National College of Ireland

Optimizing Movie Recommendations with MLOps in AWS

MSc Research Project
Cloud Computing

Priyal Patil
Student ID: 22209573

School of Computing
National College of Ireland

Supervisor: Jitendra Kumar Sharma

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Priyal Patil
Student ID:	22209573
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Jitendra Kumar Sharma
Submission Due Date:	12/08/2024
Project Title:	Optimizing Movie Recommendations with MLOps in AWS
Word Count:	6284
Page Count:	23

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:	
Date:	15th September 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):	

Optimizing Movie Recommendations with MLOps in AWS

Priyal Patil
22209573

Abstract

In the age of streaming services and online content, there has been a significant increase in the need for personalized movie suggestions. This has led to the creation of complex recommendation systems capable of managing large amounts of data and quickly adapting to evolving user preferences. This study explores enhancing movie recommendation systems by incorporating Machine Learning Operations (MLOps) in Amazon Web Services (AWS) environments. The research focuses on the increasing demand for efficient and easily expandable recommendation systems due to the growth of digital content. Conventional machine learning methods frequently encounter difficulties when being implemented in real-world situations, especially when dealing with vast amounts of data and adjusting to user behavior. The Project highlights the significance of operational elements when it comes to implementing and upkeeping machine learning models in a production environment. Utilizing AWS's strong cloud infrastructure, the project seeks to develop a streamlined MLOps pipeline to enable ongoing integration, delivery, and monitoring of machine learning models. This method guarantees that the recommendation system will be able to grow, stay dependable, and respond quickly, leading to enhanced user happiness. The project utilizes different AWS services such as SageMaker for constructing models, Lambda for executing serverless functions, and S3 for storing data, in order to establish a smooth, automated pipeline which results emphasize how MLOps can improve the performance and scalability of movie recommendation systems, offering valuable insights for future applications in this area.

1 Introduction

In this today's rapidly evolving domain of digital content, movie recommendation systems has become essential for any streaming platforms. As the volume of content available to users continues to growing exponentially, hence the need for effective and personalized recommendations is critical and important than ever. These systems are not only essential for enhancing user engagement but also for increasing overall user satisfaction by delivering content which aligns closely with individual preferences.

This thesis mainly focuses on the integration of Machine Learning Operations (MLOps) with the cloud-based environments, especifically use of poppular cloud platform Amazon Web Services (AWS), for optimizing the performance of these recommendation systems. Not like the traditional machine learning approaches that primarily concentrate on algorithmic enhancements, this research highlights its significance of the operational aspects

which is involved in deploying and maintaining machine learning models in production environments. After shifting the focus towards MLOps, the research aims for bridging the gap between model development and real-world application, ensuring that the deployment, monitoring, and continuous improvement of these models are as efficient and scalable as possible. Karamitsos et al. (2020)

The goal of this research is to create a robust pipeline of MLOps which not only streamlines the deployment of machine learning models but also improves their scalability and efficiency. with the help of this pipeline, the recommendation system can continually learn and adapt new techniques , improving its accuracy and responsiveness in delivering recommendations. This holistic approach ensures that the system remains relevant and effective in the face of changing user behaviors and preferences.

However, by utilizing AWS's popular services, the research highlights how cloud infrastructure can be used in different way to support the continuous integration and delivery (CI/CD) of machine learning models. This approach not only minimizes the time as well as effort required to managing the lifecycle of these models but also provides a scalable and reliable framework that can handle the demands of modern streaming platforms.

This thesis aims for contributing to the knowledge on MLOps by providing more advancements which can guide future implementations in the field of movie recommendation systems. The findings from this research will offer significant insights into the operational benefits of MLOps, paving the different way for more efficient and effective machine learning applications.

1.1 Background

The idea behind the development of an advanced recommendation systems came from the tremendous growth in digital content and the audience across the globe. As we know the traditional machine learning approach often falls when it comes to deployment in real-world scenario. Few models fail for transitioning from development to production. It happens due to the complexities of handling the large-scale data and ensuring that models were adapted and evolved for responsing to user feedback and behavior. However , the integration of these models into the production environments has significant challenges. It shows a significant gap between model development and operational deployment. This gap is where DevOps has evolved software development stages. It promoted faster development cycles and more reliable releases throughout the automated pipelines, continuous integration, and delivery practices.

1.2 Importance of Research

The integration of MLOps to a cloud computing frameworks have a promising answer to all of the challenges. MLOps extends DevOps principles into a machine learning domain. It aims for streamlining and automation of the machine learning lifecycle in cloud environments. This approach not only has operational bottlenecks but also uses the scalable infrastructure of cloud services like AWS to improve the performance and reliability of machine learning systems. The importance of this research fit in its potential for providing an empirical evidence and methodological advancements in the application of MLOps, especifically altered to the needs of movie recommendation systems. This thesis aims to fill the current research void by offering detailed insights into the operational benefits of MLOps, thereby guiding future implementations and innovations in this field.

1.3 Research Objectives and Question

Guided by the need to enhance operational efficiency and model efficacy, this research is structured around a central question: "How can the integration of MLOps within cloud computing environments, specifically AWS, optimize movie recommendation systems in terms of scalability, accuracy, and user satisfaction?" This question seeks to dissect the multifaceted benefits of MLOps, comparing it against traditional machine learning deployments to quantify its impact on system performance and user experience.

1.4 Report Structure

The remainder of this report is structured as follows: Section 2 is a comprehensive literature review, shows the evolution of recommendation systems and the role of MLOps for addressing existing challenges. Section 3 is the research methodology, detailing the tools, data, and procedures used in the development of the MLOps pipeline. Section 4 contains Design specification where each component is discussed while Section 5 provides detailed implementation steps. Other sections includes the evaluation in the context of the original research questions, and conclude with recommendations for future research and practical applications of MLOps in movie recommendation systems.

2 Literature Survey

The development and optimization of movie recommendation systems have been significantly evolved through advancements in machine learning and the integration of MLOps practices. This literature review aims for exploring the evolution of recommendation systems, with a focus on movie recommendation systems, the role of collaborative filtering, the architecture and significance of MLOps, and also the existing gap in the application of MLOps to movie recommendation systems

2.1 Recommendation Systems

Recommendation systems has been evolved throughout the years as crucial tools in every domains, such as music, books, and movies, driven by the need to handle large volumes of data and provide personalized user experiences. Old recommendation systems were primarily content-based and mainly they were relied on explicit user feedback for generating recommendations. Although, with the use of machine learning, these systems have become more sophisticated, use of this algorithms that can be learn from implicit user interactions and vast datasets. s collaborative filtering and hybrid models technique can be widely adopted which significantly improves the accuracy and relevance of recommendations across different industries.

2.2 Movie Recommendation System

There was need of a specific focus on Movie recommendation system. Different types of the approaches were proposed and tested by each and every researchers for improving the performance of these systems. For instance, PireciSejdiu et al. (2022) the researcher compared the performance of machine learning models such as Naïve Bayes, neural networks, and logistic regression in movie recommendation systems. The result showd the

effectiveness of these algorithms for handling the vast amount of data of movies. Similarly, other studies also explored the integration of deep learning models and hybrid approaches, combining collaborative filtering with content-based techniques to improve recommendation precision PireciSejdiu et al. (2022)

2.3 Collaborative Filtering Approach

Recently,collaborative filtering is the most popular techniques in any recommendation systems, but especially for movies. It can operate on the principle that users who have agreed in the past will continue to agree in the future, it allows the system for making predictions based on the user similarities. This method has been widely used and liked as it has simplicity and effectiveness. However, it was not without drawbacks. One prominant limitation is the cold-start problem, where the system was struggled for making accurate recommendations for new users or any items due to the lack of sufficient datasets. Additionally, collaborative filtering can also suffer from scalability issues such as the dataset grows, leading to increased computational costs and latency in generating recommendations Mu and Wu (2023). Despite these challenges, collaborative filtering remains a cornerstone of modern recommendation systems.

