National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Sai Bhanu Prasad Pacha
Student ID: x23213167

School of Computing
National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sai Bhanu Prasad Pacha
Student ID:	x23213167
Programme:	MSc in Cloud Computing **Year:** 2025-2026
Module:	MSc Research Project
Lecturer:	Yasantha Samarawickrama
Submission Due Date:	28/01/2026
Project Title:	Enhancing Cloud Security with Machine Learning Based Intrusion Detection Systems
Word Count:	1034 **Page Count:** 8

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: Sai Bhanu Prasad Pacha

Date: 28/01/2026

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

Sai Bhanu Prasad Pacha
x23213167

1 Introduction

This configuration guide is a guide to the setup, configuration, operation, and debugging of the Machine Learning-based Network Intrusion Detection System (ML-IDS), which is an ensemble-based Network Traffic classifier that is setup to detect and label network traffic as Benign or Attack. Its main characteristics are binary classification, a RESTful API with real-time predictions, deployment in Docker on AWS EC2 and monitoring stack monitoring Prometheus + Grafana with a testing suite based on the CSE-CIC-IDS-2018 dataset. This manual is prepared to help the academic students and researchers to reproduce, test, or optimize the system.

2 System Requirements

Local Development Environment:
- Processor: x86-64 CPU (Intel/AMD), ARMx64
- Memory: Minimum 8GB RAM (16 GB Recommended)
- Storage: 40 GB free disk space
- GPU: Google Colab Pro for model training (T4/L4 GPU)

Operating System:
- macOS /Ubuntu/Windows

AWS Deployment:
- AWS Account
- AWS CLI v2
- EC2 Keypair for SSH access

3 Software Dependencies

3.1 Core Softwares

Software	Version	Download Link
Python	3.10+	https://www.python.org/downloads/
Google Colab Pro	-	https://colab.research.google.com/
Visual Studio Code	Recent	https://code.visualstudio.com/download
AWS CLI	2.0+	https://aws.amazon.com/cli/
Docker	20+	https://docs.docker.com/get-started/get-docker/

3.2 Python Dependencies

Package Name	Version	Description
pyarrow	Latest	Parquet file support
numpy	2.0.2	Numerical operations
tensorflow	2.19.0	Deep learning framework
keras	3.10.0	Neural Networks API
pandas	2.2.2	Data processing
scikit-learn	1.6.1	Random Forest Model
xgboost	2.1.4	XGBoost Model
requests	Latest	API testing client
Matplotlib	Latest	Visualization
flask	2.3.3	REST API framework
gunicorn	Latest	Production WSGI server
Prometheus-client	0.18.0	Metrics collection

4 Data Preparation

This study is based on the CSE-CIC-IDS-2018 dataset created by the Canadian Institute of Cybersecurity.

Link: https://www.unb.ca/cic/datasets/ids-2018.html

- Total Samples: 723,337 network flows
- Features: 69 (after preprocessing)
- Classes: Binary (0 = Benign, 1 = Attack)
- Format: Parquet files for efficient loading

5 Model Training on Colab

- Platform: Google Colab Pro
- GPU: T4 GPU
- Runtime: Python 3.10+

5.1 Random Forest and XGBoost Training

Notebook: ML_IDS_v4.ipynb

Input: processed Parquet files (X_train_scaled.parquet, X_test_scaled.parquet, X_val_scaled.parquet, y_train.parquet, y_test.parquet)

Output files: - xgboost_ids_model.pkl, xgboost_model.json, model_metadata.json, features_names.pkl, scaler.pkl

5.2 MLP Model Training

Notebook: ML_IDS_Deep_Learning_MLP.ipynb

Input: processed Parquet files (X_train_scaled.parquet, X_test_scaled.parquet, X_val_scaled.parquet, y_train.parquet, y_test.parquet)

