National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Moiz Ahmed Nurul Hasan Khan
Student ID: 22201785

School of Computing
National College of Ireland

Supervisor: Prof Aqeel Kazmi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Moiz Ahmed Nurul Hasan Khan
Student ID:	x22201785
Programme:	MSc in Cloud Computing **Year:** 2023-2024
Module:	Research in Computing
Lecturer:	Prof Aqeel Kazmi
Submission Due Date:	16th September 2024
Project Title:	Minimizing Cold Starts in Serverless Environments with Predictive Optimization Approach Using Bi-LSTM and Genetic Algorithms
Word Count:	959 **Page Count:** 10

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: Moiz Ahmed Nurul Hasan Khan

Date: 16th 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):	

Configuration Manual

Moiz Ahmed Nurul Hasan Khan
Student ID: 22201785

1 Introduction

In this guidebook you will see how to deploy and run an optimized cold start solution in a stateless serverless environment using AWS Lambda. The primary goal is to apply the Bi-LSTM approach for building a machine learning algorithm. In order to overcome the cold start problem described, genetic algorithm is introduced to optimize the handling of Lambda function invocation. A dashboard is also provided to monitor the machine learning model and its training in the serverless mode of an application with a user-friendly interface. Through this guide, users will be able to understand how to configure, run and manage the system.

2 System Configuration

This section outlines the requirements of Hardware and Software.

2.1 Hardware Requirements

- **Brand and Model:** MacBook Pro or Dell Precision
- **Processor:** Apple M2 Max or Intel Core i7/i9 processor
- **Memory:** 16 GB RAM to handle data processing
- **Storage:** 250 GB of SSD storage for quick access to the files and data processing.
- **Internet Connection:** Stable and fast internet connection for accessing cloud services.

2.2 Software requirements

- **Operating System:** Windows 10/11, macOS, or Linux
- **Programming Languages:** Python 3, JavaScript
- **Machine Learning and Data Processing Libraries:** TensorFlow, NumPy, Pandas, Matplotlib, Scikit-learn.
- **AWS Account for deploying and managing serverless architecture**
- **Integrated Development Environment (IDE):** VS code or Jupiter Notebook

3 Project Implementation

3.1 Environment setup

3.1.1 Setup AWS Account

- Create or Login to your AWS account to access the services such as Lambda, S3, Sagemaker and CloudWatch

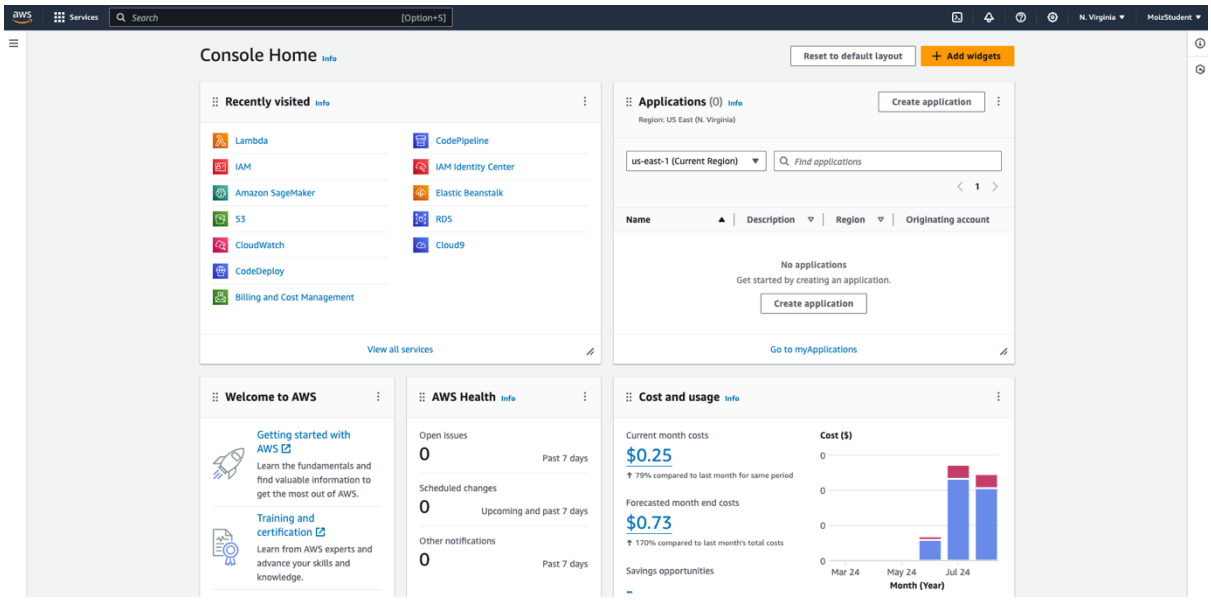

Figure 1: Creation of AWS Account

3.2 Packages & Libraries Used

- **NumPy:** For math operations and array manipulation.
- **Pandas:** For data handling and analysis.
- **TensorFlow/ Keras:** For building and training neural network models.
- **Matplotlib/Seaborn:** For graph and creating visualizations.
- **Scikit-learn:** For machine learning tasks like data pre-processing and evaluation.
- **Boto3:** AWS SDK for Python
- **Sequential, Dense, LSTM, Bi-Directional**: Components and layers in Keras for building neural network architectures.

