National College of Ireland

Configuration Manual

MSc Research Project

MSc in Cloud Computing

Avinash Ashok Lone

Student ID: x23305347

School of Computing

National College of Ireland

Supervisor: Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Avinash Ashok Lone
Student ID:	x23305347
Programme:	MSc in Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Diego Lugones
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	1044
Page Count:	8

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:	Avinash lone
Date:	11/08/2025

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):	

Contents

1 Introduction

1.1 Purpose of This Manual

This document provides detailed, step-by-step instructions for the configuration and deployment of the "Blockchain-Enabled Cloud Infrastructure Management" system. It is intended for system administrators, DevOps engineers, and developers who wish to replicate the proof-of-concept environment.

Following this guide will result in a fully functional system capable of monitoring an AWS EC2 instance, autonomously reacting to performance anomalies, and creating an immutable audit trail of its actions on a private blockchain.

1.2 System Overview

The system architecture is composed of four distinct layers:

- **Cloud Infrastructure Layer:** The AWS environment containing the monitored EC2 instance and related services (CloudWatch, IAM).
- **Automation Logic Layer:** A Python script that acts as the system's "brain", monitoring metrics and triggering actions.
- **Blockchain Trust Layer:** A local Ethereum blockchain (Ganache) and a Solidity smart contract that serves as the immutable ledger.
- **Audit and Visualization Layer:** A Flask web application that provides a user-friendly interface to view the blockchain logs.

This manual will guide the user through the setup of each of these layers in a logical sequence.

1.3 Prerequisites

Before beginning the configuration process, ensure the following software is installed and accessible from the command line on the administrator's machine:

- **AWS CLI:** The Amazon Web Services Command Line Interface.
- **Node.js and npm:** Required for installing the Truffle framework.
- **Truffle Suite:** A development framework for Ethereum.
- **Ganache:** A local personal blockchain for Ethereum development.
- **Python 3.8+ and pip:** For running the automation and web application scripts.

Additionally, the user must have access to an AWS account with administrative privileges.

2 Cloud Environment Setup (AWS)

This section details the provisioning of all necessary AWS resources. The selected region for this setup is ‘eu-north-1‘ (Stockholm).

2.1 AWS CLI Configuration

Before executing any commands, the AWS CLI must be configured with administrative credentials.

```
aws configure
```

The CLI will prompt for an AWS Access Key ID, Secret Access Key, Default region name (enter ‘eu-north-1‘), and Default output format (enter ‘json‘).

2.2 Security Group and Key Pair

These resources provide secure network access and SSH login capabilities.

2.2.1 Create an EC2 Key Pair

This command creates a key pair for SSH access. The private key material is saved to ‘ec2keypair.pem‘.

```
aws ec2 create-key-pair --key-name ec2keypair --query '
  ↪ KeyMaterial' --output text > ec2keypair.pem
```

After running this command, secure the key file’s permissions (e.g., on Linux/macOS, use ‘chmod 400 ec2keypair.pem‘).

2.2.2 Create a Security Group

This command creates a security group to act as a virtual firewall.

```
aws ec2 create-security-group \
  --group-name cloud-monitor-sg \
  --description "Security group for EC2 instance" \
  --region eu-north-1
```

Note: Record the ‘GroupId‘ from the output for your reference.

2.2.3 Authorize SSH Ingress

This command adds a rule to allow inbound SSH traffic on port 22.

```
aws ec2 authorize-security-group-ingress \
  --group-name cloud-monitor-sg \
  --protocol tcp \
  --port 22 \
  --cidr 0.0.0.0/0 \
  --region eu-north-1
```

2.3 Launching the EC2 Instance

This command launches the virtual machine that will be monitored.

```
aws ec2 run-instances \
  --image-id ami-09278528675a8d54e \
  --count 1 \
  --instance-type t3.micro \
  --key-name ec2keypair \
  --security-groups cloud-monitor-sg \
  --monitoring Enabled=true \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Role,
  ↪ Value=Monitored}]" \
  --region eu-north-1
```

Crucial Step: From the JSON output, locate and save the 'InstanceId' (e.g., 'i-0a816ee0e9b52dcbd'). This ID is required later.

2.4 IAM Role for Secure In-System Access

An IAM Role allows the system to make AWS API calls securely.

