Configuration Manual

MSc Research Project
MSc in Cloud Computing

Yamini Murugan
Student ID: x23166401

School of Computing
National College of Ireland

Supervisor: Shreyas Setlur Arun

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Yamini Murugan		
Student ID:	x23166401		
Programme:	MSc in Cloud Computing	**Year:**	1
Module:	MSc Research Project		
Lecturer:	*Shreyas Setlur Arun*		
Submission Due Date:	12/12/2024		
Project Title:	Development and Performance Evaluation of a Dockerized Flask Application for Phishing URL Detection Across AWS and Azure		
Word Count:	909	**Page Count:**	12

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: Yamini Murugan

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):	

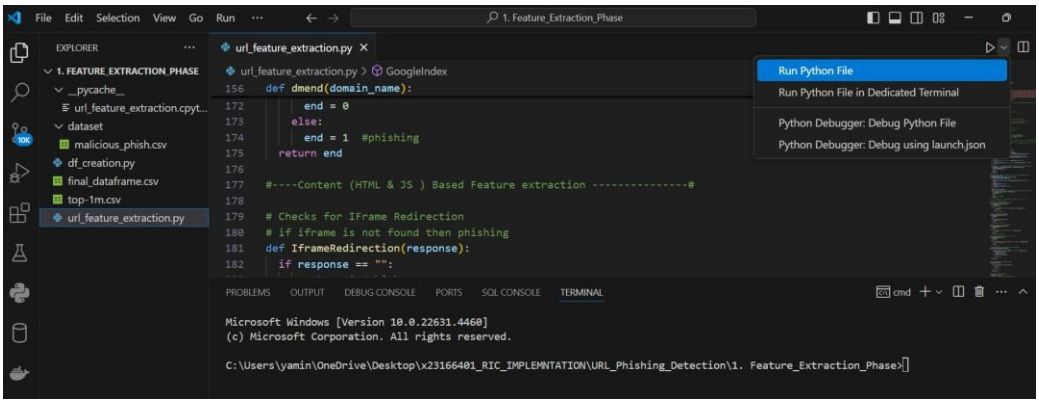
Configuration Manual

Yamini Murugan
x23166401

1 Dataset Preparation – Feature Extraction

Pre-Requisites – Vs code Installation, Python Version – 3.10.11

1. For generating the "final_dataframe.csv" after feature extraction - Run the python file "url_feature_extration.py" as shown below or,
   run command, " python url_feature_extraction.py"



2 Data Pre-Processing and Deep Learning Model Training

Pre-Requisites – Google Drive, Google Colab [1] connection.

Google Colab Link

Step 1: Upload the final_dataframe.csv to google drive

Step 2: Create a python notebook on google colab for data pre-processing and DL Model Training.

Step 3: After running the code, the best model (BiLSTM) is saved for future Flask Application development.

3 Flask Application Creation

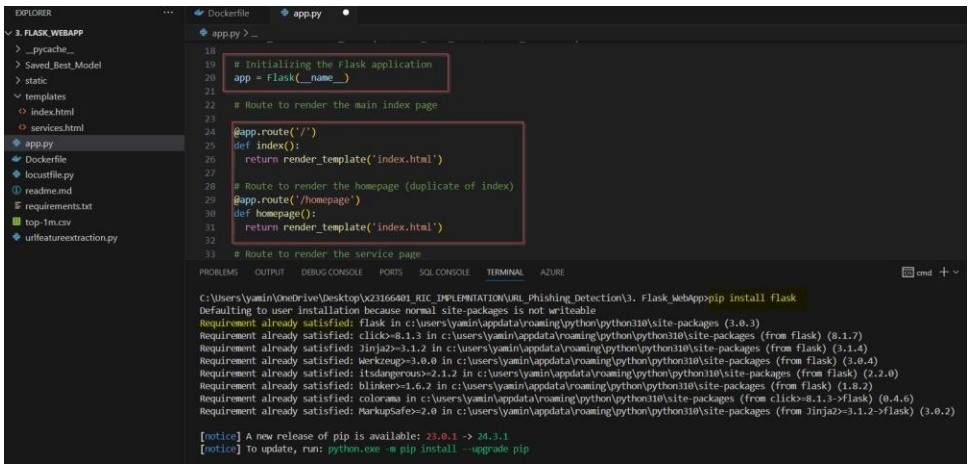
Pre-Requisites – Vs code Installation, Python Version – 3.10.11

Step 1: Install Flask [2] from Python package Manager by using the command:

pip install Flask

Step 2: Create a python file with name “app.py” in the Project Directory.

Step 3: For app.py, import Flask and define the required routes.