2.4 Content Based Filtering

Content-based filtering evolved as a more advanced method where the system suggests items that share similarities with ones the user has enjoyed in the past, focusing on item characteristics. These attributes consist of meta data such as genre, keywords or author or other brief details. The system creates profile for the user by analyzing past preferences, typically showing vector of item characteristics they have engaged in past Pazzani and Billsus (2007). Building a user profile which is vector of features which represents the user's preferences, combining characteristics of the items. The system calculates similarity of the user profile to other different items in the database. Generally used metrics, such as cosine similarity, Euclidean distance, Pearson correlation, are often utilized Sekar et al. (2023). This technique examines the content or characteristics of items and aligns them with user choices to produce recommendations Lops et al. (2011) Content-based filtering provides suggestions for new users by analyzing characteristics of the initial items they cope up with, solving the cold start issue.

2.5 MLaaS Machine Learning as a Service

The rise of Machine Learning as a Service (MLaaS) has sped up the use of recommendation systems on the cloud. MLaaS platforms like Google Cloud AI, Amazon SageMaker, and Microsoft Azure Machine Learning offer ready-made machine learning models, tools, and APIs for developers to smoothly incorporate into their applications. These services enable businesses to utilize advanced recommendation systems without requiring extensive knowledge in machine learning or managing intricate infrastructure Crankshaw et al. (2017). These cloud-based platforms provide functions such as automatic model training, hyperparameter optimization, and model deployment, which simplify the development process and facilitate quicker iteration and experimentation. With the ability to utilize pre-trained models, automated machine learning (AutoML), and sync with other cloud

services, companies can efficiently iterate on models and release them into production environments, ultimately decreasing time-to-market Crankshaw et al. (2017)

2.6 MlOps

MLOps (Machine Learning Operations) is an hot topic which applies DevOps principles for the machine learning lifecycle. It involves the automation of different steps of machine learning, from model training to deployment and monitoring. It ensures continuous integration and delivery of ML models. The architecture of MLOps typically includes following components for data ingestion, model training, validation, deployment, and monitoring, all integrated into a seamless pipeline Makinen et al. (2021). MLOps practices are important to any organizations aiming to scale their machine learning operations, as it can provide the necessary infrastructure to manage multiple models, handle large datasets, and ensure the reproducibility and reliability of ML deployments Makinen et al. (2021). The need for MLOps becames particularly evident in complex environments where frequent model retraining and deployment are required, such as in dynamic recommendation systems.

2.7 Projects using MLOps

Few projects has successfully implemented MLOps for improving the development and deployment of machine learning models. These projects highlights the benefits of MLOps, such as enhanced collaboration between teams, quicker deployment, and better model performance in production environments. For example, the application of MLOps in cloud-based environments has enabled organizations for automating their end-to-end ML pipeline, like from data preparation to model deployment, significantly reducing the time to market for new models Kumara et al. (2023). But , despite these advantages, challenges such as tool integration, managing model drift, and ensuring data privacy and security remain. however, there is still a gap in the literature regarding the application of MLOps especifically to movie recommendation systems.

2.8 Gap Identified

While searching I got to know that MLOps has been widely used, explored and implemented in different types of machine learning projects, but it is a notable gap in its application to movie recommendation systems. Despite the potential benefits of MLOps, such as improved scalability, reliability, and maintainability of recommendation systems, there is also limited research for integration of these practices into the development and operation of movie recommendation models. This gap highlights the opportunity of a future research to explore how MLOps can be leveraged to enhance the performance and scalability of movie recommendation systems, addressing the challenges identified in current implementations.

Author(s)	Title	Proposed Solution	Limitations	Gaps Identified
Zhang et al. (2019)	Deep Learning Based Recommend System: A Survey and New Perspectives	Advancements in using deep learning models like CNNs and RNNs for movie recommendation systems.	High computational demand and integration difficulties with multiple types of data.	Pointed out the need for streamlined ML operations to improve deployment efficiency.
Amershi, et al., 2019	Software Engineering for Machine Learning: A Case Study	Emphasized the complexity of managing ML data compared to software code, advocating for better data management tools.	Manual processes in ML model deployment leading to slow updates and scaling issues	Identified a critical need for continuous integration and delivery pipelines for ML models.
Karamitsos, et al., 2020	Applying DevOps Practices of Continuous Automation for Machine Learning	Integration of DevOps practices into ML life cycle to support continuous automation and update deployment.	Existing operational bottlenecks in traditional ML deployment.	Advocated for the seamless integration of MLOps to enhance operational efficiency.
Garg, et al., 2021	On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models using MLOps	Advocated for the use of CI/CD practices in ML to facilitate smoother transitions from development to production.	Operational challenges in deploying ML models, like managing dependencies and environment consistency.	Highlighted the necessity of adopting DevOps principles to boost agility and reliability of ML systems.
Kumara, et al., 2023	Architecting MLOps in the Cloud: From Theory to Practice	Discussed the evolution of MLOps as a response to applying DevOps practices to ML, supporting full ML life cycle.	Challenges in tool selection and configuring services to streamline ML workflows.	Calls for a deeper understanding of MLOps to better support data scientists' needs
Mäkinen, et al., 2021	Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help?	Focused on operational challenges faced by data scientists that can be mitigated through MLOps practices.	Did not provide specific solutions for the integration issues noted.	Need for more research on seamless integration of MLOps into existing workflows.

Figure 1: Gap Identified

3 Research Methodology

This research aims to harness the power of MLOps to improve the effectiveness of movie recommendation systems in theAWS cloud environments. By using AWS services, this project aimsto construct a robust MLOps pipeline that facilitates the continuous integration, delivery, and monitoring of machine learning models, thereby addressing scalability, efficiency, and maintenance challenges area also their.

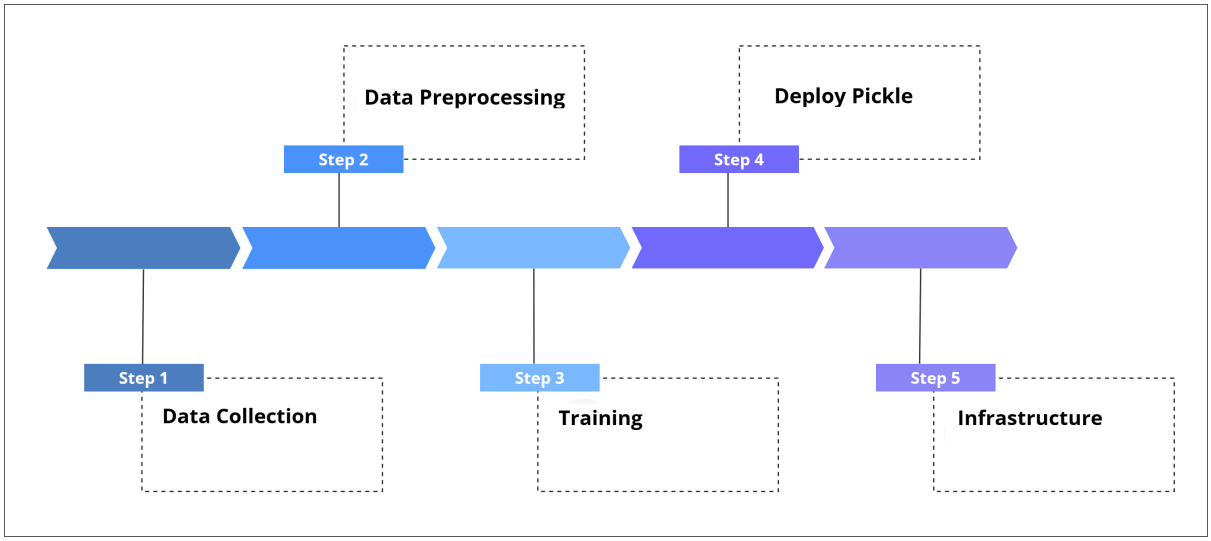

Figure 2: Process Flow

3.1 Process Flow of Methodology

3.1.1 Data Collection and Prepocessing

The data collection process has utilized the Anime movies dataset, which contains information on user ratings from 73,516 users on 12,294 anime. As it is a open dataset with CC0: Public Domain license. AWS Sagemaker notebook is been used for purpose of extracting, transforming, and load the data,. It also ensures it is cleaned and formatted correctly for use in training the recommendation models.

