REVA UNIVERSITY
Bengaluru, India

A Project Report on

Deployment of AI-Enabled Automated Solution at AWS Cloud

Submitted in Partial Fulfilment for Award of Degree of
Master of Technology
In Artificial Intelligence

Submitted By
Gaurav Karki
R20MTA02

Under the Guidance of
Dr. Rashmi Agarwal
Assistant Professor, RACE, REVA University

REVA Academy for Corporate Excellence - RACE
REVA University
Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru - 560 064
race.reva.edu.in

August, 2022

REVA
UNIVERSITY
Bengaluru, India

Candidate’s Declaration

I, **Gaurav Karki** hereby declare that I have completed the project work towards the Master of Technology in Artificial Intelligence at, REVA University on the topic entitled **“Deployment of AI-Enabled Automated Solution at AWS Cloud”** under the supervision of **Dr. Rashmi Agarwal.** This report embodies the original work done by me in partial fulfilment of the requirements for the award of degree for the academic year **2022.**

Place: Bengaluru	Name of the Student: **Gaurav Karki**
Date: 27 August, 2022	Signature of Student

REVA UNIVERSITY
Bengaluru, India

Certificate

This is to Certify that the project work entitled **"Deployment of AI-Enabled Automated Solution at AWS Cloud"** carried out by **Gaurav Karki** with **SRN R20MTA02,** is a bonafide student of REVA University, is submitting the project report in fulfilment for the award of Master of Technology in Artificial Intelligence during the academic year **2022**. The Project report has been tested for plagiarism, and has passed the plagiarism test with the similarity score less than 15%. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the said degree.

Signature of the Guide
Name of the Guide
Dr. Rashmi Agarwal

Signature of the Director
Name of the Director
Dr. Shinu Abhi

External Viva Panelists
Names of the Examiners

1. Dr. Santosh Nair, Founder, Analytic Edge
2. Rajkumar Dan, Data Scientist Consultant, Dell

Place: Bengaluru
Date: 27th Aug 2022

REVA UNIVERSITY
Bengaluru, India

Acknowledgement

I would like to acknowledge the support provided by the founder and Hon'ble Chancellor, **Dr. P Shayma Raju**, Vice-Chancellor, **Dr. M. Dhanamjaya**, and Registrar, **Dr. N Ramesh**.

I would like to express my gratitude to **Dr. Shinu Abhi**, Director of Corporate Training, for providing me significant guidance and assistance throughout the course and my project.

I would like to thank my Mentor, **Dr. Rashmi Agarwal** for the invaluable assistance she provided as my project guide in helping me to comprehend the concept and execute this project.

I am grateful to my mentors, for their great guidance and ideas in learning numerous Text Analytics aspects, as well as for their valuable guidance on a variety of project-related problems.

I would also thank all the mentors and program office at REVA Academy of Corporate Excellence for teaching us awesome methodologies and support throughout the academic year.

REVA
UNIVERSITY
Bengaluru, India

Similarity Index Report

This is to certify that this project report titled "**Deployment of AI-Enabled Automated Solution at AWS Cloud"** was scanned for similarity detection. Process and outcome is given below.

Software Used: Turnitin

Date of Report Generation: 27th August 2022

Similarity Index in %: 7%

Total word count: 3765

Name of the Guide: **DR. Rashmi Agarwal**

Place: Bengaluru

Date: 27th August 2022

Name of the Student: Gaurav Karki

Signature of Student

Verified by: M N Dincy Dechamma

Signature

Dr. Shinu Abhi,

Director, Corporate Training

List of Abbreviations

Sl. No	Abbreviation	Long Form
1	AI	Artificial Intelligence
2	PA	Prior Authorization
3	STS	Semantic Textual Similarity
4	ML	Machine Learning
5	MLOps	Machine Learning Operations
6	ECR	Elastic Container Registry
7	S3	Simple Storage Service
8	IAM	Identity and Access Management
9	OCI	Open Container Initiative

List of Figure No.s

Abstract

Machine learning (ML) has become ubiquitous in the tech sector. ML product development and rapid deployment into production is the ultimate objective of all ML operations in the industrial sector. Businesses fail to build production-level models, and as a consequence, ML fails to provide the promised (financial) value proposition. The major issue is that there is a disconnect between developing ML models and systematically integrating them into operational workflows. Businesses understand that they need people with technical skills like data engineering, cloud computing, and ML if they want to see a return on investment. And Machine Learning Operations (MLOps) is a paradigm that solves the problem.

This study is a continuation of a previous study whose objective was to develop an "AI-Enabled Automation Solution for Utilization Management in Healthcare Insurance." The purpose of this study is to investigate the feasibility of automating and operationalizing manual ML processes, with the end goal of accelerating the rate at which ML proofs of concept can be deployed.

MLOps is a framework for managing ML projects that is based on DevOps principles. To aid in the operationalization of ML, cloud providers like Amazon Web Services (AWS) have developed various services. To follow the MLOps method, the solution from the previous study is being deployed using Docker images and AWS cloud services like AWS Lambda, Amazon Elastic Container Registry (ECR), Amazon Simple Storage Service (Amazon S3), Identity and Access Management (IAM), and Amazon CloudWatch.

As a result, this study gives a comprehensive review of the required concepts, modules, and responsibilities, as well as the accompanying architecture and operations. Implementing MLOps techniques expedites time-to-market for ML initiatives by delivering repeatability, productivity, auditability, reliability, model quality, and data quality.

This study assists businesses in streamlining their end-to-end ML lifecycle and increasing the productivity of data scientists, all while increasing overall accuracy of the proposed model and improving compliance and security.

Keywords: Healthcare Insurance, Utilization Management , MLOps, Cloud Computing, Deployment, AWS Lambda, ECR, Amazon S3, IAM, Docker, Amazon CloudWatch

Table of Contents

Chapter 1: Introduction

While similar to other software systems, an ML system is distinguished by the presence of a machine learning model. ML systems may and do achieve the same rewards as those seen with DevOps. New methods, such as Data Versioning and AutoML, show a lot of promise for encapsulating the DevOps mentality because of their emphasis on automation. When excellent practices in DevOps and data engineering are in place, software systems like ML systems perform effectively and reliably. While not perfect, the ML hierarchy of demands depicted in Figure No. 1.1 is an excellent starting point for comprehending MLOps.

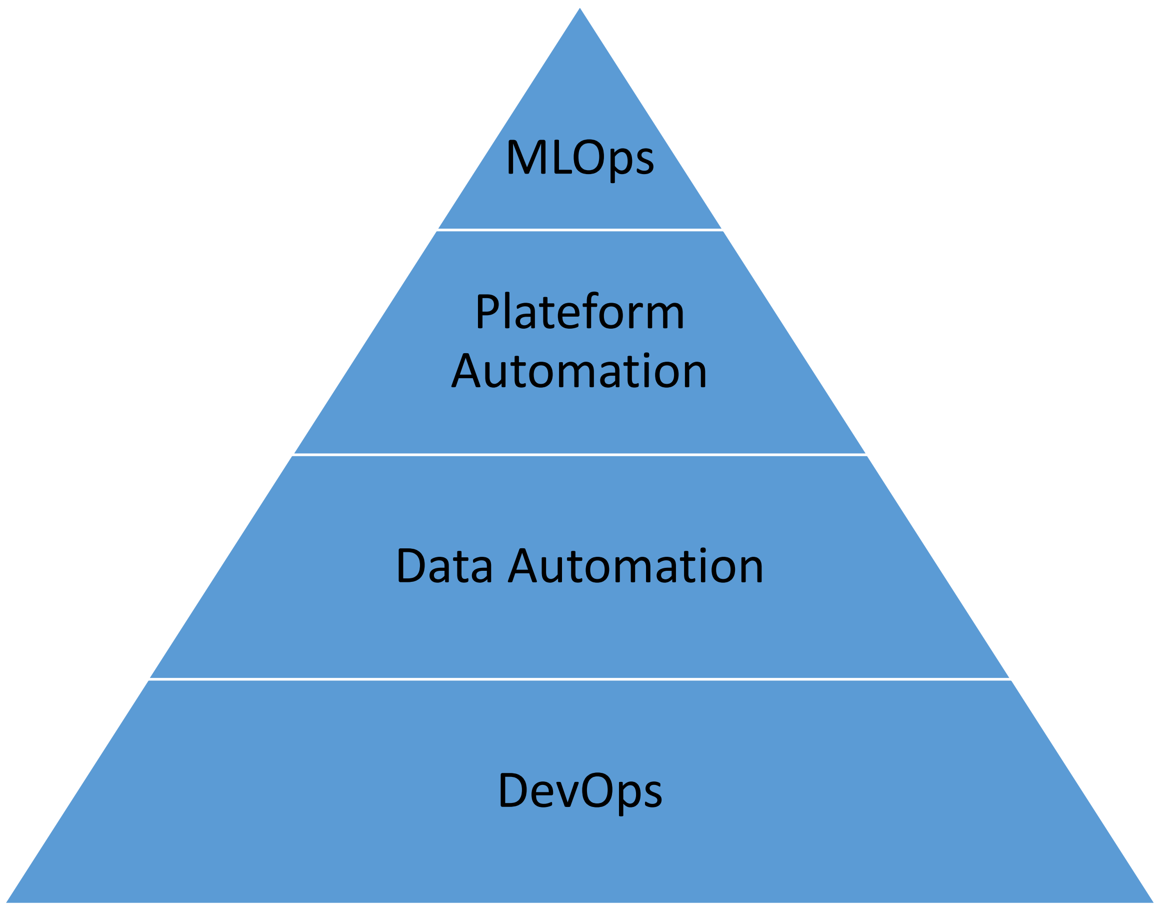


Figure No. 1.1 Ml Engineering Hierarchy of Needs

The inexperience of the data science industry is a fundamental obstacle to the production deployment of machine learning models. DevOps has been embraced by the software industry to address comparable concerns, whereas MLOps has been adopted by the machine learning community. The construction of several machine learning models and their deployment in a production setting are relatively new for the majority of conventional enterprises. Until recently, the amount of models may have been manageable on a small scale, or there may have been less company-wide interest in understanding these models and their dependencies. With decision automation (that is, a rising incidence of decisions made without human interaction), models become more significant, and controlling model risks at the executive level becomes more crucial. As depicted in Figure No. 1.2, the machine learning life cycle in an industrial context is significantly more complex in terms of requirements and tools [1].