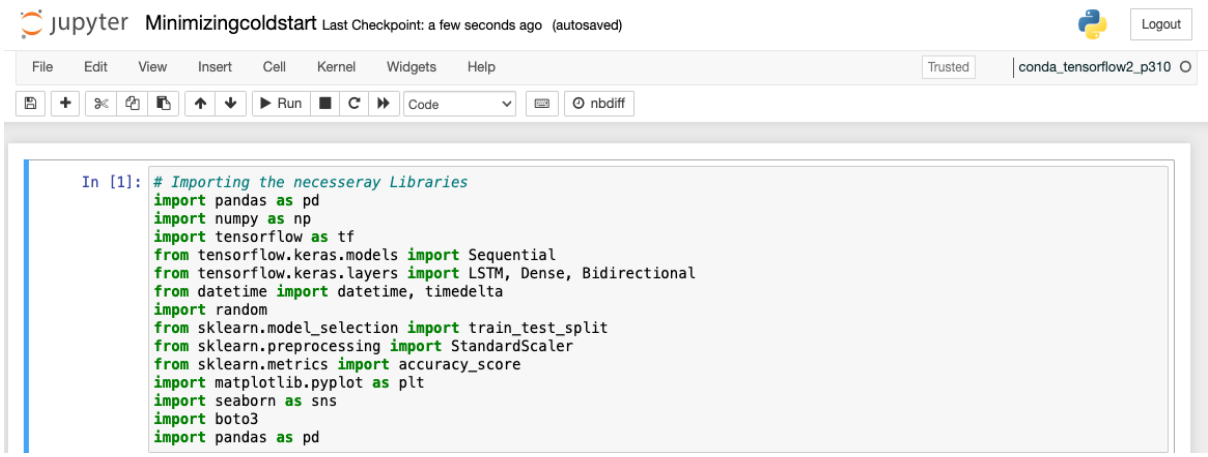

Figure 2: Importing the Libraries

4 Phases

This section has the detailed Implementation methodology for the project

4.1 Data Generation

- The Dataset used in this study is system generated as the machine learning mode is trained on the real invocations, so the study utilizes a synthetic dataset.

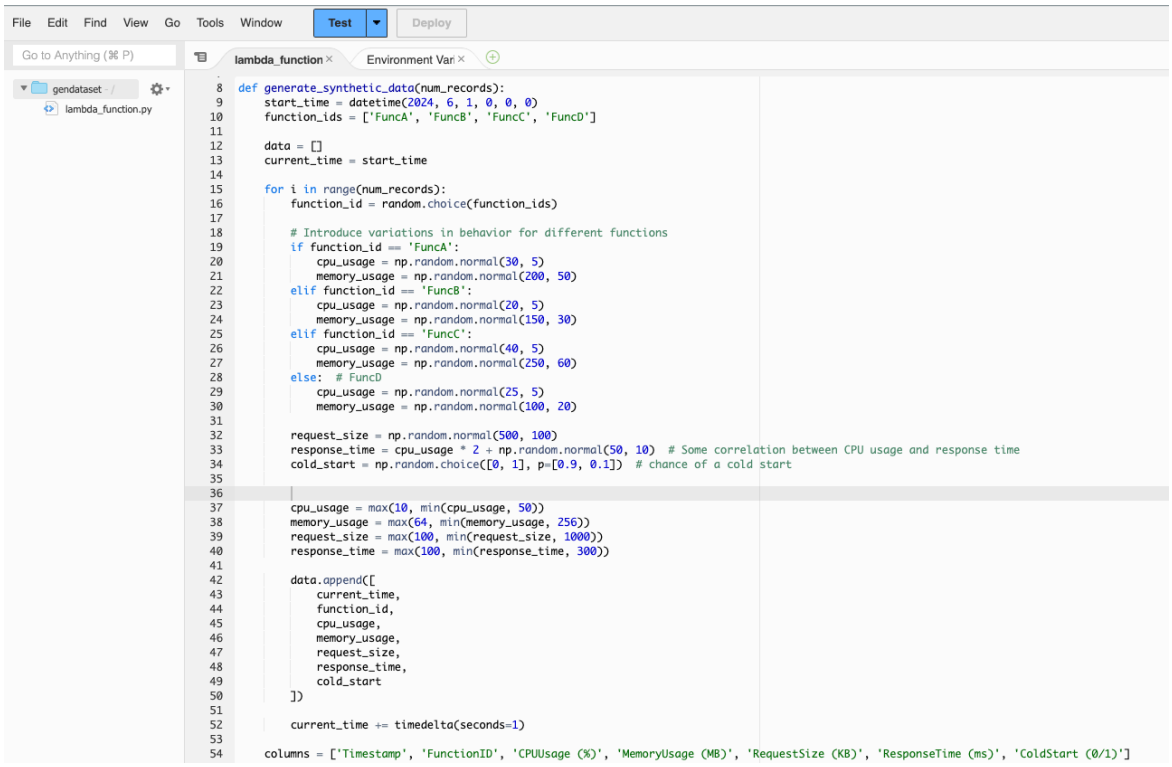

Figure 3: Synthetic Dataset Generation

- Dataset that is generated is stored in the S3 bucket named as “historical.csv” as showed in Figure 4.

