2024S-T2 CSDD1008 - Software Quality and Testing 01 (M11 Group 1)

In-Class Lab -6 - Building and Deploying a Go Application with Docker Swarm on Azure

Access the Application via Azure Public IP: http://4.248.12.230:8081/

Docker Hub Repository: https://hub.docker.com/r/saimeghana1/go-hostname-app

Git Hub Repo Link : https://github.com/MeghanaSangawar/In-Class-Lab-6

Group - A	
Shivaji Burgula	500224628
Jyothi Prasanna Kambam	500233773
Pranay Sai Chava	500236056
Yamini Kunapareddy	500234808
Sai Meghana Sangawar	500235387
Sushma Kambam	500233832

1. Development of the Go Application

Description of the Go application.

This Go application serves a web page displaying student information. It selects a random student from a predefined list and generates unique contact details (email and phone number) for the selected student. The information is displayed in a simple, styled HTML format. The program listens on port 8081 and shows the student's ID, name, email, and phone number . The server is started with the http.ListenAndServe function.

Steps to run the GoApplication Locally

1. Open and Run the Application

Open main.go in VSCode:

- Launch VSCode.
- Open the "main.go" file from your project directory.

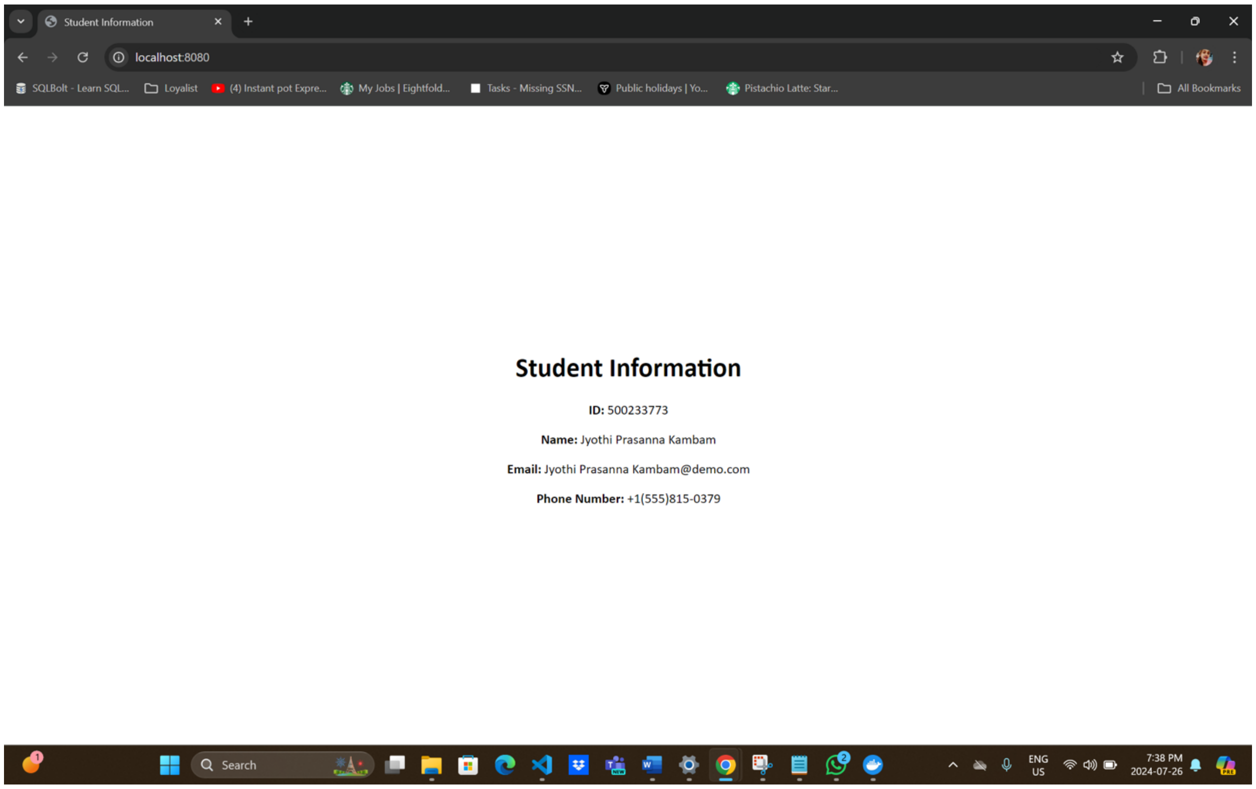
Run the Application:

Open the Terminal in VSCode

```
#In the terminal, execute the following command:
go run main.go
```

- Open any Browser in your computer
- Navigate to http://localhost:8080 to view the application running locally.

Screenshot of the application running locally.



Student Information

ID: 500233773

Name: Jyothi Prasanna Kambam

Email: Jyothi Prasanna Kambam@demo.com

Phone Number: +1(555)815-0379

2. Explanation of the code, including key functionalities.

1. **Imports and Type Definition:**
   - The code imports necessary packages like fmt, math/rand, net/http, os, and time for various functionalities such as formatting strings, generating random numbers, handling HTTP requests, accessing system information, and working with time.
   - A struct UserDetails is defined to hold a student's name, email, and phone number.
2. **Student List and Email Domains:**
   - A list of students with names and IDs is defined.
   - An array of email domains is provided to generate random email addresses.
3. **Utility Functions:**
   - generateUniqueEmail: Creates a unique email address for a student using a random domain.
   - generateUniquePhoneNumber: Generates a random phone number.
   - generateUserDetails: Populates the UserDetails struct with the student's name and the generated email and phone number.
4. **HTTP Handler Function:**
   - The handler function serves the main logic for the web server.
   - It seeds the random number generator to ensure unique random values on each request.
   - The machine's hostname is used to further randomize the selection of a student.
   - A student is randomly selected, and their details are generated.
   - The information is formatted into an HTML response, including the student's ID, name, email, and phone number.
5. **Main Function:**
   - The main function sets up the HTTP server to listen on port 8080 and uses the handler function to respond to incoming requests.
   - The server starts and waits for requests, serving the student information page when accessed.

