National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Boyang Jiang
Student ID: 23399937

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Boyang Jiang
Student ID:	23399937
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	11/12/2025
Project Title:	Configuration Manual
Word Count:	814
Page Count:	7

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:	Boyang Jiang
Date:	25th January 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

Boyang Jiang
23399937

1 Introduction

This configuration manual provides a comprehensive overview of the software, hardware, and runtime environment required to reproduce the experimental system developed in this thesis. It details the setup of the complete experimental infrastructure as well as the execution procedures for all three experiments. The environment consists of both the proposed machine learning–based monitoring system and the open-source monitoring system Prometheus Yudha Erian Saputra et al. (2024). The configuration workflow reflects a typical integration of DevOps processes and machine learning components, which commonly appear together in modern operational pipelines Jain and Kumar (2024).

2 System Overview

2.1 Hard Environment

All experiments in this study were executed in a controlled cloud environment on AWS EC2. We recommend using the following configuration.

- **Instance Type:** c5.xlarge
- **vCPU / Memory:** 4 vCPU, 8 GB Memory
- **Storage:** 40 GB
- **AMI:** Ubuntu 24.04 LTS

2.2 Software Environment

The experimental environment uses the following software components:

- Python 3.12
- Apache Spark 4.0.1
- Kafka 3.7.0 (Kraft mode)
- Docker Engine 29.1.2 and Docker Compose v5.0.0
- Prometheus 3.6.0
- Vector 0.47.0

- Chaos Toolkit 1.19.0
- FastAPI 0.121.3
- SentenceTransformer (all-MiniLM-L12-v2)

2.3 Directory Layout

The full directory structure of the experimental system is shown below. Each folder corresponds to a component of the monitoring, anomaly detection, and root cause analysis pipeline.

```
.github/                  # GitHub Actions CI/CD pipeline
chaos/                    # Chaos experiment scripts
   result/                # Chaos experiment results
fastapi-demo/             # Demo application soucre code
nginx/                    # Demo application Nginx reverse proxy
   logs/                  # Nginx logs
prometheus/               # Prometheus monitoring system
spark/                    # Spark-based data processing engine
   data/                  # Parquet datasets
   ivy2/                  # Cached JAR dependencies
   metrics/               # Experimental result files
   models/                # Trained models
   script/                # Submit Spark job scripts
vector-feeder/            # Log generator (simulated real application)
   data/                  # Feeder input dataset
```

2.4 GitHub Repository

All configuration files, scripts, and experimental resources used in this thesis are available in the public repository:

`https://github.com/BryanJiang-NCI/ml-monitoring.git`

2.5 Execution Location

To ensure reproducibility and avoid ambiguity during environment setup, this manual clearly distinguishes between commands executed on the host machine and those executed inside Docker containers. All commands starting with $ **must be executed on the host machine,** while commands beginning with ‘root@spark-master‘ **must be executed inside the Spark container**. Before executing container commands, exec command must be run to enter the container.

- **Host machine commands** Commands executed directly on the local machine.

  ```
  $
  ```

- **Exec commands** Commands used to enter a running container.

```
$ docker exec -it spark-master bash
```

- **Container commands** Commands executed inside the container after entering it.

```
root@spark-master:~#
```

2.6 System Architecture

As shown in Figure 1, the implementation follows a hierarchical structure, ensuring each layer is modular yet connected across the data flow.