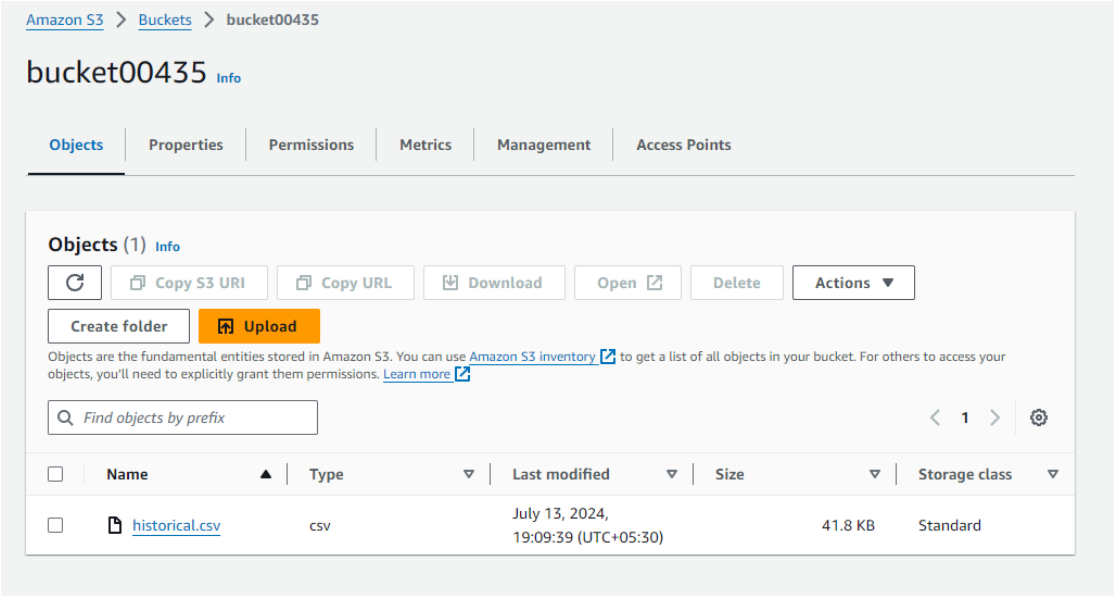

Figure 4: Dataset Stored Location

- From S3 bucket csv file is being downloaded in Amazon Sagemaker in Figure 5.

```
In [2]: import boto3
        import pandas as pd

        s3_client = boto3.client('s3')
        bucket_name = 'bucket00435'
        file_key = 'historical.csv'

        # Download the file from S3
        s3_client.download_file(bucket_name, file_key, 'historical.csv')

        # Load the data into a pandas DataFrame
        df = pd.read_csv('historical.csv')
        print(df.head())

                Timestamp FunctionID  CPUUsage (%)  MemoryUsage (MB)  \
0  2024-06-01 00:00:00      FuncA     30.561682        227.286844
1  2024-06-01 00:00:01      FuncA     29.381101        199.124056
2  2024-06-01 00:00:02      FuncA     34.583019        110.234565
3  2024-06-01 00:00:03      FuncC     45.594580        256.000000
4  2024-06-01 00:00:04      FuncC     43.940998        256.000000

   RequestSize (KB)  ResponseTime (ms)  ColdStart (0/1)
0        422.467181         106.783783                0
1        567.574967         105.540158                0
2        562.496880         121.818300                0
3        472.139546         138.608779                0
4        714.367864         150.896196                0
```

Figure 5: Dataset Information

4.2 Data Pre-processing

- Loading the dataset from S3 into a Pandas DataFrame to perform pre-processing. Perform cleaning, removing outliers and transformations to prepare the data for model training.

```
In [12]: #outlier Detection
import matplotlib.pyplot as plt
import seaborn as sns

# Box plot for each numerical column
plt.figure(figsize=(15, 10))
sns.boxplot(data=df[['CPUUsage (%)', 'MemoryUsage (MB)', 'RequestSize (KB)', 'ResponseTime (ms)']])
plt.show()
```