3.1.2 Training

Model was trained initially in the notebook with the help of content-based filtering approach and afterwards converted into a python script for executing it with sagemaker pipelines. Cosine similarity was used to get similarity matrices which could recommend movies having similar values.

3.1.3 Deploy Pickle

Then, The trained model was then converted into a pickle file after executing and dumped into a S3 bucket as a source for the frontend application to load and recommend movies once invoked.

3.1.4 Infrastructure

The proposed system architecture has the integration of various AWS services for supporting the lifecycle of a movie recommendation model. AWS SageMaker was used for model building and training as well while AWS Lambda, AWS Elastic Container Service (ECS), and Amazon Elastic Container Registry handled the whole deployment of frontend and backend . This integrated environment has goal to be sure that all components work seamlessly to supporting continuous training, evaluation, and deployment of the recommendation models and flask application for user to consume.

4 Design Specification

4.1 Architecture Diagram

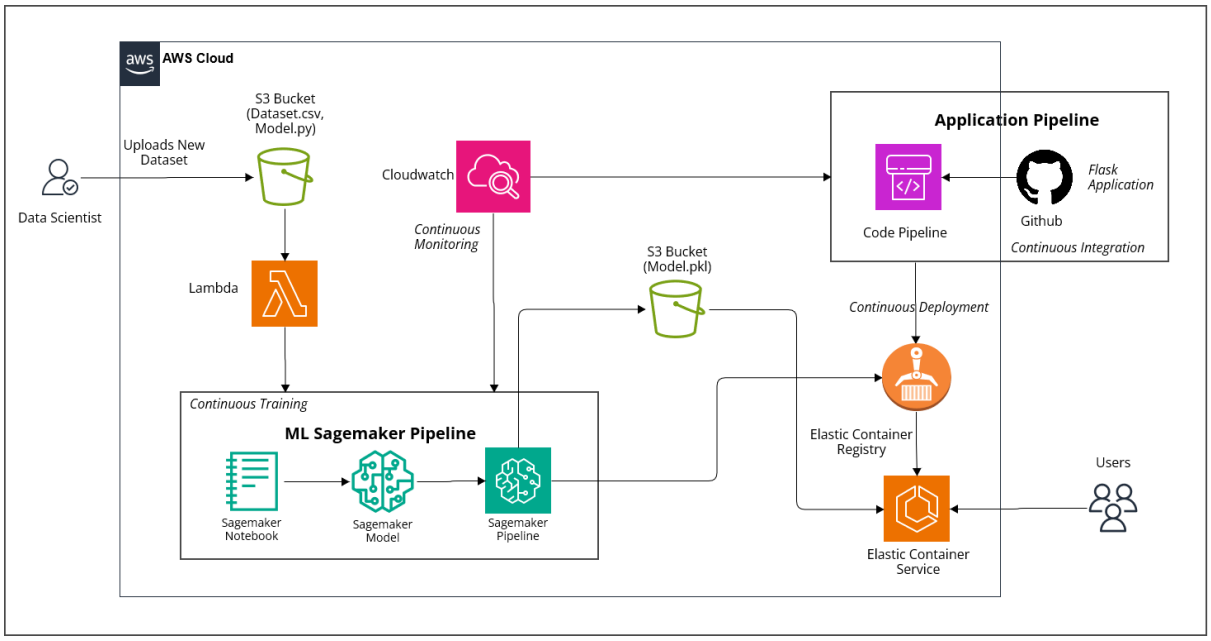

Figure 3: Architecture Diagram of the Project (Original Illustration

The architecture diagram 3 explains complete flow of my MLOps project where each step is independent of human intervention and each pipeline flows without interruption. Each component of this project is explained in detail below with reason why they were used in this project:

1. **Amazon Web Services (AWS) Cloud Platform**: AWS Cloud is leading public cloud provider across the globe. It provides on-demand computing resources and APIs to individuals, companies, and governments, on a pay-as-you-go basis *Amazon Web Services* (2024). It is chosen for its extensive global infrastructure, reliability, scalability, and integration options, making it an ideal environment for deploying machine learning operations (MLOps) and handling complex data workflows.

2. **Identity Access Management (IAM)**: IAM provides fine-grained access control to AWS resources, ensuring that only authorized and authenticated users and services can access your AWS resources. In MLOps projects, IAM is crucial for defining roles and permissions for AWS services and users to interact securely with resources like S3 buckets, SageMaker, Container service (ECS) and Lambda functions.

    - **SageMaker Execution Role**: "AmazonSageMaker-ExecutionRole-20240630T161935" execution role acts on behalf of SageMaker to perform call other AWS services without any issue. I have Provided S3 Full access policy to fetch input data from S3 bucket and push artifacts to S3. Additionally, provided AWS ECR access to build image and train model on training container

- **Lambda Execution Role**: "Lambda-SageMaker-Pipeline-Execution-Role" allows Lambda functions to trigger SageMaker Pipeline. Provided SageMaker access to this role to invoke pipeline in SageMaker once the lambda function is invoked due to PUT API called on S3 bucket (i.e New Dataset is added in S3 bucket).

3. **Sagemaker**: SageMaker is a fully managed machine learning (ML) as a service in AWS *What is Amazon SageMaker? - Amazon SageMaker* (n.d.). SageMaker simplifies the process of developing, training, and deploying machine learning models on a large scale. In just a few steps, we can launch a model into a secure and scalable environment using the SageMaker console. *What is Amazon SageMaker? - Amazon SageMaker* (n.d.)

    - **Jupyter Notebook**: Integrated directly within SageMaker, these notebooks are used for interactive data exploration, visualization, and model development.
    - **Sagemaker Pipeline**: This feature automates the stages of a machine learning process, like data preprocessing, model creation, and implementation, to guarantee consistent and effective procedures.

4. **Simple Storage Solution (S3)**: S3 is a service for storing objects that provides scalability, data availability, security, and performance. It is utilized in this project for the storing and accessing of various data types, like training datasets, model artifacts, and output results. Use of s3 with the mlops is a great idea because it helps in flexibility, automation and reliability as well. Also, S3 acts as a centralized storage of a datasets, model artifacts, logs, and other resources which are necessary throughout the ML lifecycle.

5. **CloudWatch**: CloudWatch keeps track of the AWS resources and the applications that run on AWS. It is utilized for gathering and monitoring metrics, logs, setting alarms, and automatically responding to changes in AWS resources, it is essential for managing and optimizing machine learning models. In this project, Cloudwatch is used to store logs of Lambda function, sagemaker training process etc.cloudwatch is also used for monitoring the performance and health of the sagemaker endpoints where ML models are deployed. CLoudwatch metrics and alarm can be configured in such a way that it can trigger autoscaling actions for AWS resources such as remove or add instances.

6. **EventBridge**: EventBridge is a serverless platform that links application data from your own applications, software as a service (SaaS), and Amazon Web Services (AWS) offerings. Automating workflows and enabling application integration are facilitated by automatically routing data between software applications, a crucial element for initiating processes in MLOps pipelines *Amazon EventBridge Documentation* (n.d.). EventBridge can also triggers specific actions in the MLOps pipelines which are based on real time events such as new data involving in S3 or changes committed in databases. it also facilities the integration of AWS resources such as Lambda, step function and SageMaker with third party SaaS applications like Datadog.

