National College of Ireland

Configuration Manual

MSc Research Project
MSCCLOUD_B

Sai Varshitha Sanagari
Student ID: 23131012

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	SAI VARSHITHA SANAGARI
Student ID:	X23131012
Programme:	MSCCLOUD_B **Year:** 2024-2025
Module:	RESEARCH IN COMPUTING
Lecturer:	SHAGUNA GUPTA
Submission Due Date:	28-01-2025
Project Title:	Design and Implementation of a Hybrid Cloud-Edge Computing Architecture for Real-Time IoT Data Processing with Blockchain Integration
Word Count:	1234 **Page Count:** 11

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 VARSHITHA SANAGARI
Date:	28-01-2025

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

1 System Setup

In this section we describe the configuration necessary to run the hybrid cloud edge computing architecture with blockchain support to deploy, test. The hardware and software configuration is part of the setup, and it is required to be the successful implementation of the system.

1.1 Hardware Configuration

The system requires the following hardware components:

1. **Edge Devices:** Prove that there is no lullaby on Earth that could relax the brains of Mr Snowden and Mr Cole. To accomplish this, we suggest the use of AWS EC2 t2.medium instances.

2. **Cloud Nodes:** For the computationally intensive tasks use AWS EC2 m5.large instances.

3. **IoT Devices:** Data for sensors (e.g., temperature and humidity) are generated by scripts such that they are simulated.

The network setup must have API Gateway which sits in the gap between the IoT devices and the processing nodes. Make sure that all devices are connected to the internet through a safe and stable connection.

1.2 Software Configuration

Install and configure the following software:

- **AWS Cloud Services:** Setup Amazon S3, Lambda, RDS, API Gateway, Athena. Each service must be correctly provisioned using AWS Identity and Access Management (IAM), correctly provisioned and the permissions must be assigned.

- **Blockchain Framework:** For data integrity and security use Hyperledger Fabric. Item 1: Set up 2 - 4 peer nodes, a certificate authority (CA) and smart contracts (chaincode).

- **Programming Environments:** To code task scheduling algo and Hyperledger Fabric chaincode in Hyperlode Fabric, python for task scheduling algorithm and Golang is installed.

- **Containerization:** Embedded your Hyperledger Fabric components using Docker to deploy and manage.

- **Development Tools:** Performance analysis and visualisation Jupyter Notebooks.

To ensure all installations run smoothly and securely try ensuring all installations are running on their latest stable versions.

2 Implementation Steps

In this section, the step by step procedure to implement the system and deploy its components is delineated.

2.1 Task Scheduling Setup

1. Using Python develop the Weighted Round Robin (WRR) algorithm.

    - We assign weights to edge and cloud nodes based on their computational capacities.

    - The algorithm needs to be integrated into AWS Lambda functions.

2. Make WRR algorithm dynamic and configure AWS Lambda to allocate tasks in that fashion.

    - Edge nodes do low latency tasks.

    - Cloud nodes are told to do computationally intensive tasks.

3. Simulate task loads and monitor task distribution with the algorithm.

2.2 Blockchain Integration

1. Deploy Hyperledger Fabric using Docker:

    - Set up two organizations (Org1 and Org2), with peer nodes and a CA, configured.

    - Efficiently validate transaction with Raft consensus mechanism set up.

2. Write and deploy smart contracts (chaincode) in Go:

    - Implement functions for adding and querying data on the blockchain.

        - Enforce access control policies and validate data integrity.

3. Log IoT data transactions by connecting blockchain to AWS Lambda.

2.3 Data Storage and Querying

1. Store raw IoT data into Amazon S3.
2. Store structured data in Amazon RDS.
3. Query processed data stored in S3, RDS and use Amazon Athena.
    - When it comes to analyzing data, create schemas and SQL queries as the most effective ways.
4. Test the end-to-end data flow from ingestion to storage and querying.

3 Performance Evaluation

The third section describes the procedure to compute the system performance with respect to key metrics.

3.1 Latency Testing

1. Use scripts to simulate the IoT sensor data as real time.
2. Time tasks are processed edge and cloud nodes are measured.
    - We capture task initiation and completion timestamps using logging tools.
3. Find out how low latency and high computation tasks compare.

3.2 Resource Utilization Analysis

1. Measuring CPU + memory usage of edge and cloud nodes during task execution.
    - Track resource usage on AWS CloudWatch.
2. Analyze the resource usage trends to evaluate effort of task allocation.