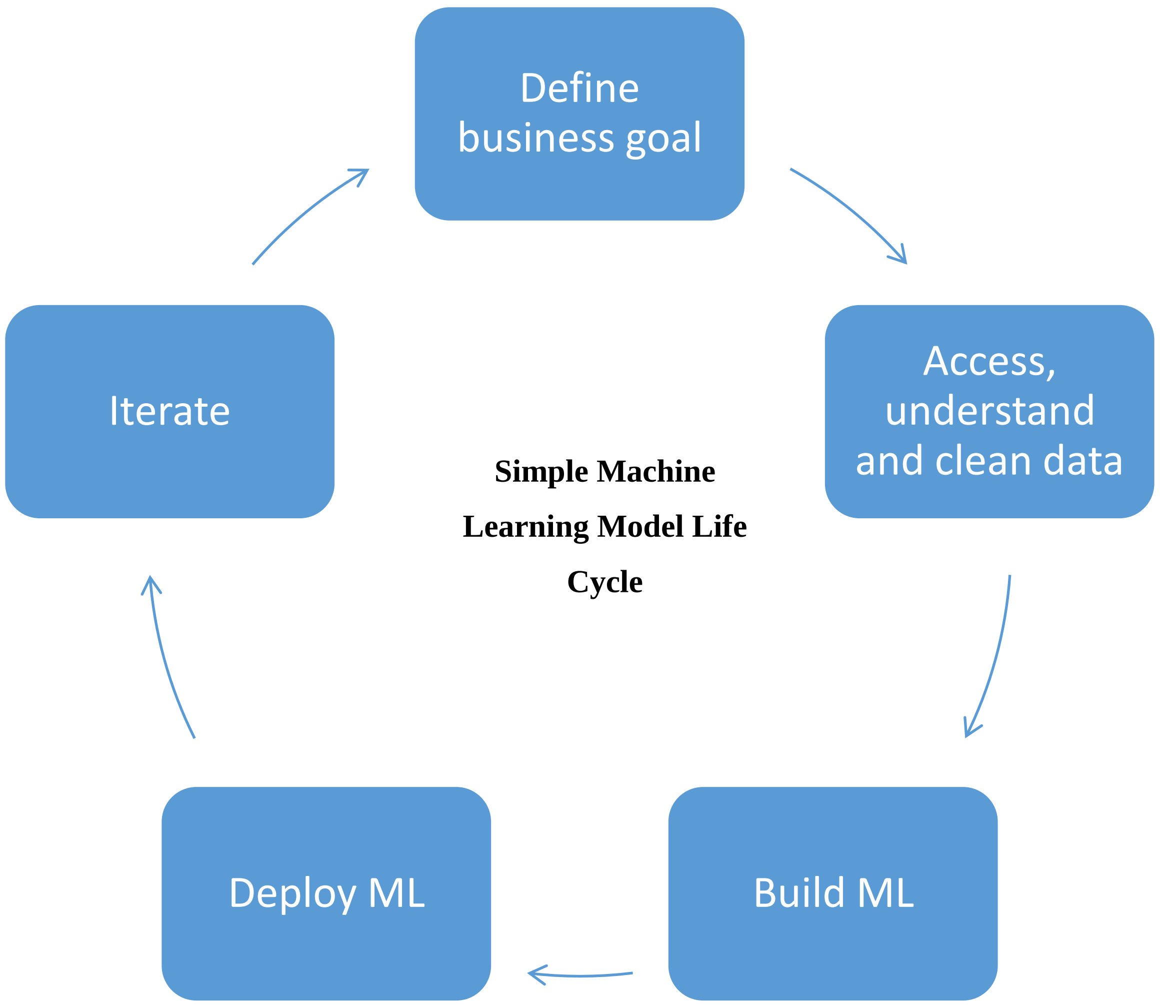

Define business goal
Access, understand and clean data
Build ML
Deploy ML
Iterate
Simple Machine Learning Model Life Cycle

Figure No. 1.2 A Simple Representation of the Machine Learning Model Life Cycle

MLOps, like DevOps, is based on a collaborative and streamlined approach, where the convergence of people, process, and technology optimises end-to-end processes, when it comes to establishing, constructing, and running machine learning workloads. MLOps, which is founded on the intersection of data science and data engineering with existing DevOps approaches, streamlines model delivery throughout the entire machine learning development lifecycle.

MLOps refers to the process of integrating machine learning workloads into regular practises such as release management and continuous integration/delivery. Without the collaboration of software engineering, operations, data engineering, and data science, there can be no MLOps. MLOps is the study of combining machine learning workflows with other operational processes, such as release management, continuous integration/delivery, and operations. MLOps requires the consolidation of software engineering, operations, data engineering, and data science [2].

As more enterprise organizations use ML, manual operations for generating, training, and deploying ML models seem to become innovation bottlenecks. To overcome this, businesses must develop a clear operating model that specifies how multiple personalities, such as ML engineers, data engineers, data scientists, and business stakeholders, should co - operate and interact, how to separate concerns, responsibilities, and skills, and how to use AWS services optimally. The history of automation suggests that individuals are most valuable when using technology as architects and practitioners, rather than while performing repetitive tasks. Similarly, developers, models, and operations must collaborate through transparent teamwork and fruitful cooperation.

Chapter 2: Literature Review

ML is now a must-have tool for maximizing data's worth, which in turn helps firms innovate, save money, and stay in business for the long haul [3][4]. Despite this potential, many actual ML applications fall short of expectations [5]. The majority of machine learning attempts fail, and many ML proofs of concept are never implemented [6]. From a theoretical standpoint, this is not surprising because the ML community has focused primarily on ML model development and has paid little attention to (a) creating ML products ready for mass production and (b) coordinating the resulting, often complex, ML system components and infrastructure, including the roles required to automate and operate a real-world ML system [7]. In many industrial applications, for instance, data scientists still handle ML operations manually to a significant degree, resulting in several operational concerns with the particular ML solution.

This is required to explore the new ML engineering practice in order to solve this issue. MLOps precisely addresses the creation and maintenance of productive ML. To reach a common knowledge of the related components, concepts, duties, and architectures, we use a holistic viewpoint [8]. MLOps' emergence as a major industry standard is spurred by the inability to move models to production. MLOps is similar to DevOps in that, from a philosophical sense, DevOps demands automation. It is often believed that everything that is not automated is defective. Similarly, the MLOps system cannot have any components that exploit humans as machine levers. The history of automation reveals that humans are most valuable when utilizing technology as architects and practitioners, rather than while performing monotonous tasks. Likewise, coordination between developers, models, and operations requires open teamwork and effective collaboration. Consider MLOps to be the automation of machine learning using DevOps principles [2].

MLOps refers to the process of automating machine learning with DevOps approaches. The process of developing machine learning is known as machine learning engineering. MLOps is therefore a behaviour, just as DevOps is a behaviour. While some people work as DevOps engineers, a software engineer is more likely to utilize DevOps best practices when performing tasks. Similarly, a machine learning engineer should design machine learning systems using MLOps best practices [9].

MLOps builds upon these approaches and expands particular elements to directly target machine learning systems. One way to describe these best practices is to note that they provide repeatable models with robust packaging, validation, and deployment. These factors also improve the capacity to describe and observe model performance as shown in Figure No. No. 2.1.

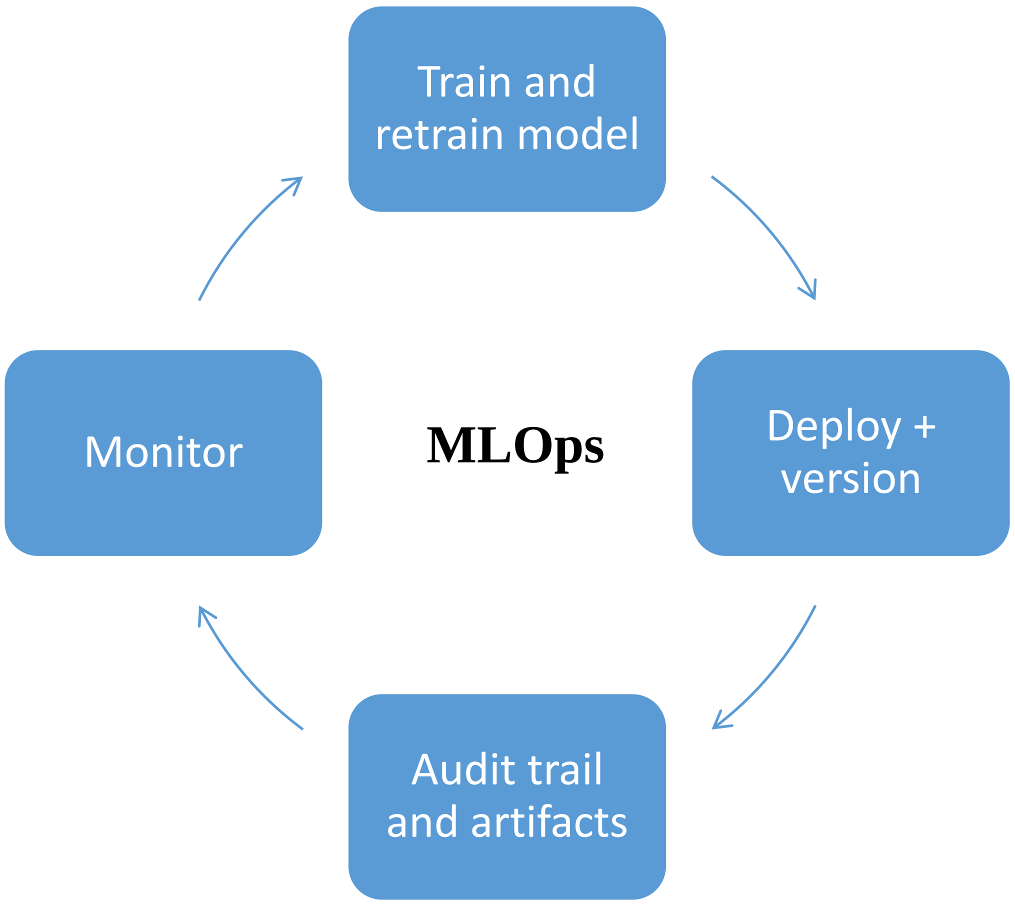


Figure No. 2.1 MLOps Feedback Loop

Chapter 3: Problem Statement

Every company's machine learning efforts should ultimately result in the rapid release of a commercial machine learning product. However, many ML projects fail to meet their objectives because automating and operationalizing ML solutions is so challenging. Therefore, the issue that we have at this stage is to deploy the solution that was found in the earlier study by employing cloud services. There are three main obstacles to handling machine learning life cycles at scale:

1. Multiple interdependencies exist. Not only are data continually evolving, but so are business requirements. Continuous feedback must be provided to the business to ensure that the actuality of the model in production and on production data fits with expectations and, most importantly, addresses the original issue or achieves the original objective.

2. Not all individuals speak the same language. Even while members of the business, data science, and IT teams are involved in the machine learning life cycle, none of these groups use the same technologies or, in many cases, possess the same core communication skills.