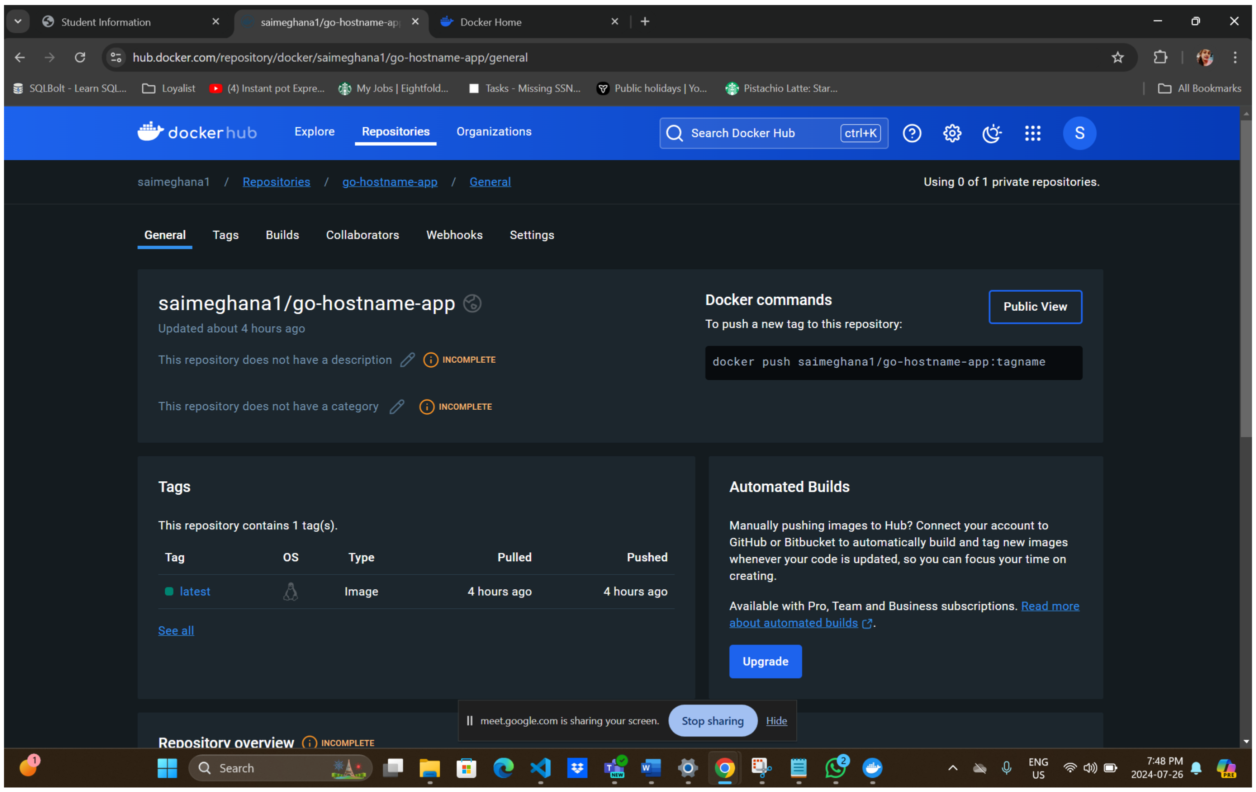
3. Creating the Docker Image

Dockerfile contents and explanation.

This Dockerfile sets up a Docker image for a Go application. It uses the Go version 1.22 image as the starting point. Inside the container, it sets the working folder to /app. Then, it copies all the files from the current folder on your computer into this folder in the container. It builds the Go application into an executable file named main with specific settings to make it work on Linux. The Dockerfile also tells Docker that the application will use port 8081. Finally, it sets the container to run the main file by default when it starts. This process ensures the Go application is ready to run in a Docker container.

```
#Commands to Build, Test, and Push a Docker Image

docker build -t saimeghana1/go-hostname-app:latest .

docker run -d -p 8081:8080 saimeghana1/go-hostname-app:latest

#Command to push the Docker Image to Docker Hub

docker push saimeghana1/go-hostname-app:latest
```

Screenshot of the Docker Hub repository page



hub.docker.com/repository/docker/saimeghana1/go-hostname-app/general

saimeghana1 / Repositories / go-hostname-app / General

Using 0 of 1 private repositories.

General | Tags | Builds | Collaborators | Webhooks | Settings

saimeghana1/go-hostname-app

Updated about 4 hours ago

This repository does not have a description — INCOMPLETE

This repository does not have a category — INCOMPLETE

Docker commands

To push a new tag to this repository:

```
docker push saimeghana1/go-hostname-app:tagname
```

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	4 hours ago	4 hours ago

See all

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. Read more about automated builds.

