

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)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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):	

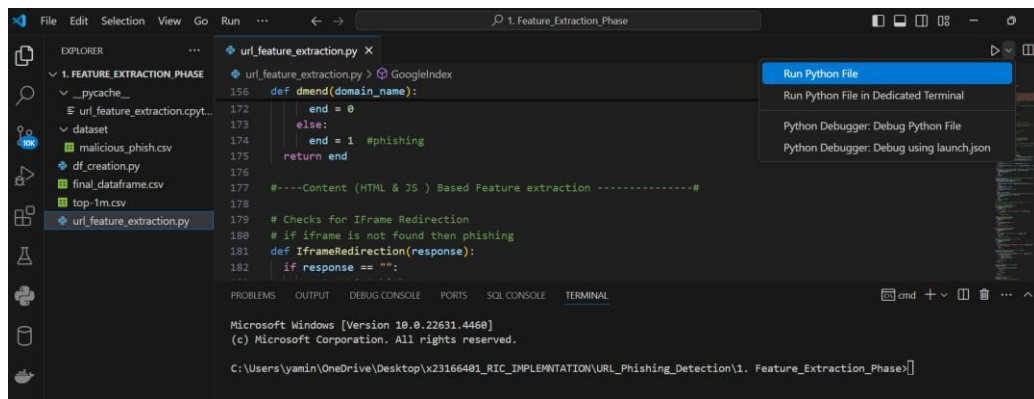
Configuration Manual

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_extraction.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

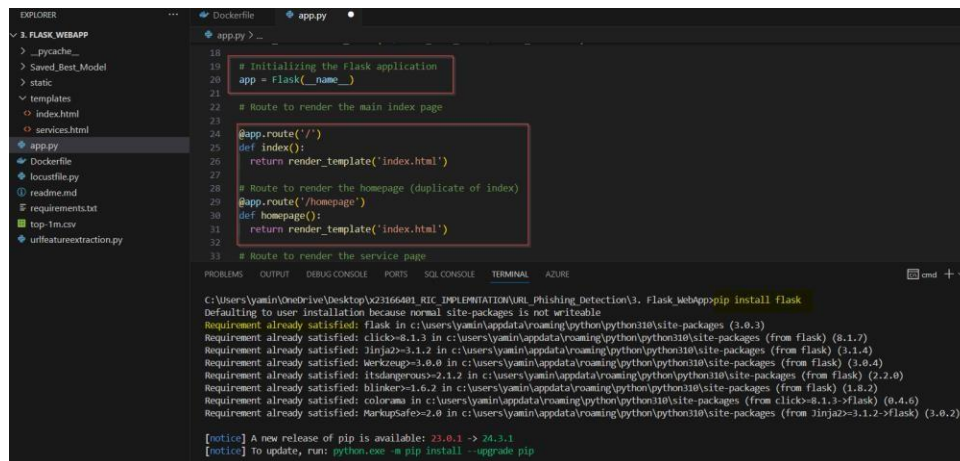
Pre-Requisites – Vs code Installation, Python Version – 3.10.11

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.



The screenshot shows a VS Code editor with a file explorer on the left showing a project named '3. FLASK_WEBAPP'. The 'app.py' file is open in the editor. The code in 'app.py' is as follows:

```
18 # Initializing the Flask application
19 app = Flask(__name__)
20
21 # Route to render the main index page
22
23 @app.route('/')
24 def index():
25     return render_template('index.html')
26
27 # Route to render the homepage (duplicate of index)
28 @app.route('/homepage')
29 def homepage():
30     return render_template('index.html')
31
32 # Route to render the service page
33
```

The terminal at the bottom shows the command 'pip install flask' and its output, which lists various requirements and their versions.

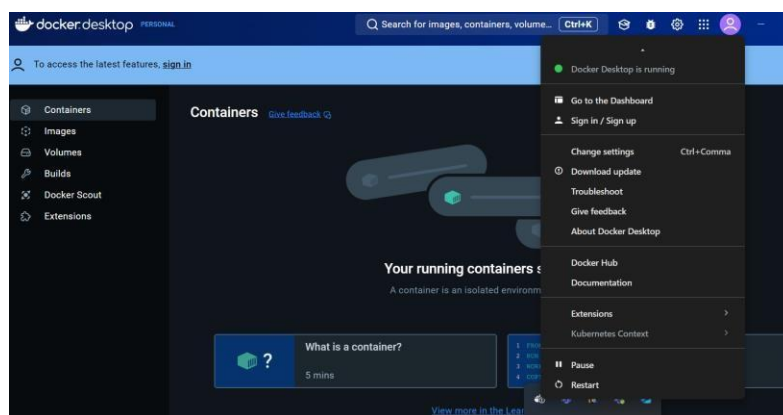
Step 4: Run “app.py” flask application with command

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.

