National College of Ireland

Configuration Manual

MSc Research Project
MSc Cloud Computing

Apurva Ghodekar
Student ID: x23304995

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Apurva Ghodekar
Student ID:	x23304995
Programme:	MSc Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	11/12/2025
Project Title:	Configuration Manual
Word Count:	747
Page Count:	6

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:	Apurva Ghodekar
Date:	11th December 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

Apurva Ghodekar
x23304995

1 Introduction

This installation guide outlines the entire setup, execution process and requirements to develop and run the Cloud-Migrate-AI framework. The system combines several cloud automation and validation features such as Infrastructure-as-Code with the *Terraform Documentation* (2024), serverless deployment with the Serverless Framework platform *Serverless Framework Docs* (2024), policy validation with Open Policy Agent (OPA) *Open Policy Agent Docs* (2024), extracting telemetry from Amazon CloudWatch *Amazon CloudWatch Metrics* (2024), and AI-assisted anomaly detection using the PyTorch framework *PyTorch Documentation* (2024).

This manual will aim to make all experiments carried out in this project reproducible, that is infrastructure deployment, model training, validation gates, and CI/CD-driven automation.

2 System Requirements and Setup

2.1 Hardware Requirements

Cloud-Migrate-AI framework is not based on GPU acceleration and can run on any standard compute hardware. The recommended minimum specifications will be:

- **Processor:** Quad-core 64-bit CPU
- **Memory:** 16 GB RAM
- **Storage:** 20 GB free disk space
- **GPU:** Not required (all models train and evaluate on CPU)

2.2 Software Requirements

Supported operating systems include:

- **Linux:** Rocky Linux 9, Ubuntu 22.04
- **macOS:** Compatible with standard Python environments
- **Windows:** Recommended via WSL2 for consistent tooling

2.2.1 Programming Runtimes

- **Python:** Version 3.9 or later
- **Node.js:** Version 18+ (required for the Serverless Framework)
- **Terraform:** Version 1.9+ *Terraform Documentation* (2024)

2.2.2 CLI and Tooling Dependencies

- **AWS CLI:** Required for programmatic AWS access
- **Serverless Framework:** For packaging and deploying Lambda functions *Serverless Framework Docs* (2024)
- **Open Policy Agent (OPA):** Used for policy validation *Open Policy Agent Docs* (2024)
- **Utility Tools:** `jq`, `curl`, `unzip`

2.3 Python Environment Setup

All Python dependencies are defined in `requirements.txt`. The environment is created using:

```
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

Key libraries used in the project include:

- **NumPy:** NumPy is used to do basic computations and calculations in numbers *NumPy Documentation* (2024).
- **Pandas:** This is to clean, convert and sort the telemetry data into a workable format. *Pandas Documentation* (2024).
- **PyTorch:** Python library to construct and train the LSTM and Autoencoder models in the presence of anomaly. *PyTorch Documentation* (2024).

3 Project Structure

Cloud-Migrate-AI is a project that is guided by a modular DevOps and MLOps design comprising of infrastructure automation, serverless application deployment, telemetry extraction, and AI-based validation.

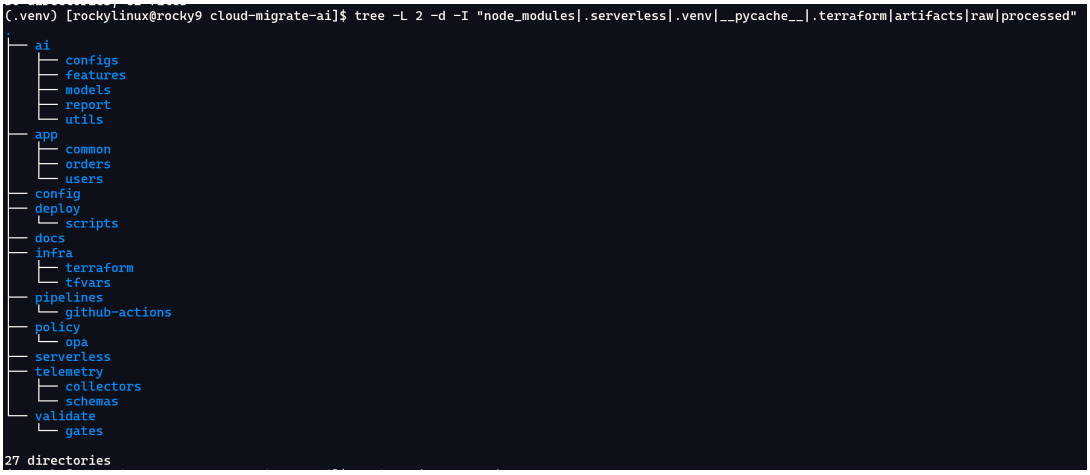

Figure 1: Directory structure

- **infra**: This directory contains the Terraform code that is used to deploy all cloud resources, such as networking resources, IAM roles, and the RDS database.
- **serverless**: Contains the configuration and application handlers of the Serverless Framework and FastAPI which are deployed as AWS Lambda functions.
- **policy/opa**: This holds the OPA Rego policies and related tests that authenticate security, compliance, and configuration guidelines at deployment.
- **telemetry**: Contains scripts to gather CloudWatch metrics, cost information, and schema definitions to be used in storing system telemetry.
- **ai**: This is the folder where the machine learning pipeline is stored which consists of feature engineering scripts, model training code (LSTM and Autoencoder) as well as prediction modules and reporting utilities.
- **validate/gates**: This gives automated validation scripts that are used in CI/CD, i.e. the OPA gate, smoke tests, cost and carbon checks, and AI anomaly tests.
- **deploy**: Helper scripts that are employed in infrastructure deployment, artifact export, smoke testing, and rollback.
- **app**: This has the FastAPI application structure of the users and orders including database connection and route handlers.
- **pipelines**: The pipeline workflow files are kept here and are used to automate the CI/CD of the project.

Separation of concerns will allow independent testing, automation as well as integration of every subsystem.

4 Execution Workflow

4.1 Step 1 — Provision AWS Infrastructure

Terraform is used to deploy the networking and database components:

```
cd infra/terraform/aws
terraform init
terraform apply
```

This generates:

- VPC, subnets, route tables
- Security groups
- RDS PostgreSQL database
- IAM roles and SSM parameters

4.2 Step 2 — Deploy Serverless Application

The Serverless Framework *Serverless Framework Docs* (2024) packages the FastAPI handlers into AWS Lambda:

```
cd serverless
sls deploy --stage dev
```

4.3 Step 3 — Smoke Tests

Application correctness is verified via:

```
curl https://<api-url>/healthz
```

4.4 Step 4 — Telemetry Collection

CloudWatch *Amazon CloudWatch Metrics* (2024) and Cost Explorer *AWS Cost Explorer* (2024) metrics are collected using custom scripts:

```
python telemetry/collectors/cw_collect.py
python telemetry/collectors/ce_cost_collect.py
```

4.5 Step 5 — Feature Engineering

Telemetry is merged into a unified dataset:

```
python ai/features/build_features.py
```

4.6 Step 6 — Train LSTM Model

```
python ai/models/train_lstm.py
```

4.7 Step 7 — Train Autoencoder

```
python ai/models/train_autoencoder.py
```

4.8 Step 8 — Produce Visualisations

```
python ai/report/plots.py
```

4.9 Step 9 — AI Validation Gate

```
./validate/gates/ai_gate.sh
```

4.10 Step 10 — CI/CD Execution

GitHub Actions automates:

- OPA compliance validation *Open Policy Agent Docs* (2024)
- Infrastructure drift checks
- Serverless deployment
- Telemetry collection
- AI anomaly decision

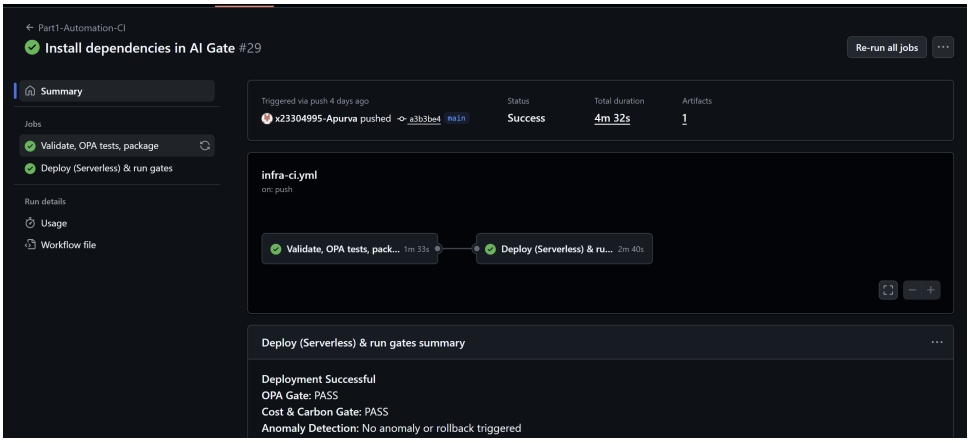

Figure 2: Github Actions

5 Generated Artifacts

The system outputs several reproducibility artifacts essential for experiment tracking and reporting:

- `migration_aws_dev.json`: Infrastructure descriptor
- CloudWatch telemetry (`cw_*.parquet`)
- Cost telemetry (`cost_*.parquet`)
- Feature dataset (`features.parquet`)

- Trained models (`lstm_best.pt`, `ae_best.pt`)
- Model scalers and metrics JSON files
- Visualisation outputs for analysis
- AI anomaly report (`ai_score.json`)

```
(.venv) [rockylinux@rocky9 validate]$ cat ai_score.json
{
  "anomaly_score": 0.6624233964230333,
  "threshold": 3.0,
  "flag": false,
  "lstm": {
    "raw_error": 0.00047435022589497494,
    "normalized_error": 0.9740859279636473,
    "baseline": 0.0004869695909544826,
    "feature_columns": [
      "db_cpu",
      "db_conn",
      "db_write_ms",
      "cost_daily_usd",
      "hour_of_day",
      "day_of_week"
    ],
    "target_columns": [
      "db_read_ms"
    ]
  },
  "autoencoder": {
    "raw_error": 0.010687612653666799,
    "normalized_error": 0.35076086488241925,
    "baseline": 0.030469797869980338,
    "feature_columns": [
      "db_cpu",
      "db_conn",
      "db_write_ms",
      "cost_daily_usd",
      "hour_of_day",
      "day_of_week"
    ]
  }
```

Figure 3: `ai_score.json`

References

Amazon CloudWatch Metrics (2024). `https://docs.aws.amazon.com/cloudwatch`.

AWS Cost Explorer (2024). `https://docs.aws.amazon.com/cost-explorer`.

NumPy Documentation (2024). `https://numpy.org/doc/`.

Open Policy Agent Docs (2024). `https://www.openpolicyagent.org/docs/`.

Pandas Documentation (2024). `https://pandas.pydata.org/docs/`.

PyTorch Documentation (2024). `https://pytorch.org/docs/`.

Serverless Framework Docs (2024). `https://www.serverless.com/framework/docs`.

Terraform Documentation (2024). `https://developer.hashicorp.com/terraform/docs`.