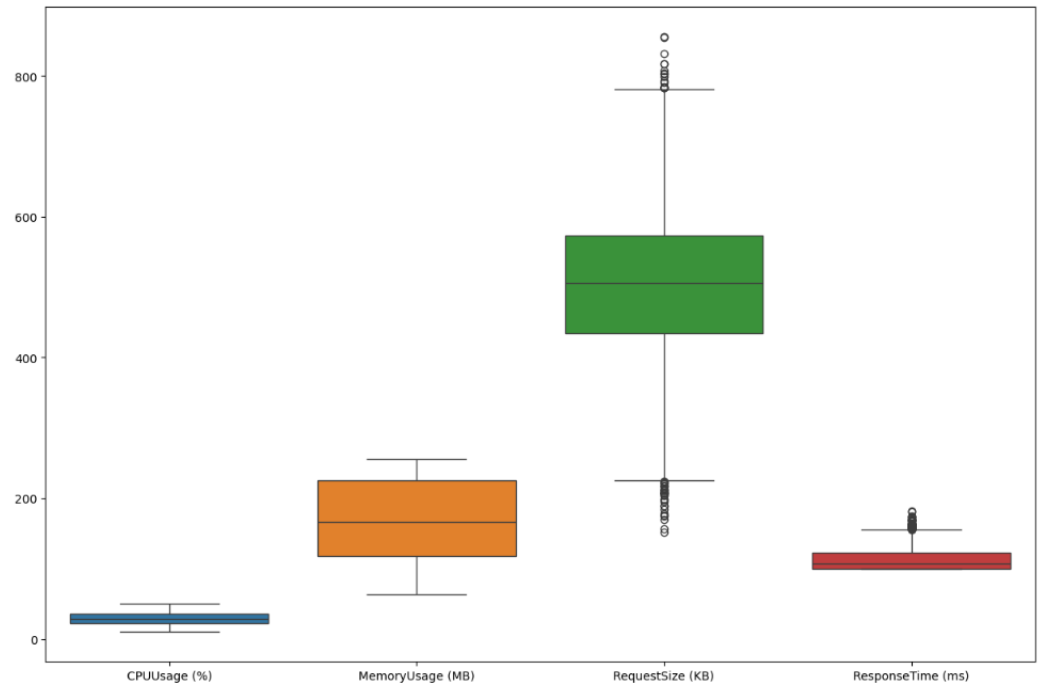

Figure 6: Outlier Detection and Cleaning

```
In [16]: # Identify outliers
outliers = (df[['CPUUsage (%)', 'MemoryUsage (MB)', 'RequestSize (KB)', 'ResponseTime (ms)']] < (Q1 - 1.5 * IQR)) |
df_outliers = df[outliers.any(axis=1)]
print(df_outliers.head(3))
```

```
                     FunctionID  CPUUsage (%)  MemoryUsage (MB)  \
Timestamp
2024-06-01 00:00:19      FuncC     50.000000             256.0
2024-06-01 00:01:28      FuncC     46.738806             256.0
2024-06-01 00:03:08      FuncC     50.000000             256.0

                     RequestSize (KB)  ResponseTime (ms)  ColdStart (0/1)
Timestamp
2024-06-01 00:00:19         493.808821         167.073657                1
2024-06-01 00:01:28         691.935942         156.010319                1
2024-06-01 00:03:08         463.646208         161.613317                0
```

Figure 7: Identified Outlier

4.3 Training Dummy Machine Learning Model

- Run the code in the Jupiter notebook named “Minimizingcoldstart.ipynb” in AWS Sagemaker for the results.
- Ensure the accuracy of the dummy model and generation of “initial_results.csv” in your s3 bucket

```
In [32]: # Build the model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Bidirectional, LSTM, Dense
model = Sequential()
model.add(Bidirectional(LSTM(50, return_sequences=True), input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Bidirectional(LSTM(50)))
model.add(Dense(1, activation='relu'))
optimizer = Adam(clipvalue=1.0)
model.compile(optimizer=Adam(learning_rate=0.01), loss='binary_crossentropy', metrics=['accuracy'])
```

```
2024-08-08 22:49:42.477424: E external/local_xla/xla/stream_executor/cuda/cuda_driver.cc:282] failed call to cuIni
t: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
```

```
In [33]: # Train the model
history = model.fit(X_train, y_train, epochs=8, batch_size=10, validation_data=(X_test, y_test))
```

```
Epoch 1/8
560/560 [==============================] - 37s 12ms/step - loss: 0.9137 - accuracy: 0.8989 - val_loss: 1.0099 - val
_accuracy: 0.9064
Epoch 2/8
560/560 [==============================] - 4s 7ms/step - loss: 0.4112 - accuracy: 0.9013 - val_loss: 0.3194 - val_a
ccuracy: 0.9064
Epoch 3/8
560/560 [==============================] - 3s 6ms/step - loss: 0.3232 - accuracy: 0.9023 - val_loss: 0.3144 - val_a
ccuracy: 0.9064
Epoch 4/8
560/560 [==============================] - 4s 6ms/step - loss: 0.3224 - accuracy: 0.9023 - val_loss: 0.3137 - val_a
ccuracy: 0.9064
Epoch 5/8
560/560 [==============================] - 4s 7ms/step - loss: 0.3225 - accuracy: 0.9023 - val_loss: 0.3200 - val_a
ccuracy: 0.9064
Epoch 6/8
560/560 [==============================] - 4s 6ms/step - loss: 0.3232 - accuracy: 0.9023 - val_loss: 0.3231 - val_a
ccuracy: 0.9064
Epoch 7/8
560/560 [==============================] - 4s 7ms/step - loss: 0.3225 - accuracy: 0.9023 - val_loss: 0.3213 - val_a
ccuracy: 0.9064
Epoch 8/8
560/560 [==============================] - 4s 7ms/step - loss: 0.3319 - accuracy: 0.9023 - val_loss: 0.6732 - val_a
ccuracy: 0.9064
```

```
44/44 [==============================] - 2s 2ms/step
Model Accuracy: 90.64%
```