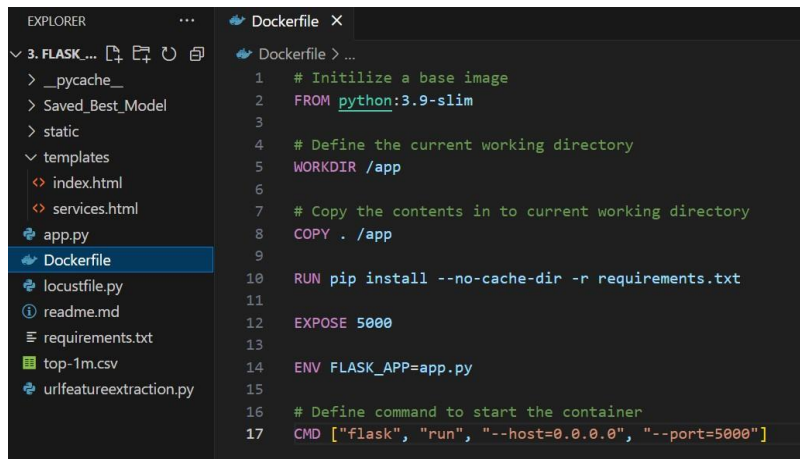


Step 2: Build Dockerfile

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.

The image shows a VS Code window with the Explorer sidebar on the left and a Dockerfile open in the editor. The Explorer sidebar shows a file tree for a project named '3. FLASK...', with files like __pycache__, Saved_Best_Model, static, templates, index.html, services.html, app.py, Dockerfile, locustfile.py, readme.md, requirements.txt, top-1m.csv, and urlfeatureextraction.py. The Dockerfile in the editor contains the following content:

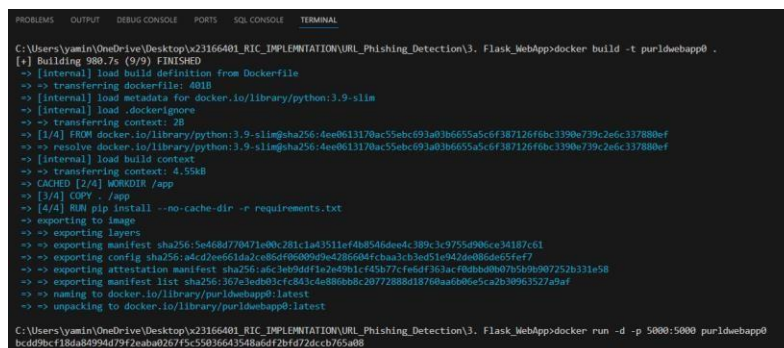
```
1 # Initialize a base image
2 FROM python:3.9-slim
3
4 # Define the current working directory
5 WORKDIR /app
6
7 # Copy the contents in to current working directory
8 COPY . /app
9
10 RUN pip install --no-cache-dir -r requirements.txt
11
12 EXPOSE 5000
13
14 ENV FLASK_APP=app.py
15
16 # Define command to start the container
17 CMD ["flask", "run", "--host=0.0.0.0", "--port=5000"]
```