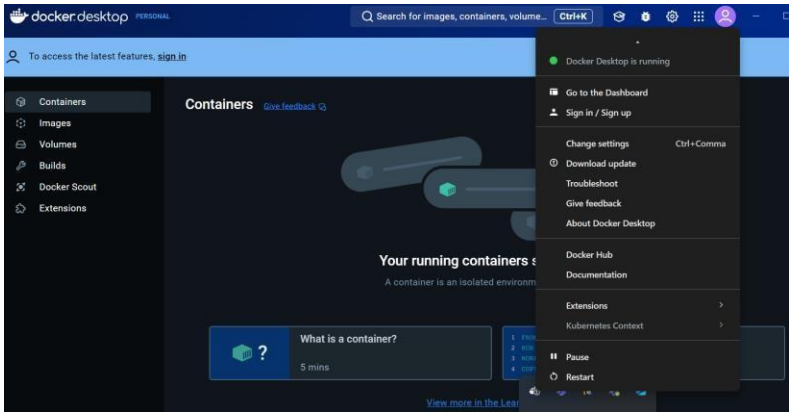
Step 4: Run “app.py” flask application with command

python app.py

4 Docker Image Creation

Pre-Requisites – Docker Desktop Downloaded [3]

Step 1: Docker must be installed following the installation guidelines from: docker_installation_guide and must be in running state.

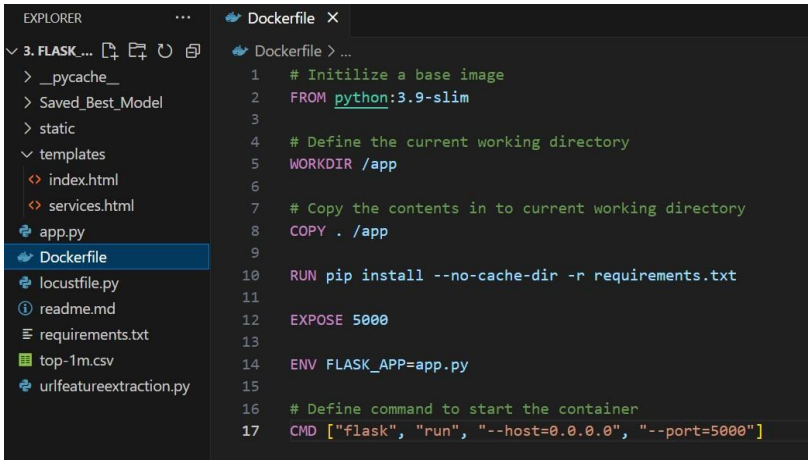


Step 2: Build Dockerfile

Pre-Requisites: requirements.txt file with a list of libraries to be imported

The below Dockerfile (case sensitive - with “D” capitalized) is created for the flask application.

The workings of this file are explained with the comments.

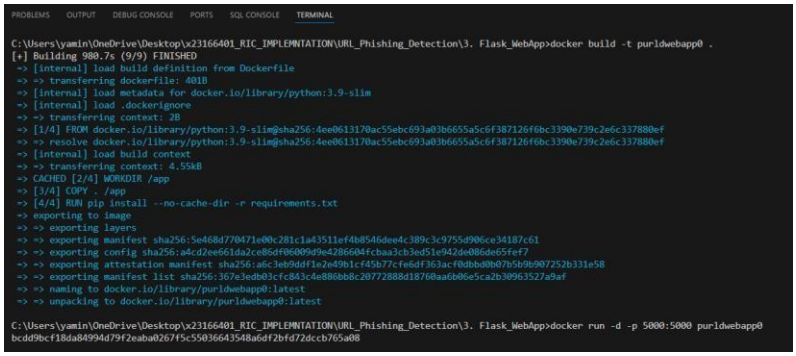


Step 3: Build Docker Image with command

docker build -t purldwebapp0 .