2.4.1 Create the Trust Policy

Create a local file named 'ec2-trust-policy.json' with the following content. This policy allows the EC2 service to assume the role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": { "Service": "ec2.amazonaws.com" },
            "Action": "sts:AssumeRole"
        }
```

```
    ]
}
```

2.4.2 Create the IAM Role

```
aws iam create-role \
  --role-name CloudMonitorRole \
  --assume-role-policy-document file://ec2-trust-policy.json
```

2.4.3 Attach Required Policies

```
aws iam attach-role-policy \
  --role-name CloudMonitorRole \
  --policy-arn arn:aws:iam::aws:policy/
  ↪ CloudWatchReadOnlyAccess

aws iam attach-role-policy \
  --role-name CloudMonitorRole \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2FullAccess
```

2.4.4 Create and Associate an Instance Profile

An instance profile is a container for the role.

1. Create the instance profile:

```
aws iam create-instance-profile \
  --instance-profile-name CloudMonitorProfile
```

2. Add the role to the profile:

```
aws iam add-role-to-instance-profile \
  --instance-profile-name CloudMonitorProfile \
  --role-name CloudMonitorRole
```

3. Associate the profile with the EC2 instance. Replace ‘¡YOUR$_{I}NSTANCE_{I}D >$ ‘$withtheIDsavedearlier$.

3 Blockchain Environment Setup

3.1 Project Setup

(a) Create a main project directory. Inside it, create a subdirectory named ‘cloud-chain‘.

(b) Navigate into the ‘cloud-chain‘ directory.

(c) Initialize a new Truffle project:

```
truffle init
```

(d) Inside the ‘contracts/‘ folder, create ‘CloudActionLogger.sol‘ and populate it with the Solidity source code.

(e) Inside the ‘migrations/‘ folder, create ‘2_deploy_contracts.js‘ and populate it with the deployment script.

3.2 Smart Contract Deployment

(a) **Start Ganache.** Launch the Ganache application and ensure it runs on ‘http://127.0.0.1:7545‘.

(b) In the terminal, navigate to the ‘cloud-chain‘ directory.

(c) Compile the smart contract:

```
truffle compile
```

(d) Deploy the contract to Ganache:

```
truffle migrate --network development
```

Crucial Step: From the output, locate and save the ‘contract address‘.

4 Automation Script Configuration

4.1 Python Environment Setup

(a) In the main project directory, create a Python virtual environment:

```
python -m venv venv
```

(b) Activate the environment (e.g., on Windows: ‘venv‘).

(c) Install required libraries:

```
pip install boto3 web3
```

4.2 Script Configuration

(a) Create a Python file named ‘monitor_and_log.py‘ in the main project directory.

(b) Populate this file with the Python source code for the automation script.

(c) **Edit the configuration variables** at the top of the script:

- Set ‘INSTANCE_ID‘ to the ID saved from Section 2.
- Set ‘REGION‘ to ‘eu-north-1‘.
- Set ‘contract_address‘ to the address saved from Section 3.
- Verify the path to the ABI JSON file is correct.

4.3 Execution and Testing

4.3.1 Baseline Test (Normal Load)

Run the script while the EC2 instance is idle. The expected output is a message indicating that CPU usage is normal.

```
python monitor_and_log.py
```

4.3.2 High Load Test

(a) SSH into the EC2 instance.

(b) Install and run a stress testing tool to generate high CPU load.

```
sudo dnf install stress-ng -y
stress-ng --cpu 2 --timeout 600
```

(c) While the stress test is running, execute the monitoring script again. The expected output is a message indicating high CPU usage and a confirmation that an action was logged, including a transaction hash.

```
python monitor_and_log.py
```

5 Audit Web Interface Setup

5.1 Application Configuration

(a) In the main project directory, create ‘app.py‘.

(b) Create a subdirectory ‘templates‘, and inside it, create ‘index.html‘.

(c) Populate both files with their respective source code.

(d) **Edit the configuration variables** at the top of ‘app.py‘, setting the contract address and ABI path.

5.2 Running the Application

With the virtual environment active, run the following command.

```
flask run
```

This will start a local web server, typically at ‘http://127.0.0.1:5000‘.

5.3 Verification

Open a web browser and navigate to ‘http://127.0.0.1:5000‘. The page should display an HTML table. If the high-load test was performed, the table will contain the details of the logged action.

6 Conclusion

By following the steps outlined in this manual, a user has successfully deployed a fully functional proof-of-concept for a blockchain-enabled cloud management system. The environment consists of a monitored AWS EC2 instance, a local Ethereum blockchain acting as a secure ledger, an autonomous script capable of reacting to system metrics, and a web interface for auditing the immutable log.

This configured system serves as a powerful demonstration platform for the principles of trustworthy autonomous operations. From this baseline, the user can now explore extensions, such as implementing more complex automation logic or testing different blockchain platforms.