3. Data scientists are not computer programmers. Most are skilled in model construction and evaluation, but are not always professionals in application authoring. This may change in the future when some data scientists become deployment or operational specialists, but for the time being, many data scientists must juggle multiple tasks, making it difficult to do any of them thoroughly. With an increasing number of models to handle, it becomes increasingly troublesome for data scientists to be spread too thin at scale. Considering the employee turnover on data teams, the complexity increases exponentially, and suddenly data scientists must manage models they did not design.

Chapter 4: Objectives of the Study

This study relies on a previous effort to develop a "AI-Enabled Automation Solution for Utilization Management in Healthcare Insurance." In order to expedite the implementation of ML proofs of concept, the objective of the study is to deploy solution from the previous study using the cloud service AWS and examines the possibility for automating and operationalizing manual ML operations. While MLOps provides useful tools for scaling your business, it might be difficult to incorporate them into your machine learning projects. To address the foregoing difficulties, the sub-objectives are as follows:

1. **Project Management:** For ML initiatives, data scientists are infrequently included into cross-functional teams, despite the fact that they are an increasingly rare vocation. The difficulty of converting business objectives into technical requirements is exacerbated when new team members lack a common technical vocabulary with product owners and software developers.

2. **Collaboration and Communication:** Data scientists are rarely included in cross-functional teams for ML initiatives, despite the fact that their profession is becoming increasingly uncommon. When new team members lack a technical language shared by product owners and software engineers, the problem of translating business objectives into technological needs is increased.

3. **Monitoring and logging**: In addition to the normal endpoint stability and performance indicators, deployed machine learning models must monitor the data provided to the model for inference. The monitoring system must also capture the output quality of the model, as determined by an acceptable metric for machine learning.

Chapter 5: Project Methodology

To establish an MLOps framework is difficult that can satisfy the operational, human resource, and technology needs of enterprise clients. Therefore, we describe a maturity model that divides the requisite MLOps capabilities into four separate stages as shown in Figure No. 5.1 [10].

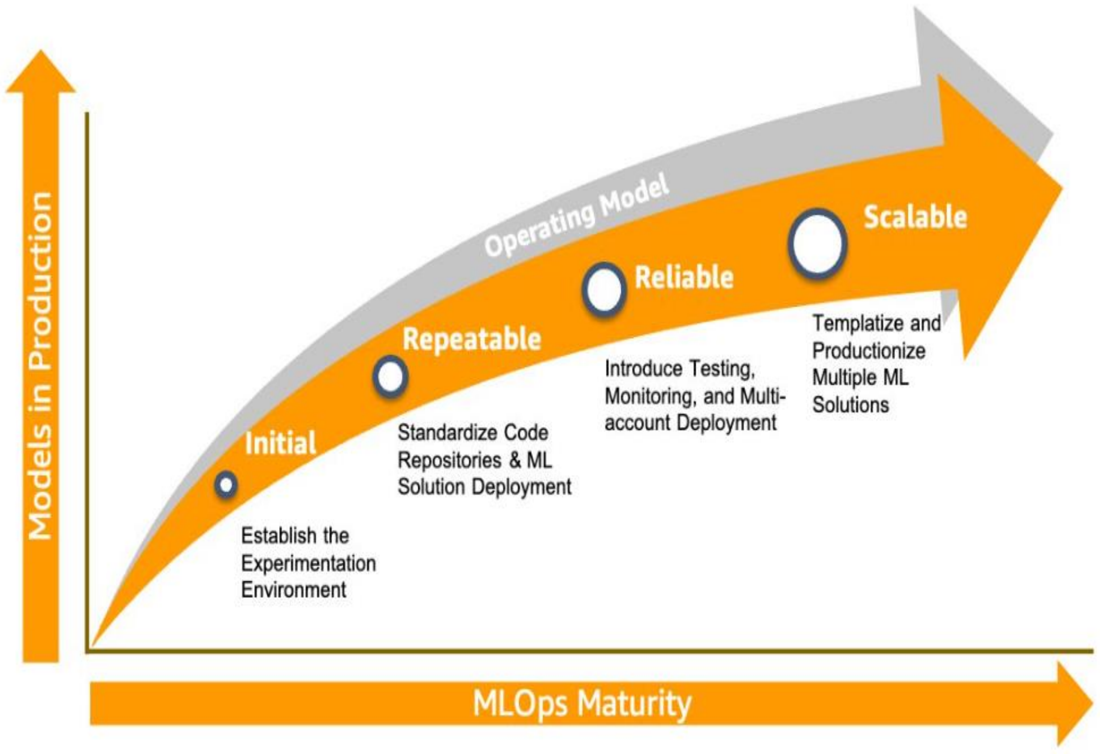

Models in Production
Operating Model
Initial
Establish the Experimentation Environment
Repeatable
Standardize Code Repositories & ML Solution Deployment
Reliable
Introduce Testing, Monitoring, and Multi-account Deployment
Scalable
Templatize and Productionize Multiple ML Solutions
MLOps Maturity

Figure No. 5.1 Maturity Model

5.1 Initial Phase

During this phase, data scientists can experiment with SageMaker services to construct, train, and deploy models on AWS. Amazon SageMaker Studio is indicated as the development environment in which data scientists can experiment and collaborate using Studio notebooks.

5.2 Repeatable Phase

With the capacity to experiment on AWS, the next step is to develop workflows for automatically pre-processing data and building and training models (ML pipelines). In a separate environment, data scientists and ML engineers construct production-ready algorithms and source code, coordinated by Amazon SageMaker Pipelines. The Amazon SageMaker model registry stores and evaluates the created models.

5.3 Reliable phase

Even if the models are built by the ML pipelines, they must be tested prior to deployment. In this phase, a staging environment that mimics the production environment is created, and an automated testing technique is established for both the model and the infrastructure that triggers the model. Upon satisfactory completion of the testing phase, the models advance to the manufacturing phase. To push models across the multiple ecosystems, manual evaluation and permissions are required.

5.4 Scalable phase

After the first machine learning solution has been deployed, the MLOps framework must be extended to accommodate several data science teams collaborating to deploy tens or hundreds of ML use cases. Here, we provide solution templatization, which accelerates time-to-value by reducing the time required to generate new production solutions from weeks to days. To further minimize tech sector’s cost and dependence, we automate the instantiation of secure MLOps settings that allow several teams to simultaneously operate on their data.

Chapter 6: Resource Requirement Specification

There are two distinct sorts of required resources.

1. Data Resources
2. Technical Resources

6.1 Data Resources

This study requires only Prior Authorization (PA) cases in csv format; our earlier research allowed us to generate data resources, so this study only needs those examples.

6.2 Technical Resources

The following are the requirements for this resource.

1. All the files from the previous study.
2. Python environment with required libraries
   - I. NLTK
   - II. Pandas
   - III. Numpy
   - IV. Json
   - V. Tensorflow
   - VI. Scipy
   - VII. Re
3. Docker
4. Pre-trained model - Universal Sentence Encoder
5. Semantic Textual Similarity (STS) benchmark data for evaluation
6. AWS CLI
7. AWS management console
8. AWS cloud services
   - I. AWS Lambda,
   - II. Amazon Elastic Container Registry (ECR),
   - III. Amazon Simple Storage Service (Amazon S3),
   - IV. Identity and Access Management (IAM),
   - V. Amazon CloudWatch

Chapter 7: Software Design

This study intends to deploy the AI-enabled automated solution at the cloud platform AWS. Figure No. 7.1 shows the overview of the whole process that is the combination of docker and lambda function and as a result it replicates lambda environment inside a Docker container.

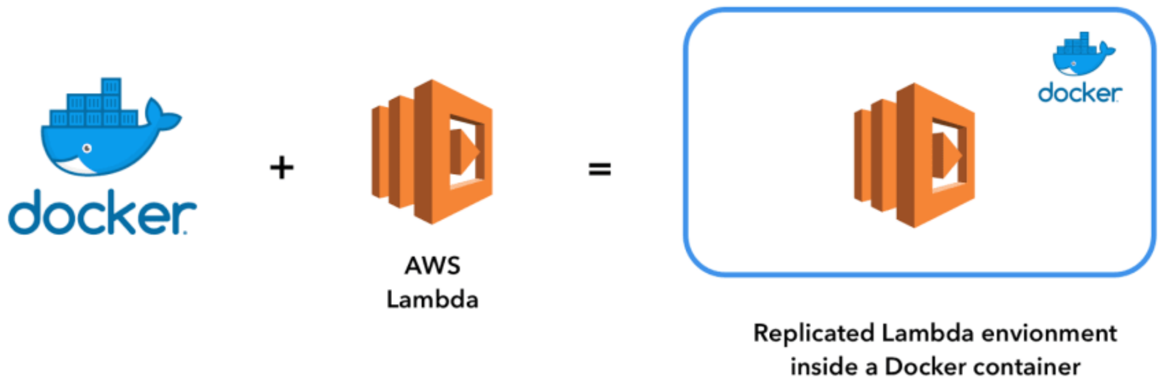


Figure No. 7.1 Overview of Deployment Process

Figure No. 7.2 shows that docker is used to create an docker image from the application.

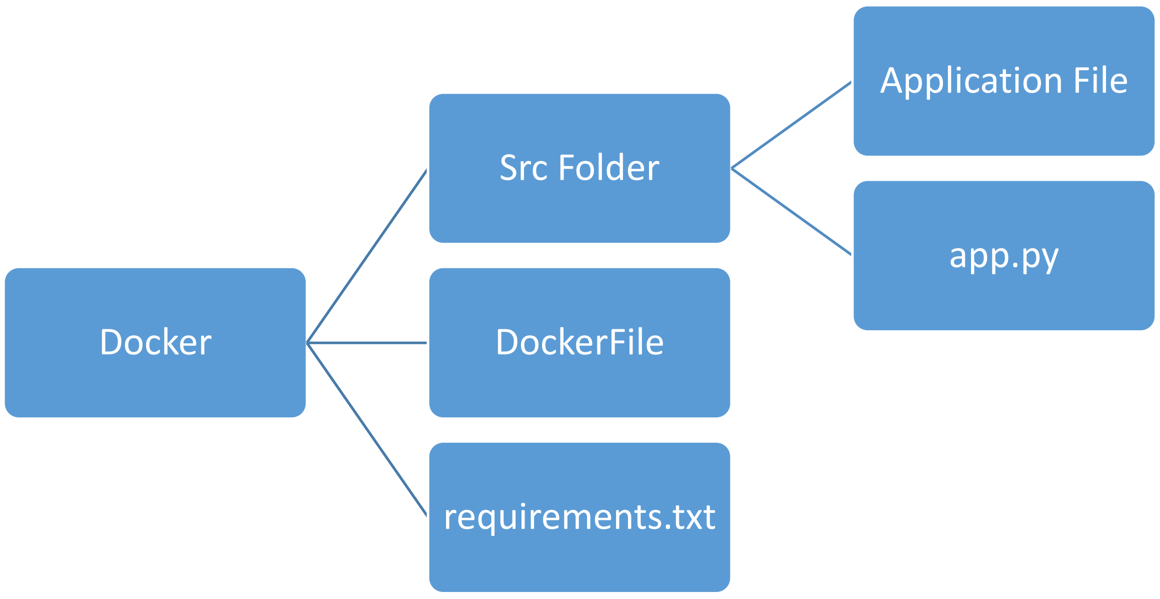


Figure No. 7.2 Required Files for Docker Image Creation

Figure No. 7.3 shows the whole process with required each AWS service.

