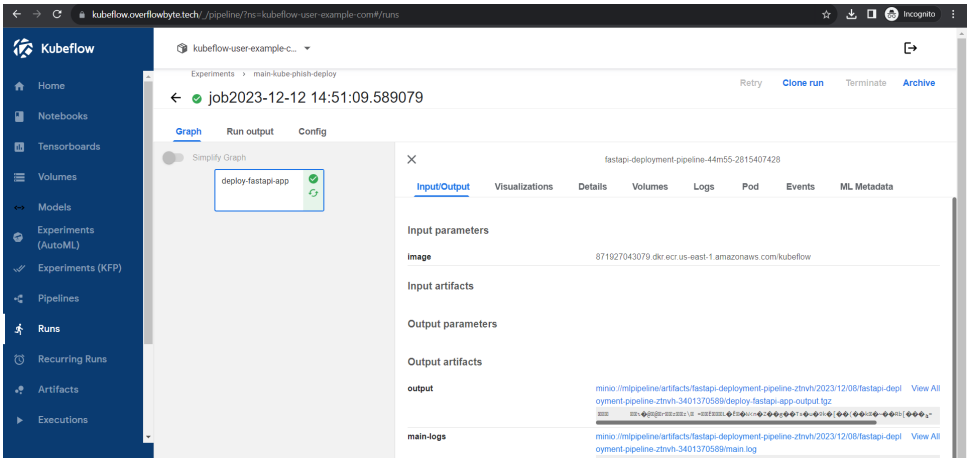

Figure 7: Kubeflow Pipeline

Automatic procedures that enabled frequent and dependable deployments allowed the software to develop continuously and iteratively. The time it required to bring new features and upgrades to market can be reduced by the capacity to release them more often. Because of the CI/CD method's automation, manual mistakes have greatly decreased, leading to a more stable and reliable deployment cycle. Because developers could spend more time on innovation and less time on the operational aspects of deployment, the software program improvement lifecycle as a whole can be more efficient.

6.5 Discussion

The system has demonstrated high scalability, high reliability, and robust performance for cloud-native field deployments with the integration of MLOps with CI/CD. The ML models performed a great job at phishing hyperlink detection, and the containerization technique made it possible to continuously deploy in a variety of environments. The cloud architecture made possible by AWS EKS offered excellent scalability and uptime. However, there is always opportunity for improvement in areas such as real-time processing of data and the use of more advanced ML algorithms to increase accuracy. Future study will focus on more advanced security measures and AI-driven analytics for predictive remodeling.

7 Conclusion and Future Work

The purpose of this project was to build Machine Learning Operations in order to overcome the issues related to machine learning and cloud-native deployments. The project successfully developed an automated and efficient pipeline for creating, managing, and deploying machine learning models within containerized systems by using MLOps. The research delved into the relatively unclear concept of MLOps and clarified its significance for professionals and researchers. The project contributed to the actual implementation of MLOps in the context of cloud-native architectures by using rigorous research methodologies, including a detailed literature survey. The containerization of the application using Docker and its deployment through an Elastic Kubernetes Service with Kubeflow

pipelines showcased the project's commitment to delivering efficient, scalable, and cloud-native solutions. The project essentially and successfully managed integrating machine learning with DevOps concepts, which has an important effect on the efficient deployment of machine learning models in modern cloud-native workflows. Future research has to focus upon including additional and significant data sets in order to improve the versatility of machine learning models for phishing detection. To improve field deployment and management in cloud-local environments, it would also be possible to investigate the integration of additional technology and tools. Deployment methods and models can be improved continuously based on user input and real-world performance by using continuous feedback systems.

References

Cepuc, A., Botez, R., Craciun, O., Ivanciu, I.-A. and Dobrota, V. (2020). Implementation of a continuous integration and deployment pipeline for containerized applications in amazon web services using jenkins, ansible and kubernetes, *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, IEEE, pp. 1–6.

di Laure and Sessione, I. (2021). *Mlops-standardizing the machine learning workflow*, PhD thesis, University of Bologna Bologna, Italy.

Garg, S., Pundir, P., Rathee, G., Gupta, P., Garg, S. and Ahlawat, S. (2021). On continuous integration/continuous delivery for automated deployment of machine learning models using mlops, *2021 IEEE fourth international conference on artificial intelligence and knowledge engineering (AIKE)*, IEEE, pp. 25–28.

John, M. M., Olsson, H. H. and Bosch, J. (2021). Towards mlops: A framework and maturity model, *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, pp. 1–8.

Kohler and Anders (2022). Evaluation of mlops tools for kubernetes: A rudimentary comparison between open source kubeflow, pachyderm and polyaxon.

Kreuzberger, D., Kühl, N. and Hirschl, S. (2023). Machine learning operations (mlops): Overview, definition, and architecture, *IEEE Access* .

Krishna, M. Y. S. and Gawre, S. K. (2023). Mlops for enhancing the accuracy of machine learning models using devops, continuous integration, and continuous deployment, *Research Reports on Computer Science* pp. 97–103.

Kumar, Deepak, S. A. (2022). Orchestration of ml/ai models using mlops/aiops frameworks, *AIOps Frameworks (June 19, 2022)* .

Mäkinen, S., Skogström, H., Laaksonen, E. and Mikkonen, T. (2021). Who needs mlops: What data scientists seek to accomplish and how can mlops help?, *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, IEEE, pp. 109–112.

Makinen, S. et al. (2021). Designing an open-source cloud-native mlops pipeline.

Openja, M., Majidi, F., Khomh, F., Chembakottu, B. and Li, H. (2022). Studying the practices of deploying machine learning projects on docker, *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, pp. 190–200.

Poloskei and Istvan (2022). Mlops approach in the cloud-native data pipeline design, *Acta Technica Jaurinensis* **15**(1): 1–6.

Ratilainen, K.-M. (2023). Adopting machine learning pipeline in existing environment.

Sinde, S. P., Thakkalapally, B., Ramidi, M. and Veeramalla, S. (2022). Continuous integration and deployment automation in aws cloud infrastructure, *International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN* pp. 2321–9653.

Tamburri, D. A. (2020). Sustainable mlops: Trends and challenges, *2020 22nd international symposium on symbolic and numeric algorithms for scientific computing (SYNASC)*, IEEE, pp. 17–23.

Testi, M., Ballabio, M., Frontoni, E., Iannello, G., Moccia, S., Soda, P. and Vessio, G. (2022). Mlops: A taxonomy and a methodology, *IEEE Access* **10**: 63606–63618.

Zhou, Y., Yu, Y. and Ding, B. (2020). Towards mlops: A case study of ml pipeline platform, *2020 International conference on artificial intelligence and computer engineering (ICAICE)*, IEEE, pp. 494–500.