Output files: deep_learning_mlp_best.h5

6 Project Structure

```
bhanuprasad-thesis/
│
├── deployment/                    # Deployment files
│   ├── app.py                     # Flask API application
│   ├── Dockerfile                 # Docker container definition
│   ├── docker-compose.yml         # multi-container orchestration
│   ├── requirements.txt           # Python packages
│   ├── prometheus.yml             # Configuration for Prometheus
│   ├── test_api.py                # API test suite
│   ├── real_data_simulator.py     # Demo run script
│   ├── traffic_simulator.py       # traffic simulator
│   ├── ids_analytics.py           # Performance analytics
├── models/                        # Trained ML models
│   ├── random_forest_model.pkl
│   ├── xgboost_model.json
│   ├── scaler.pkl
│   ├── feature_names.pkl
│   └── model_metadata.json
├── processed_data/                # Preprocessed datasets
│   ├── X_train_scaled.parquet
│   ├── X_test_scaled.parquet
│   ├── y_train.parquet
│   └── y_test.parquet
├── ML_IDS_v4.ipynb                        # Main training notebook
├── ML_IDS_Deep_Learning_MLP.ipynb         # Deep learning notebook
├── ML_IDS_v5_model_export.ipynb           # Model export notebook
└── README.txt
```

7 AWS Deployment

7.1 AWS Configuration

$ aws configure
AWS Access Key ID: [Access Key ID]
AWS Secret Access Key: [Secret Key ID]
Default region name: us-east-1/eu-west-1
Default output format: json

7.2 Launch EC2 Instance

```
$ aws ec2 run-instances \
  --image-id ami-0c1c30571d2dae5c9 \
  --instance-type t3.medium \
  --key-name ids-deploy-key \
  --security-group-ids sg-xxx \
  --block-device-mappings DeviceName=/dev/sda1,Ebs={VolumeSize=20} \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=ids-api-server}]'
```

7.3 Install Docker on EC2

- Connect to EC2 instance:
  $ chmod 400 ids-deploy-key.pem
  $ ssh -i ids-deploy-key.pem ubuntu@<instance-ip>

- Install Docker:
  $ curl -fsSL https://get.docker.com -o get-docker.sh
  $ sudo sh get-docker.sh
  $ sudo usermod -aG docker ubuntu

- Install Docker Compose:
  $ sudo apt install docker-compose -y

- Verify installation:
  $ docker --version
  $ docker-compose --version

7.4 Upload Models and Files

From local machine, upload files to EC2:

- Upload deployment directory:
  $ scp -i ids-deploy-key.pem -r deployment/ ubuntu@<instance-ip>:~/

- Upload models directory:
  $ scp -i ids-deploy-key.pem -r models/ ubuntu@<instance-ip>:~/deployment/

- Verify upload on EC2:
  $ ssh -i ids-deploy-key.pem ubuntu@<instance-ip>
  $ cd deployment

Model files:
  - random_forest_model.pkl
  - xgboost_model.json
  - scaler.pkl
  - feature_names.pkl
  - model_metadata.json

7.5 Deploy with docker-compose

On EC2 instance:

- Navigate to deployment directory:
  $ cd ~/deployment

- Build Docker image:
  $ docker-compose build

- Start all services:
  $ docker-compose up -d

  This starts:
  - ids-api: Flask API (port 5000)
  - prometheus: Metrics scraper (port 9090)
  - grafana: Monitoring dashboard (port 3000)

- Check running containers:
  $ docker-compose ps

- View logs:
  $ docker-compose logs -f ids-api

  Expected output should show:
  - Metadata loaded
  - Scaler loaded
  - Feature names loaded
  - Random Forest loaded
  - XGBoost loaded

7.6 Verify Deployment

From your local machine:

1. Test health endpoint:
   $ curl http://34.242.155.220:5000/health #34.242.155.220 is the EC2 Instance IP

   Expected response:

```
{
  "status": "healthy",
  "models_loaded": {
    "random_forest": true,
    "xgboost": true,
    "deep_mlp": false
  },
  "features_count": 69,
  "version": "1.0.0"
}
```

2. Test API info endpoint:
   $ curl http://34.242.155.220:5000/

   This will return API endpoints with description

3. Test model info:
   $ curl http://34.242.155.220:5000/model/info

This will return model metadata with accuracy, AUC metrics

7.7 Deployment Testing

The real_data_simulator.py script tests the API with CSE-CIC-IDS-2018 test data.