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`

The image shows a VS Code window with the Terminal sidebar on the left and a terminal window open. The terminal shows the output of the Docker build and run commands. The build command is `docker build -t purldwebapp0 .` and the run command is `docker run -d -p 5000:5000 purldwebapp0`. The terminal output shows the build process, including the transfer of the Dockerfile, the build context, and the installation of requirements. The run command output shows the container being started and the port mapping.

```
C:\Users\yamin\OneDrive\Desktop\vx23166401_RIC_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp>docker build -t purldwebapp0 .
[+] Building 988.7s (9/9) FINISHED
-> [internal] load build definition from Dockerfile
-> [internal] load metadata for docker.io/library/python:3.9-slim
-> [internal] load .dockerignore
-> [internal] load build context
-> [1/4] FROM docker.io/library/python:3.9-slim@sha256:4ee0613170ac55ebc693a03b6655a5c6f387126f6bc3390e739c2e6c337880ef
-> [2/4] COPY . /app
-> [3/4] RUN pip install --no-cache-dir -r requirements.txt
-> [4/4] EXPOSE 5000
-> exporting layers
-> exporting manifest sha256:5e468d770471e00c281c1a3511ef4b0546deed4389c3c975d906ce34187c61
-> exporting config sha256:a4cd2ee61da2ce8d6f86089d3e4286604fcaa33ed5e942de085de65fef7
-> exporting attestation manifest sha256:a6c3e8d0f1e2e4901c45b77c7e6d4f363ac40bda0807b569090725b331e58
-> exporting manifest list sha256:30723ed0b0cfc843c4e80b0bc2077208051870aad686c5ca2b30963527a9af
-> naming to docker.io/library/purldwebapp0:latest
-> unpacking to docker.io/library/purldwebapp0:latest
C:\Users\yamin\OneDrive\Desktop\vx23166401_RIC_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp>docker run -d -p 5000:5000 purldwebapp0
bcd9bcf18da84994d79f2eaba0267f5c55036643548a6d72bf472dccb765a08
```

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

Configure environment [Info](#)

Environment tier [Info](#)

Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.

☒ Web server environment

Run a website, web application, or web API that serves HTTP requests. [Learn more](#)

☐ Worker environment

Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. [Learn more](#)

Application information [Info](#)

Application name

Maximum length of 100 characters.

► Application tags (optional)

Environment information [Info](#)

Choose the name, subdomain and description for your environment. These cannot be changed later.

Environment name

Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.

Domain

.eu-west-2.elasticbeanstalk.com

[Check availability](#)

Environment description

Platform [Info](#)

Platform type

☒ Managed platform

Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#)

Platform

Platform branch

Platform version

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

[Alt+S]

Source

Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (via OAuth app)

Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.

Connected

You have successfully authenticated your account.

The GitHub (via OAuth app) action is not recommended

The selected action uses OAuth apps to access your GitHub repository. This is no longer the recommended method. Instead, choose the GitHub (via GitHub App) action to access your repository by creating a connection. Connections use GitHub Apps to manage authentication and can be shared with other resources. [Learn more](#)

Repository

Yamini-Murugan/URL_Phishing_Detection_App

Branch

main

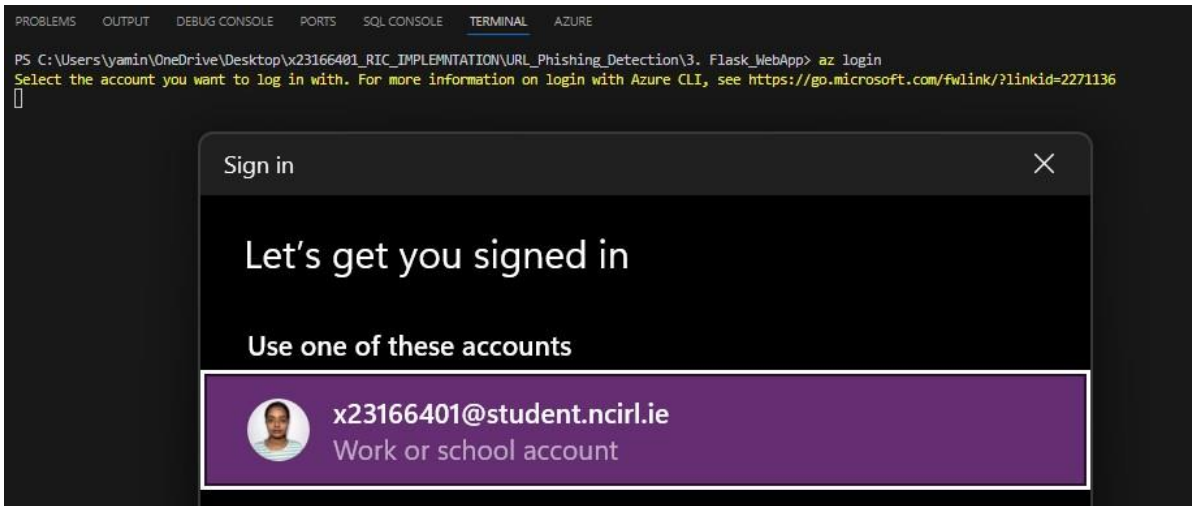
Step 3: Set up a CI/CD pipeline on AWS using AWS CodePipeline and establish a connection with GitHub during configuration.

