Configuration Manual

MSc Research Project
Cloud Computing

Amruth Kumar kori
StudentID:23308591

School of Computing
National College of Ireland

Supervisor: Aqeel kazmi

National College of Ireland

Project Submission Sheet
School of Computing

Student Name:	Amruth Kumar kori
Student ID:	23308591
Programme:	Msccloud
Year:	1 year
Module:	MSc Research Project
Supervisor:	Aqeel kazmi
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	260
Page Count:	6

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:	Amruth kumar
Date:	August 11, 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECK-LIST:

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

Amruth Kumar
23308591

1 Introduction

This manual provides the complete technical configuration and operational procedures required to deploy and manage the multi-cloud infrastructure detailed in this research. The primary objective of this document is to offer a clear, precise, and repeatable process for a qualified DevOps engineer to set up the necessary development environment, provision all cloud resources on both Amazon Web Services (AWS) and Microsoft Azure using Terraform, and execute the custom Python observability suite to gather performance data.

Following this guide meticulously will result in a fully functional and validated instance of the tactical framework. This includes IaC-managed infrastructure, fully automated CI/CD pipelines ready for future changes, and active performance monitoring capabilities. This document assumes the operator has access to the project's source code repository and has installed the prerequisite tools listed below.

2 Prerequisites

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

- **Git:** For version control and interacting with the GitHub repository.
- **Terraform (v1.6.3 or compatible):** The core IaC tool used for provisioning all cloud resources.
- **AWS CLI (v2):** The official command-line interface for interacting with AWS services.
- **Azure CLI (v2):** The official command-line interface for interacting with Azure services.

- **Python (v3.9+):** The programming language used for the custom observability suite, along with its package installer, ‘pip‘.

3 AWS Environment Configuration

This section details the step-by-step process for configuring and deploying the Amazon Web Services environment.

3.1 AWS CLI Authentication

First, configure the AWS Command-Line Interface (CLI) with the credentials for the ‘terraform-user‘ IAM user. These credentials are provided in the

```
aws configure
AWS Access Key ID [None]: <Enter Access Key from CSV>
AWS Secret Access Key [None]: <Enter Secret Key from CSV>
Default region name [None]: eu-north-1
Default output format [None]: json
```

‘terraformuser$_c$*redentials.csv‘fileandshouldbetreatedassensitive.*

AWS CLI Configuration

3.2 Python Monitoring Environment Setup

To ensure dependency isolation, a dedicated Python virtual environment must be created for the AWS monitoring scripts. Navigate to the appropriate directory and

```
cd   E:\amruth_thesis\aws\   python   -m   venv
aws_venv     .\aws_venv\Scripts\activate     pip
install -r requirements.txt
```

execute the following commands.

AWS Monitoring Environment Setup

This creates a self-contained environment named ‘aws venv‘, activates it, and installs all required Python libraries, such as ‘boto3‘ and ‘pandas‘.

3.3 Infrastructure Deployment

Navigate to the AWS Infrastructure as Code directory. The Terraform configuration is designed to use a pre-configured S3 bucket and DynamoDB table for remote state management, which facilitates team collaboration and security.

- terraform init: Initializes the working directory, downloading the AWS provider plugin and configuring the S3 backend.

- terraform plan: Creates an execution plan, showing what resources will be created, modified, or destroyed. This is a critical review step.

- terraform apply: Applies the changes to your AWS account. The ‘-autoapprove‘ flag is used here for simplicity but should be used with caution in production environments.

```
cd E:\amruth_thesis\aws-iac\ terraform init
terraform plan terraform apply -auto-
approve
```

Deploying AWS Infrastructure

3.4 Running the Unified Observability Suite

Once the infrastructure is active, execute the monitoring script to collect performance data and generate the analytical reports. The target EC2 instance ID

```
cd  E:\amruth_thesis\aws\  python
monitor.py
```

is pre-configured in the ‘.env‘ file within the ‘aws/‘ directory.

Executing AWS Monitoring Script

Upon successful execution, this script will produce three files in the ‘aws/plots/‘ subdirectory: ‘aws performance dashboard.png‘, ‘aws interactive dashboard.html‘, and ‘aws _performance report.json‘.

4 Azure Environment Configuration

This section details the setup for the Microsoft Azure environment, demonstrating the framework’s multi-cloud capabilities.

4.1 Azure CLI Authentication

Authentication with Azure is handled via an interactive browser-based login, which is more secure than storing user credentials locally. After logging in, set the active

```
az login
az account set --subscription <Your-Subscription-ID>
```

subscription to the one intended for this project.

Azure CLI Login

4.2 Python Monitoring Environment Setup

As with the AWS setup, a separate Python virtual environment is required for the

```
cd  E:\amruth_thesis\azure\  python  -m  venv
azure_venv
.\azure_venv\Scripts\activate pip install -r
requirements.txt
```

Azure monitoring tools to avoid dependency conflicts.

Azure Monitoring Environment Setup

4.3 Infrastructure Deployment

Navigate to the Azure IaC directory. The deployment process requires passing the content of your public SSH key as an input variable for the virtual machine. It is

```
cd E:\amruth_thesis\azure-iac\ terraform init terraform plan -
var="admin_ssh_public_key=< content_of_your_public_key>"
terraform apply -auto-approve -var="admin_ssh_public_key=<
      content_of_your_public_key>"
```

best practice to manage this variable securely.

Deploying Azure Infrastructure

4.4 Running the Unified Observability Suite

Execute the Azure monitoring script. The target VM details are specified in the '.env'

```
cd   E:\amruth_thesis\azure\   python
monitor.py
```

file located in the 'azure/' directory.

Executing Azure Monitoring Script

This will produce a parallel set of outputs in the azure/plots/ directory: azure vm _performance _dashboard.png, azure interactive dashboard.html, and azure performance report.json.

This will produce a parallel set of outputs in the azure/plots/ directory: azure

_vm_performance_dashboard.png, azure_interactive_dashboard.html, and azure_performance_report.json.

5 CI/CD Pipeline Configuration

The automation pillar of the framework relies on GitHub Actions. For the workflows to authenticate securely with the cloud providers, secrets must be configured in the project's GitHub repository settings under Settings > Secrets and variables > Actions. The required secrets and their purposes are detailed in Table 1.

Table 1: Required GitHub Actions Secrets

Secret Name	**Platform**	**Description**
AWS ACCESS KEY ID	AWS	The access key for the dedicated 'terraform-ci' IAM user.
AWS SECRET ACCESS KEY	AWS	The secret key for the dedicated 'terraform-ci' IAM user.
ARM CLIENT ID	Azure	The Application (client) ID of the Azure Service Principal used for automation.
ARM CLIENT SECRET	Azure	The password (client secret) generated for the Service Principal.
ARM SUBSCRIPTION ID	Azure	The unique identifier of the target Azure subscription.
ARM TENANT ID	Azure	The Directory (tenant) ID of the

		Azure Active Directory instance.
AZURE SSH PUBLIC KEY	Azure	The public SSH key content required for creating the Azure virtual machine.

6 Conclusion

By completing the procedures outlined in this manual, an operator can successfully instantiate the entire multi-cloud environment described in this paper. The infrastructure on both AWS and Azure is deployed and managed via a consistent, version-controlled, and automated process. The use of remote state backends ensures secure and effective collaboration. Furthermore, the CI/CD pipelines are fully configured to automate future infrastructure changes, and the custom observability suite is operational, capable of generating standardized performance reports for both distinct cloud platforms. This complete setup represents a working, tangible, and validated instance of the proposed tactical framework.