7. **Lambda**: Lambda enables you to execute code without the need to allocate or oversee servers. It is utilized to carry out behind-the-scenes operations in reaction to AWS service events (such as modifications in an S3 bucket), vital for activities needing instant processing such as real-time data processing or request handling.AWS Lambda helps for automatically scaling the volume of incoming events which make it ideal for handling bursts of activity in MLOps tasks. Lambda operates on a pay as you go concept which only charges for computed time during task execution.this is really cost effective.

8. **AWS Elastic Container Service (ECS)**: ECS is a container management service that is highly scalable and high-performance, it supports Docker containers and enables you to run applications on a managed cluster of servers. ECS is utilized for launching containerized apps, such as web apps and backend services.ECS integrates smooth;y with the CI/CD pipelimes and also enables automated deployments of the containerized applications. This is particularly useful in MLOps where models can automatically containerized tested or deployed in scalable environments.

9. **Fargate**: Fargate is a feature for Amazon ECS that enables running containers without the need to handle servers or clusters. It eliminates the necessity of setting up and adjusting the compute, allowing for a more concentrated effort on creating and developing applications. Fargate automtically manages the ongoing computing resources which are required for running in your container for example scale up or scale down on workload demand. also with the help for applications task to run on its own isloated environment as well.

10. **AWS Elastic Container Registry (ECR)**:ECR is a Private Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. In this project, it is used to:

    - **Flask Docker Image**: Manage the Docker image for the Flask application to facilitate simple deployment and version control of my web application.
    - **Sagemaker Training Image**: Store custom Docker images used for model training in SageMaker. It contains all required dependencies installed in that image.

11. **Cloud9**: Cloud9 is an online integrated development environment (IDE) based on the AWS that allows to write, execute, and troubleshoot code in a web browser. It offers a platform for coding, testing, and troubleshooting code while developing and testing applications in AWS. Cloud9 was used in AWS due to its feature of deploying or pushing our code into AWS services seamlessly as AWS keys and access keys are not exposed outside of AWS.cloud9 is allows multiple developers to work along with in realtime on same code base which enables efficient teamwork and code reviews. It is available in AWS by default and allows to interact with other AWS services directly.

12. **CodePipeline**: CodePipeline performs the automated tasks to continuously deploy software updates in the phase setup in the pipeline. In this project, code pipeline is used to create a CI/CD pipeline for Flask application. CodePipeline is used over Jenkins, CircleCI or other CI tool because it has benefit of being part of AWS environment which can seamlessly access other AWS services within the

network (Reducing access-based issues). Codepipeline integrates seamlessly with IAM helping for fine grained access control over anyone who can access and modify it. also, it works with other AWS developer tools. this native integration simplifies the setup and management of entire CI/CD

13. **GitHub**: GitHub is a hosting service for Git repositories that also offers tools for version control and teamwork. Enables multiple developers to collaborate on projects remotely, a critical aspect of team-based software development. I preferred GitHub over CodeCommit as my SCM tool as it is matured and user friendly.as it integrates with a wide range of third party tools such as GitHub actions for CI/CD pipeline, project management tools and also with the communication tools. Githubs Pulls request system is powerful which has code review and collaboration feature

14. **Flask Application Framework**: Flask is a light-weight WSGI python based web application framework. Its purpose is to facilitate a quick and easy start, and it can also grow to handle advanced applications. In this project, Flask is utilized to develop a user-friendly web interface for interacting with the machine learning model.Flask Modular design allows an developer to easily extends the applications with the third party plugins which makes it highly adaptable for different project needs. Its lightweight nature and simplicity makes it easy for integration with popular ML libraries such as Tensorflow or scikit-learn

5 Implementation

The project begins with development of a robust MLOps pipeline which automates the end-to-end lifecycle of machine learning models from data preprocessing and model training to final deployment and monitoring. Using AWS services such as SageMaker, Lambda, and ECS, the pipeline improves the scalability, efficiency of the recommendation system. There are two major steps in implementation: ML Pipeline and CI/CD Pipeline

5.1 ML Pipeline

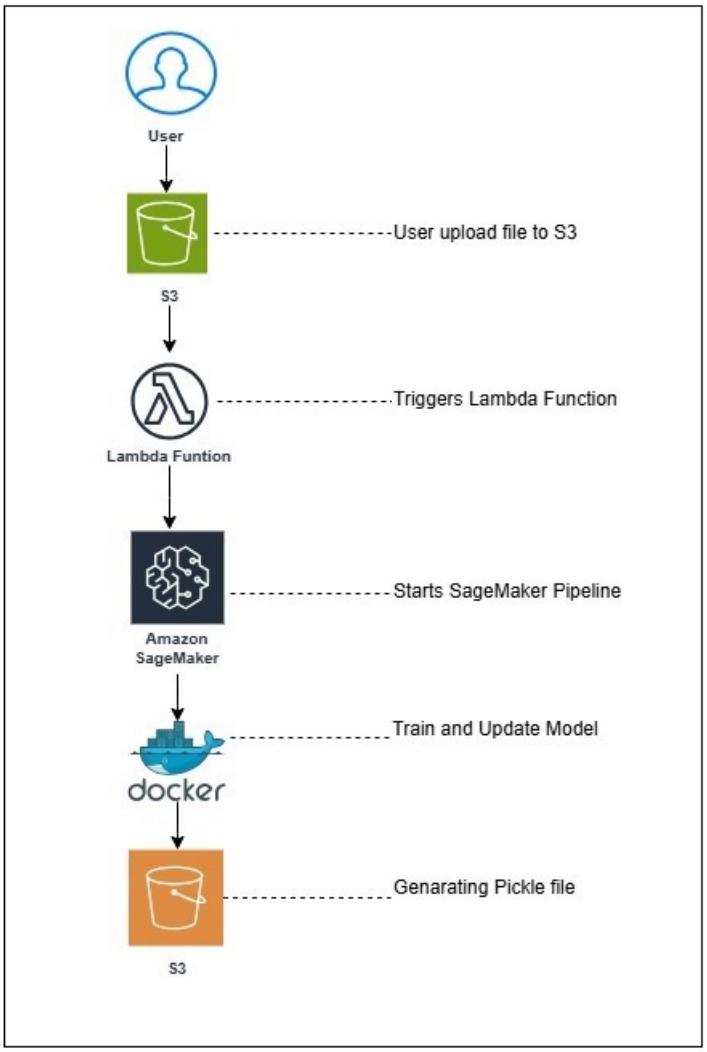

Figure 4: SageMaker ML Pipeline (Original Illustration)

1. **Sagemaker Domain**: I created a SageMaker domain to isolate my project from other machine learning projects within the organization. This is a standard procedure when there are numerous projects in progress. This newly created space includes all the required SageMaker tools such as Jupyter notebooks, pipelines, model registry, MLFlow, Data Wrangler and others which supports project development. *What is Amazon SageMaker? - Amazon SageMaker* (n.d.)

2. **JupyterLab Notebook**: I started a JupyterLab environment using the "ml.m5.large" instance to manage the required computing power for the initial model training in the notebook. I developed notebook with exploratory data analysis (EDA), data preprocessing, and model training codes using the dataset stored in S3. I have submitted the notebook (.ipynb) file in code artifact submission.

3. **Training Script**: Upon completing data preprocessing, exploratory data analysis, and model training in the notebook, I converted the notebook into an executable

script (.py file) and transferred it to S3 bucket. The S3 bucket is connected to the pipeline and uses the .py file for training the model with new datasets uploaded in S3. I have submitted the executable python script (.py) file in code artifact submission.

