National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cloud Computing

Yash Santosh Bidkar
Student ID: X23232251

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: …Yash Santosh Bidkar……

Student ID: …x23232251……..……

Programme: …MSc in Cloud Computing…… **Year:** …2024-25……..

Module: …MSc Research Project…….……

Lecturer: …Sean Heeney…….……

Submission Due Date: …11th December 2025…….……

Project Title: …Latency-Aware Edge-Cloud Collaboration For Real Time Video Analytics.……

Word Count: …1735…… **Page Count:** …5……..……

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: …Yash Santosh Bidkar……

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

Yash Santosh Bidkar
Student ID: x23232251
Video Presentation- https://youtu.be/BBbgEYOkzP0

1 Introduction

As part of the "Latency Aware Edge-Cloud Collaboration for Real-Time Video Analytics" project, this Configuration Manual serves as your guide to configure the required software environment, Amazon Web Services (AWS) cloud resources and parameters that govern how the system operates.

The end user of the manual will include, but is not limited to; Researchers, Developers and System Administrators, looking to recreate experiments performed, develop additional features and deploy the system elsewhere. Following the instructions within this document is essential in achieving success in executing and reproducing the research.

2 System Prerequisites

Before proceeding with the configuration, ensure the following software is installed and accessible from the system's command line interface:

- **Python:** Version 3.8 or higher.
- **pip:** The Python package installer.
- **AWS Command Line Interface (CLI):** Version 2. The CLI is required for authenticating with AWS.
- **Docker and Docker Compose:** Required for running the resource-constrained edge experiments.
- **Git:** For version control and cloning the project repository.

Additionally, an active Amazon Web Services (AWS) account with programmatic access and permissions to create and manage EC2 instances, security groups, and key pairs is mandatory.

3 Environment Configuration (.env file)

The primary configuration for the project is managed through an environment variable file named .env located in the root directory of the project. This file must be created before running any setup scripts.

Create a file named .env and populate it with the following key-value pairs.

```
# --- AWS Configuration ---
# Specifies the AWS region for resource deployment.
# This must match the region where the EC2 Key Pair is created.
AWS_REGION=eu-north-1

# The name of the EC2 Key Pair file (without the .pem extension).
# This must exactly match the key pair name created in the AWS console.
KEY_PAIR_NAME=cloud-key-pair

# The EC2 instance type for the cloud node.
# t3.small is recommended for balancing cost and performance.
INSTANCE_TYPE=t3.small
```

# The Amazon Machine Image (AMI) ID for the EC2 instance.
# This ID is specific to the region (eu-north-1) and OS (Amazon Linux 2023).
AMI_ID=ami-0c7d68785ec07306c

# --- Cloud Server Configuration ---
# The network port on which the cloud inference server (Flask API) will listen.
# The security group must allow inbound traffic on this port.
CLOUD_API_PORT=8000
The table below provides a detailed description of each variable in the .env file.

Table 1: Environment Variable Descriptions

Variable	Description	Default Value	Example
AWS_REGION	The AWS geographical region for deploying the EC2 instance and other resources.	eu-north-1	us-east-1
KEY_PAIR_NAME	The name assigned to the EC2 key pair used for SSH access.	cloud-key-pair	my-project-key
INSTANCE_TYPE	The hardware profile of the EC2 instance. Affects cost and performance.	t3.small	t2.micro
AMI_ID	The unique identifier of the Amazon Machine Image to launch the instance.	ami-0c7d68785ec07306c	ami-xxxxxxxxxxxxxxxxx
CLOUD_API_PORT	The TCP port for the cloud inference server API.	8000	5000

4 AWS Configuration

Proper configuration of the AWS environment is essential for the Cloud Node. Configuring AWS Configuration will be done in 2 steps: Configure the AWS CLI with your user credentials and create an EC2 Key Pair for secure access to your instances.

4.1 AWS CLI Configuration

The AWS CLI provides the underlying authentication mechanism that is used by the scripts in this project to provision resources.

1. **Generate Access Keys:** Sign in to the AWS IAM Console, select your user account, click the Security Credentials tab, and create a new access key. Keep the Access Key ID and Secret Access Key securely stored.
2. **Configure the CLI:** Open a terminal or command prompt and enter the following command to configure the AWS CLI:

aws configure

3. You will be asked for credentials by the command window to input 4 points of information. You'll enter the info like this.

- **AWS Access Key ID -** You will paste the AWS Access Key ID that you copied from step one.
- **AWS Secret Access Key** - You will copy and paste the AWS Secret Access Key that you copied from step one.

- **Default Region Name -** You'll put the same region here as you specified in your .env file (For example: eu-north-1)
- **Default Output Format -** You'll hit enter and accept the default option (JSON).