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

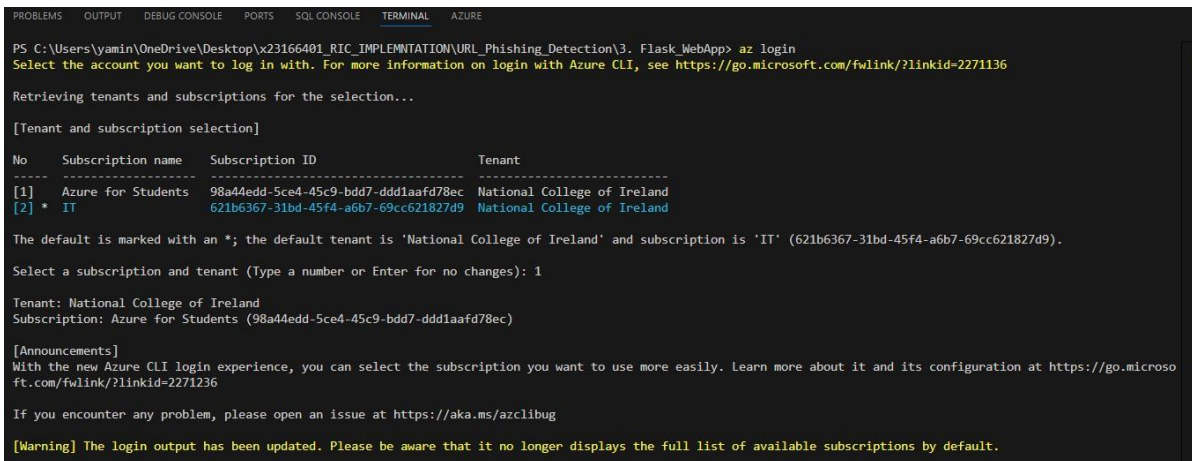
Step 5: Select the Elastic Beanstalk environment and the application, as the Deployment platform during configuring CodePipeline, this triggers the CI/CD pipeline.

6 Azure Cloud Deployment

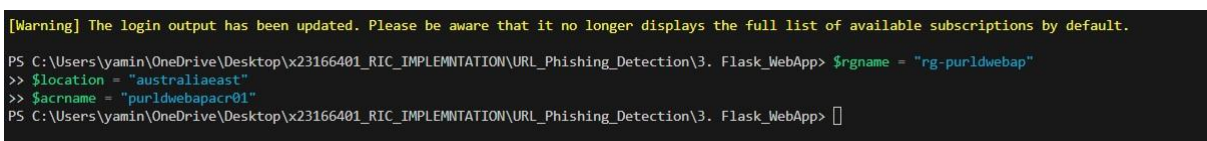
Pre-Requisites – Docker Desktop, Azure CLI and its VS Code extension, and Microsoft Azure [\[6\]](#) Account Set Up



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_IMPLEMENTATION\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_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp> az acr create --resource-group $rgname --name $acrname --sku Basic
>>
{
  "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 \$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_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp>
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp> az acr login --name $acrname
>>
Login Succeeded
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMENTATION\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_IMPLEMENTATION\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_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp> docker tag purldwebapp0 purldwebapacr01.azurecr.io/purldwebap
>>
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMENTATION\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 purldwebapp0 purldwebapacr01.azurecr.io/purldwebap”