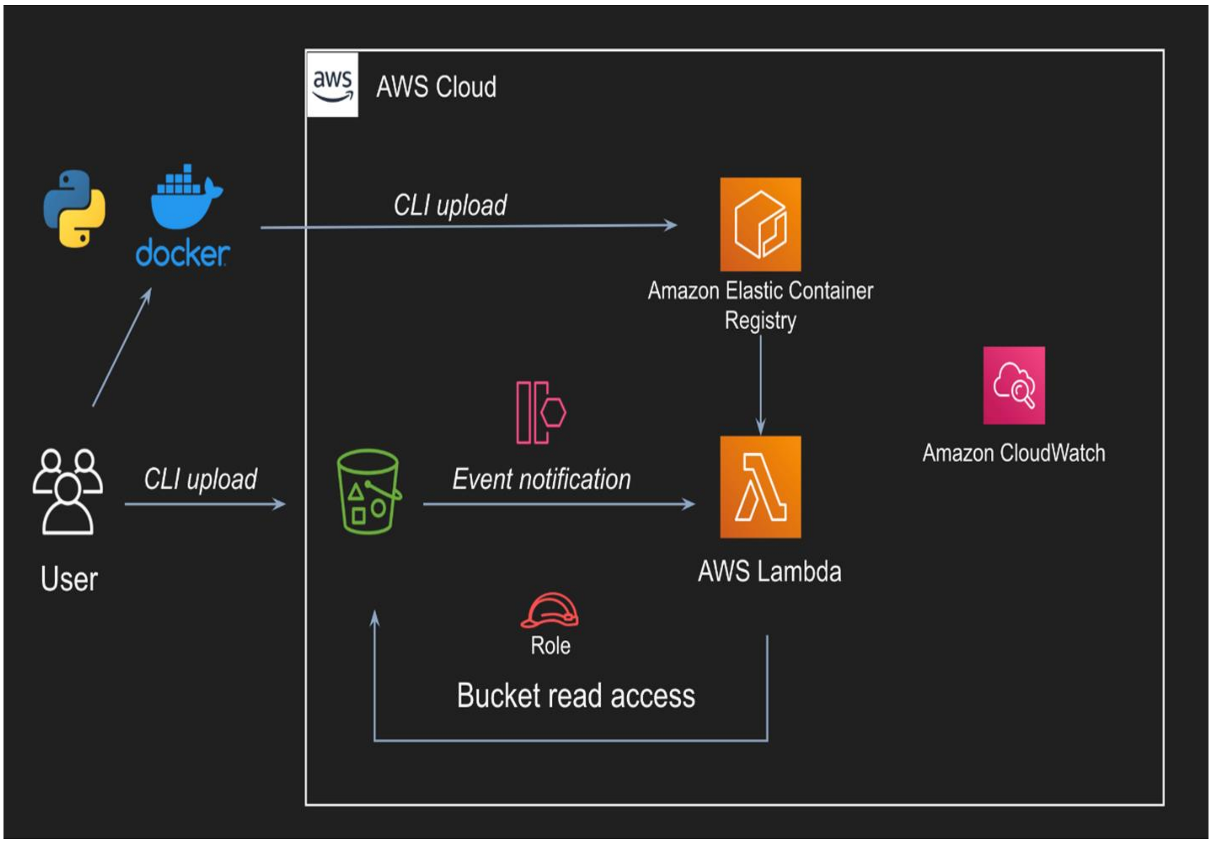


Figure No. 7.3 Deployment Process with Each AWS Service Connection

The following are descriptions of various AWS services.

7.1 Docker

Docker is a containerization technology that enables quick software development, testing, and release. Docker is a software packaging technology that places applications and their accompanying code, libraries, system tools, and runtime environments into portable containers. Docker enables the dependable, quick deployment and scalability of applications across a variety of infrastructures and deployment patterns. Docker on AWS enables the development, distribution, and operation of distributed applications of any size with ease and efficiency.

7.2 AWS Command Line Interface (AWS CLI)

Utilize the AWS CLI to centrally manage all of your AWS resources. After downloading and configuring a single tool, several AWS services can be managed and programmed from the command line using a single tool.

7.3 Amazon Simple Storage Service (S3)

Amazon's S3 is unparalleled with regards to scalability, data availability, security, and performance. Amazon S3 can be utilized by enterprises of all sizes and in a wide range of industries for a variety of reasons, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, Internet of Things (IoT) devices, and big data analytics. With Amazon S3, you can optimize, manage, and conFigure No. access to your data to fit the requirements of your business, organization, and any compliance standards applicable to your industry.

7.4 IAM role

To provide specific permissions to a user in your account, you must utilize IAM roles. As with IAM users, IAM roles are AWS identities with authorization policies that determine what actions the identity can and cannot perform within AWS. A job, however, is not intended to be permanently associated with a specific individual, but rather to be filled by whoever is in need of it. A job, on the other hand, lacks permanent credentials such as a password or a set of keys. Rather, it gives temporary security credentials that are job-specific whenever you assume a position.

7.5 Amazon Elastic Container Registry

Amazon Elastic Container Registry (Amazon ECR) is a service on Amazon Web Services that stores container images and is controlled, highly available, and scalable. ECR allows users to construct private repositories and apply resource-based permissions with AWS IAM. The goal is to limit who can access your container repositories and images, whether that be specific users or Amazon EC2 instances. You can use your favorite command line interface to publish, pull, and manage Docker images, Open Container Initiative (OCI) images, and OCI compatible artifacts.

7.6 AWS Lambda

As a computing service, Lambda enables the execution of code without the need to conFigure No. or manage underlying servers. In an always-on computing environment, Lambda executes your code and manages all compute resources on your behalf. This involves repairing and upgrading servers and operating systems, adding additional capacity as necessary, scaling up or down as necessary, and maintaining accurate logs. Using Lambda, almost any type of service or application backend can have its code performed. The only need is code, which can be submitted in any of the supported languages by Lambda.

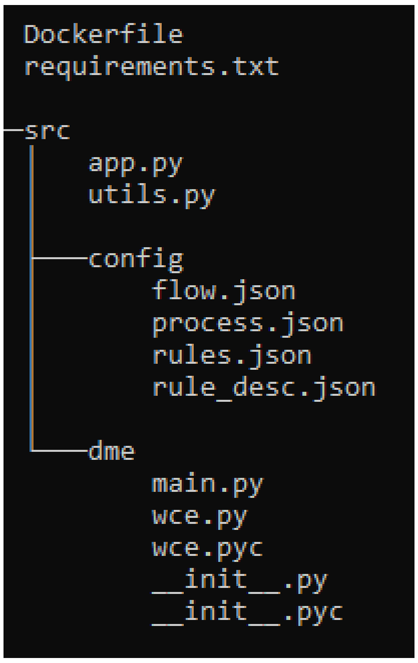
7.7 Amazon CloudWatch

Amazon CloudWatch enables real-time monitoring of your AWS infrastructure and any apps you've deployed there. With CloudWatch, metrics or quantitative data may be collected and monitored. Infrastructure components and software packages can be measured.

Chapter 8: Implementation

The steps required to implement the AI-enabled solution on the AWS cloud service are outlined below.

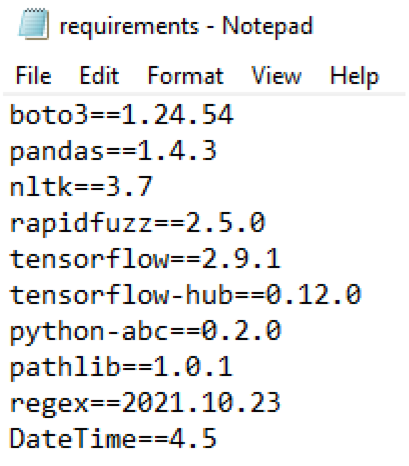
Step 1. To preapare the application for creating docker image. Figure No. 8.1 shows the tree of the folder.



```
Dockerfile
requirements.txt

─src
 │   app.py
 │   utils.py
 │
 ├───config
 │       flow.json
 │       process.json
 │       rules.json
 │       rule_desc.json
 │
 └───dme
         main.py
         wce.py
         wce.pyc
         __init__.py
         __init__.pyc
```

Figure No. 8.1 Application Directory Tree

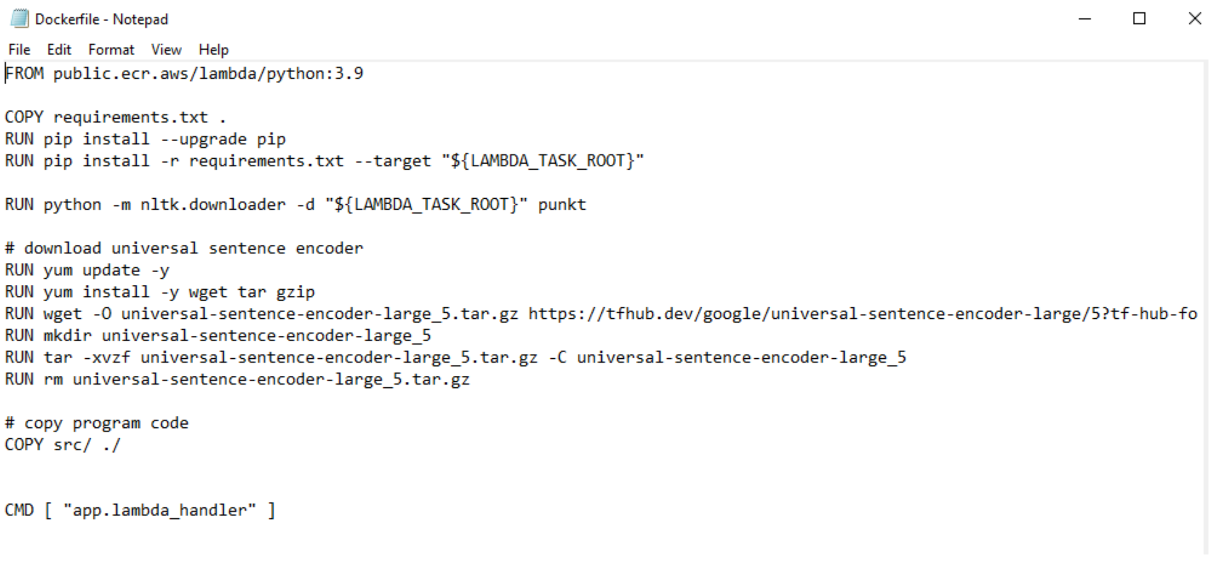
Step 2. To create dependencies file that is requirements.txt as shown in Figure No. 8.2.



```
boto3==1.24.54
pandas==1.4.3
nltk==3.7
rapidfuzz==2.5.0
tensorflow==2.9.1
tensorflow-hub==0.12.0
python-abc==0.2.0
pathlib==1.0.1
regex==2021.10.23
DateTime==4.5
```