Step 4: To Check Docker Image use command

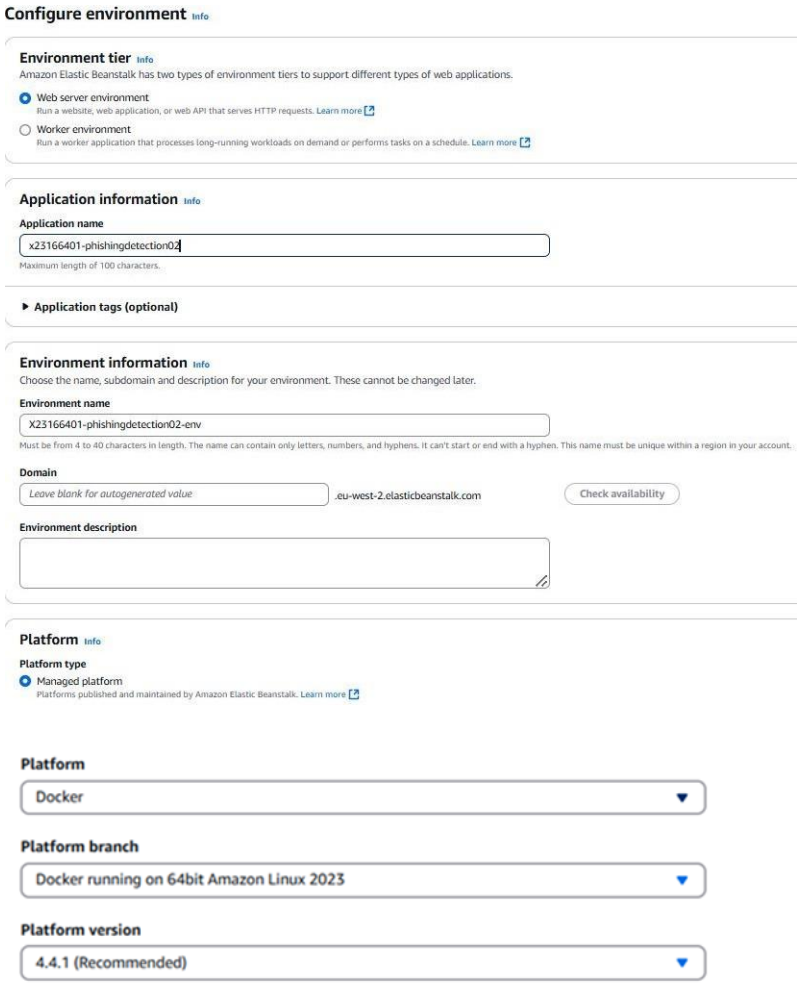
docker run -d -p 5000:5000 purldwebapp0



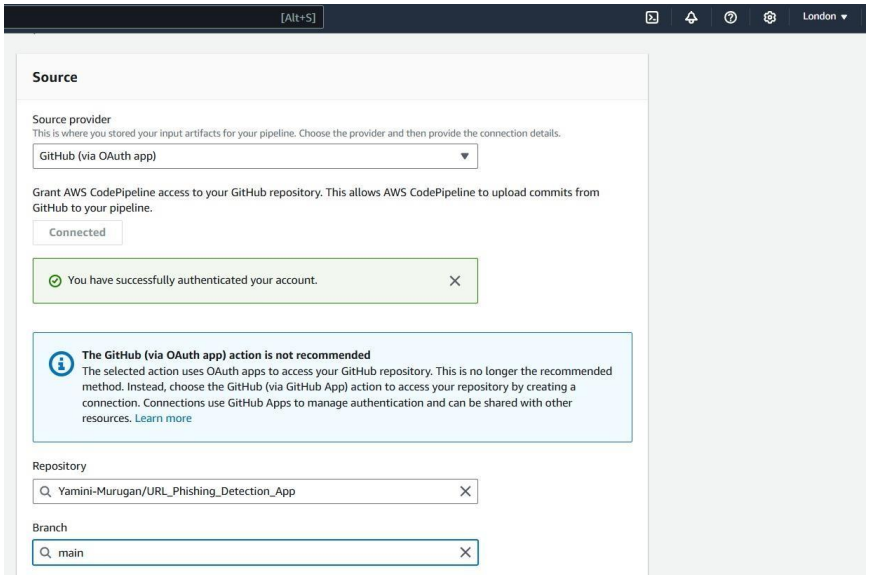
5 AWS Deployment

Pre-Requisites – GitHub [4] Account and AWS [5] Account Set Up

Step 1: Upload the codebase along with the Dockerfile to the GitHub repository



Step 2: Create a sample AWS Elastic Beanstalk environment for phishing URL detection application and choose Docker as the Platform.



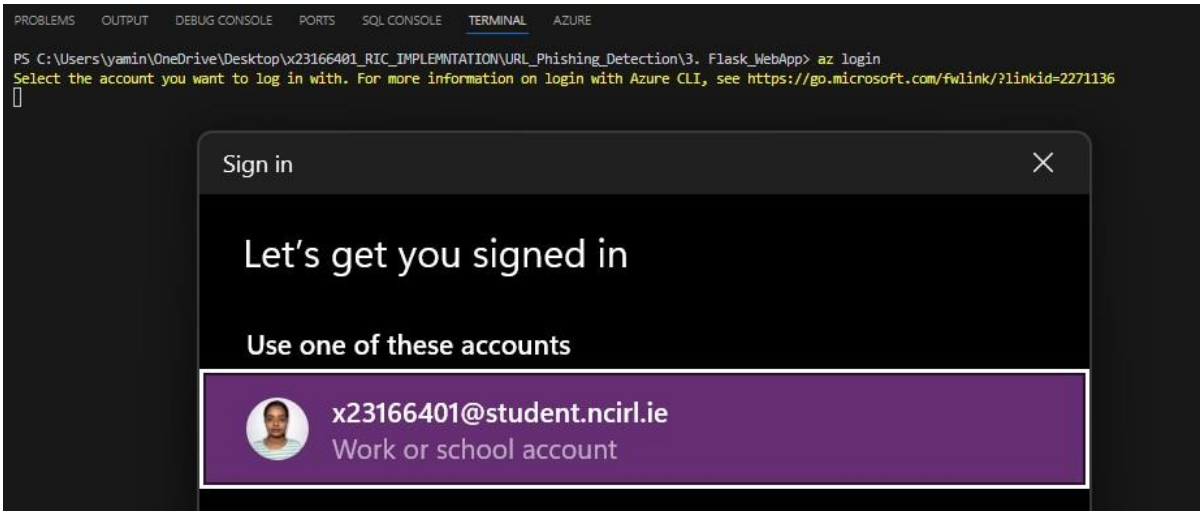
Step 3: Set up a CI/CD pipeline on AWS using AWS CodePipeline and establish a connection with GitHub during configuration.