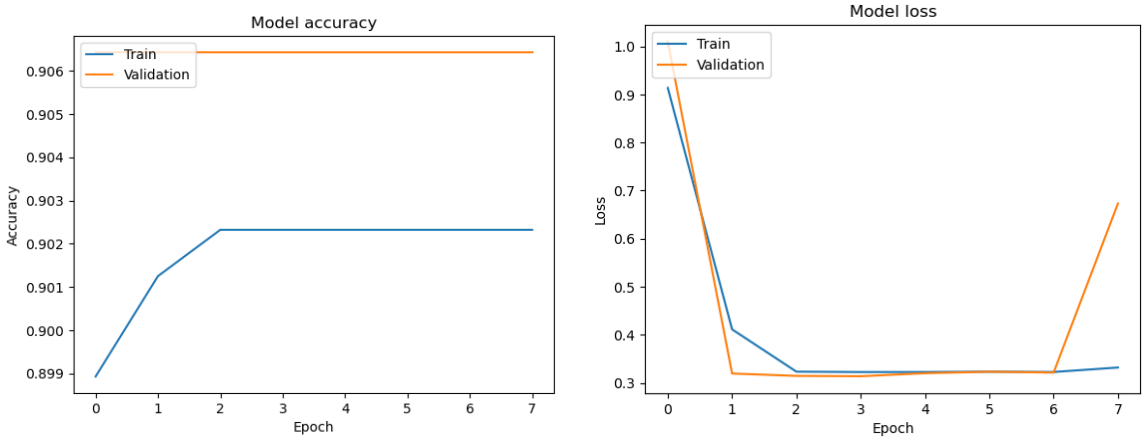

Figure 8: Model Training

4.4 Train Optimized Machine Learning Model

- Browse recent invocations for the “gendata” lambda functions to manually save it as csv
- View it in CloudWatch logs and Run the query Figure