Figure No. 8.2 Dependencies File

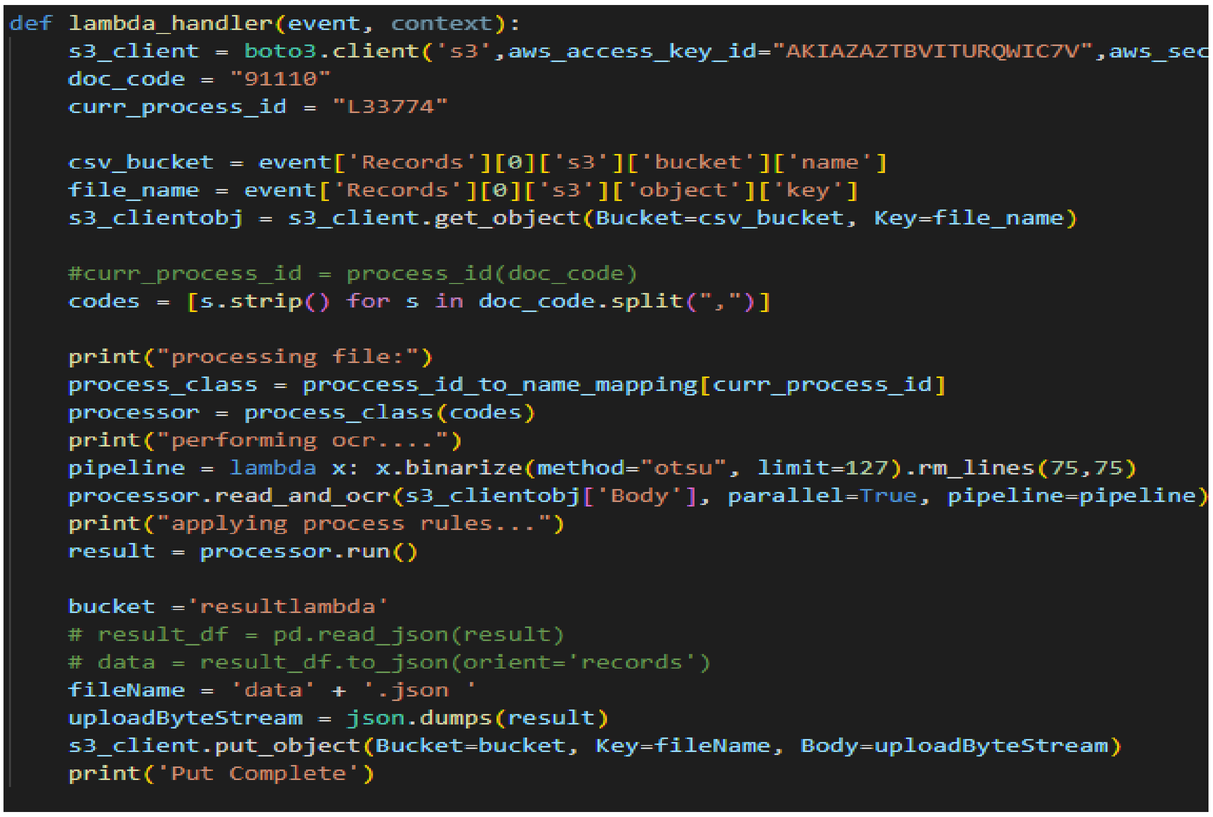
Step 3. To create docker file as shown in Figure No. 8.3.



```
FROM public.ecr.aws/lambda/python:3.9

COPY requirements.txt .
RUN pip install --upgrade pip
RUN pip install -r requirements.txt --target "${LAMBDA_TASK_ROOT}"

RUN python -m nltk.downloader -d "${LAMBDA_TASK_ROOT}" punkt

# download universal sentence encoder
RUN yum update -y
RUN yum install -y wget tar gzip
RUN wget -O universal-sentence-encoder-large_5.tar.gz https://tfhub.dev/google/universal-sentence-encoder-large/5?tf-hub-fo
RUN mkdir universal-sentence-encoder-large_5
RUN tar -xvzf universal-sentence-encoder-large_5.tar.gz -C universal-sentence-encoder-large_5
RUN rm universal-sentence-encoder-large_5.tar.gz

# copy program code
COPY src/ ./


CMD [ "app.lambda_handler" ]
```

Figure No. 8.3 Docker File

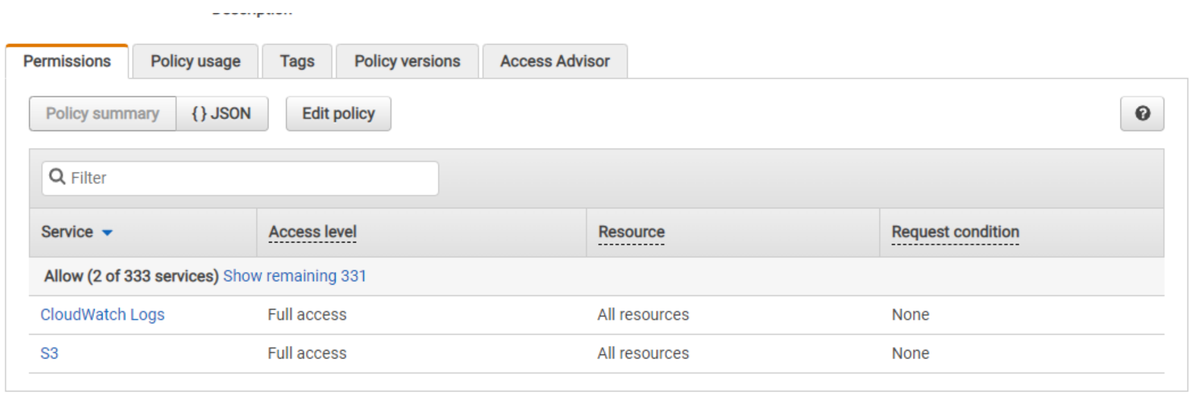
Step 4. To create lambda handler function for the application invocation as shown in Figure No. 8.4.



```
def lambda_handler(event, context):
    s3_client = boto3.client('s3',aws_access_key_id="AKIAZAZTBVITURQWIC7V",aws_sec
    doc_code = "91110"
    curr_process_id = "L33774"

    csv_bucket = event['Records'][0]['s3']['bucket']['name']
    file_name = event['Records'][0]['s3']['object']['key']
    s3_clientobj = s3_client.get_object(Bucket=csv_bucket, Key=file_name)

    #curr_process_id = process_id(doc_code)
    codes = [s.strip() for s in doc_code.split(",")]

    print("processing file:")
    process_class = proccess_id_to_name_mapping[curr_process_id]
    processor = process_class(codes)
    print("performing ocr....")
    pipeline = lambda x: x.binarize(method="otsu", limit=127).rm_lines(75,75)
    processor.read_and_ocr(s3_clientobj['Body'], parallel=True, pipeline=pipeline)
    print("applying process rules...")
    result = processor.run()

    bucket ='resultlambda'
    # result_df = pd.read_json(result)
    # data = result_df.to_json(orient='records')
    fileName = 'data' + '.json '
    uploadByteStream = json.dumps(result)
    s3_client.put_object(Bucket=bucket, Key=fileName, Body=uploadByteStream)
    print('Put Complete')
```

Figure No. 8.4 Lambda Handler Function Creation

Step 5. To creating IAM role with S3 and AWS cloudwatch logs full access as shown in Figure No. 8.5.



Service	Access level	Resource	Request condition
Allow (2 of 333 services) Show remaining 331			
CloudWatch Logs	Full access	All resources	None
S3	Full access	All resources	None

Figure No. 8.5 IAM Role with Policy

Step 6. To generate repository at ECR and pushing docker image as shown in Figure No. 8.6.

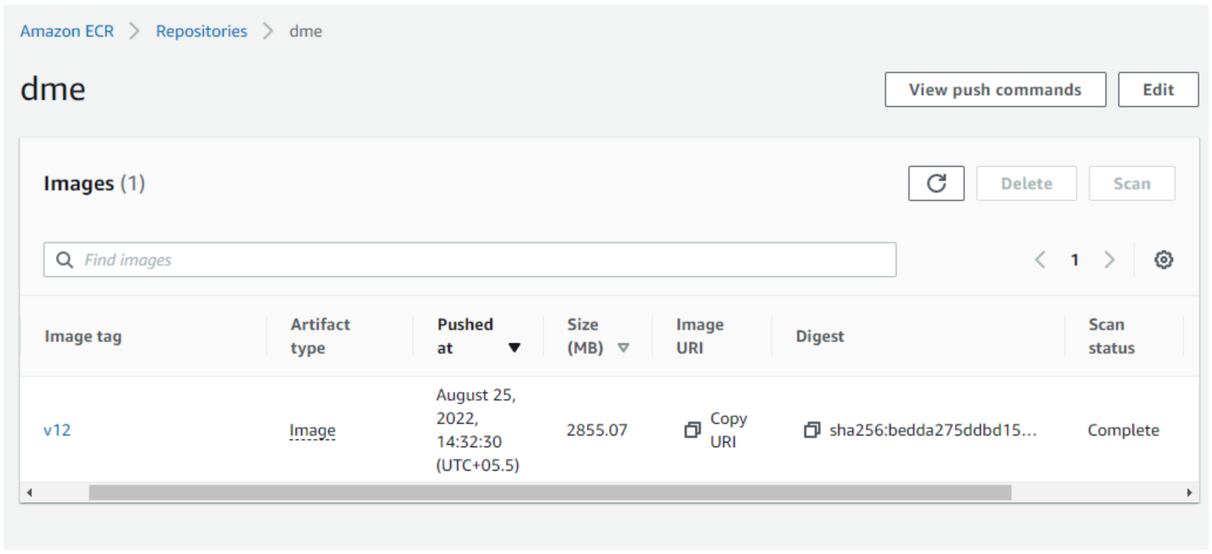
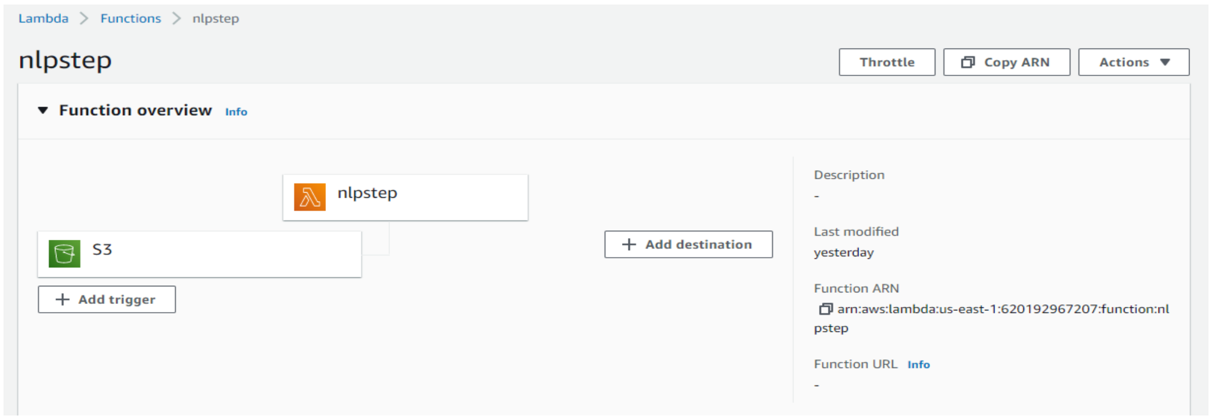


Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status
v12	Image	August 25, 2022, 14:32:30 (UTC+05.5)	2855.07	Copy URI	sha256:bedda275ddbd15...	Complete

Figure No. 8.6 Docker Image Pushed at ECR

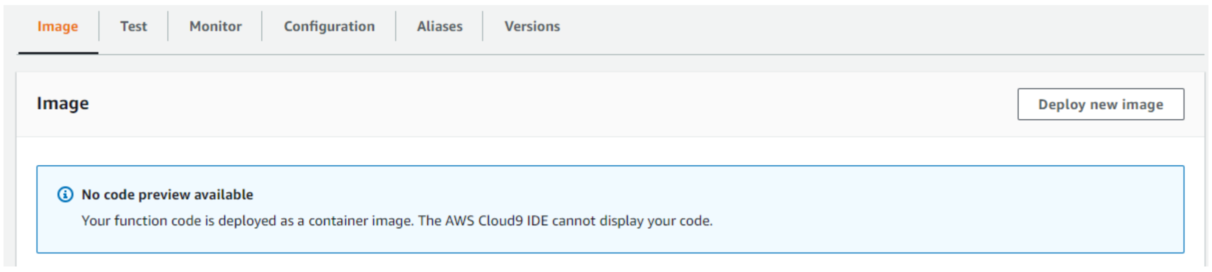
Step 7. To Create lambda function using docker image with Trigger point S3 as shown in Figure No. 8.7.



Description
-
Last modified
yesterday
Function ARN
arn:aws:lambda:us-east-1:620192967207:function:nlpstep
Function URL Info
-

Figure No. 8.7 AWS Lambda Function

Step 8. Now this study can proceed for testing of the function after clicking on deployment as shown in Figure No. 8.8.



No code preview available
Your function code is deployed as a container image. The AWS Cloud9 IDE cannot display your code.

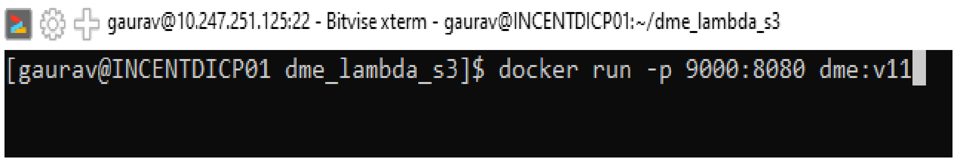
Figure No. 8.8 AWS Lambda Deployment Step

Chapter 9: Testing and validation

At this stage, this study needs to test and validate docker image as well as lambda function.

9.1 Docker image test:

1. The docker run command begins the Docker image as shown in Figure No. 9.1.

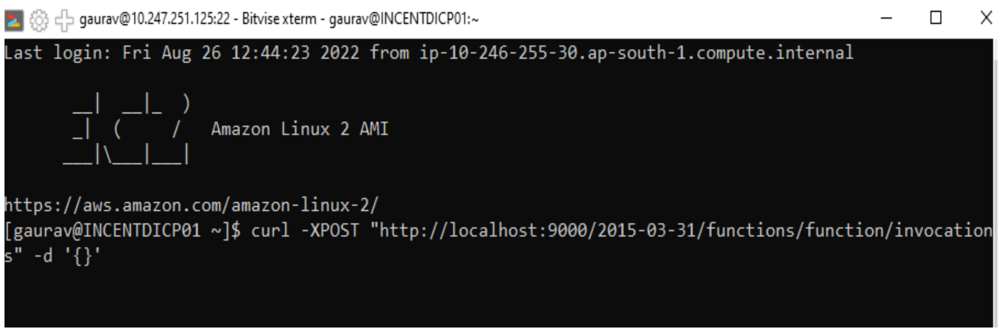


gaurav@10.247.251.125:22 - Bitvise xterm - gaurav@INCENTDICP01:~/dme_lambda_s3

```
[gaurav@INCENTDICP01 dme_lambda_s3]$ docker run -p 9000:8080 dme:v11
```

Figure No. 9.1 Docker RUN Command

2. You may test your app without having to deploy it by using the runtime interface emulator. Utilizing the curl command in a separate terminal, send the event to the following endpoint. Invoking the container image's function with this command yields a response similar to that seen in Figure No. 9.2.

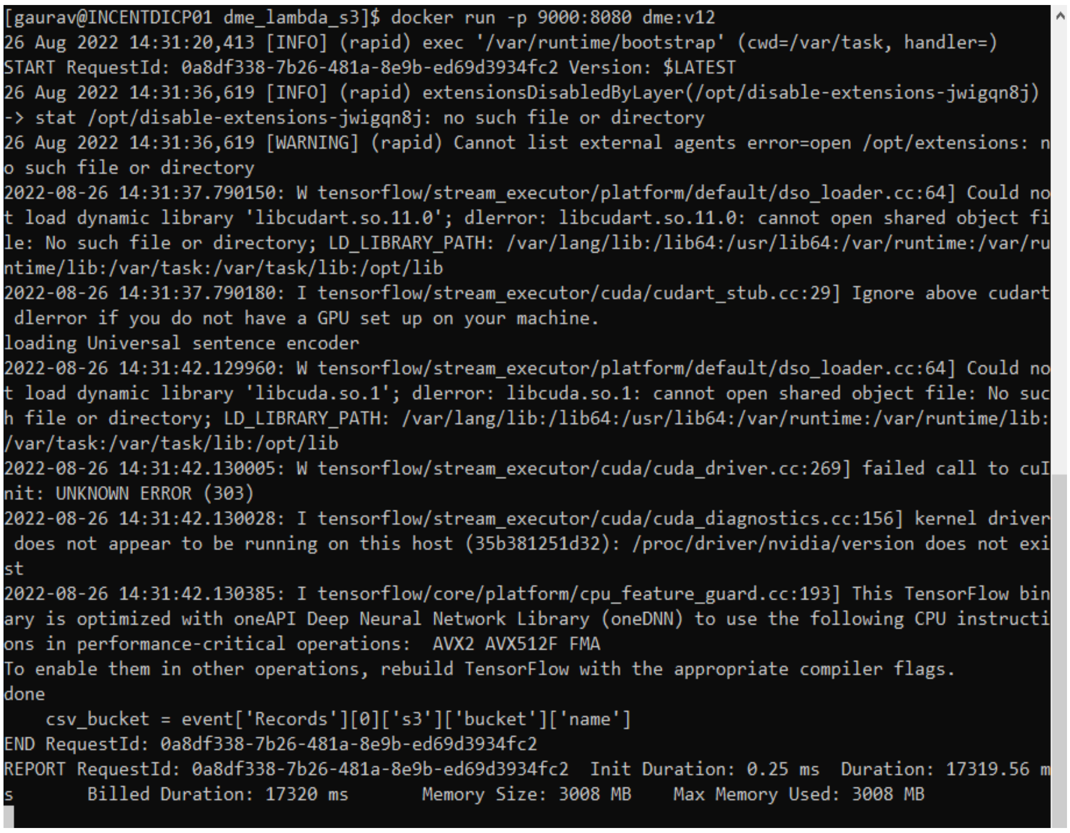


gaurav@10.247.251.125:22 - Bitvise xterm - gaurav@INCENTDICP01:~

```
Last login: Fri Aug 26 12:44:23 2022 from ip-10-246-255-30.ap-south-1.compute.internal

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[gaurav@INCENTDICP01 ~]$ curl -XPOST "http://localhost:9000/2015-03-31/functions/function/invocation
s" -d '{}'
```

Figure No. 9.2 Lambda Function Invocation Running in the Container

3. Test result is shown in Figure No. 9.3.

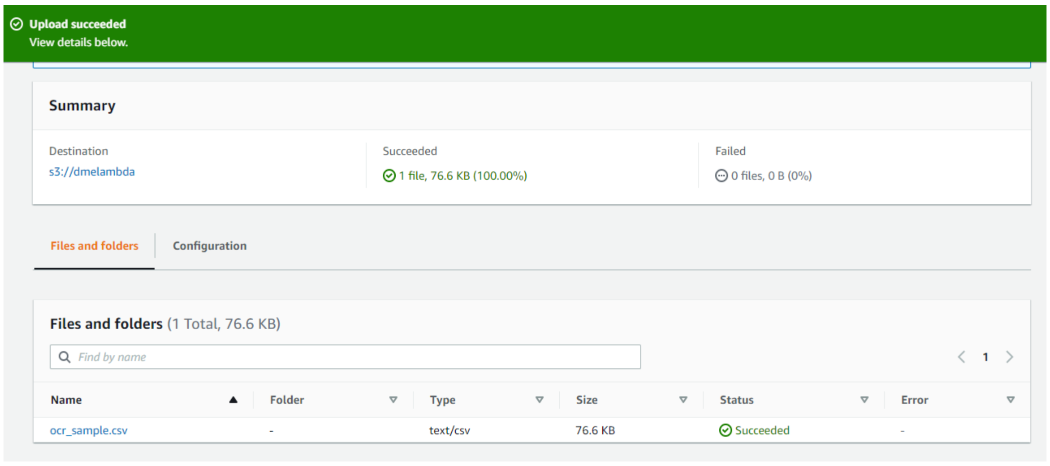


```
[gaurav@INCENTDICP01 dme_lambda_s3]$ docker run -p 9000:8080 dme:v12
26 Aug 2022 14:31:20,413 [INFO] (rapid) exec '/var/runtime/bootstrap' (cwd=/var/task, handler=)
START RequestId: 0a8df338-7b26-481a-8e9b-ed69d3934fc2 Version: $LATEST
26 Aug 2022 14:31:36,619 [INFO] (rapid) extensionsDisabledByLayer(/opt/disable-extensions-jwigqn8j)
-> stat /opt/disable-extensions-jwigqn8j: no such file or directory
26 Aug 2022 14:31:36,619 [WARNING] (rapid) Cannot list external agents error=open /opt/extensions: n
o such file or directory
2022-08-26 14:31:37.790150: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could no
t load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open shared object fi
le: No such file or directory; LD_LIBRARY_PATH: /var/lang/lib:/lib64:/usr/lib64:/var/runtime:/var/ru
ntime/lib:/var/task:/var/task/lib:/opt/lib
2022-08-26 14:31:37.790180: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
 dlerror if you do not have a GPU set up on your machine.
loading Universal sentence encoder
2022-08-26 14:31:42.129960: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could no
t load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object file: No suc
h file or directory; LD_LIBRARY_PATH: /var/lang/lib:/lib64:/usr/lib64:/var/runtime:/var/runtime/lib:
/var/task:/var/task/lib:/opt/lib
2022-08-26 14:31:42.130005: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuI
nit: UNKNOWN ERROR (303)
2022-08-26 14:31:42.130028: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver
 does not appear to be running on this host (35b381251d32): /proc/driver/nvidia/version does not exi
st
2022-08-26 14:31:42.130385: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow bin
ary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructi
ons in performance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
done
    csv_bucket = event['Records'][0]['s3']['bucket']['name']
END RequestId: 0a8df338-7b26-481a-8e9b-ed69d3934fc2
REPORT RequestId: 0a8df338-7b26-481a-8e9b-ed69d3934fc2  Init Duration: 0.25 ms  Duration: 17319.56 m
s       Billed Duration: 17320 ms       Memory Size: 3008 MB    Max Memory Used: 3008 MB
```