Step 4: Select the repository containing the code pertaining to this project, to be the source.

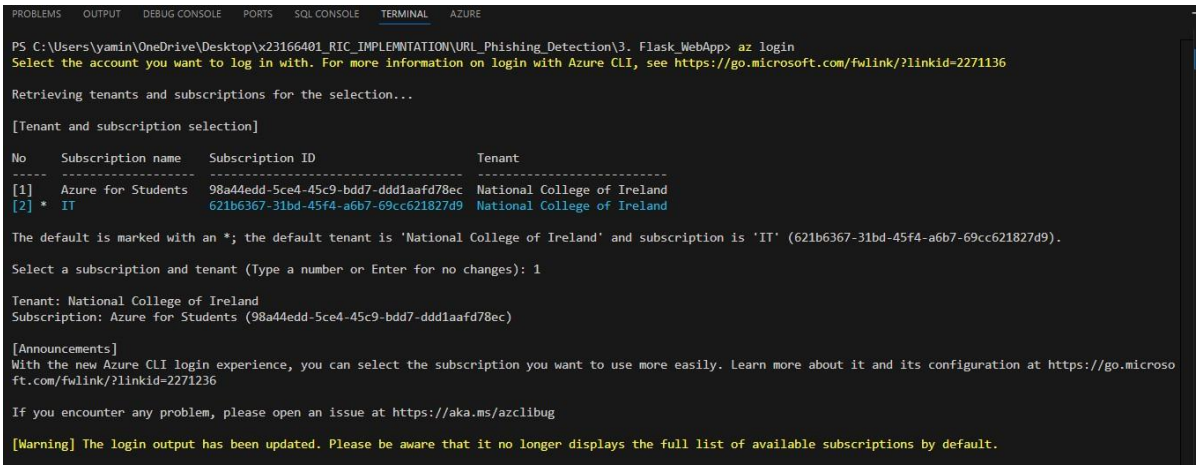
Step 5: Select the Elastic Beanstalk environment and the application, as the Deploymentplatform during configuring CodePipeline, this triggers the CI/CD pipeline.

6 Azure Cloud Deployment

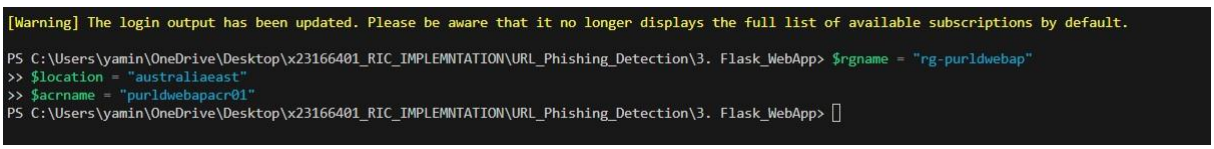
Pre-Requisites – Docker Desktop, Azure CLI and its VS Code extension, and Microsoft Azure [6] Account Set Up



Step 1: Connect to Azure account using the command "az login" from powershell.



Step 2: Choose the correct subscription "Azure for students" by entering 1.



Step 3: Set the variables for resource group name, location and container registry name.

```
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> az group create --name $rgname --location $location
>>
{
  "id": "/subscriptions/98a44edd-5ce4-45c9-bdd7-ddd1aafd78ec/resourceGroups/rg-purldwebap",
  "location": "australiaeast",
  "managedBy": null,
  "name": "rg-purldwebap",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> az acr create --resource-group $rgname --name $acrname --sku B
asic
>>
{
  "adminUserEnabled": false,
  "anonymousPullEnabled": false,
  "creationDate": "2024-12-09T11:41:09.802370+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
    "status": "disabled"
  },
  "id": "/subscriptions/98a44edd-5ce4-45c9-bdd7-ddd1aafd78ec/resourceGroups/rg-purldwebap/providers/Microsoft.ContainerRegistry/registries/purldwebapacr01",
  "identity": null,
  "location": "australiaeast",
  "loginServer": "purldwebapacr01.azurecr.io",
  "metadataSearch": "Disabled",
  "name": "purldwebapacr01",
  "networkRuleBypassOptions": "AzureServices",
  "networkRuleSet": null,
  "policies": {
    "azureAdAuthenticationAsArmPolicy": {
      "status": "enabled"
    },
    "exportPolicy": {
      "status": "enabled"
```

Step 4: Create an Azure resource group using “az group create --name $rgname --$location”

Step 5: Create an ACR repository using the command, “az acr create --resource-group $rgname --name $acrname --sku Basic”

```
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp>
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> az acr login --name $acrname
>>
Login Succeeded
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> docker image ls
REPOSITORY     TAG       IMAGE ID       CREATED       SIZE
purldwebapp0   latest    367e3edb03cf   4 hours ago   4.59GB
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp>
```

Step 6: Log onto azure container registries by running the command “az acr login –name $acrname”