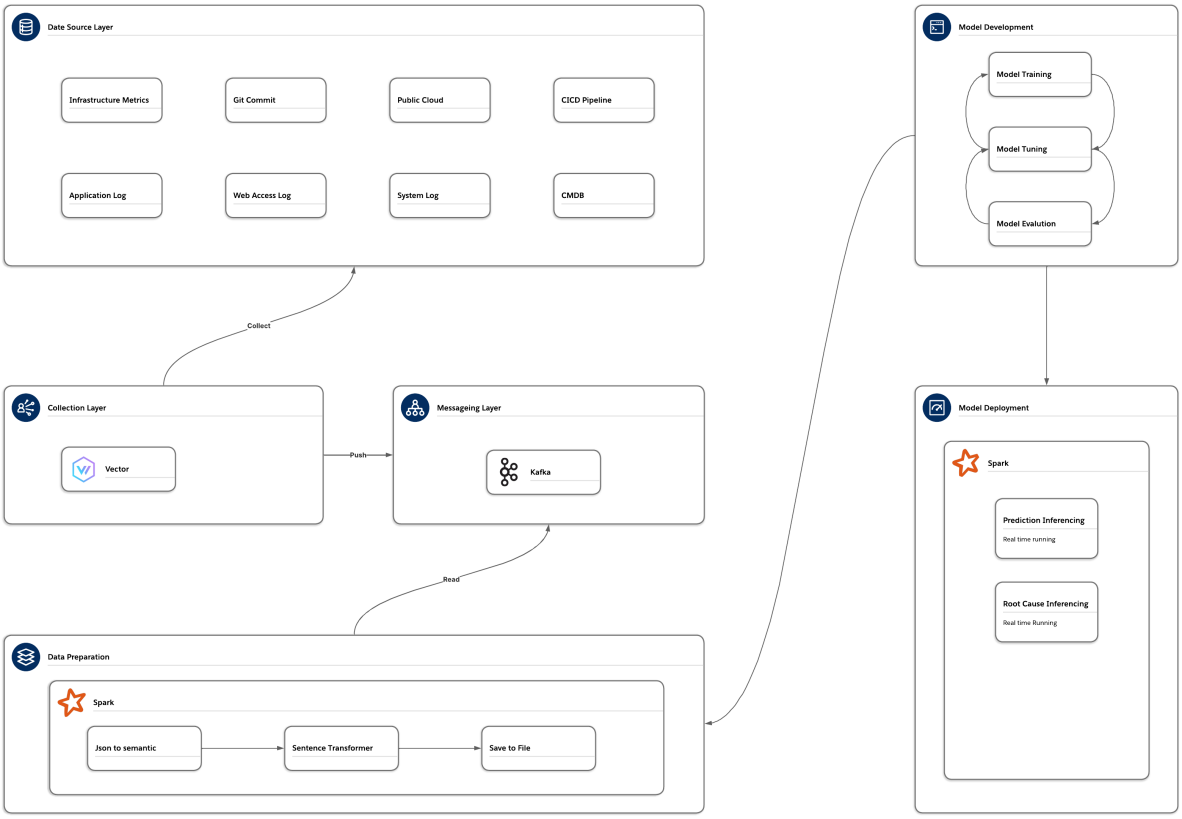

Figure 1: ML-Monitoring Architecture

2.7 Structure of the Remaining Document

The following sections are organized to support both full transparency of the experimental workflow and fast reproducibility:

Section 3 presents the complete manual execution steps. It provides a detailed, step-by-step breakdown of the entire experimental pipeline.

Section 4 introduces the automated execution workflow. All procedures are encapsulated into shell scripts, enabling a streamlined "one-click" execution experience. For simple and quick replication of the experiment, one can directly refer to the section.

3 Manual Execution (Full Step-by-Step Instructions)

3.1 Pull Source Code

Pull the source code from GitHub. This step is required for both manual and automated execution.

```
$ git clone https://github.com/BryanJiang-NCI/ml-monitoring.git
$ cd ml-monitoring
```

3.2 Container Startup

Before setting up the experimental environment, ensure that the Docker Engine and Docker Compose are properly installed on the host machine. The entire environment can be launched with a single command:

```
~/ml-monitoring$ docker compose up -d
[+] Running 9/9
  Network ml-monitoring_ml_experiment  Created  0.0s
  Container vector-feeder              Started  0.5s
  Container kafka-kraft                Started  0.5s
  Container prometheus                 Started  0.6s
  Container spark-master               Started  0.6s
  Container fastapi-demo               Started  0.5s
  Container nginx                      Started  0.6s
  Container spark-worker               Started  0.7s
  Container vector                     Started  0.7s
```

3.3 CMDB Dataset Preparation

We directly use files as the database for the CMDB and generate some data for use by the root cause analysis model. The output file is `spark/data/cmdb.jsonl`.

```
~/ml-monitoring$ python3 spark/generate_cmdb.py
```

