Configuration Manual

MSc Research Project

Threat Detection and Intrusion Prevention in Cloud-Based Infrastructures

Abhishek Joshy
Student ID: X23323507

School of Computing
National College of Ireland

Supervisor: Sean Heaney

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Abhishek Joshy		
Student ID:	X2323507		
Programme:	Msc Cloud Computing	**Year:**	2024-2025
Module:	Msc Research Project		
Lecturer:	Sean Heaney		
Submission Due Date:	11/08/2025		
Project Title:	Threat detection and Intrusion prevention in cloud based infrastructure		
Word Count:	1821	**Page Count:**	5

Signature: Abhishek Joshy

Date: 11/08/2025

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

Abhishek Joshy
Student ID: X23323507

1 Introduction

This document provides detailed instructions for the setup, configuration, and deployment of the "Threat Detection and Intrusion Prevention in Cloud-Based Infrastructures" system. The project consists of a machine learning model trained for network threat detection, a Flask web application that serves the model, and a cloud infrastructure on Amazon Web Services (AWS) where the system is hosted.
Following this manual will guide the user through setting up a local development environment, provisioning the necessary AWS resources, deploying the application code, and verifying that the system is fully operational and integrated with AWS CloudWatch for monitoring. The goal is to create a replicated environment that matches the one used for the development and evaluation of this research project.

1.1 System Prerequisites

Before beginning the configuration process, ensure the following software and accounts are available on your local machine:

- **Git:** For cloning the project repository.
- **Python:** Version 3.8 or higher.
- **pip:** The Python package installer (usually included with Python).
- **AWS Account:** An active Amazon Web Services account with administrative privileges to create IAM users, VPCs, and EC2 instances.
- **AWS CLI:** The AWS Command Line Interface, installed and configured on your local machine.
- **SSH Client:** A terminal or application capable of establishing an SSH connection (e.g., OpenSSH, PuTTY).

2 Local Environment Setup

This section describes how to set up and run the application on a local machine for development and testing purposes.

2.1 Extract Zip

First, clone the project repository from GitHub to your local machine.
Reach to the folder

```
cd thesis
```

2.2 Create a Python Virtual Environment

It is highly recommended to use a virtual environment to manage project dependencies and avoid conflicts with other Python projects.

```
# Create a virtual environment named 'venv'
python -m venv venv

# Activate the virtual environment
# On Windows:
```

```
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate
```

2.3 Install Required Dependencies

Install all the necessary Python packages using the pip command. A requirements.txt file should be present in the repository.

```
pip install -r requirements.txt
```

(Note: If a requirements.txt file is not available, install the core packages manually: pip install flask pandas numpy scikit-learn joblib plotly boto3)

2.4 Verify Project Structure

Ensure that the directory structure is correct, especially the location of the pre-trained models. The saved_models/ directory must be present in the project's root and contain the following files:

- decision_tree_model.pkl
- scaler.pkl
- pca_model.pkl
- label_encoder.pkl

2.5 Run the Application Locally

Start the Flask application from the project's root directory.

```
python app.py
```

You should see output indicating that the models have been loaded and the Flask server is running, typically on http://127.0.0.1:5000.

2.6 Verify Local Operation

Open a web browser and navigate to http://127.0.0.1:5000. You should see the web dashboard. The charts will start populating as the background packet scanner runs. Note that CloudWatch integration will be disabled in this local mode, which is expected.

3 AWS Cloud Infrastructure Configuration

This section details the steps to provision the required cloud infrastructure on AWS. The following details are based on the project's existing setup in the eu-north-1 region.

3.1 Configure AWS CLI

If you haven't already, configure your AWS CLI with credentials that have administrative access.

```
aws configure
AWS Access Key ID [None]: YOUR_ACCESS_KEY
AWS Secret Access Key [None]: YOUR_SECRET_KEY
Default region name [None]: eu-north-1
Default output format [None]: json
```

3.2 Provision EC2 Key Pair

You will need an EC2 Key Pair to securely connect to your instance.

1. Navigate to the EC2 service in the AWS Management Console.
2. Go to **Key Pairs** under the **Network & Security** section.
3. Click **Create key pair**.
4. Name it ec2keypair and select the .pem format.

5. Download the ec2keypair.pem file and save it in a secure location. You will need to set its permissions to be read-only for your user.

```
chmod 400 ec2keypair.pem
```

3.3 Provision Core Infrastructure

The following resources need to be created. You can do this via the AWS Management Console in the eu-north-1 region.

- **VPC:** A Virtual Private Cloud with CIDR block 10.0.0.0/16. (VPC ID: vpc-07cbe4f30f3a2f518)
- **Subnet:** A public subnet within the VPC with CIDR block 10.0.1.0/24. (Subnet ID: subnet-0ae0b1401435a2043)
- **Internet Gateway:** Create and attach an Internet Gateway to your VPC. (IGW ID: igw-0367e9ff7fd6aa70b)
- **Route Table:** Ensure the main route table for your VPC has a route (0.0.0.0/0) pointing to the Internet Gateway. (RTB ID: rtb-0095a787d34dd48f7)

3.4 Configure Security Group

Create a Security Group that will act as a virtual firewall for the EC2 instance.

- **Group Name:** flask-app-sg
- **VPC:** Select the VPC created in the previous step.
- **Inbound Rules:** Add the following rules to allow necessary traffic.