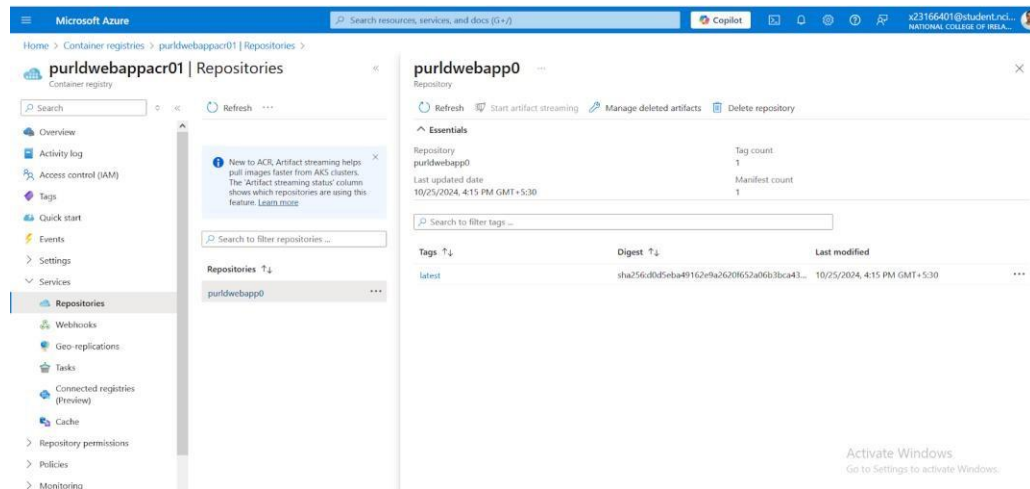
Step 8: Check for the presence of Docker image and the image tag on Docker Desktop using “docker image ls”

```

PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMENTATION\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]
e898f07f68a: Pushed
3705d759815b: Pushed
f2dc2395bdd2: Pushed
8d9d4ebf9cc3: Pushed
16eb7cf39d27: Pushed
bc0965b23a04: Pushed
c3edffebd723: Pushed
c07f3a6e2bd8: Pushed
latest: digest: sha256:367e3edb03cf843c4e886bb8c2077288bd18760aa6b06e5ca2b30963527a9af size: 856
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMENTATION\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_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp>
>> $resourceGroupName = "rg-purldwebap"
>> $acrName = "purldwebapacr01"
>> $acrImage = "$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 = "purldwebapacr01"
$acrImage = "$acrName.azurecr.io/purldwebap:latest"
$location = "australiaeast"
$containerAppEnv = "purldwebap-capenv"
$containerAppName = "purldwebap-webapp"
```