Upgrade

Repository overview — INCOMPLETE

meet.google.com is sharing your screen. Stop sharing Hide

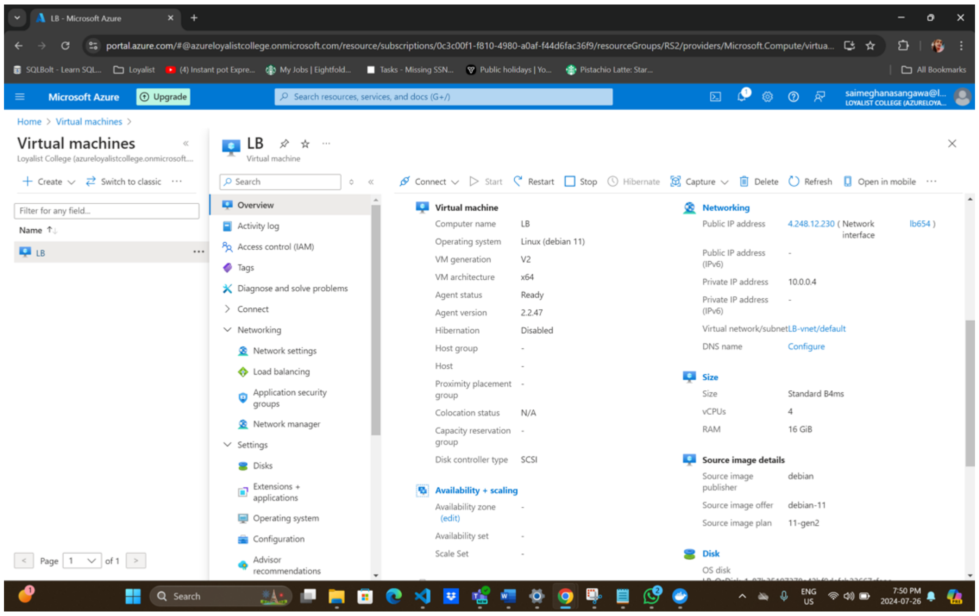
4. Setting Up Azure Virtual Machine

1. Create a Debian-Based VM:

1. Log in to the Azure Portal.
2. Go to “Virtual machines” and click “Add”.
3. Enter VM details:
    - Select subscription.
    - Create or choose a resource group.
    - Name VM.
    - Choose a region.
    - Select “Debian” as the image.
    - Pick the VM size.
    - Choose “Password” for authentication.
    - Enter your username and password.
4. Configure disks and networking.
5. Review your settings and click “Create”.

2. Configure Security Groups:

1. Go to the “Networking” tab of your VM.
2. Add inbound rules to allow:
    - Port 22 for SSH.
    - Port 80 for HTTP.
        - Port 443 for HTTPS.


portal.azure.com/#@azureloyalistcollege.onmicrosoft.com/resource/subscriptions/0c3c00f1-f810-4980-a0af-f44d6fac36f9/resourceGroups/RS2/providers/Microsoft.Compute/virtua...

saimeghanasangawa@l... LOYALIST COLLEGE (AZURELOYA...

Home > Virtual machines >

Virtual machines
Loyalist College (azureloyalistcollege.onmicrosoft....

LB
Virtual machine

Virtual machine

Computer name	LB
Operating system	Linux (debian 11)
VM generation	V2
VM architecture	x64
Agent status	Ready
Agent version	2.2.47
Hibernation	Disabled
Host group	-
Host	-
Proximity placement group	-
Colocation status	N/A
Capacity reservation group	-
Disk controller type	SCSI

Availability + scaling

Availability zone (edit)	-
Availability set	-
Scale Set	-

Networking

Public IP address	4.248.12.230 (Network interface lb654)
Public IP address (IPv6)	-
Private IP address	10.0.0.4
Private IP address (IPv6)	-
Virtual network/subnet	LB-vnet/default
DNS name	Configure

Size

Size	Standard B4ms
vCPUs	4
RAM	16 GiB

Source image details

Source image publisher	debian
Source image offer	debian-11
Source image plan	11-gen2

Disk

OS disk

5. Deploying Application with Docker Swarm

1.Initialize Docker Swarm: Start Docker Swarm mode on your VM to enable container orchestration.

2.Deploy the Service: Create a Docker service to run your application, specifying the number of replicas and the ports for exposure.