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	For SSH access
HTTP	TCP	80	0.0.0.0/0	Standard web traffic
HTTPS	TCP	443	0.0.0.0/0	Secure web traffic
Custom TCP	TCP	5000	0.0.0.0/0	For the Flask application

3.5 Launch EC2 Instance

Launch the virtual server that will host the application.

1. Navigate to the EC2 service and click **Launch instances**.
2. **AMI:** Select an Amazon Linux 2 AMI (or a similar Linux distribution). (Project AMI ID: ami-09d840fad48a1395e).
3. **Instance Type:** Select t3.micro (this is eligible for the AWS Free Tier).
4. **Key Pair:** Select the ec2keypair created earlier.
5. **Network Settings:**
   - Select your custom VPC and subnet.
   - Enable **Auto-assign public IP**.
   - Select the flask-app-sg security group.
6. Launch the instance. Once it is running, take note of its **Public IPv4 address** (e.g., 16.16.25.121).

4 Application Deployment to AWS EC2

This section covers deploying the application code to the newly created EC2 instance.

4.1 Connect to the EC2 Instance

Use your SSH client and the .pem key file to connect to the instance. Replace 16.16.25.121 with your instance's public IP.

```
ssh -i "path/to/ec2keypair.pem" ec2-user@16.16.25.121
```

4.2 Transfer Project Files

From your **local machine's terminal**, use the scp command to recursively copy the entire project directory to the EC2 instance's home directory.

```
scp -i "path/to/ec2keypair.pem" -r . ec2-user@16.16.25.121:/home/ec2-user/abhishek-thesis
```

4.3 Set Up Environment on EC2

Once connected to the EC2 instance via SSH, perform the following steps:

1. **Install Python and Git:**

```
sudo yum update -y
sudo yum install git python3 -y
```

2. **Navigate to the project directory:**

```
cd abhishek-thesis
```

3. **Install Dependencies:**

```
pip3 install -r requirements.txt --user
```

4.4 Configure IAM Credentials for CloudWatch

For the application to send data to CloudWatch, it needs AWS credentials. The best practice is to use an IAM Role attached to the EC2 instance.

1. **Create an IAM Role:** In the AWS Console, go to IAM > Roles > Create role. Select **AWS service** and **EC2** as the use case.
2. **Add Permissions:** Attach the CloudWatchFullAccess policy.
3. **Name and Create:** Name the role (e.g., EC2CloudWatchAccessRole) and create it.
4. **Attach Role to EC2:** Go to your EC2 instance in the console, select Actions > Security > Modify IAM role, and attach the newly created role.

This method is more secure than storing access keys on the instance. The application's boto3 library will automatically discover and use the role's permissions.

4.5 Run the Application Persistently

To ensure the Flask application continues running after you disconnect your SSH session, use a tool like nohup.

```
nohup python3 app.py > flask.log 2>&1 &
```

This command will:

- Run python3 app.py.
- Redirect all output (stdout and stderr) to a file named flask.log.
- The & at the end runs the process in the background.

To check the application's logs, you can use:

```
tail -f flask.log
```

To stop the application, you will need to find its process ID (pid) and kill it:

```
ps -fA | grep python
kill <pid>
```

4.6 Verification and Monitoring

After deployment, verify that the entire system is functioning correctly.

4.7 Access the Web Dashboard

Open your web browser and navigate to your instance's public IP on port 5000: http://16.16.25.121:5000

The dashboard should load and display live, updating data.

4.8 Verify CloudWatch Integration

1. **Check Logs:** Navigate to the CloudWatch service in the AWS Console. Go to **Log groups** and find /aws/ec2/abhishek-flask-app. You should see log streams containing output from the running application.
2. **Check Metrics:** In CloudWatch, go to **All metrics**. Under the **Custom namespaces** section, you should find ThreatDetection/Flask. Click on it to see and graph the custom metrics being sent (e.g., AttackPackets, PredictionConfidence).
3. **Check CloudWatch Dashboard:** Access the pre-configured dashboard via its URL to see a consolidated view of the system's performance and security posture. https://eu-north-1.console.aws.amazon.com/cloudwatch/home?region=eu-north-1#dashboards:name=abhishek-threat-detection-dashboard

5 Troubleshooting Common Issues

- **Connection Timed Out:** If you cannot access the web application, double-check the **Security Group inbound rules**. Ensure port 5000 is open to your IP or 0.0.0.0/0.
- **Internal Server Error (500):** Check the application logs (flask.log on the EC2 instance) for Python errors. This is often due to missing dependencies or file path issues.
- **Models Not Loading:** Ensure the scp command in step 5.2 copied the saved_models/ directory and all its contents correctly.
- **CloudWatch Not Receiving Data:** This is almost always an IAM permissions issue. Ensure the IAM Role is correctly created and attached to the EC2 instance, and that it has the CloudWatchFullAccess policy.

Conclusion

By following this manual, you have successfully deployed the Threat Detection and Intrusion Prevention system on a dedicated AWS cloud infrastructure. The application is now running, performing real-time predictions, and is fully integrated with AWS CloudWatch for robust monitoring. The system is configured for continuous operation and provides a powerful platform for analyzing and visualizing network security threats in a cloud environment.