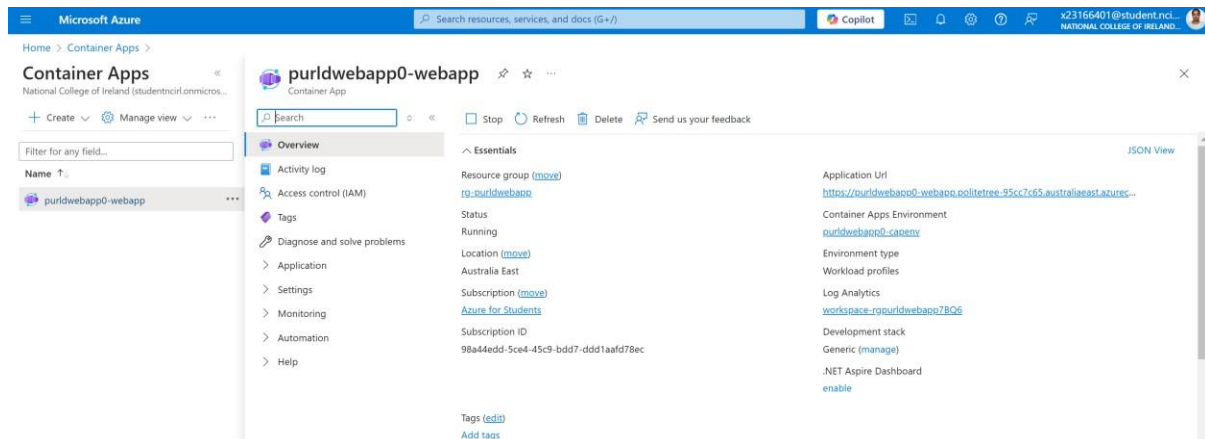
```
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp> $acrUsername = az acr credential show --name $acrName --query "
username" --output tsv
>> $acrPassword = az acr credential show --name $acrName --query "passwords[0].value" --output tsv
>>
PS C:\Users\yamin\OneDrive\Desktop\x23166401_RIC_IMPLEMENTATION\URL_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_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp> az containerapp create --name $containerAppName --resource-group
p $resourceGroupName --environment $containerAppEnv --image $acrImage --registry-server "$acrName.azurecr.io" --registry-username $acrUsername --registry-password $ac
rPassword --target-port 5000 --ingress 'external' --cpu 0.5 --memory 1.0Gi
>>
Adding registry password as a secret with name "purldwebappacr01azurecr.io-purldwebappacr01"
Container app created. Access your app at https://purldwebapp-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, “\$acrUsername = 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”

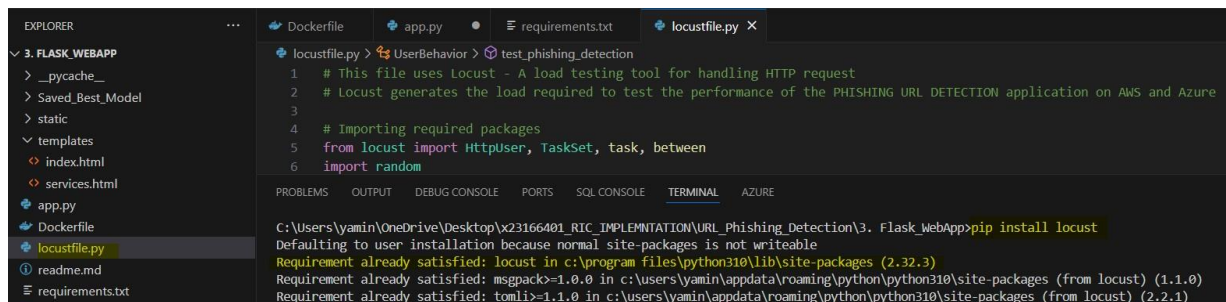


The creation of Azure Container Apps environment is seen in Figure above

7 Locust Load Testing Set up