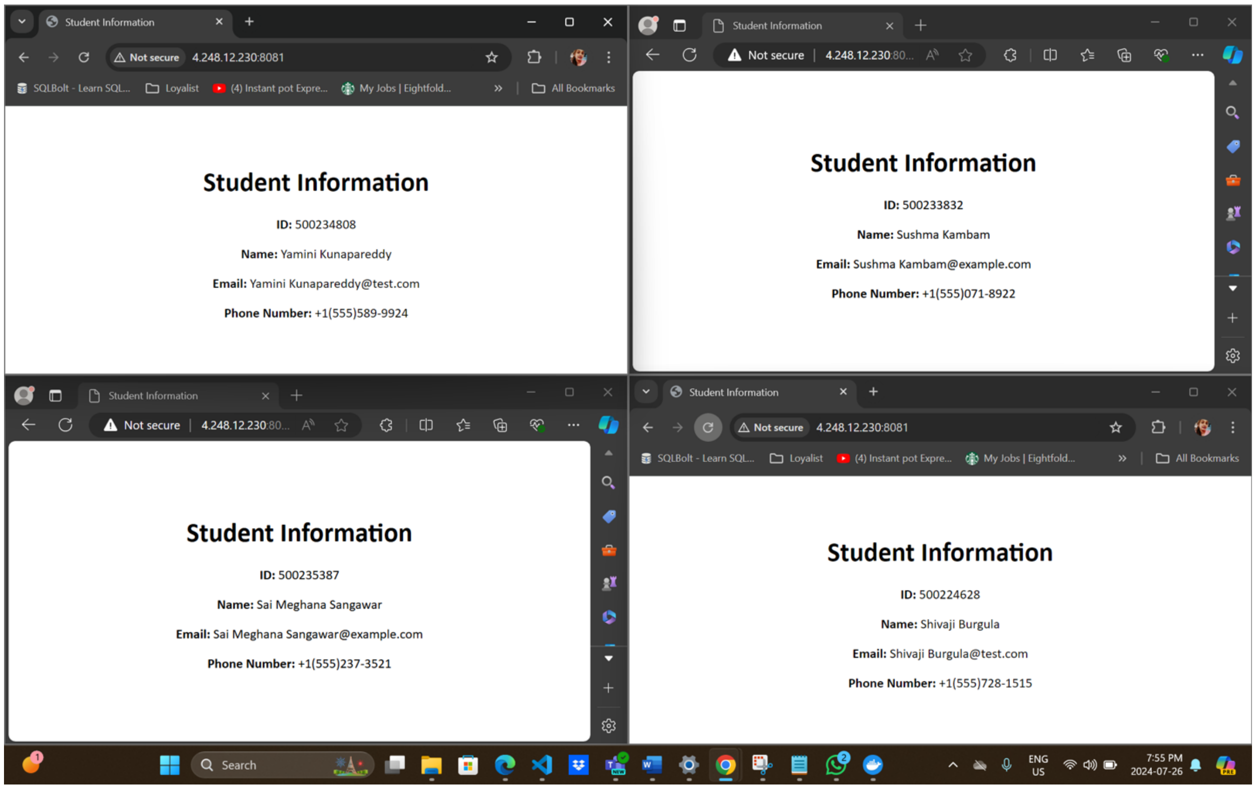
3.Test the Application: Verify that your application is running correctly by accessing it via your VM's public IP address and specified port.

4.Verify the Deployment: Inspect the service to ensure it is running as expected.

Commands to verify the deployment

```
# Initialize Docker Swarm mode on your VM

sudo docker swarm init

# Deploy a Docker service with 4 replicas, exposing port 8081 on the VM and mapping it to
port 8080 in the container

sudo docker service create --name myservice --publish 8081:8080 --replicas 4
saimeghana1/go-hostname-app:latest

# Check the status of the deployed service

sudo docker service ls

# Update the service to use the latest image version

sudo docker service update --image saimeghana1/go-hostname-app:latest myservice

# Test the application by accessing it via your VM's public IP address and port 8081

# Open a web browser and go to: http://<VM_PUBLIC_IP>:8081

# Recreate the service with the same settings

sudo docker service create --name myservice --publish 8081:8080 --replicas 4
saimeghana1/go-hostname-app:latest
```

Running multiple containers under one public IP Address in different browsers



Student Information

ID: 500234808

Name: Yamini Kunapareddy

Email: Yamini Kunapareddy@test.com

Phone Number: +1(555)589-9924

Student Information

ID: 500233832

Name: Sushma Kambam

Email: Sushma Kambam@example.com

Phone Number: +1(555)071-8922

Student Information

ID: 500235387

Name: Sai Meghana Sangawar

Email: Sai Meghana Sangawar@example.com

Phone Number: +1(555)237-3521

Student Information

ID: 500224628

Name: Shivaji Burgula

Email: Shivaji Burgula@test.com

Phone Number: +1(555)728-1515