4. **Pipeline**: After finishing above steps, I created the pipeline notebook where I used ScriptProcessor module of sagemaker to process executable file created earlier. I have also provided the image URI for the container to use execute .py file for model training. 5

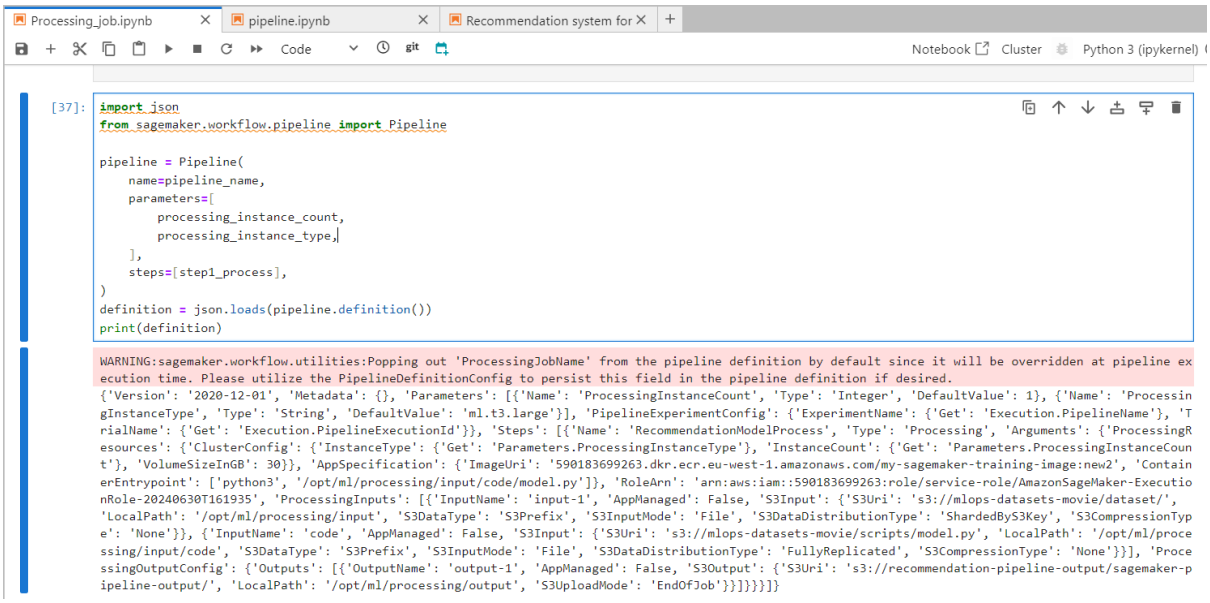

Figure 5: SageMaker ML Pipeline

5. **Pickle File**: Once the model is trained, The pickle file of the model is dumped into a S3 bucket which is accessed by the Flask application. 6

model.py 9+

model.py > ...

```python
# Function to upload file to S3
def upload_file_to_s3(bucket_name, object_name, data):

    s3 = boto3.client('s3')

    try:
        # Upload the file
        s3.put_object(Bucket=bucket_name, Key=object_name, Body=data)
        print(f"File {object_name} uploaded to {bucket_name}")
        return True
    except Exception as e:
        print(f"Upload failed: {e}")
        return False

model_data = (indices, similarity, movie)
buffer = BytesIO()
pickle.dump(model_data, buffer)
buffer.seek(0)

upload_file_to_s3('recommendation-pipeline-output', 'model.pkl', buffer.getvalue())
```

Figure 6: Pickle Dump Code

6. **S3 bucket for input dataset, script**: S3 is used to store the input dataset, script and the model pickle file. S3 is the main component in both the pipelines and it has source files to trigger the sagemaker pipeline and the pickle file which is loaded in flask application for user interaction. 7

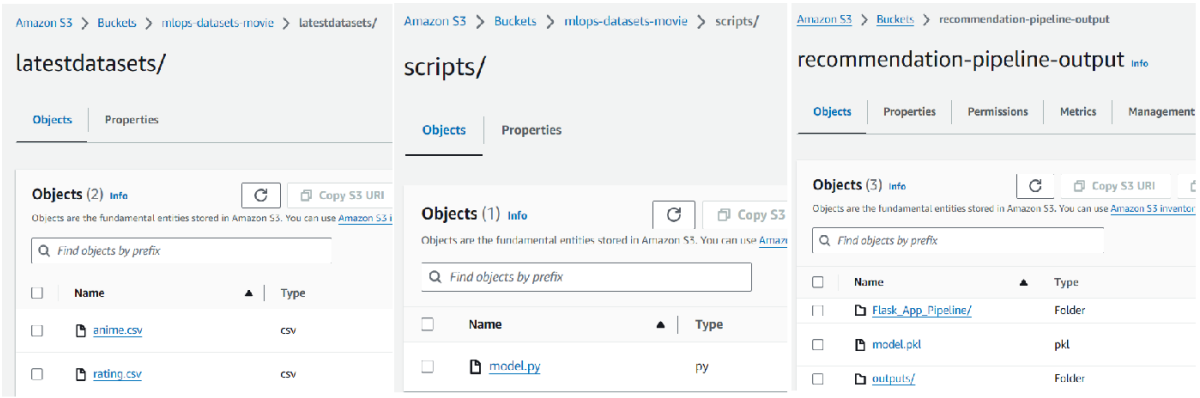

Figure 7: S3 Buckets

7. **Lambda Trigger On S3 Bucket PUT API**: This ML pipeline developed is actually triggered by a PUT API called on S3 bucket where a new dataset in inserted. In response, lambda is triggered and sagemaker pipeline is invoked. 8

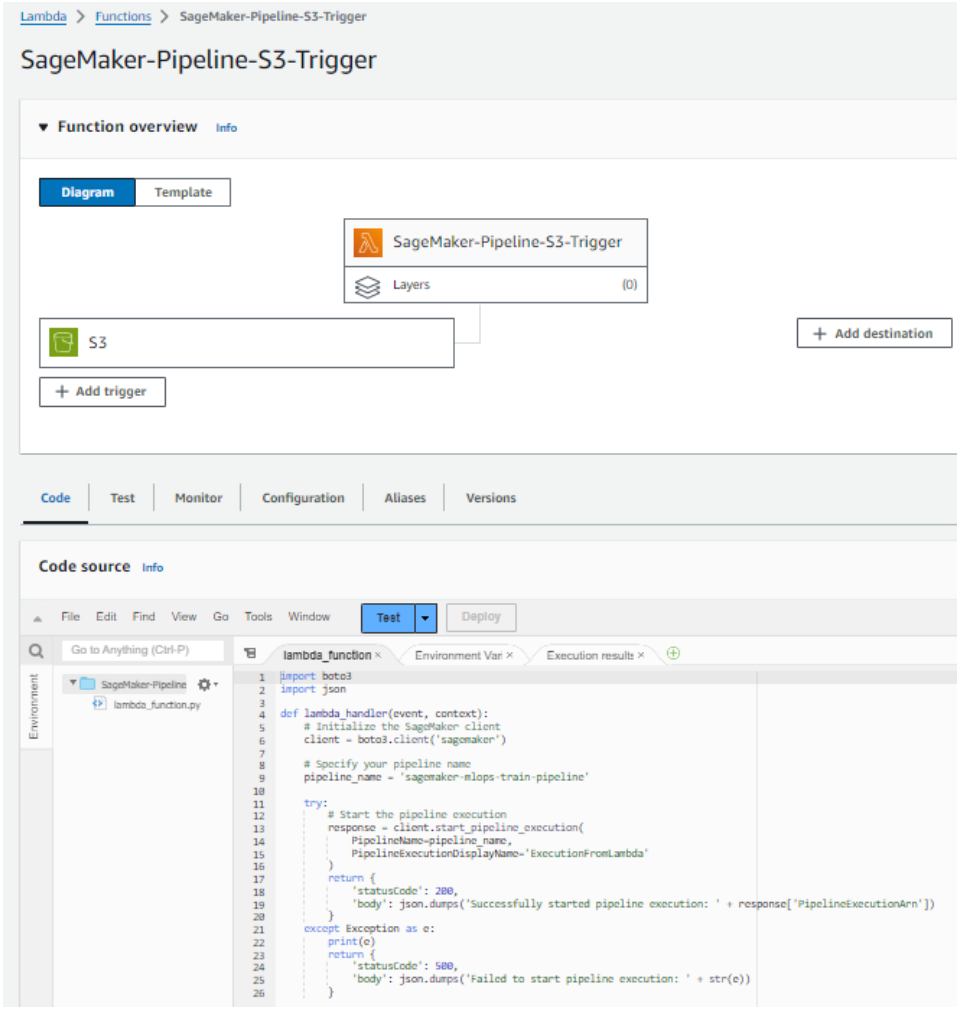

Figure 8: SageMaker Pipeline Trigger Lambda Function

8. **Training Image On ECR**: The image URI provided in step 4 image 9, is built and stored in AWS ECR which contains dependencies required for the successful training of model. The dockerfile used to build the docker image is given below.