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

pip install locust



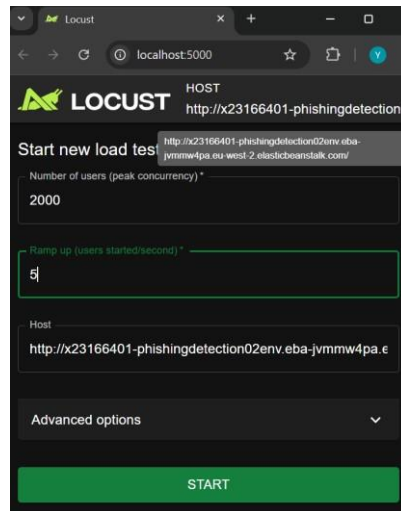
Step 2: Create a locust file with name “locustfile.py” in the Project Directory.

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

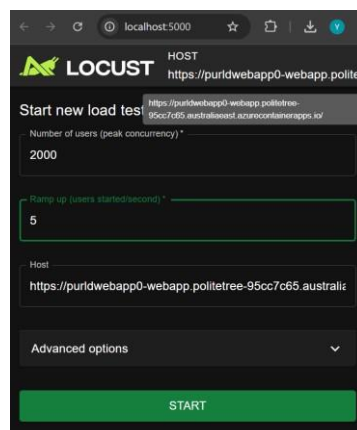
```
C:\Users\yamin\OneDrive\Desktop\X23166401_RIC_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp>locust -f locustfile.py --host=http://x23166401-phishingdetection02env.eba-jvmmw4pa.eu-west-2.elasticbeanstalk.com/ --web-port 5000
[2024-12-11 01:18:55,195] DESKTOP-FJK2R3S/INFO/locust.main: Starting Locust 2.32.3
[2024-12-11 01:18:55,196] DESKTOP-FJK2R3S/INFO/locust.main: Starting web interface at http://localhost:5000 (accepting connections from all network interfaces)
```



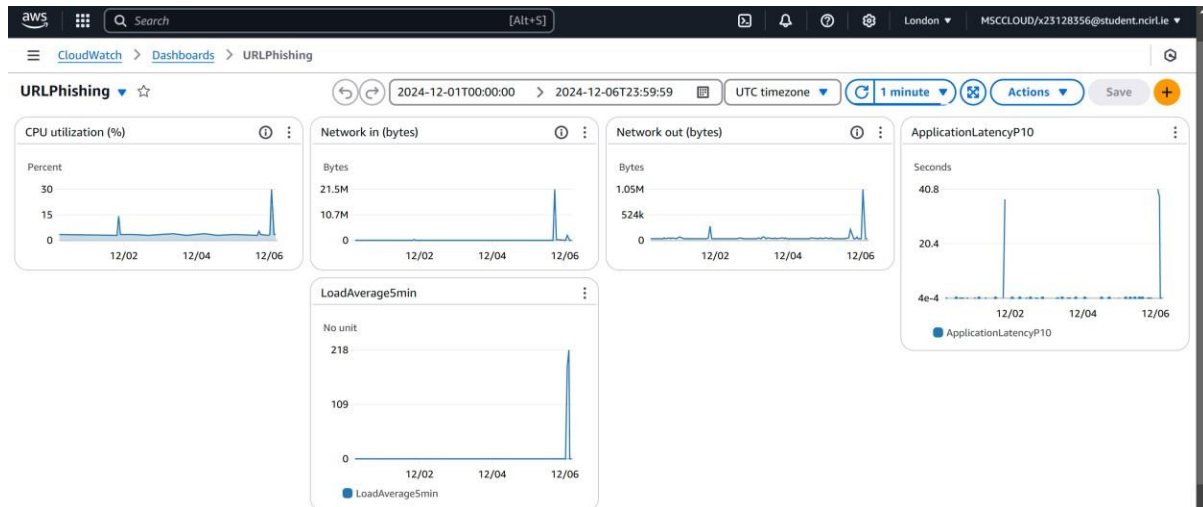
For Azure:

locust -f locustfile.py --host=https://purlwebapp0-webapp.politetree-95cc7c65.australiaeast.azurecontainerapps.io/ --web-port 5000

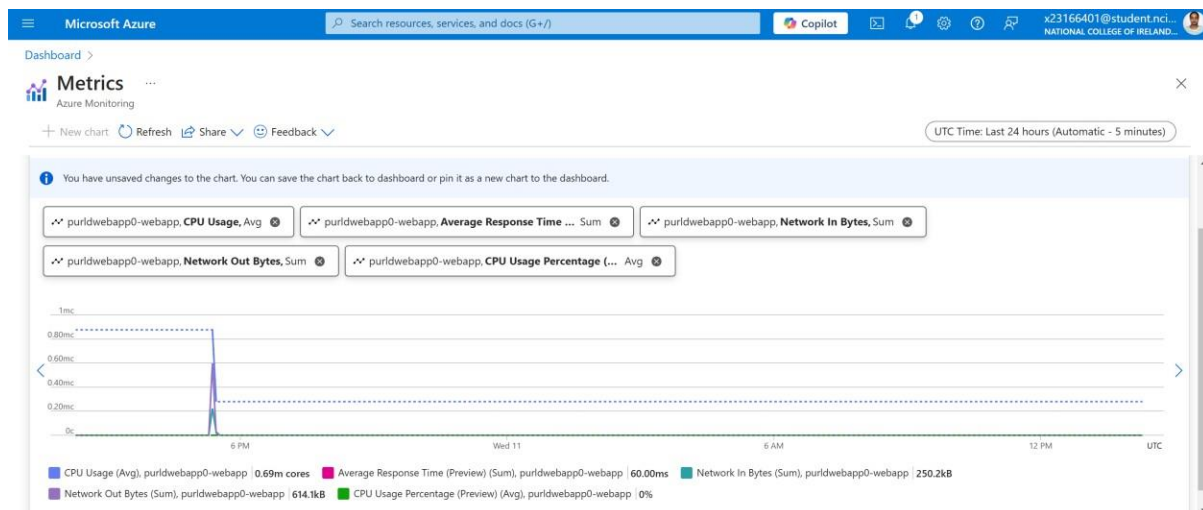
```
C:\Users\yamin\OneDrive\Desktop\X23166401_RIC_IMPLEMENTATION\URL_Phishing_Detection\3. Flask_WebApp>locust -f locustfile.py --host=https://purlwebapp0-webapp.politetree-95cc7c65.australiaeast.azurecontainerapps.io/ --web-port 5000