3.3 Scalability Testing

1. Simulate from 10 to 1000 of IoT devices slowly, with an increase in those.
2. We observe the performance of the system in terms of latency as well as throughput.
    - Make sure that the WRR algorithm can automatically adapt to load increase.

3. Another example of why to run Machine Learning models in the cloud: Identify any bottlenecks too and optimize task scheduling or resource scaling.

3.4 Blockchain Security Validation

1. Making any unpermitted changes to the ledger of the blockchain.

2. Verify that Hyperledger Fabric prevents data tampering and enforces data integrity.

3. Test audit trail functionality by retrieving and validating transaction logs.

References

Bandyopadhyay, S., & Sen, J. (2011). Internet of Things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1), 49–69.

Chiang, M., & Zhang, T. (2016). Fog and IoT: An overview of research opportunities. *IEEE Access*, 4, 7886–7899.

Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation Review*, 2, 6–10.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660.

Lin, C. Y., Lee, S. G., & Lee, T. Y. L. (2017). A weighted round-robin scheduling algorithm for cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, 6(1), 1–14.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Retrieved from https://bitcoin.org/bitcoin.pdf

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1), 30–39.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5), 637–646.

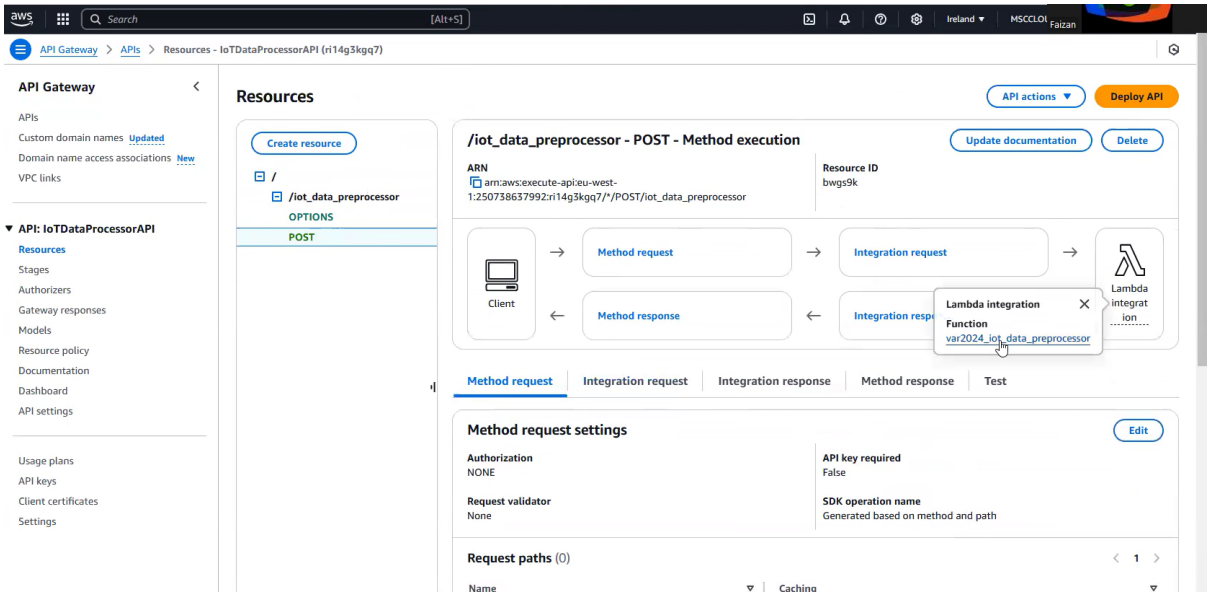
Zanella, A., Bui, N., Castellani, A. P., Vangelista, L., & Zorzi, M. (2014). Internet of Things for smart cities. *IEEE Internet of Things Journal*, 1(1), 22–32.

Zhang, L., Li, M., Li, Z., & Lin, X. (2017). Towards a hybrid edge-cloud architecture for IoT applications: A case study of video surveillance. *IEEE Access*, 5, 13343–13354.

