National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cloud Computing

Roshan Jawahar
Student ID: x23128356

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Roshan Jawahar
Student ID:	x23128356
Programme:	MSc in Cloud Computing **Year:** 1
Module:	MSc Research Project
Lecturer:	*Abubakr Siddig*
Submission Due Date:	12/12/2024
Project Title:	Multi-Cloud Deployment and Performance Benchmarking of a Dockerized Fake News Detection Application
Word Count:	876 **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: Roshan Jawahar

Date: 12/12/2024

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

Roshan Jawahar
x23128356

1 Dataset Pre-Processing and Model Training:

Step 1: Dataset should be uploaded onto Google Drive

Step 2: Create a Google Colab notebook and link it with Google Drive

Step 3: Perform data cleaning, data visualization tasks.

Step 4: Train the Machine Learning models using the pre-processed dataset.

Step 5: Save the best model for creating the application with the model as its base.

Google Colab

2 Application Creation

Step 1: Using pip, install Flask onto the work environment, VS Code.

Step 2: create a python file “app.py”

Step 3: Develop the application by creating the UI and configuring the Fake news classification using the Machine Learning model.

Step 4: Run the application using “python app.py”

3 Docker Image Creation

Step 1: Download Docker Desktop.

Step 2: Build the “Dockerfile” on the working environment.

Step 3: Enter the specifics of the Application in the Dockerfile to successfully create the Image.

Step 4: Build the Image by running “docker build -t fakenewsclfapp .”

4 AWS Deployment

Step 1: Push the flask application codebase and the accompanying Dockerfile onto a GitHub repository.

Step 2: Log into AWS and Create a sample environment on AWS Elastic Beanstalk with the deploy platform set as Docker by uploading the Terraform script on the AWS CloudShell environment and running it.

Step 3: Create a CI/CD pipeline using AWS CodePipeline.

Step 4: Set the source as the GitHub repository and select the correct branch while configuring the pipeline.

Step 5: Set the deployment platform as AWS Elastic Beanstalk. The CI/CD pipeline is initiated once the pipeline is set up.

5 Azure Deployment

Step 1: Download Azure CLI and its extension on VS Code

Step 2: Connect to the Microsoft Azure account by running the command “az login” on powershell terminal.

Step 3: Choose the correct Azure subscription, which is “1” for student accounts.

Step 4: Set the appropriate variables for the setting the resource group name, server location, and container registry name.

```
$rgname = "rg-fakenewsclfapp"
$location = "australiaeast"
$acrname = "fakenewsclfappacr01"
```

Step 5: Create the resource group by running the command “az group create --name $rgname --$location”

Step 6: Create Azure Container registries repository by running the command “az acr create --resource-group $rgname --name $acrname --sku Basic”

Step 7: Log into the ACR by running the command “az acr login –name $acrname”

Step 8: A Docker tag is created using the command “docker tag fakenewsclfapp fakenewsclfappacr01.azurecr.io/fakenewsclfapp”

Step 9: The created Docker tag can be viewed by running “docker image ls”

Step 10: Enable admin access for the created Azure Container Registry by running the command “az acr update -n fakenewsclfappacr01 --admin-enabled true”

Step 11: Push the docker image onto the created registry using the command “docker push fakenewsclfappacr01.azurecr.io/fakenewsclfapp”

This finishes the ACR repository creation and the Docker image storage part of the Azure deployment process.

Step 12: The variables for the Azure Container Apps are set using the commands

```
$resourceGroupName = "rg-fakenewsclfapp"
$acrName = "fakenewsclfappacr01"
$acrImage = "$acrName.azurecr.io/fakenewsclfapp:latest"
$location = "australiaeast"
```

$containerAppEnv = "fakenewsclfapp-capenv"
$containerAppName = "fakenewsclfapp-webapp"

Step 13: The Azure Container App environment is setup by running the command “az containerapp env create --name $containerAppEnv --resource-group $resourceGroupName --location $location”

Step 14: The ACR repository username and password are fetched by running the below two commands one by one.

“$acrUsername = az acr credential show --name $acrName --query "username" --output tsv”
“$acrPassword = az acr credential show --name $acrName --query "passwords[0].value" --output tsv”

Step 15: The Azure Container Apps environment is created for the Docker image deployment using the command “az containerapp create --name $containerAppName --resource-group $resourceGroupName --environment $containerAppEnv --image $acrImage --registry-server "$acrName.azurecr.io" --registry-username $acrUsername --registry-password $acrPassword --target-port 5000 --ingress 'external' --cpu 0.5 --memory 1.0Gi”

6 Locust setup

Step 1: Install Locust on the work environment on VS Code by running the command
“pip install locust”

Step 2: Create a file named “locustfile.py” in the directory containing the project.

Step 3: To run the file and create load on Azure application, run the command

“locust -f locustfile.py --host=https://fakenewsclfapp-webapp.happypebble-97bb53d5.australiaeast.azurecontainerapps.io/ --web-port 5000”

Step 4: To create load on the AWS application, run the command
“locust -f locustfile.py --host=http://fakenewsclfflaskappenv.eba-4p9z6dkr.eu-west-2.elasticbeanstalk.com/ --web-port 5000”

Step 5: Enter the parameters on the Locust tool such as number of users accessing the site, and the frequency at which they should join the load testing sequence and start the load testing process.

7 Monitoring tools setup

Part 1: Microsoft Azure

Step 1: Utilize Azure Monitor Dashboard to create monitoring metrics for the application deployed on Azure Container Apps.

Step 2: Choose the resource group, subscription and the application environment to enable the metrics to be added to the dashboard.

Step 3: The metrics chosen for Azure dashboard are CPU Usage, Response Time, and Network I/O.

Part 2: AWS

Step 1: Use AWS CloudWatch to monitor the operational health of the application deployed on AWS Elastic Beanstalk.

Step 2: Select the region, service to open the metrics available to the deployed application, which can be later added to the CloudWatch dashboard.

Step 3: The metrics added to the dashboard are CPU Utilization, Application Latency, and Network I/O.