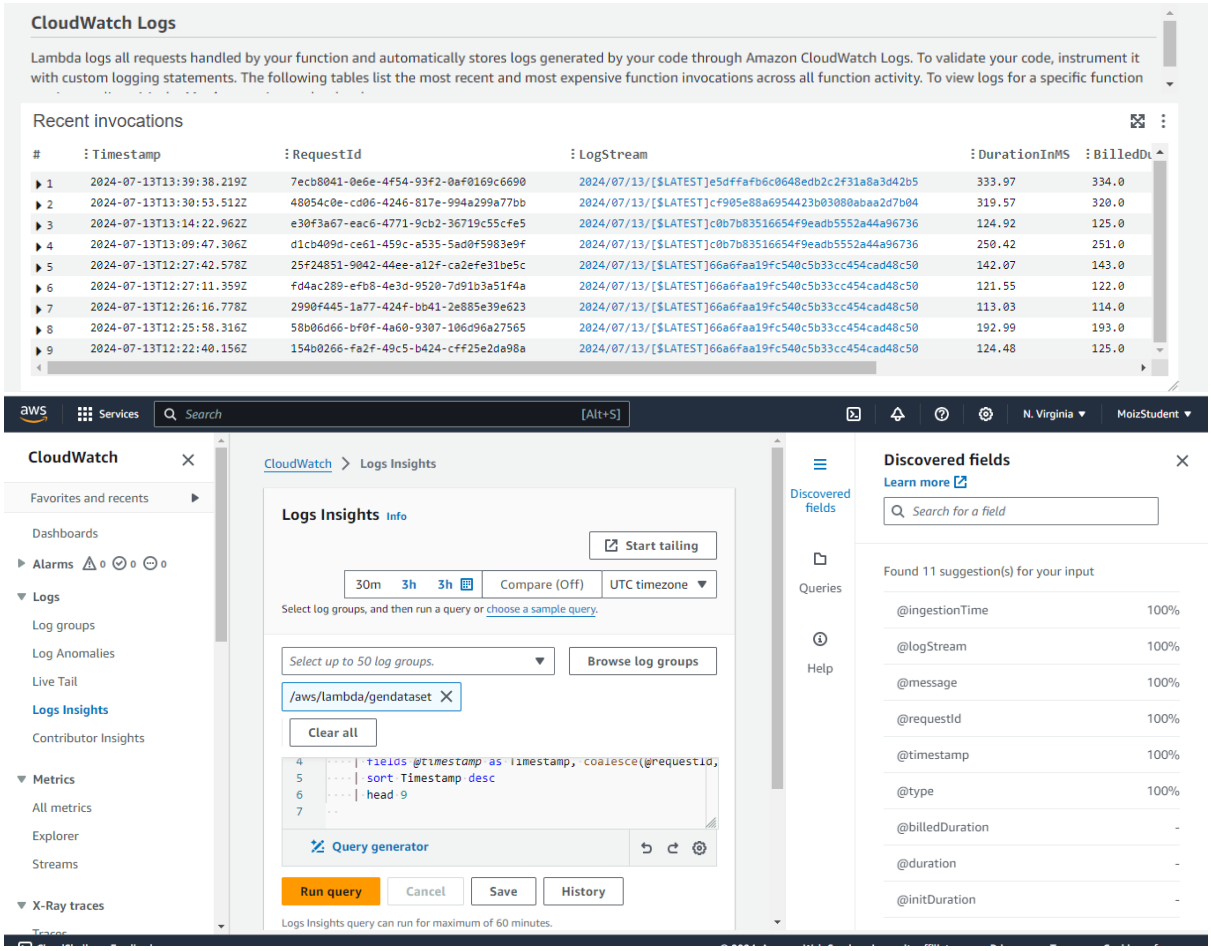

Figure 9: CloudWatch Log

- Export the data in csv format to your Computer

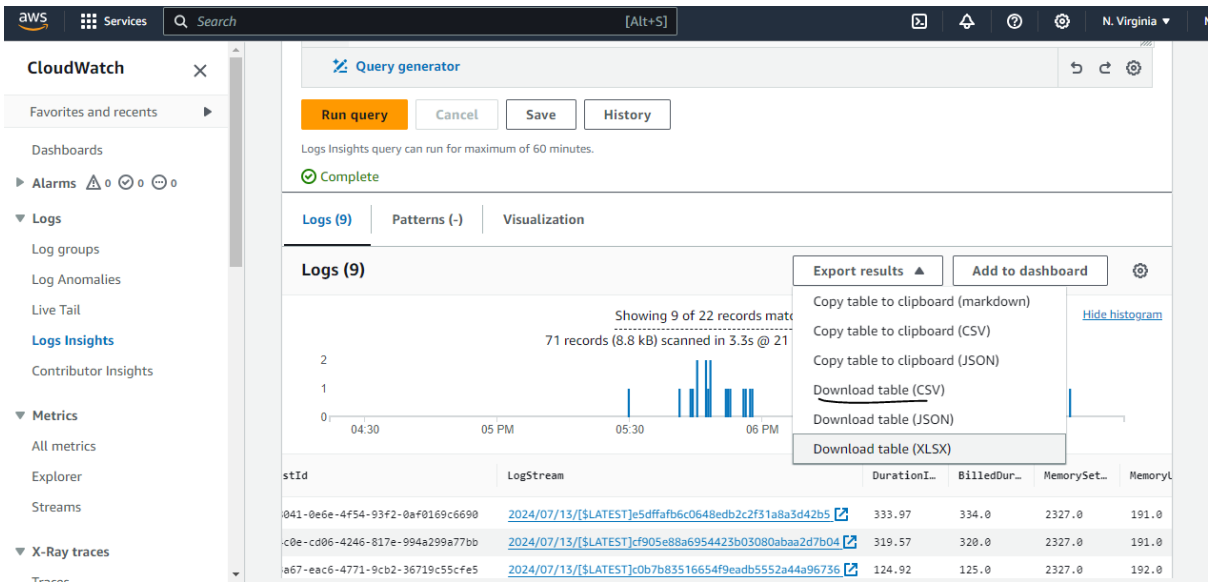

Figure 10

- Upload the csv to your S3 Bucket

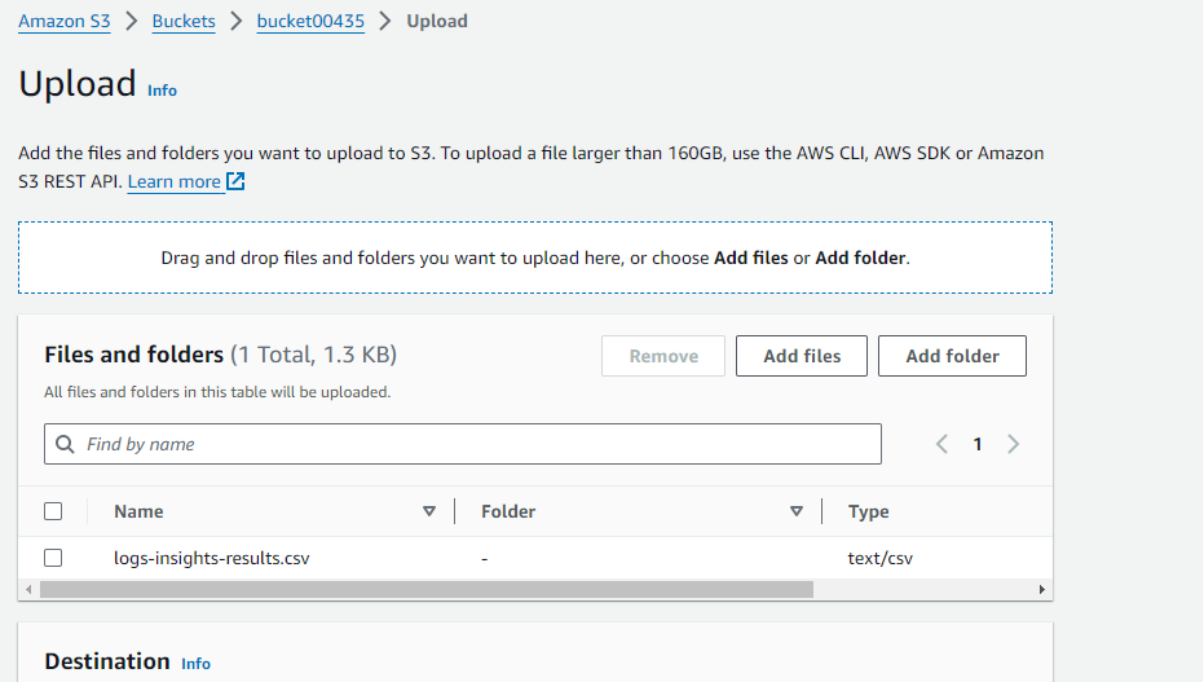

Figure 11

- Ensure the file has been uploaded successfully to S3