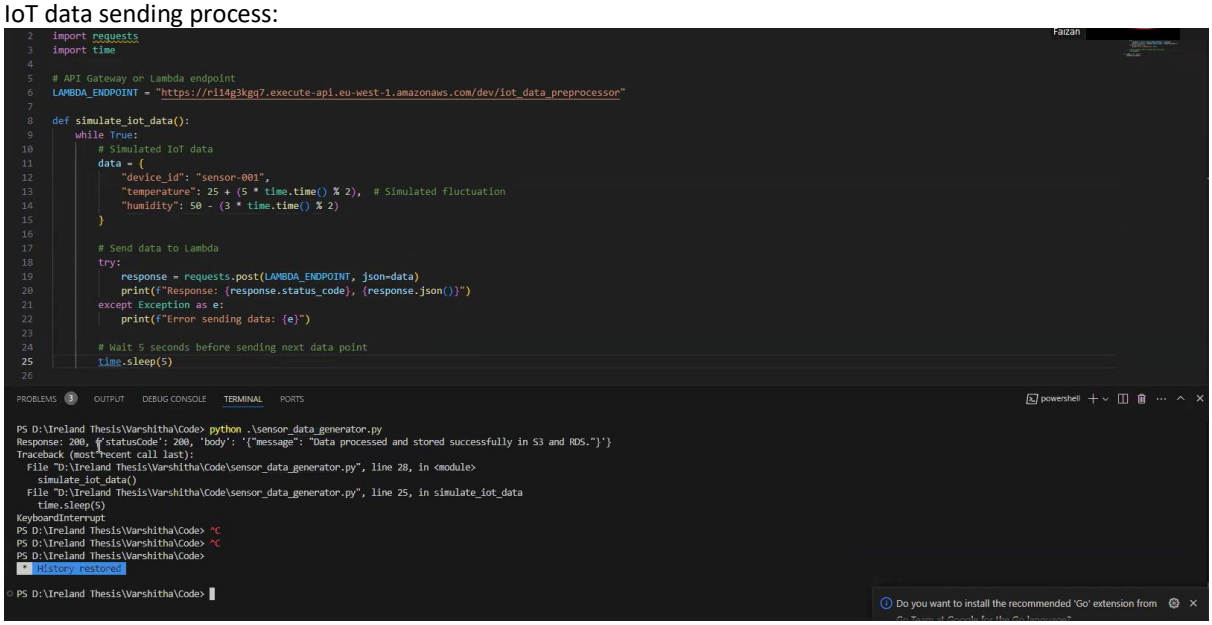
4 Appendix

Screenshots of the setup:

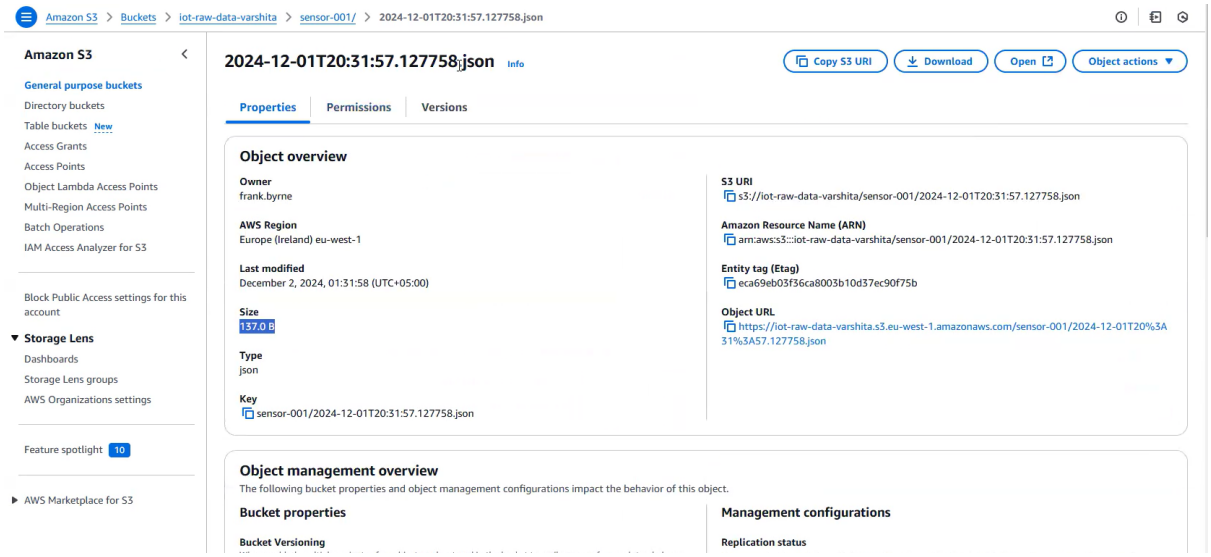
API gateway method for receiving iot data:



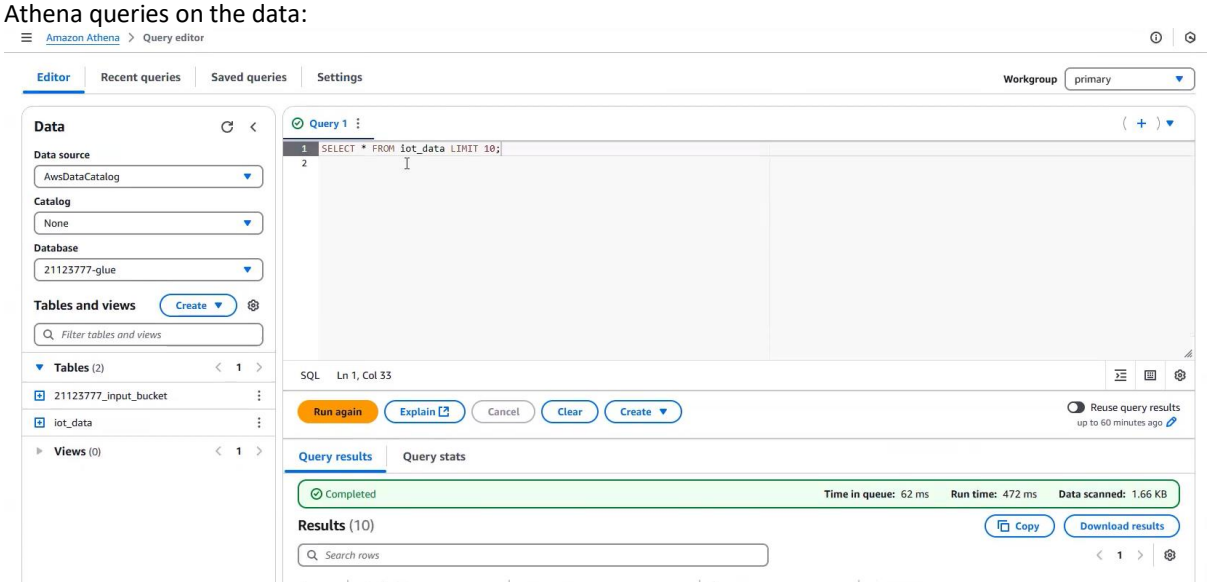
IoT data sending process:



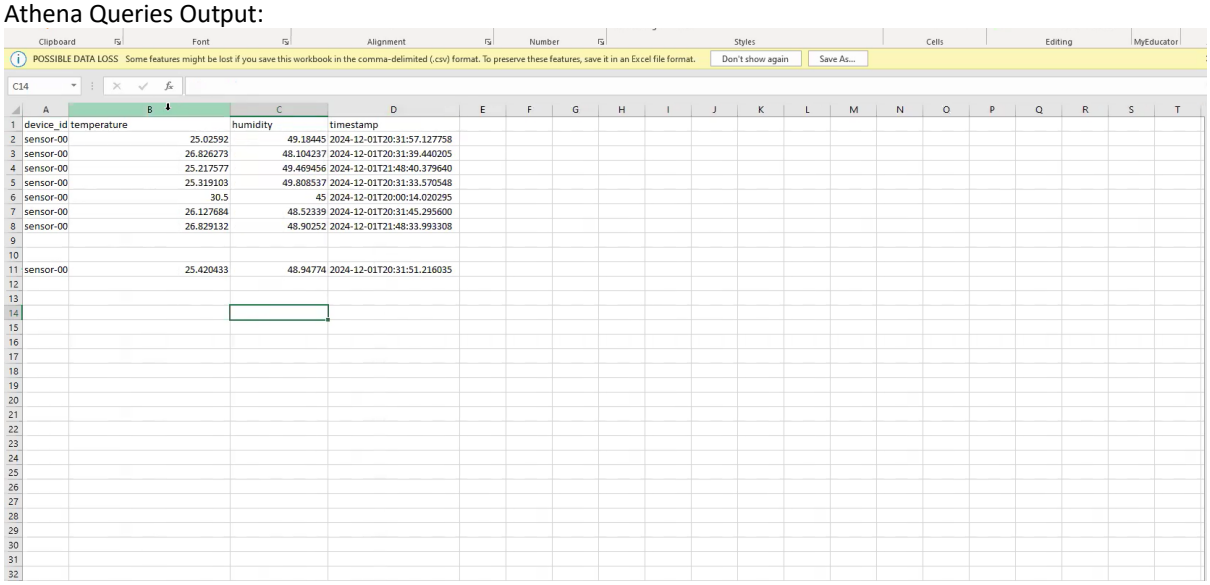
Data stored as JSON on aws s3 buckets:



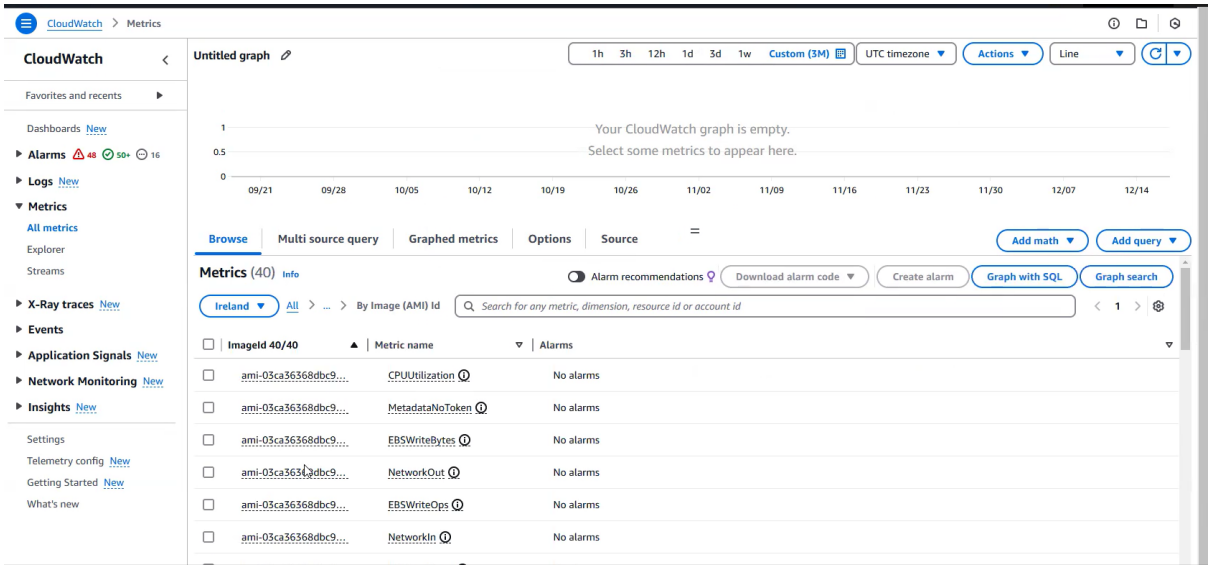
Athena queries on the data:



Athena Queries Output:



Cloudwatch evaluation metrics:


CloudWatch > Metrics
CloudWatch
Favorites and recents
Dashboards New
Alarms 48 50+ 16
Logs New
Metrics
All metrics
Explorer
Streams
X-Ray traces New
Events
Application Signals New
Network Monitoring New
Insights New
Settings
Telemetry config New
Getting Started New
What's new
Untitled graph
1h 3h 12h 1d 3d 1w Custom (3M)
UTC timezone
Actions
Line
Your CloudWatch graph is empty.
Select some metrics to appear here.
09/21 09/28 10/05 10/12 10/19 10/26 11/02 11/09 11/16 11/23 11/30 12/07 12/14
Browse
Multi source query
Graphed metrics
Options
Source
Add math
Add query
Metrics (40) Info
Alarm recommendations
Download alarm code
Create alarm
Graph with SQL
Graph search
Ireland
All
By Image (AMI) Id
Search for any metric, dimension, resource id or account id
ImageId 40/40
Metric name
Alarms
ami-03ca36368dbc9...
CPUUtilization
MetadataNoToken
EBSWriteBytes
NetworkOut
EBSWriteOps
NetworkIn
No alarms