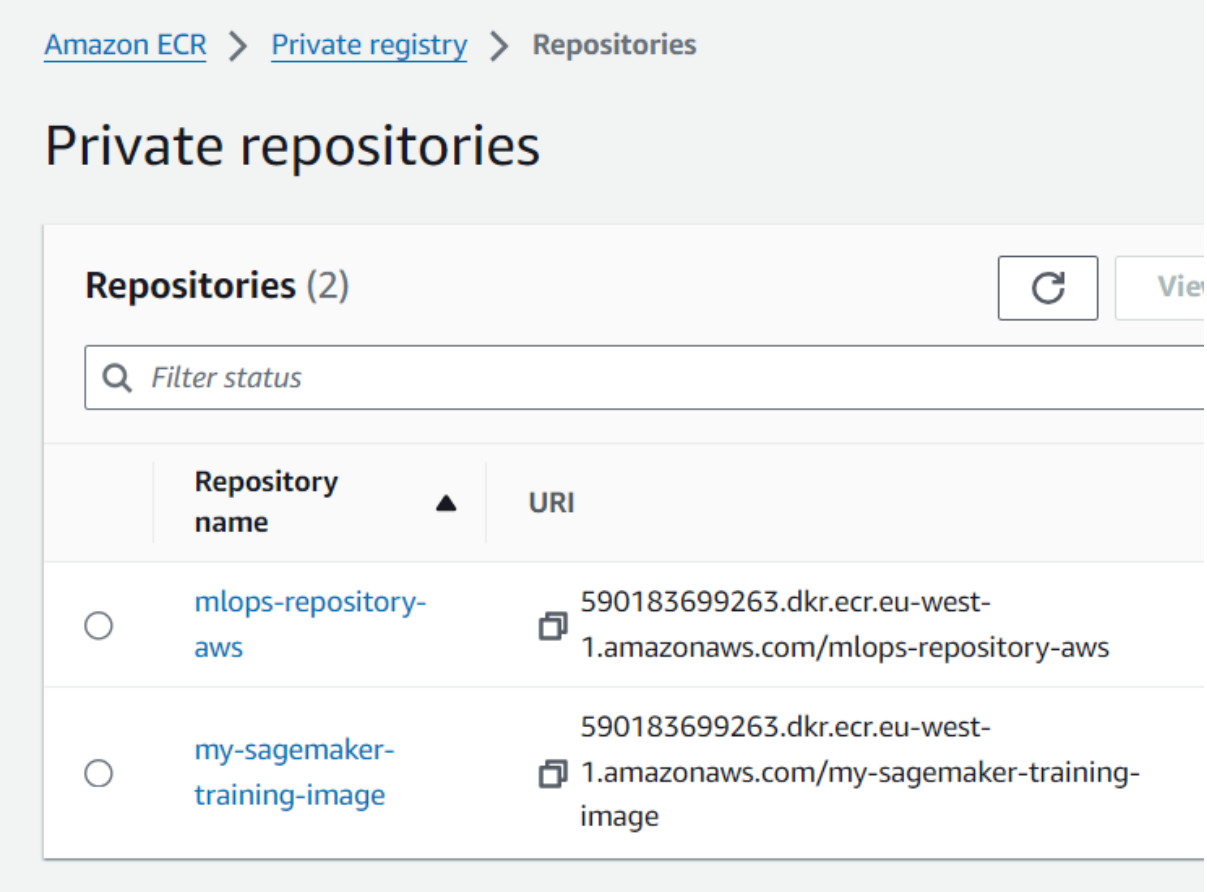

Figure 9: ECR Repository

9. **Cloudwatch Logs**: There are log groups created to monitor lambda trigger functionality and model training job as "/aws/lambda/SageMaker-Pipeline-S3-Trigger" and "/aws/sagemaker/ProcessingJobs" respectively.

5.2 Flask Application CI/CD

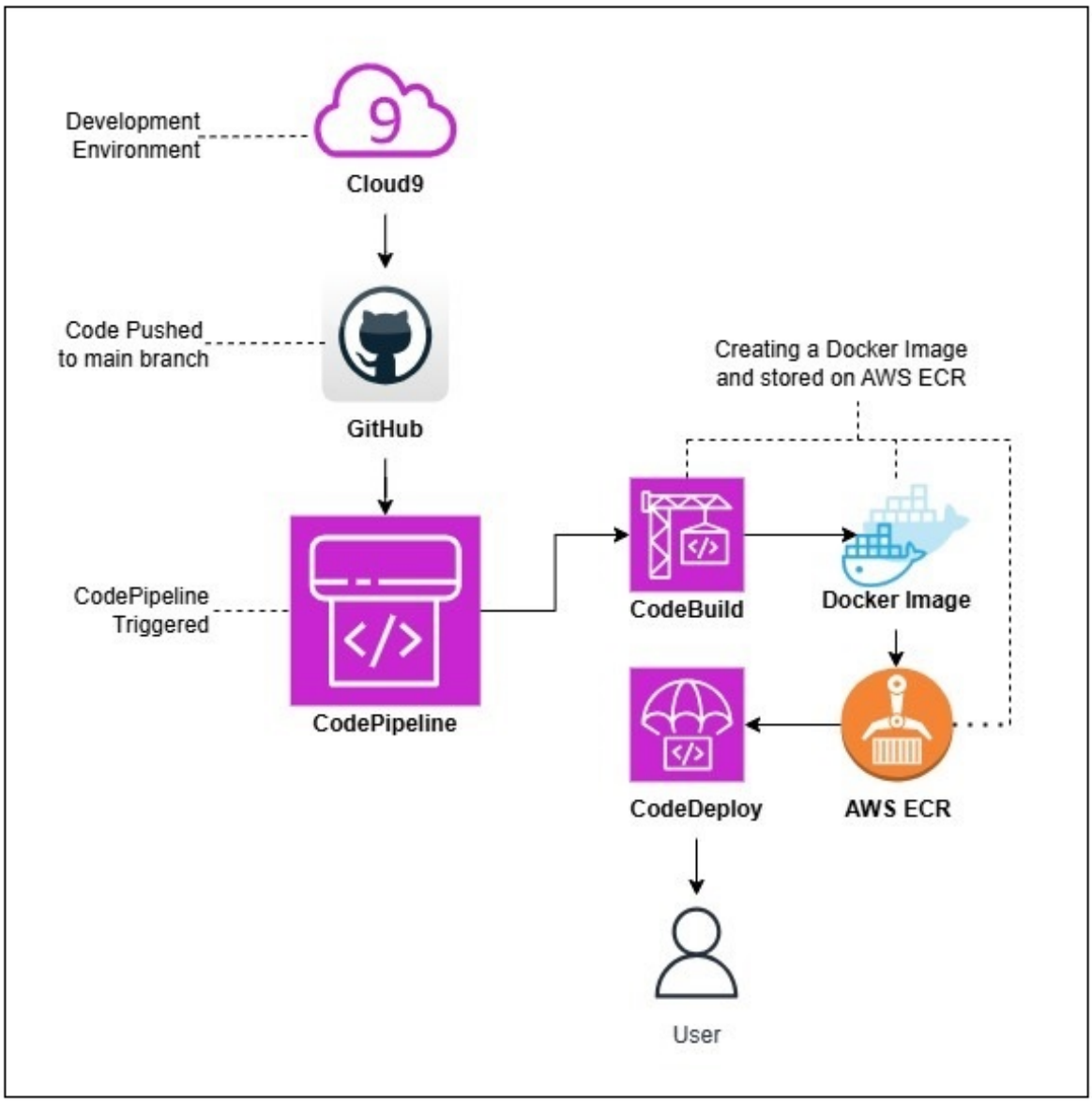

Figure 10: CI/CD Pipeline of Flask Application

1. **Cloud9**: AWS Cloud9 is an online IDE which is used to develop my flask application and push it to git repository. Cloud9 spin off a EC2 instance in background to host the IDE. The image 11 depicts code deployed.