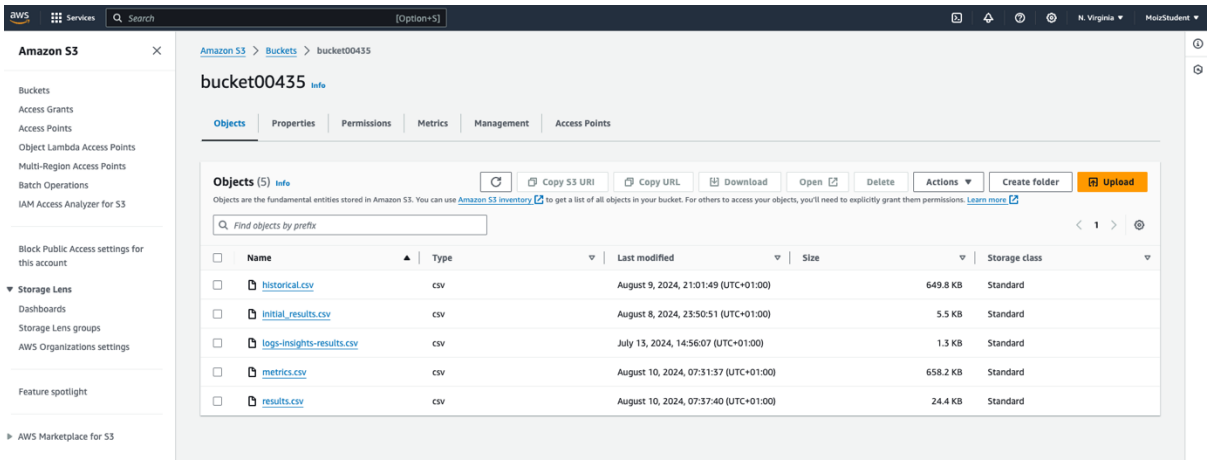

Figure 12

- Run all the cells in “Optimizing Cold” Start notebook in sagemaker

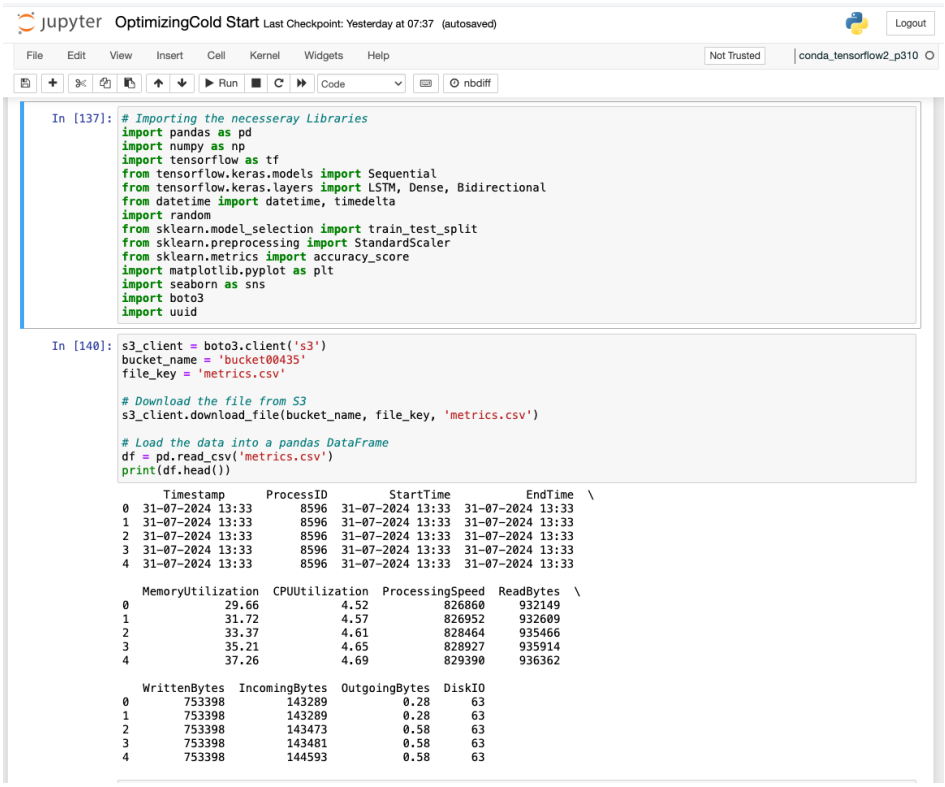

Figure 13

- Test the optimization function in lambda to pre-warm the main function with the help of Genetic Algoritm