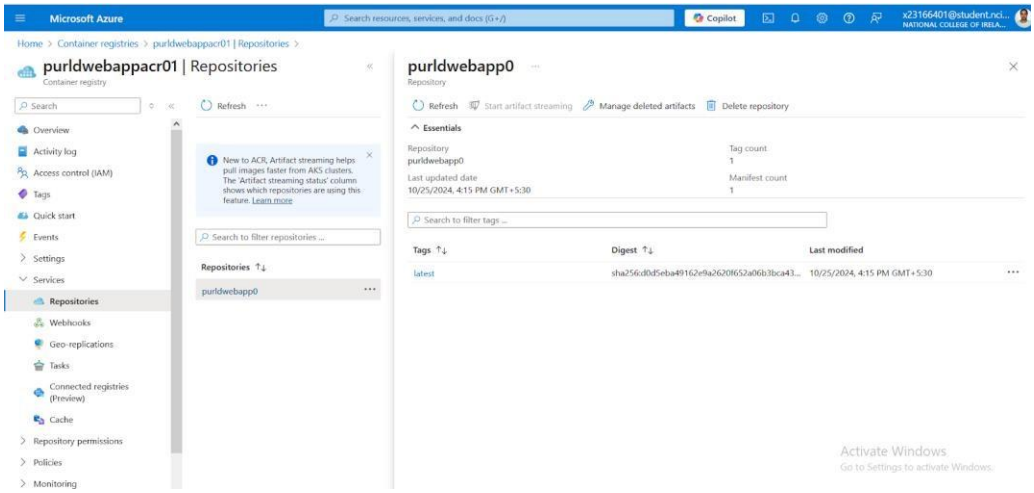
```
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> docker tag purldwebapp0 purldwebapacr01.azurecr.io/purldwe
bap
>>
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> docker image ls
>>
REPOSITORY                              TAG       IMAGE ID       CREATED        SIZE
purldwebapp0                            latest    367e3edb03cf   47 hours ago   4.59GB
purldwebapacr01.azurecr.io/purldwebap   latest    367e3edb03cf   47 hours ago   4.59GB
```

Step 7: Create a Docker tag for the image using “docker tag puridwebapp0 purldwebapacr01.azurecr.io/purldwebap”

Step 8: Check for the presence of Docker image and the image tag on Docker Desktop using “docker image ls”

```
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> docker push purldwebapacr01.azurecr.io/purldwebap
>>
Using default tag: latest
The push refers to repository [purldwebapacr01.azurecr.io/purldwebap]
e898f07f768a: Pushed
3705d759815b: Pushed
f2dc2395bdd2: Pushed
8d9d4ebf9cc3: Pushed
16eb7cf39d27: Pushed
bc0965b23a04: Pushed
c3edffebd723: Pushed
c07f3a6e2bd8: Pushed
latest: digest: sha256:367e3edb03cfc843c4e886bb8c20772888d18760aa6b06e5ca2b30963527a9af size: 856
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp>
```

Step 9: Push the Docker image to ACR using “docker push purldwebapacr01.azurecr.io/purldwebap”



A repository has been created on Azure Container Registry as seen in Figure above

```
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp>
>> $resourceGroupName = "rg-purldwebap"
>> $acrName = "purldwebpacr01"
>> $acrImage = "$acrName.azurecr.io/purldwebap:latest"
>> $location = "australiaeast"
>> $containerAppEnv = "purldwebap-capenv"
>> $containerAppName = "purldwebap-webapp"
```

Step 10: Set the variables for deploying the image on Azure Container Apps,

$resourceGroupName = "rg-purldwebap"
$acrName = "purldwebpacr01"
$acrImage = "$acrName.azurecr.io/purldwebap:latest"
$location = "australiaeast"
$containerAppEnv = "purldwebap-capenv"
$containerAppName = "purldwebap-webapp"

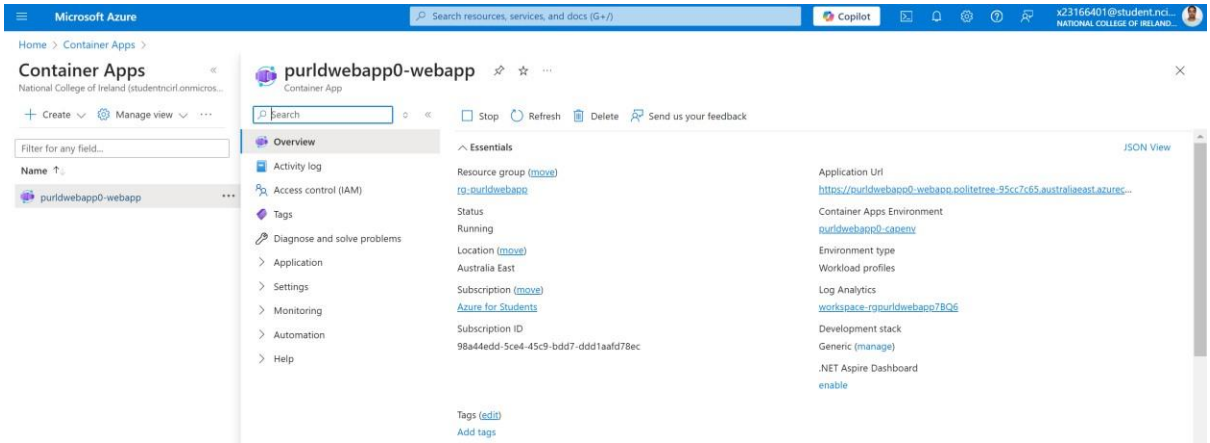
```
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> $acrUsernamе = az acr credential show --name $acrName --query "
username" --output tsv
>> $acrPassword = az acr credential show --name $acrName --query "passwords[0].value" --output tsv
>>
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> $acrPassword = az acr credential show --name $acrName --query "
passwords[0].value" --output tsv
>>
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp> az containerapp create --name $containerAppName --resource-grou
p $resourceGroupName --environment $containerAppEnv --image $acrImage --registry-server "$acrName.azurecr.io" --registry-username $acrUsernamе --registry-password $ac
rPassword --target-port 5000 --ingress 'external' --cpu 0.5 --memory 1.0Gi
>>
Adding registry password as a secret with name "purldwebapppacr01azurecrio-purldwebapppacr01"

Container app created. Access your app at https://purldwebappp-webapp.nicepebble-3240e10b.australiaeast.azurecontainerapps.io/
```