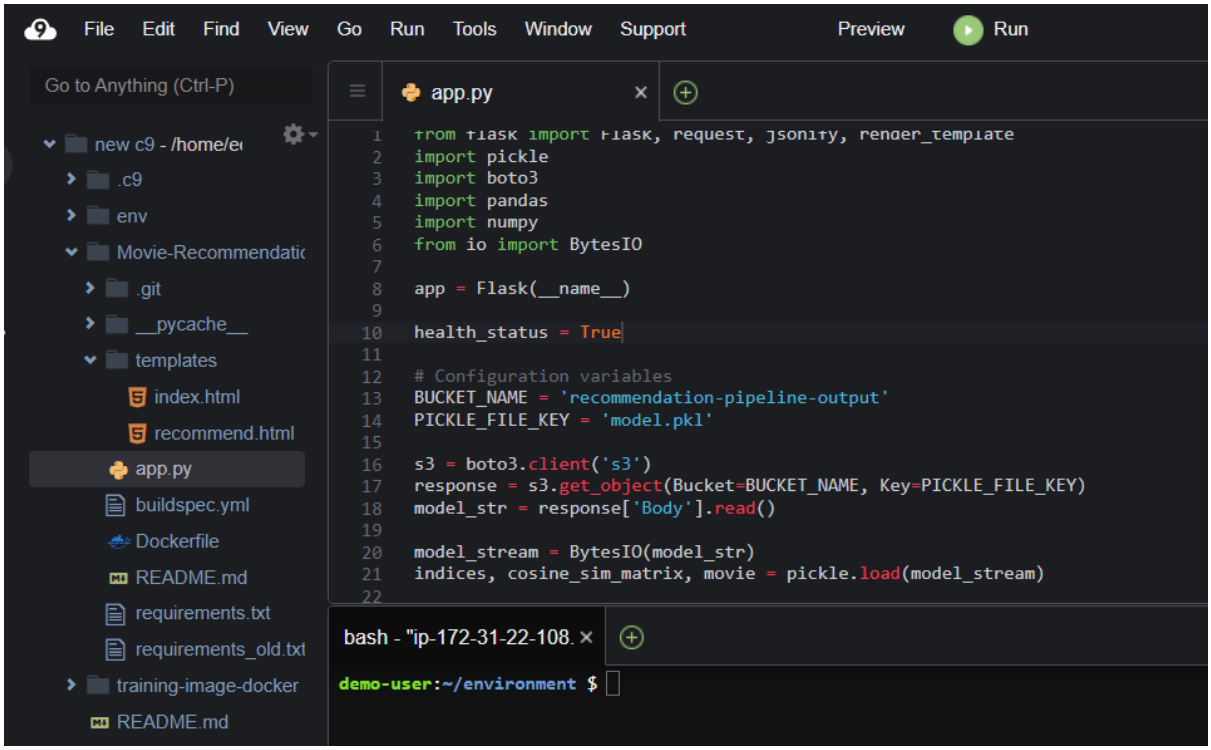

Figure 11: Flask Application in Cloud9

2. **GitHub**: Github is used as a source code repository for the flask application. This is a source for the AWS CodePipeline to start its execution.

3. **CodePipeline**: CodePipeline is used to establish a CI/CD pipeline to host the flask application on the docker container on AWS ECS Container service. There are 3 phases in this CI/CD pipeline which are.

    - **Source Phase**: This acts as the starting point of the pipeline which is invoked by the push on the main branch of the above github repository.
    - **CodeBuild**: Build Docker image using docker file- install dependencies and move project code and run application on boot. It also pushes the latest image on ECR repository.
    - **Deployment**: Container service AWS ECS used with Fargate which acts as serverless container to host the flask application. It saves cost of running containers 24x7.

      Below 12 is the screenshot of the successful execution of my CI/CD pipeline:

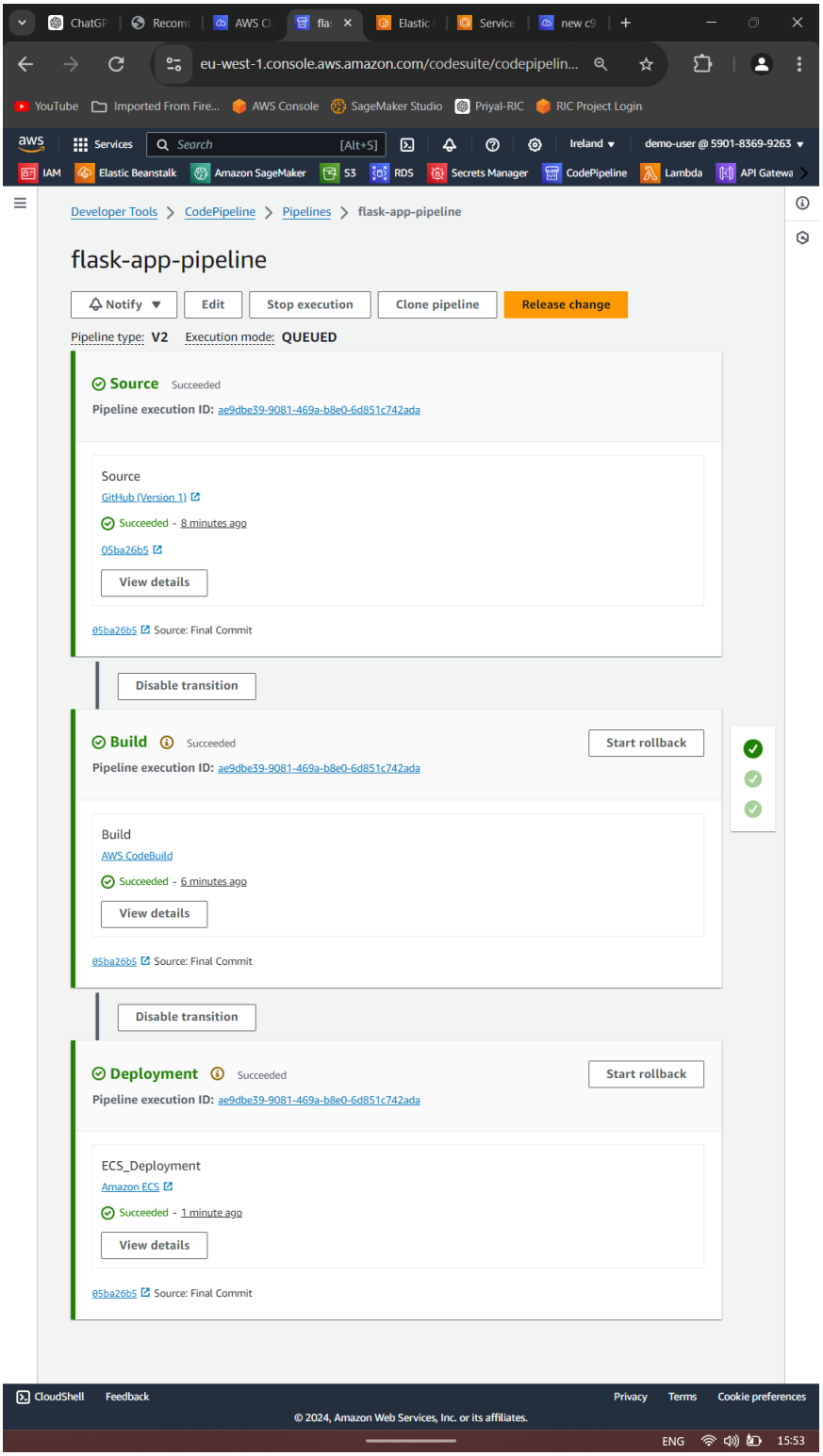

Figure 12: CI/CD Pipeline of Flask Application (Original Illustration)

- **AWS ECR**: Private Docker repo for prod image of my flask app and training image of sagemaker.
- **CloudWatch Logs**: Monitoring is enabled for the build phase of pipeline in cloudwatch under log group "/aws/codebuild/mlopsbuild". ECS container in-

sights logs are named as "/aws/ecs/containerinsights/MLOpsProdClusterECS/performance"

6 Evaluation

The evaluation of the MLOps pipeline and the movie recommendation system was carried out by examining several key metrics that highlight the benefits of integrating MLOps practices within AWS.