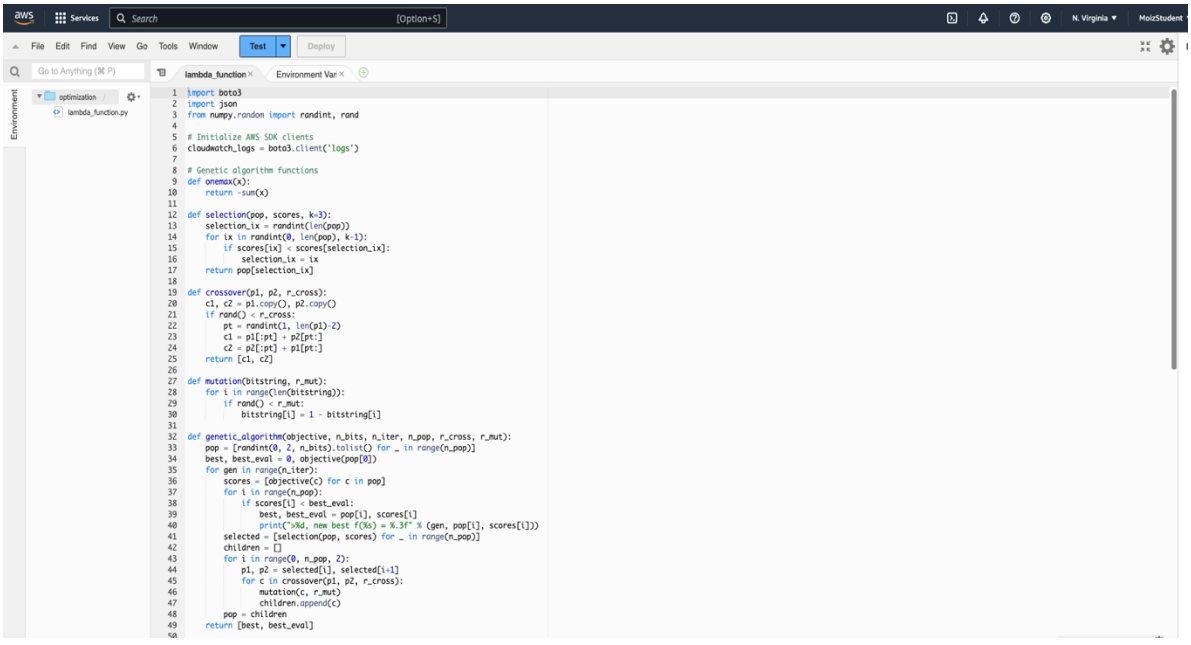

Figure 14

- Optimized cold start time and Original Invocation Time

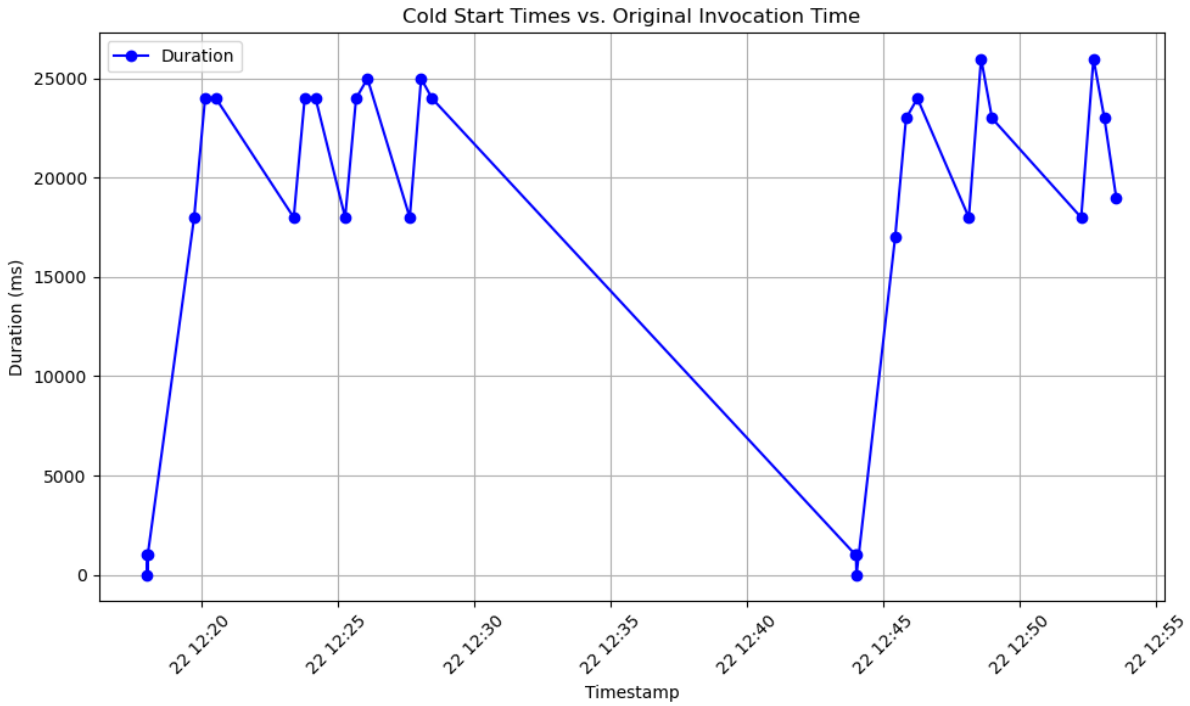

Figure 15

References

Amazon Web Services, Inc. (n.d.). *AWS Lambda Documentation*. Retrieved from https://docs.aws.amazon.com/lambda/

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362. Retrieved from https://numpy.org/

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95. Retrieved from https://matplotlib.org/