National College of Ireland

Real-Time Cloud-Based Anomaly Detection

MSc Research Project

Cloud Computing

Gokul Girish Kumar

Student ID: 23297379

School of Computing

National College of Ireland

Supervisor: Punit Gupta

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Gokul Girish Kumar		
Student ID:	23297379		
Programme:	MSc Cloud Computing	**Year:**	2024
Module:	MSc Research Project		
Supervisor:	Punit Gupta		
Submission Due Date:	15/09/2025		
Project Title:	Real-Time Cloud Based Anomaly Detection		
Word Count:	948	**Page Count:**	4

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: Gokul Girish Kumar

Date: 15/09/2025

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

Gokul Girish Kumar

23297379

1. Introduction

This document provides a complete configuration guide for setting up and deploying the **Real-Time Cloud-Based Anomaly Detection** system. The project's core objective is to deploy a scalable and intelligent intrusion detection system using a deep learning model within a fully serverless AWS architecture.

This manual will walk you through every step, from preparing the dataset to deploying the Long Short-Term Memory (LSTM) model on AWS SageMaker Serverless Inference and exposing it via a public API. It is intended for developers and researchers looking to reproduce the project environment.

2. System Requirements

Before starting, ensure your environment meets the following requirements:

- **AWS Account**: An active AWS account with permissions to manage S3, SageMaker, Lambda, API Gateway, and IAM roles.
- **AWS CLI**: The AWS Command Line Interface must be installed and configured with your credentials.
- **Python**: Version 3.9 or higher is required.
- **Docker**: Required for building and packaging certain environments, though not explicitly for this serverless deployment model. It is good practice to have it installed.
- **cURL or Postman**: A tool for sending HTTP requests is needed to test the final API endpoint.

3. Required Libraries and Packages

The project relies on a set of Python libraries for data science and cloud integration. These should be listed in a `requirements.txt` file in the project repository.

- **boto3**: The AWS SDK for Python, used for programmatic interaction with AWS services like S3.
- **sagemaker**: The SageMaker Python SDK for deploying and managing models.

- **pandas**: For data manipulation, including reading and merging the dataset's CSV files.
- **numpy**: For numerical operations and preparing data for the model.
- **scikit-learn**: Used for essential preprocessing tasks such as `LabelEncoder`, `StandardScaler`, and `train_test_split`.
- **tensorflow**: The deep learning framework used to build, train, and save the LSTM model.

4. Configuration Steps

This section provides a step-by-step guide to deploying the anomaly detection system.

Step 1: Data Ingestion and Storage

1. **Download the Dataset**: Obtain the **CICIDS2017** dataset. It consists of multiple CSV files.
2. **Upload to S3**: Create a bucket in Amazon S3. Upload all the CSV files from the dataset into a directory within that bucket. For example:

   `s3://your-bucket-name/cicids2017/.`

Step 2: Data Processing and Model Training

This phase is typically performed within an AWS SageMaker Notebook, which provides the necessary environment and integrations.

1. Launch a SageMaker Notebook instance.
2. Run the Jupyter Notebook responsible for data preparation (`merger.ipynb` and `preprocess.ipynb`). The notebook will:
   - List and merge all CSVs from S3 into a single pandas DataFrame.
   - Clean the data by stripping whitespace from headers and dropping empty or corrupted rows.
   - Encode the target `Label` column using `LabelEncoder`.
   - Scale the 78 feature columns using `StandardScaler`.
   - Split the data into training and testing sets.
3. Run the training notebook (`train.ipynb`). This script defines the 1D LSTM architecture, compiles it, and trains the model for 5 epochs.
4. After training, the script will save the model as **`model.h5`** and a sample input vector as **`sample_input.npy`**.

Step 3: Model Packaging for Deployment

SageMaker requires a specific packaging format (`.tar.gz`) for deployment.

1. Execute the following shell commands from your SageMaker Notebook terminal to package the trained model.

```
# Create the required directory structure
mkdir -p model_dir/1

# Use a short Python script to load the H5 model and save it in
TensorFlow's SavedModel format
python - <<EOF
import tensorflow as tf
m = tf.keras.models.load_model('model.h5')
m.save('model_dir/1', save_format='tf')
EOF

# Create the final gzipped tarball
tar -czvf model.tar.gz -C model_dir 1
```

This sequence creates the

`model.tar.gz` artifact required by SageMaker.

Step 4: SageMaker Serverless Deployment

1. From a deployment notebook (

   `deploy.ipynb`), upload the `model.tar.gz` artifact to your S3 bucket.

2. Define the SageMaker

   `TensorFlowModel`, providing the S3 path to your model artifact, an IAM execution role, and the framework version (`2.11`).

3. Define the `ServerlessInferenceConfig`, specifying the memory size (e.g., 2048 MB) and max concurrency (e.g., 5).
4. Deploy the model by calling the `.deploy()` method. Name the endpoint **`anomaly-detect-sls`**.

```
# Example code from the deployment notebook
predictor = tf_model.deploy(
    serverless_inference_config=serverless_cfg,
    endpoint_name='anomaly-detect-sls'
)
```

Step 5: API Layer Configuration

To make the model publicly accessible, create an API wrapper using Lambda and API Gateway.

1. **Create AWS Lambda Function**:
   - Navigate to the AWS Lambda console and create a new function named **`invoke-anomaly-endpoint`**.

- Set the runtime to **Python 3.9**.
- Create and attach an IAM role that grants the function `sagemaker:InvokeEndpoint` permissions on the `anomaly-detect-sls` endpoint.
- Use the following code for the Lambda handler:

```
import json
import boto3

runtime = boto3.client('sagemaker-runtime')

def lambda_handler(event, context):
    payload = json.loads(event['body'])
    response = runtime.invoke_endpoint(
        EndpointName='anomaly-detect-sls',
        ContentType='application/json',
        Body=json.dumps(payload)
    )
    result = response['Body'].read().decode()
    return {'statusCode': 200, 'body': result}
```

2. **Create API Gateway Endpoint**:
   - Navigate to the Amazon API Gateway console and create a new **HTTP API** named `anomaly-api`.
   - Create a **`POST`** route for the path **`/predict`**.
   - Attach an integration to this route that invokes your `invoke-anomaly-endpoint` Lambda function.
   - Deploy the API to a stage named `prod`.

Step 6: Live Testing

1. From the API Gateway console, copy the **Invoke URL** for your `prod` stage.
2. Create a `payload.json` file containing a sample data point for prediction. You can generate this from the `sample_input.npy` file saved earlier.
3. Use `cURL` to send a prediction request to your live endpoint.

```
curl -X POST "https://<api-id>.execute-
api.<region>.amazonaws.com/prod/predict" \
-H "Content-Type: application/json" \
-d @payload.json
```

A successful request will return a JSON response from the model with the prediction.

```
C:\Users\gokul\Downloads\Anomaly Payload>curl -X POST "https://mhmv246f8i.execute-api.us-east-1.amazonaws.com/prod/predict" -H "Content-Type: application/json" -
d @payload.json
{"predictions": [[1.0]]}
```