1. Precision: Precision is an important evaluation for checking the performance of a recommendation system. It is also defined as the ratio of relevant recommendations I.e true positives to the total number of items recommended I.e sum of true positives and false positives. With respect to movie recommendations, a high precision score indicates that the system predominantly suggests good movies that users will likely to find appealing. This metric is particularly important in scenarios where the cost of false positives—recommending irrelevant movies—is high, it will directly impacts user satisfaction.

2. Monitoring: This evaluation metric consists of two metrics:

   - CPU Utilization: CPU utilization method will measures the percentage of available CPU capacity used during different stages of the MLOps pipeline, such as data processing, model training, and inference. High CPU utilization will typically indicate that the system is performing intensive computational tasks, such as complex calculations or processing large datasets. on the other hand, low CPU utilization can suggest that resources are underutilized, which could indicate inefficiencies in the pipeline. Monitoring CPU utilization is essential for optimizing the allocation of computational resources, hence ensuring that the system operates efficiently without unnecessary resource expenditure.
   - Memory Usage: Memory usage will tracks the amount of RAM consumption during the execution of tasks in the pipeline, particularly during model training and inference. Efficient memory management is crucial asit helps for handling large models and datasets, as insufficient memory can lead to performance bottlenecks, such as increased latency or system crashes. By monitoring and optimizing memory usage, the system can easily maintain high performance. If there is heavy workloads or with complex models that it require significant memory resources.

3. Cost Efficiency: Cost efficiency is an important metric which evaluates the relationship between resource utilization and the cost of running the MLOps pipeline. It involves computation of used resources, memory, and storage consumed during the pipeline's operation translate into meaningful improvements in model performance or user satisfaction. After correlating these metrics with the associated costs, the system can be fine-tuned to maximize performance while minimizing expenses, ensuring that the recommendation system remains both effective and cost-efficient.

4. Re-Training: Model is retrained within few minutes once the new dataset is placed in the S3 bucket. On average it takes 2 minutes 30 seconds to train the model on the container and dumps new pickle file. The time taken to train the model can be improved by using large container size.

5. User Feedback: This evaluation metrics was not addressed in this project due to time constraints. I propose this as a future scope to be implemented.

7 Conclusion and Future Work

This thesis was successfully developed and implemented a robust MLOps pipeline within the AWS environment, mainly goal was to optimizing movie recommendation systems. The integration of different poppular AWS services such as SageMaker, Lambda, and ECS. it helps for the continuous training, deployment, as well as the monitoring of machine learning models, significantly improving its scalability, efficiency, and reliability of the recommendation system. The implementation results shows that the MLOps framework not only enhances the accuracy of recommendations but also streamlined the operational workflow. It helps to reduce the complexity and manual intervention which are specifically associated with deploying machine learning models in the production.

The evaluation of the system showed promising improvement and in performance validating the effectiveness of the MLOps approach in a real-world application. Also, there are different opportunities for further enhancement . Future work could primarily focus on integrating real-time user feedback for updating it dynamically also improving recommendation relevance and accuracy. Second task is, the exploration of multi-modal data sources can also offer a richer, more personalized user experience by incorporating diverse types of user data.

7.1 Future Work

Future work should be focus on improvement of the adaptability and customization of the movie recommendation system by integrating the concept of real-time user feedback mechanisms. This would also involve the development of a system which is capable of continuously learning and updating recommendations based on live user interactions. It will lead to more accurate and responsive suggestions. Advances in machine learning and natural language processing continue to enhance the effectiveness of content-based filtering, making it a key component of modern recommendation systems.

On the other hand, exploring the integration of multi-modal data sources, like as user-generated content, social media activity, or visual media analysis, can significantly enhances the capabilities of recommendation engine's . By use different types of the data , the system can offer a more holistic and personalized user experience. Solve cold start problem

This research not only build the bridges the gap between model development and operational deployment but also provides a scalable framework which can be adapted for various recommendation systems. Future work can also explore the real integration of real-time user feedback and multi-modal data sources to further refine the system. The findings also suggest potential commercial applications, particularly for small and medium-sized enterprises seeking to leverage advanced machine learning capabilities without significant upfront investment.

7.1.1 Commercialization Approach

From a commercial perspective, these advancements can be packaged in the scalable Software-as-a-Service (SaaS) platform which will offer a robust, customizable recom-

mendation solutions to any businesses across various industries,for example streaming services or an e-commerce.

Furthermore, there is also potential for commercialization of this MLOps design developed in this project by offering enterprise-level consultancy services, which will help an organizations to implement and optimize their machine learning operations with best practices derived from this research.

In summary, this project not only achieved its primary goals but also gave a solid foundation for future advancements in the field of MLOps and machine learning-driven recommendation systems, with clear paths for both academic research and commercial exploitation.

References

Amazon EventBridge Documentation (n.d.).
URL: *https://docs.aws.amazon.com/eventbridge/*

Amazon Web Services (2024). Page Version ID: 1239117327.
URL: *https://en.wikipedia.org/w/index.php?title=Amazon*$_{W}$*eb*$_{S}$*servicesoldid* = 1239117327

Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E. and Stoica, I. (2017). Clipper: A {Low-Latency} online prediction serving system, *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pp. 613–627.

Karamitsos, I., Albarhami, S. and Apostolopoulos, C. (2020). Applying DevOps Practices of Continuous Automation for Machine Learning, *Information* **11**(7): 363. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.
URL: *https://www.mdpi.com/2078-2489/11/7/363*

Kumara, I., Arts, R., Di Nucci, D., Van Den Heuvel, W. J. and Tamburri, D. A. (2023). Requirements and reference architecture for mlops: insights from industry, *Authorea Preprints* .

Lops, P., De Gemmis, M. and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends, *Recommender systems handbook* pp. 73–105.

Makinen, S., Skogstrom, H., Laaksonen, E. and Mikkonen, T. (2021). Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help?, pp. 109–112.

Mu, Y. and Wu, Y. (2023). Multimodal movie recommendation system using deep learning, *Mathematics* **11**(4).
URL: *https://www.mdpi.com/2227-7390/11/4/895*

Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems, *The adaptive web: methods and strategies of web personalization*, Springer, pp. 325–341.

PireciSejdiu, N., Ristevski, B. and Jolevski, I. (2022). Performance Comparison of Machine Learning Algorithms in Movie Recommender Systems, *2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)*, pp. 1–4.
URL: *https://ieeexplore.ieee.org/document/9828583*

Sekar, S., Dhanasekaran, D., Charlyn, G. and Latha, C. (2023). Content-Based Movie Recommendation System Using MBO with DBN, *Intelligent Automation and Soft Computing* .

What is Amazon SageMaker? - Amazon SageMaker (n.d.).
URL: *https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html*