3.4 Model Training Dataset Preparation

Prepare a semantic and structural training dataset. Each contains approximately 5,000 rows. After execution, the following two directories will be generated:

spark/data/semantic_data/
spark/data/structured_data/

```
root@spark-master:~# run_spark semantic_preprocessing.py

root@spark-master:~# run_spark structure_preprocessing.py
```

3.5 Test Set (labeled parquet) Generation

A total of **1000 log samples, which included 50 anormalies** were collected from Kafka using an automated Spark streaming script. This set includes at least 50 exception messages. Output file is `spark/data/test_labeled_parquet`.

```
root@spark-master:~# run_spark generate_testset.py
```

3.6 Experiment / Case Study 1 - Traditional vs Machine Learning-based Monitoring

An experiment comparing traditional monitoring systems and intelligent monitoring systems. After successful execution, the result is in the **chaos/result** folder.

```
root@spark-master:~# run_spark semantic_train.py
```

Important Prerequisite: For any scripts in this experiment that contain *inference.py, it is recommended to run them in a separate window. This is because real-time inference tasks run persistently within Spark.

```
root@spark-master:~# run_spark semantic_inference.py
```

```
$ ./chaos/run_chaos
```

3.7 Experiment / Case Study 2 - Structured Data vs Semantic Embedding

A comparison of model performance between structured processing and semantic processing methods. After successful execution, the result is in the **spark/data/metrics** folder.

```
root@spark-master:~# run_spark semantic_train.py
root@spark-master:~# run_spark structure_train.py
root@spark-master:~# run_spark evaluate_models.py
```

3.8 Experiment / Case Study 3 - Root Cause Analysis and Feedback Learning

This part of the experiment includes the workflow of the root cause analysis model and a performance comparison before and after training the root cause analysis model using the feedback dataset.

3.8.1 Root Cause Process

Real-time inference must be running. See Section 3.6 for the Important Prerequisite.

```
root@spark-master:~# run_spark root_cause_inference.py
```

```
$ tail -n 1 spark/data/anomaly.jsonl
{
```

```
"timestamp": "2025-11-30T03:14:48.004448",
"semantic_text": "remote_addr=192.168.65.1 request_method=GET
request_uri=/
status=200 body_bytes_sent=16
http_referer= http_user_agent=python-requests/2.32.3",
"prediction": "low_anomaly",
"mse": 0.34809,
"threshold": 0.212097
}

$ python spark/feedback_labeler.py --label true
--timestamp 2025-11-30T03:14:48.004448
```

3.8.2 Feedback Learning Process

For the feedback dataset, we have already simulated three data points and placed them in feedback_sample.jsonl. Simply execute the following code.

```
root@spark-master:~# run_spark root_cause_train.py
root@spark-master:~# run_spark evaluate_root.py
```

4 Automated Execution (Fast Reproduction Workflow)

4.1 Install environment

```
$ ./00_setup_environment.sh
```

4.2 Prepare All dataset

```
$ ./01_prepare_dataset.sh
```

4.3 Train all models

```
$ ./02_train_models.sh
```

4.4 Experiment / Case Study 1 - Traditional vs Machine Learning-based Monitoring

Real-time inference must be running. See Section 3.6 for the Important Prerequisite.

```
root@spark-master:~# run_spark semantic_inference.py
```

```
$ ./chaos/run_chaos
```

4.5 Experiment / Case Study 2 - Structured Data vs Semantic Embedding

```
root@spark-master:~# run_spark evaluate_models.py
```

4.6 Experiment / Case Study 3 - Root Cause Analysis and Feedback Learning

4.6.1 Root Cause Process

See Section 3.8.1 for the Root Cause Process.

4.6.2 Feedback Learning Process

```
root@spark-master:~# run_spark evaluate_root.py
```

References

Jain, S. and Kumar, P. (2024). Devops practices into machine learning, *2024 IEEE International Conference on Intelligent Systems, Smart and Green Technologies (ICISSGT)* pp. 97–101.

Yudha Erian Saputra, M., Noprianto, Noor Arief, S., Nur Wijayaningrum, V. and Syaifudin, Y. W. (2024). Real-time server monitoring and notification system with prometheus, grafana, and telegram integration, *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)* pp. 1808–1813.