Step 11: Create the Container App environment by using this command, “az containerapp env create --name $containerAppEnv --resource-group $resourceGroupName --location $location”

Step 12: Fetch the ACR repository username by using this command, “$acrUsernamе = az acr credential show --name $acrName --query "username" --output tsv”

Step 13: Fetch the ACR repository password by using this command, “$acrPassword = az acr credential show --name $acrName --query "passwords[0].value" --output tsv”

Step 14: Create a Container Apps environment 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”

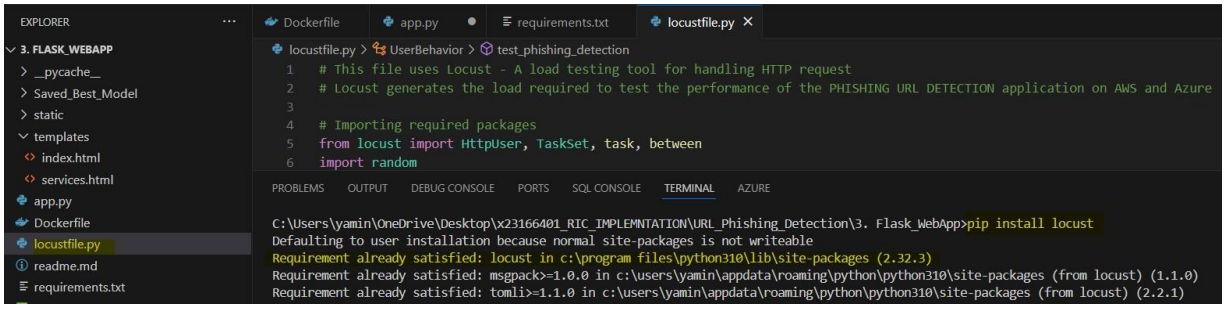


The creation of Azure Container Apps environment is seen in Figure above

7 Locust Load Testing Set up

Step 1: Install locust [7] from Python package Manager by using the command

pip install locust



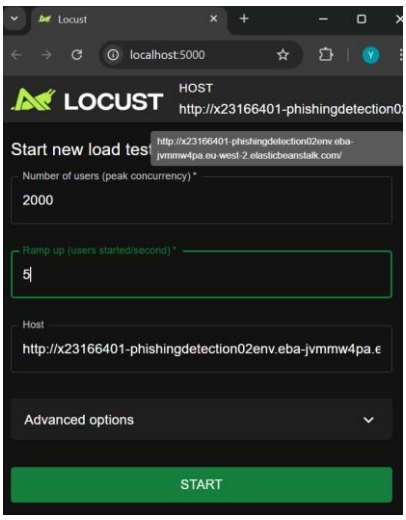
Step 2: Create a locust file with name “locustfile.py” in the Project Directory.

Step 3: To run - use command

For AWS:

locust -f locustfile.py --host=http://x23166401-phishingdetection02env.eba-jvmmw4pa.eu-west-2.elasticbeanstalk.com/ --web-port 5000

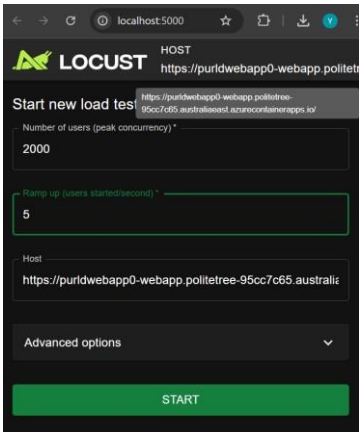
```
C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp>locust -f locustfile.py --host=http://x23166401-phishingdetection02env.eba-jvmm
w4pa.eu-west-2.elasticbeanstalk.com/ --web-port 5000
[2024-12-11 01:18:55,195] DESKTOP-FJK2RJS/INFO/locust.main: Starting Locust 2.32.3
[2024-12-11 01:18:55,196] DESKTOP-FJK2RJS/INFO/locust.main: Starting web interface at http://localhost:5000 (accepting connections from all network interfaces)
```