Figure No. 9.3 Docker Image Test Result

9.2 AWS Lambda Test:

1. Uploading the file at S3 bucket as shown in Figure No. 9.4.



Upload succeeded
View details below.

Summary

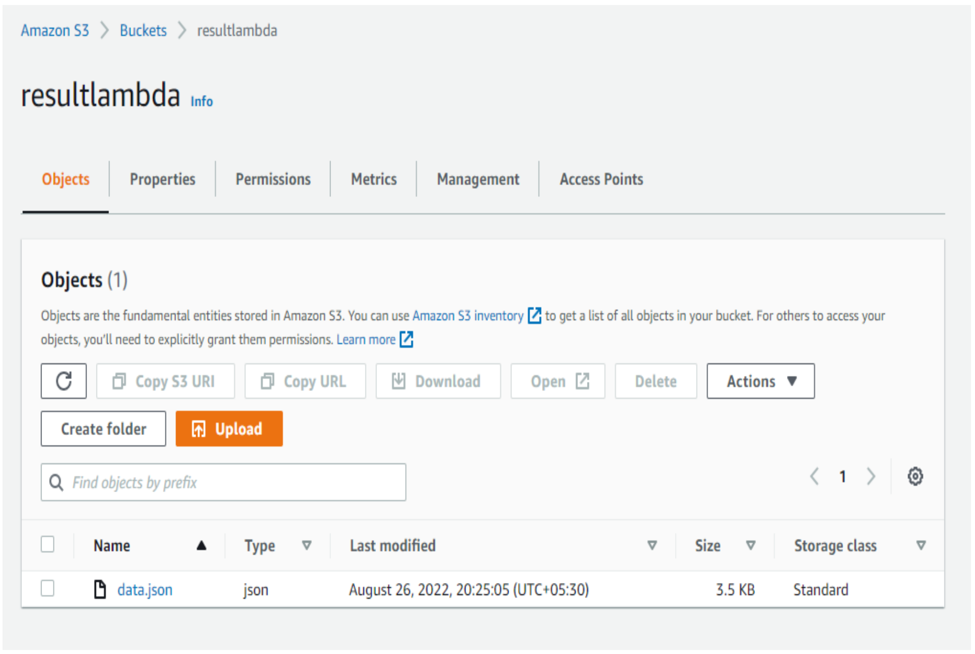
Destination	Succeeded	Failed
s3://dmelambda	1 file, 76.6 KB (100.00%)	0 files, 0 B (0%)

Files and folders (1 Total, 76.6 KB)

Name	Folder	Type	Size	Status	Error
ocr_sample.csv	-	text/csv	76.6 KB	Succeeded	-

Figure No. 9.4 File uploaded at Bucket S3

2. Result file at json format will get saved at S3 bucket as shown in Figure No. 9.5.



Amazon S3 > Buckets > resultlambda

resultlambda Info

Objects | Properties | Permissions | Metrics | Management | Access Points

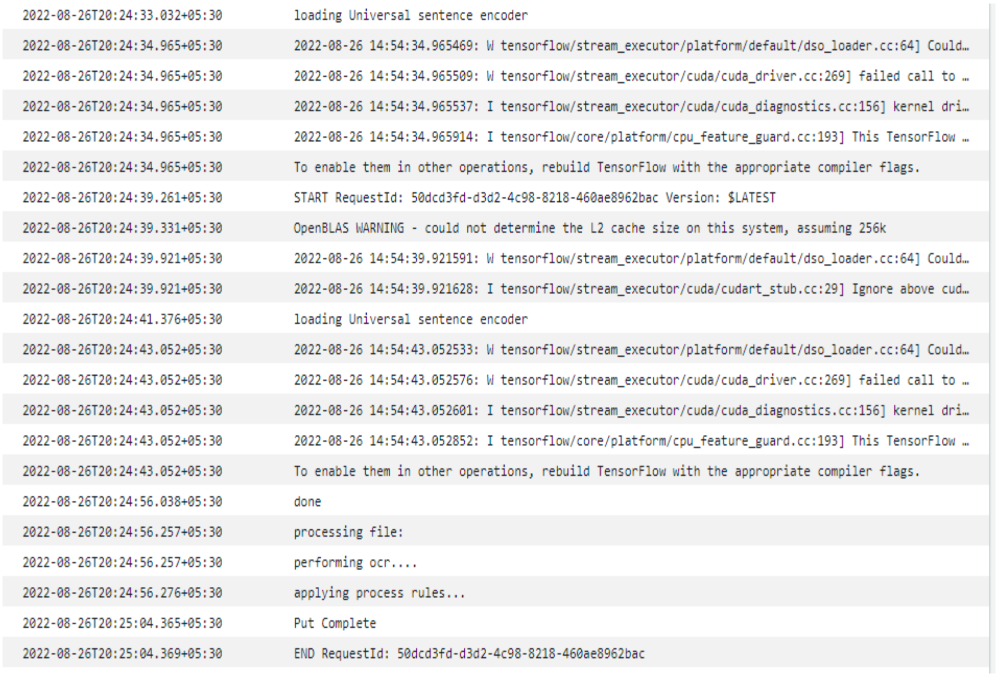
Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

	Name ▲	Type	Last modified	Size	Storage class
☐	data.json	json	August 26, 2022, 20:25:05 (UTC+05:30)	3.5 KB	Standard

Figure No. 9.5 Result File Uploaded at S3 Bucket as Json Format

3. CloudWatch logs shows each logs of each event as shown in Figure No. 9.6.



```
2022-08-26T20:24:33.032+05:30    loading Universal sentence encoder
2022-08-26T20:24:34.965+05:30    2022-08-26 14:54:34.965469: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could…
2022-08-26T20:24:34.965+05:30    2022-08-26 14:54:34.965509: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to …
2022-08-26T20:24:34.965+05:30    2022-08-26 14:54:34.965537: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel dri…
2022-08-26T20:24:34.965+05:30    2022-08-26 14:54:34.965914: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow …
2022-08-26T20:24:34.965+05:30    To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-08-26T20:24:39.261+05:30    START RequestId: 50dcd3fd-d3d2-4c98-8218-460ae8962bac Version: $LATEST
2022-08-26T20:24:39.331+05:30    OpenBLAS WARNING - could not determine the L2 cache size on this system, assuming 256k
2022-08-26T20:24:39.921+05:30    2022-08-26 14:54:39.921591: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could…
2022-08-26T20:24:39.921+05:30    2022-08-26 14:54:39.921628: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cud…
2022-08-26T20:24:41.376+05:30    loading Universal sentence encoder
2022-08-26T20:24:43.052+05:30    2022-08-26 14:54:43.052533: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could…
2022-08-26T20:24:43.052+05:30    2022-08-26 14:54:43.052576: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to …
2022-08-26T20:24:43.052+05:30    2022-08-26 14:54:43.052601: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel dri…
2022-08-26T20:24:43.052+05:30    2022-08-26 14:54:43.052852: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow …
2022-08-26T20:24:43.052+05:30    To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-08-26T20:24:56.038+05:30    done
2022-08-26T20:24:56.257+05:30    processing file:
2022-08-26T20:24:56.257+05:30    performing ocr....
2022-08-26T20:24:56.276+05:30    applying process rules...
2022-08-26T20:25:04.365+05:30    Put Complete
2022-08-26T20:25:04.369+05:30    END RequestId: 50dcd3fd-d3d2-4c98-8218-460ae8962bac
```

Figure No. 9.6 CloudWatch Logs

Chapter 10: Analysis and Results

In this study, the CloudWatch Metrics listed below are used for analysis.

10.1 Invocations

Tally of all unsuccessful lambda function calls. If the request for information about the invocation is throttled or fails, no information about the invocation will be retained. Therefore, this equals the total number of requests for which payment was made.

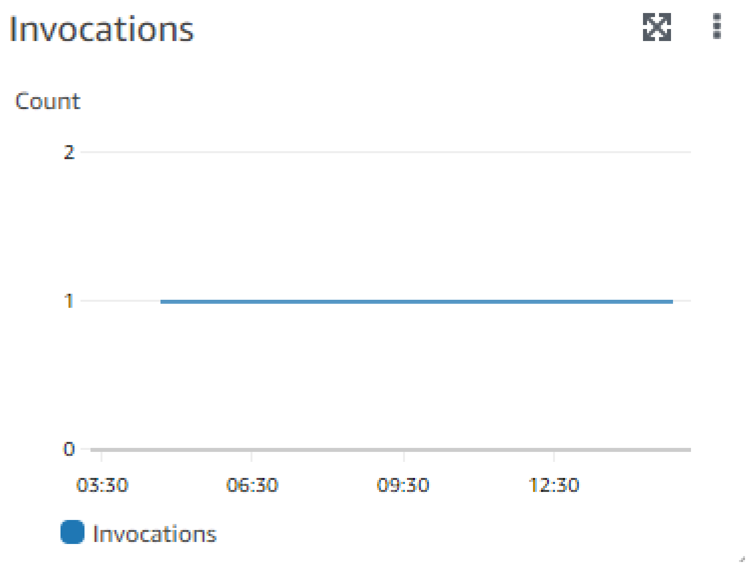


Figure No. 10.1 Invocation Metric

10.2 Duration

Time spent in milliseconds while your function code dealt with an event. Time spent in an invocation is measured in milliseconds and rounded up to the next whole number as shown in Figure No. 10.2.