4. **Confirm Your Configuration:** To confirm you have correctly authenticated via the CLI, execute the following command as shown (You will see your account details).

```
aws sts get-caller-identity
```

4.2 Create an EC2 Key Pair

A key pair needs to be created for secure access to an EC2 instance via SSH.

1. **Go to the EC2 console:** Access the OSS Management console and navigate to the EC2 service. When viewing the EC2 service in the console, ensure that the region selected in the upper-right corner of your screen matches the AWS Region specified in your .env file.

2. **Create Key Pair:** In the left-hand navigation pane, under "Network & Security," select "Key Pairs." Click "Create key pair."
3. **Configure Key Pair:**
    - **Name:** Enter the exact name specified for KEY_PAIR_NAME in the .env file (e.g., cloud-key-pair).
    - **Key pair type:** Select **RSA**.
    - **Private key file format:** Select **.pem** (for use with OpenSSH).
4. **Download and Secure:** Click "Create key pair." The private key file (e.g., cloud-key-pair.pem) will be automatically downloaded. **Move this file into the root directory of the project.** This is the only time this file can be downloaded.
5. **(For Linux/macOS Users) Set File Permissions:** For security, SSH requires that the private key file is not publicly viewable. Set the correct permissions by running the following command in the project's root directory:

```
chmod 400 cloud-key-pair.pem
```

5 Experimental Parameter Tuning

Several key parameters that control the behavior of the schedulers and the experiments are defined as constants within the Python source code. To modify the experimental conditions, these files must be edited directly.

The table below lists the most important tunable parameters, their location, and their function.

Table 2: Key Experimental Parameters

Parameter	File Location	Description	Default Value
latency_threshold	scheduler.py, run_experiments.py	The end-to-end latency (in ms) above which the scheduler will prefer edge processing. It acts as the deadline.	150
frame_rate	run_experiments.py	The target frames per second	10

		(FPS) for the video stream processing loop.	
warm_up_frames	run_experiments.py	The number of initial frames to process without collecting metrics, allowing the system to stabilize.	10
history_size	latency_predictor.py	The number of recent latency measurements the ARIMA model uses for prediction.	60
arima_order	latency_predictor.py	The (p, d, q) parameters for the ARIMA model. Tuning these can affect prediction accuracy.	(2, 1, 2)
min_samples	latency_predictor.py	The minimum number of latency samples required before the ARIMA model is used (falls back to mean otherwise).	10
CONFIDENCE_THRESHOLD	cloud_inference_server.py, edge_inference.py	The confidence score threshold for object detections.	0.5

		Detections below this value are discarded.	

6 Docker Environment Configuration

The docker-compose.yml file is used to configure and run the experiment within a resource-constrained Docker container, simulating a realistic edge device.

The most critical section for configuration is the edge-node service definition:

```
services:
  edge-node:
    build: .
    container_name: edge-inference-node
    # --- Resource Constraints ---
    cpus: '2.0'  # Restrict to 2 virtual CPUs
    mem_limit: 4g  # Restrict to 4 GB of RAM
    # --------------------------
    volumes:
      - ./videos:/app/videos
      - ./results:/app/results
      - ./logs:/app/logs
    # ...
```

To modify the simulated edge device's capabilities, adjust the following parameters:

- cpus: Defines the number of CPU cores the container is allowed to use. Reducing this value (e.g., to '1.0') will increase local inference time and stress the scheduler more.
- mem_limit: Sets a hard limit on the amount of memory the container can use.

7 Resetting and Cleanup

It is crucial to properly clean up the cloud resources after completing the experiments to avoid incurring unnecessary AWS charges.

7.1 Resetting Local Configuration

In order for the installation scripts to create a new EC2 instance instead of using an existing one, you must delete the file aws_config.json located in the root directory of your project. If you run the Python script aws_setup.py again after doing this, the provisioning process will be performed in full.

7.2 Terminating Your Cloud Resources

Terminating your EC2 instance is by far the easiest method of clean-up. Please note that terminating your EC2 instance cannot be undone and will permanently remove the instance and its associated data from existence.

1. **Find the Instance ID:** Open the aws_config.json file to find the ID of the running instance (e.g., "instance_id": "i-0c938c6209973e553").
2. **Terminate via AWS CLI:** Run the following command, replacing <instance_id> with the ID from the previous step:

```
aws ec2 terminate-instances --instance-ids <instance_id>
```

3. **Terminate via AWS Console:** Alternatively, navigate to the EC2 Dashboard in the AWS Console, select the instance, click "Instance state," and choose "Terminate instance."

Note: The security group created by the script can also be deleted from the "Security Groups" section of the EC2 Console, but it does not incur charges if left idle.