Experiment 1: Verbose mode (see each prediction):
   $ python deployment/real_data_simulator.py --samples 20 -v --api http://34.242.155.220:5000

   Shows individual predictions with:
   - Sample ID
   - Actual label (Benign/Attack)
   - Predicted label
   - Confidence score
   - Latency (ms)

Experiment 2: Batch performance test:
$ python deployment/real_data_simulator.py --samples 500 --batch-size 10 --api http://34.242.155.220:5000

Output includes:
- Confusion matrix
- Accuracy, Precision, Recall, F1-Score
- Average latency per sample
- Throughput (samples/second)

Experiment 3: Attack-only test:
$ python deployment/real_data_simulator.py --samples 100 --mode attack --api http://34.242.155.220:5000

Experiment 4: Benign-only test: (For false-positive tests)
$ python deployment/real_data_simulator.py --samples 100 --mode benign --api http://34.242.155.220:5000

7.8 Performance Analytics (Visualizations)

Run analytics:

$ python deployment/ids_analytics.py --samples 200 --api http://34.242.155.220:5000

This provides latency plots, confusion matrix, performance dashboard figure, and a performance report.

8 Monitoring

Access Prometheus metrics: http://34.242.155.220:9090

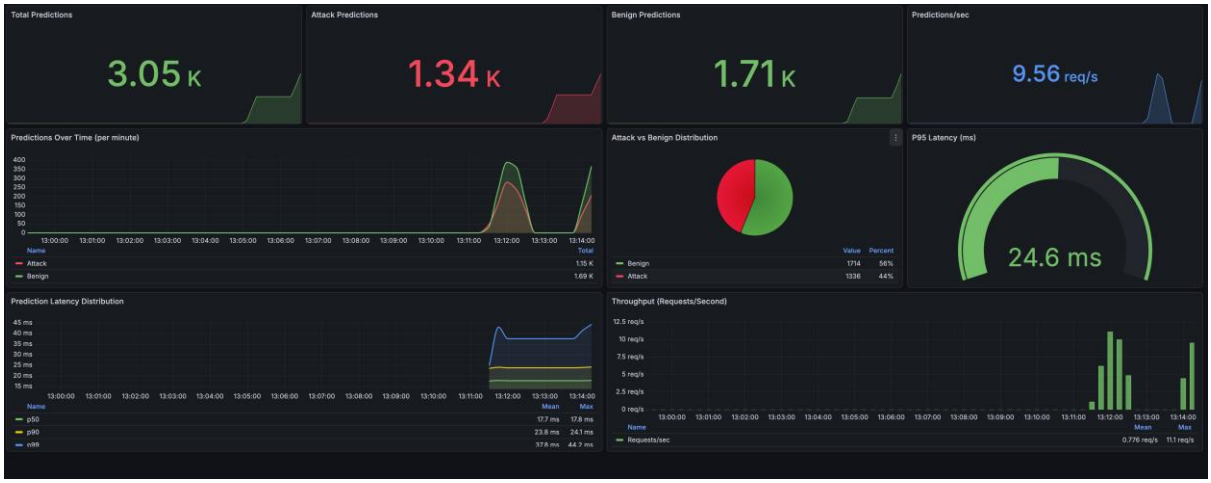
Key metrics:
- predictions_total{prediction_type="Benign"}
- predictions_total{prediction_type="Attack"}
- prediction_latency_seconds

Sample queries:
1. Total predictions: sum(predictions_total)
2. Attack rate: predictions_total{prediction_type="Attack"} / sum(predictions_total) * 100
3. Average latency: rate(prediction_latency_seconds_sum[5m]) /

rate(prediction_latency_seconds_count[5m])

Grafana Dashboard: http://34.242.155.220:3000
Login: admin / Bhanu@nci25

To add Prometheus data source:
1. Configuration → Data Sources → Add data source
2. Select Prometheus
3. URL: http://prometheus:9090
4. Save & Test



9 References

Google Colab: https://colab.research.google.com/
AWS CLI v2: https://docs.aws.amazon.com/cli/v1/userguide/install-windows.html
Flask Documentation: https://flask.palletsprojects.com/
Grafana Documentation: https://grafana.com/docs/