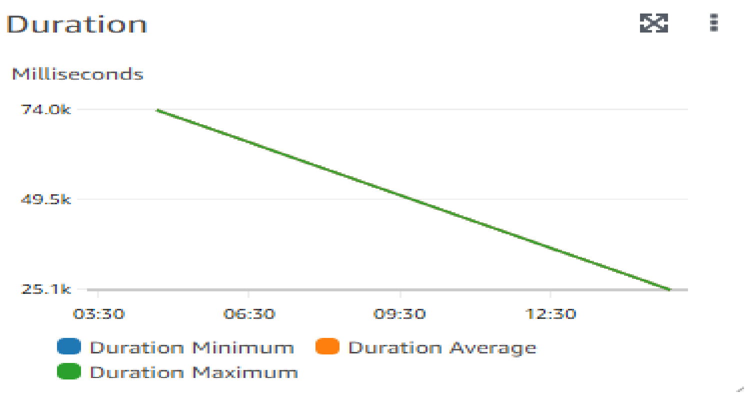


Figure No. 10.2 Duration Metric

10.3 Error count and success rate

The total number of times a function call failed. It's possible for both your code and the Lambda runtime to generate errors within a function. When problems occur during the execution, such as timeouts or incorrect setup settings, errors are reported. Just divide Errors by Invocations to get the error rate. Despite popular belief, the timestamp on an error metric indicates not the time the problem occurred but the time the function was called as shown in Figure No. 10.3.

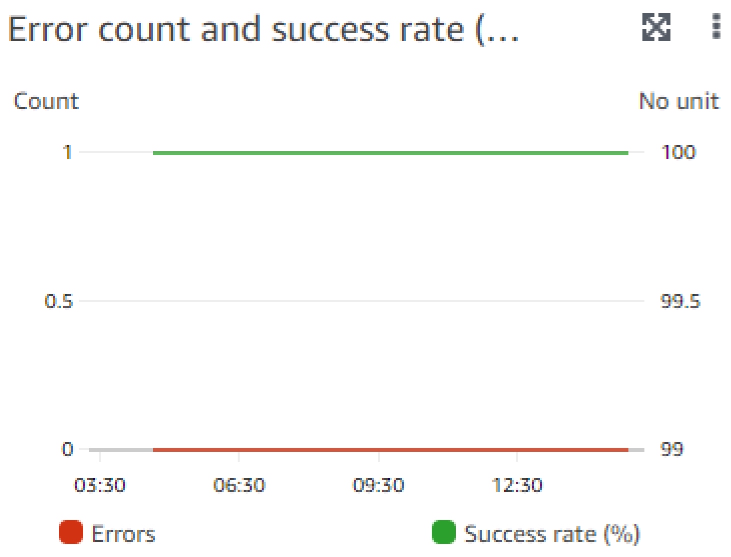


Figure No. 10.3 Error Count and Success Rate Metric

Chapter 11: Conclusions and Future Scope

The following advantages are gained by using MLOps methods, which in turn expedite the time to market for ML projects.

1. **Productivity:** Data engineers and data scientists can work more quickly and spend less time fixing inaccurate or incomplete data if they have access to curated data sets in a self-service environment.

2. **Reproducibility:** By automating the MLDC, you can guarantee a reproducible process for training, evaluating, and deploying the model.

3. **Reliability:** Implementing CI/CD best practices not only speeds up deployment times, but also improves the quality and consistency of such deployments.

4. **Auditability:** It is important to keep track of versions of everything related to data science, including experiments, raw data, and the trained model, so that it can be shown how each element contributed to the final result and where it was deployed.

5. **Data and model quality:** MLOps allows us to monitor the statistical features of our data and the quality of our models over time, as well as implement restrictions designed to prevent model bias.

A greater number of machine learning solutions are being created than ever before as a result of the increasing availability of data, the enhancement of analytical tools, and the continual need to incorporate fresh concepts. Nevertheless, relatively few of these proofs of concept are utilized and manufactured. In practice, data scientists continue to handle ML operations manually. MLOps is a novel paradigm capable of handling such situations.

Future Scope

To explore feature mentioned below:

Termination Policy – API response consistent will not go for loop

Python Profiling – Performance can get increase from less resources

Bibliography

[1] N. Omont and M. Treveil, *Introducing MLOps [Book]*. O'Reilly Media, 2020.

[2] "Why Should You Use MLOps? - Amazon SageMaker." https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-projects-why.html (accessed Aug. 26, 2022).

[3] M. Aykol, P. Herring, and A. Anapolsky, "Machine learning for continuous innovation in battery technologies," *Nat. Rev. Mater. 2020 510*, vol. 5, no. 10, pp. 725–727, Jun. 2020, doi: 10.1038/s41578-020-0216-y.

[4] M. Pandey and S. S. Rautaray, Eds., "Machine Learning: Theoretical Foundations and Practical Applications," vol. 87, 2021, doi: 10.1007/978-981-33-6518-6.

[5] R. Kocielnik, S. Amershi, and P. N. Bennett, "Will you accept an imperfect AI? Exploring Designs for Adjusting End-user Expectations of AI Systems," *Conf. Hum. Factors Comput. Syst. - Proc.*, May 2019, doi: 10.1145/3290605.3300641.

[6] "Gartner Says Nearly Half of CIOs Are Planning to Deploy Artificial Intelligence." https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence (accessed Aug. 26, 2022).

[7] A. Posoldova, "Machine Learning Pipelines: From Research to Production," *IEEE Potentials*, vol. 39, no. 6, pp. 38–42, Nov. 2020, doi: 10.1109/MPOT.2020.3016280.

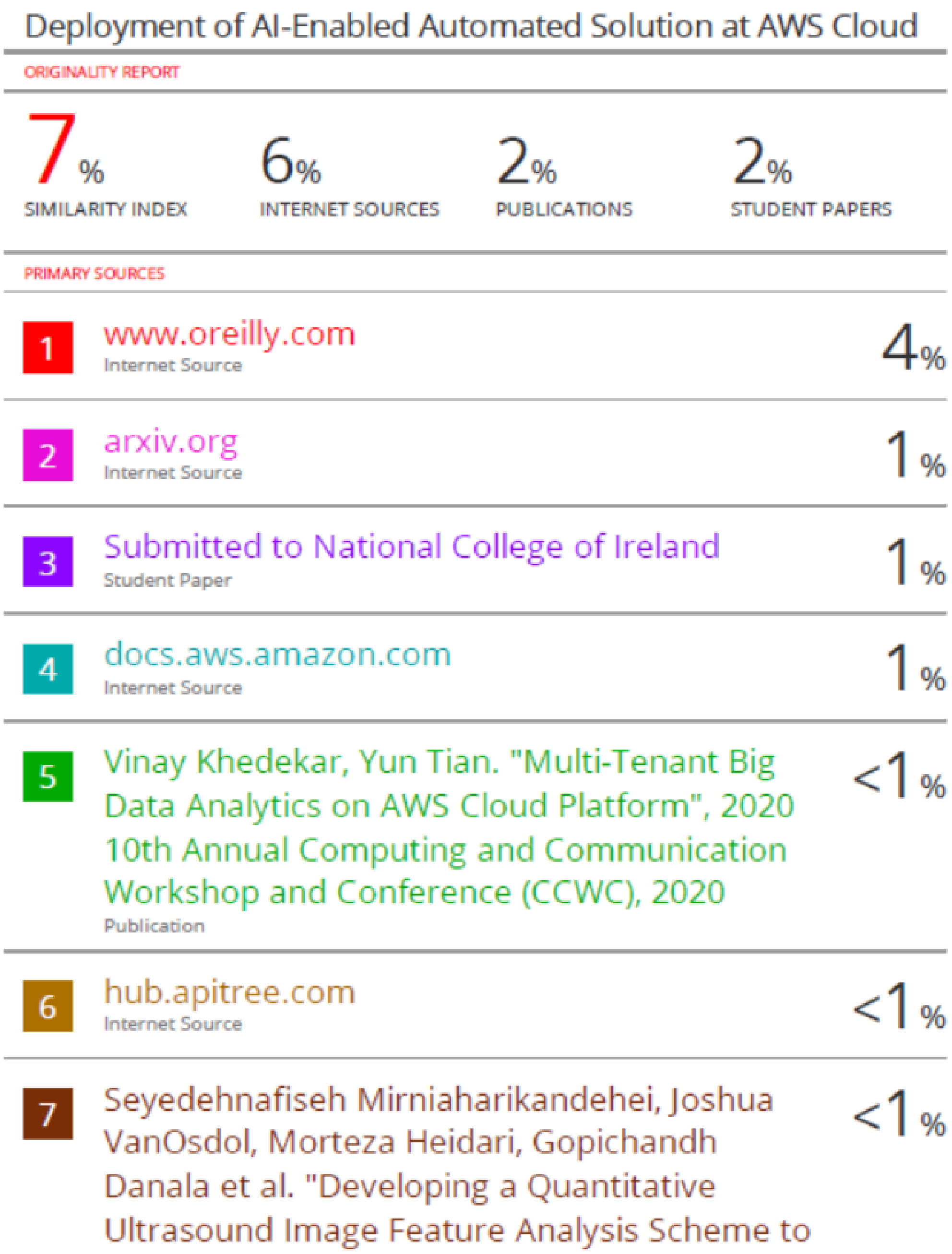
[8] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," 2022, [Online]. Available: http://arxiv.org/abs/2205.02302.

[9] N. Gift and A. Deza, *Practical MLOps*, vol. 1. 2021.

[10] "MLOps foundation roadmap for enterprises with Amazon SageMaker | AWS Machine Learning Blog." https://aws.amazon.com/blogs/machine-learning/mlops-foundation-roadmap-for-enterprises-with-amazon-sagemaker/ (accessed Aug. 26, 2022).

Appendix

Plagiarism Report[1]



Deployment of AI-Enabled Automated Solution at AWS Cloud

ORIGINALITY REPORT

7%	6%	2%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.oreilly.com Internet Source	4%
2	arxiv.org Internet Source	1%
3	Submitted to National College of Ireland Student Paper	1%
4	docs.aws.amazon.com Internet Source	1%
5	Vinay Khedekar, Yun Tian. "Multi-Tenant Big Data Analytics on AWS Cloud Platform", 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), 2020 Publication	<1%
6	hub.apitree.com Internet Source	<1%
7	Seyedehnafiseh Mirniaharikandehei, Joshua VanOsdol, Morteza Heidari, Gopichandh Danala et al. "Developing a Quantitative Ultrasound Image Feature Analysis Scheme to	<1%

[1] Turnitn report to be attached from the University.

Assess Tumor Treatment Efficacy Using a Mouse Model", Scientific Reports, 2019
Publication

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		