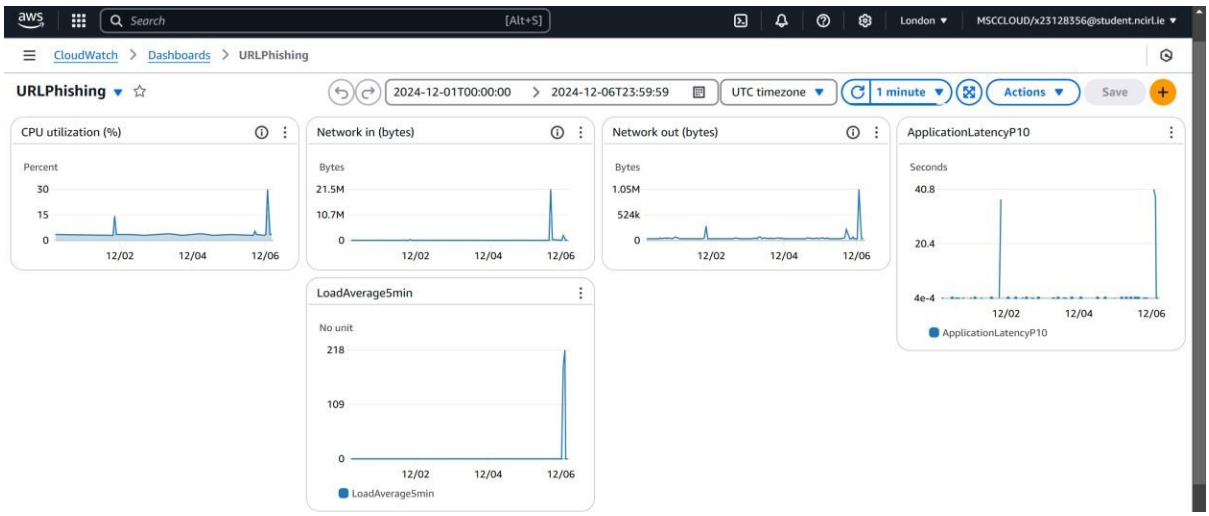
For Azure:

locust -f locustfile.py --host=https://purldwebapp0-webapp.politetree-95cc7c65.australiaeast.azurecontainerapps.io/ --web-port 5000

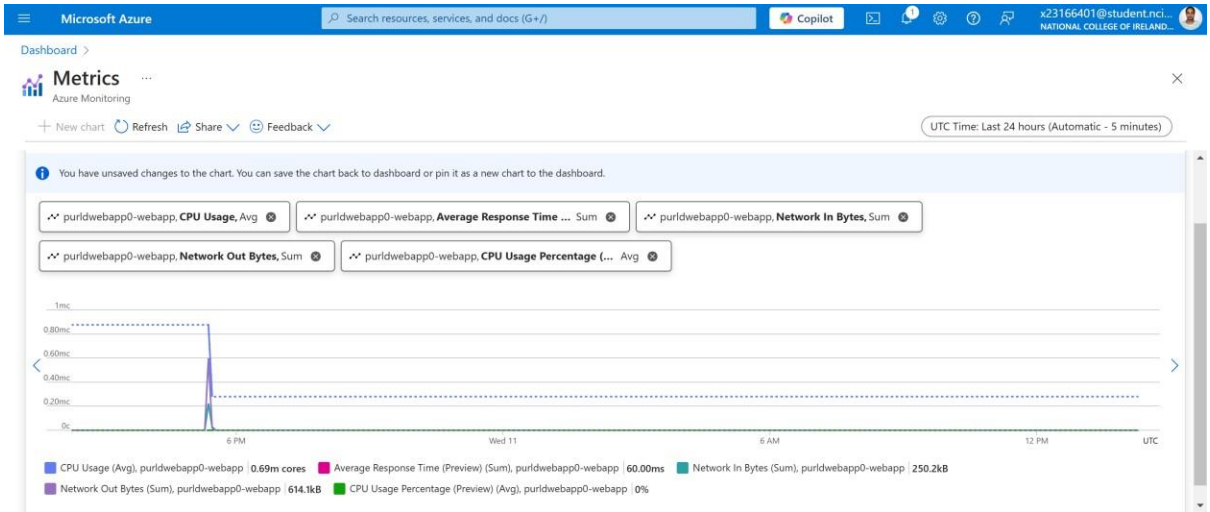
```
C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMNTATION\URL_Phishing_Detection\3. Flask_WebApp>locust -f locustfile.py --host=https://purldwebapp0-webapp.politetree-95cc7c65.
australiaeast.azurecontainerapps.io/ --web-port 5000
[2024-12-11 01:04:04,446] DESKTOP-FJK2RJS/INFO/locust.main: Starting Locust 2.32.3
[2024-12-11 01:04:04,448] DESKTOP-FJK2RJS/INFO/locust.main: Starting web interface at http://localhost:5000 (accepting connections from all network interfaces)
```