[2024-12-11 01:04:04,446] DESKTOP-FJK2R3S/INFO/locust.main: Starting Locust 2.32.3
[2024-12-11 01:04:04,448] DESKTOP-FJK2R3S/INFO/locust.main: Starting web interface at http://localhost:5000 (accepting connections from all network interfaces)
```



8 AWS CloudWatch and Azure Monitor

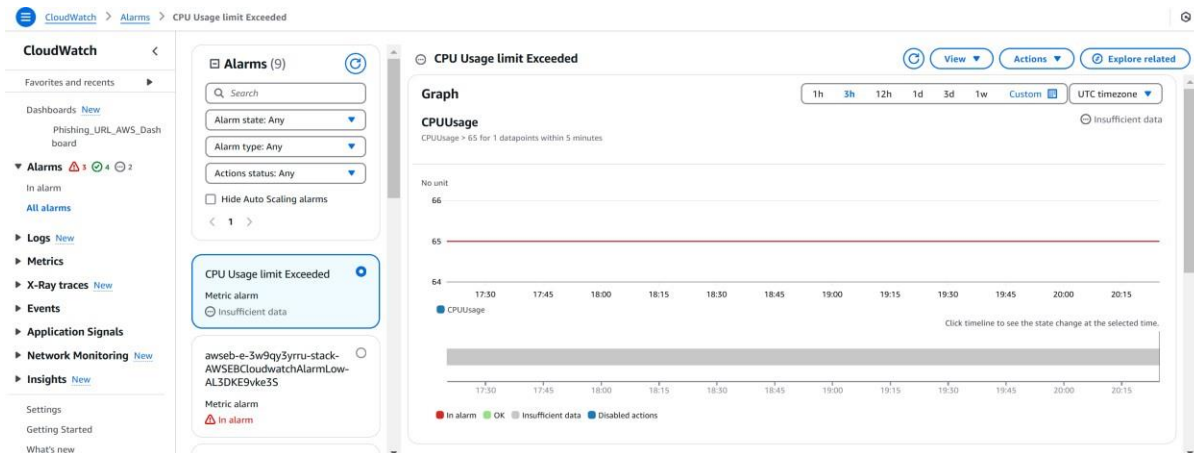


AWS CloudWatch dashboard is set up and the metrics such as CPU usage, Network I/O, Latency and AverageLoad is collected.

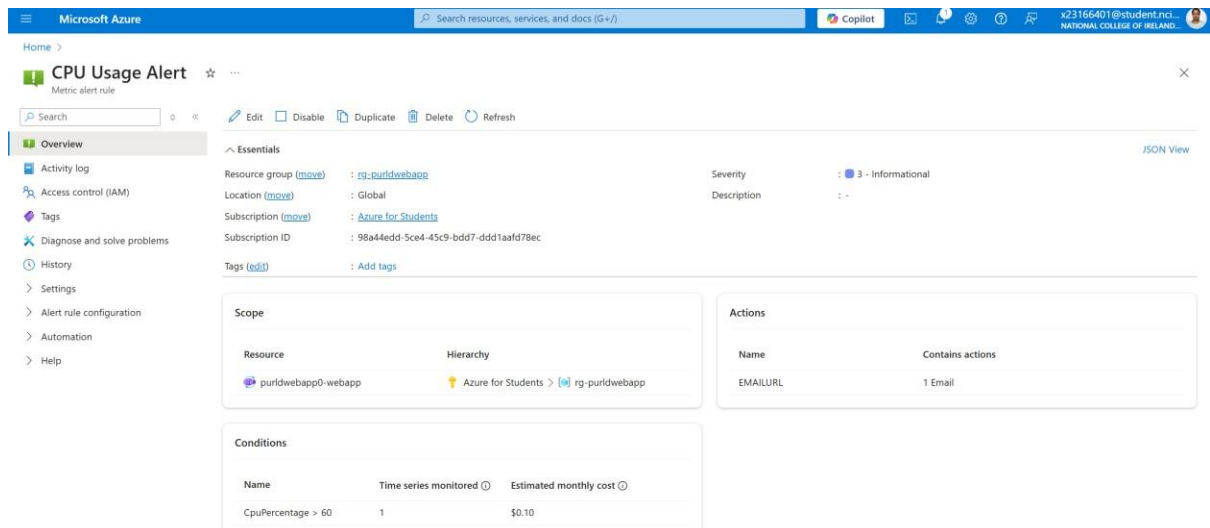


Azure Monitor Dashboard is set up to gather insights from the deployed application on Azure Container Apps. Metrics include CPU Usage, Average Response time, Network I/O and CPU usage Percentage.

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://colab.research.google.com/>.
- [2] "Flask Framework," [Online]. Available: <https://flask.palletsprojects.com/en/stable/>.
- [3] "Docker Desktop," [Online]. Available: <https://www.docker.com/>.
- [4] "AWS Documentation," [Online]. Available: https://aws.amazon.com/?nc2=h_lg.
- [5] "Azure Documentation," [Online]. Available: <https://azure.microsoft.com/en-us/>.
- [6] "Locust," [Online]. Available: <https://locust.io/>.
- [7] "GitHub," [Online]. Available: <https://github.com/>.