National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Muhammad Arsil Khan
Student ID: x23308737

School of Computing
National College of Ireland

Supervisor: Ahmed Makki

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Muhammad Arsil Khan
Student ID:	x23308737
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Ahmed Makki
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	863
Page Count:	3

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:	Muhammad Arsil Khan
Date:	10th August 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

Muhammad Arsil Khan
x23308737

This manual will provide a guide on how to configure the environment and run the Python code provided to reproduce the work done in this project.

Prerequisites

1) AWS Account with access to EC2 and Security Groups.
2) Azure account with access to Virtual machine.
3) Python 3.10
4) Any IDE (VS code, PyCharm)

1 Setup AWS EC2 Instance

1) Create a new AWS instance (t2.large) with Amazon Linux using AWS console.
2) Install and start docker as a service using the following commands:
    1) sudo apt update && sudo apt install docker.io -y
    2) sudo systemctl start docker && sudo systemctl enable docker
3) Install and start Prometheus to collect metrics using the following commands:
    1) wget https://github.com/prometheus/prometheus/releases/download /v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz
    2) tar xvf prometheus-2.52.0.linux-amd64.tar.gz
    3) cd prometheus-2.52.0.linux-amd64
    4) nohup ./prometheus –config.file=prometheus.yml &

    This will help in installation and running Prometheus on the machine. Replace the prometheus.yml file with the one provided in code artifact so it can connect to the flask apps.
4) Install and start Node exporter to collect system metrics using the following commands:
    1) wget https://github.com/prometheus/node_exporter/releases/download /v1.8.1/node_exporter-1.8.1.linux-amd64.tar.gz

2) tar xvf node_exporter-1.8.1.linux-amd64.tar.gz

3) cd node_exporter-1.8.1.linux-amd64

4) nohup ./node_exporter &

This will help in installation and running node exporter on the machine.

2 Setup Azure VM instance

1) Create an Azure instance with Linux installed in it using Azure management console.

2) Follow the steps mentioned in previous section to setup docker, prometheus and node exporter.

3 Run RL-based Service

To run the RL-based service on your machine (AWS or Azure) copy the "RL-App" folder provided in the code-artifacts to any directory of your machine. The folder contains all the necessary files to build the docker image. Step into this folder and run the following command to build a docker image of RL-based service.

docker build -t llm-rl .

The dockerfile in the folder contains necessary commands to copy required files to the image working directory and dependencies required to run the image.

Once the image build is complete, run the following command to start a docker container that will use the image created in the above step.

docker run -d –network host -p 80:80 llm-rl

This will run the llm-rl image in the container on port 80 accessible on the network.

4 Run Static Interval-based Service

To run the static interval-based service on your machine (AWS or Azure) copy the "Static-App" folder provided in the code-artifacts to any directory of your machine. The folder contains all the necessary files to build the docker image. Step into this folder and run the following command to build a docker image of static interval-based service.

docker build -t llm-static .

The dockerfile in the folder contains necessary commands to copy required files to the image working directory and dependencies required to run the image.

Once the image build is complete, run the following command to start a docker container that will use the image created in the above step.

docker run -d –network host –rm -e STATIC_MODE=true -e STATIC_INTERVAL=900 -p 8080:8080 llm-static:latest

This will run the llm-static image in the container on port 8080 accessible on the network.

5 Run Load Test Script

To reproduce the experiment and get the results, run the "load_test.py" script available in code-artifacts. To run the script you can either use terminal or an IDE of your choice such as Visual Studio Code. Run the following command in the terminal and it will execute the "load_test.py" and call the RL-based and interval-based services for the specified time and produce workload.

python load_test.py –static-url http://[server-ip]:8080/predict –rl-url http://[server-ip]:80/predict –rps [requests per second] –duration [duration in seconds] –prompt [testing prompt] –output-csv comparison.csv

Replace the placeholders with the values of your choice and press enter. This will start the load test script and it will run for the time specified in the command.

6 Collect Metrics

Once the load_test.py script has ran successfully, "export_metrics.py" can be used to collect the results from the Prometheus. The below command will execute the script.

python export_metrics.py –start "[date time]" –end "[date time]" –outdir ./csv_exports

This script will fetch the results and store the in their respective csv files in csv_exports folder in the same directory. These files can further be used for plotting the results.

7 Visualize Results

To visualize the results and plot graphs for retrain and performance comparison "plots.ipynb" can be used available in code-artifacts. Open the file in the IDE preferably VS Code. The file will read the export metrics and plot them. There will be multiple code sections in the file. Run them sequentially to avoid any errors. Further analysis can be done using these plots and files.