8 AWS CloudWatch and Azure Monitor

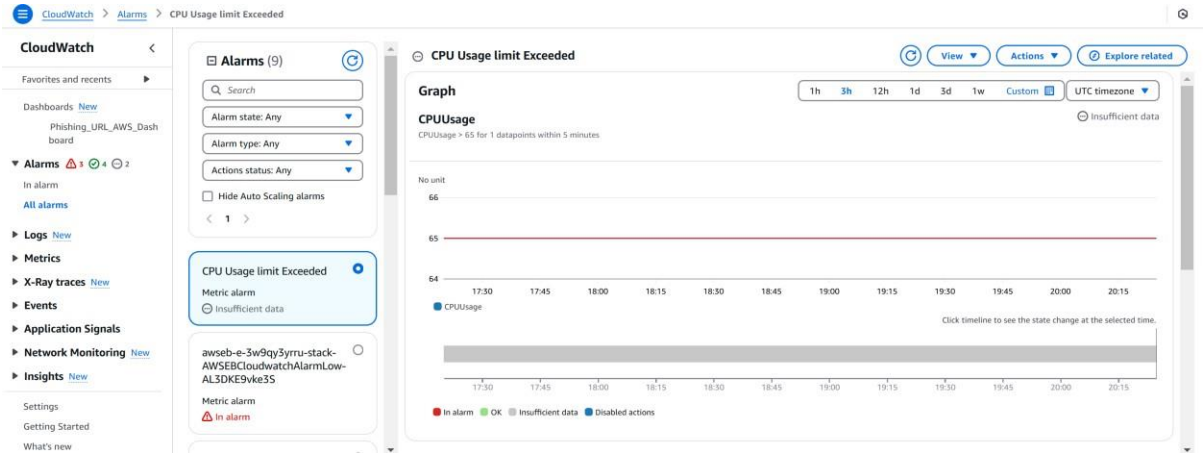


AWS CloudWatch dashboard is set up and the metrics such as CPU usage, Network I/O, Latency and AverageLoad is collected.

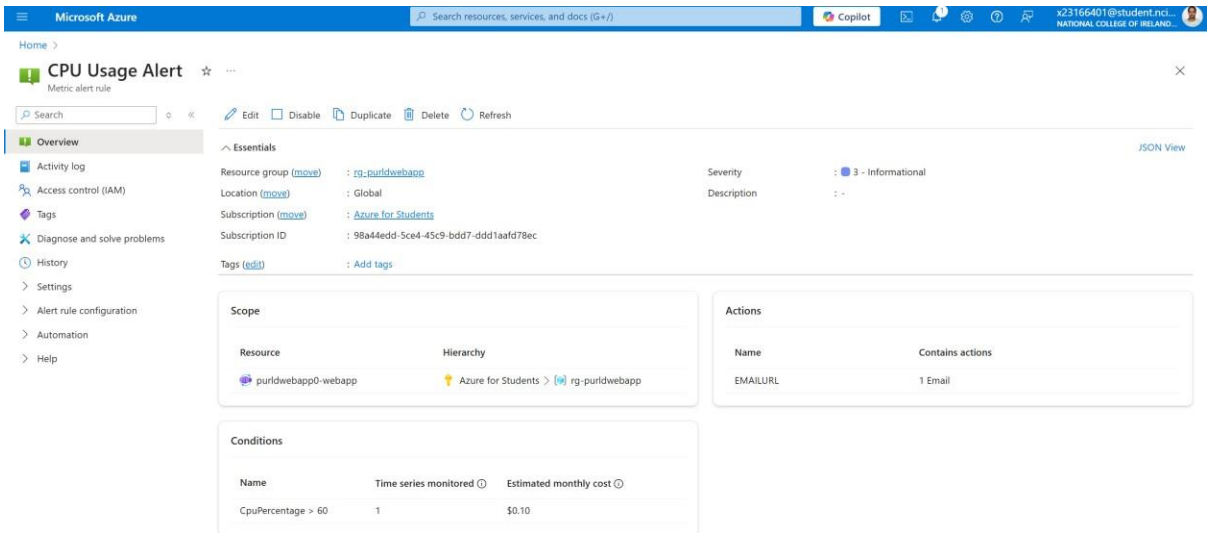


Azure Monitor Dashboard is set up to gather insights from the deployed application on Azure Container Apps. Metrics include CPU Usage, Average Response time, Network I/O and CPU usage Percentage.

9 Alert Set up



AWS SNS subscription is used to trigger an alert to the configured email if the CPU utilization percentage crosses 65% over a time period of 5 minutes.



Microsoft Azure Monitor Alerts is set up to trigger email alerts if the CPU Usage Percentage of the Azure application crosses 60%.

10 References

[1] "Google Colab," [Online]. Available: https://col b.research.google.com/.

[2] "Flask Framework," [Online]. Available: https://flask.palletsprojects.com/en/stable/.

[3] "Docker Desktop," [Online]. Available: https://ww.docker.com/.

[4] "AWS Documentation," [Online]. Available: https://aws.amazon.com/?nc2=h_lg.

[5] "Azure Documentation," [Online]. Available: htps://azure.microsoft.com/en-us/.

[6] "Locust," [Online]. Available: https://locust.io/.

[7] "GitHub," [Online]. Available: https://github.com/.