**Code Generator Using CI/CD Pipeline**

**ABSTRACT**

Continuous Integration/Continuous Deployment (CI/CD) is a development approach that helps streamline the software development process by automating various stages of the software development lifecycle. More than that, the primary aim of this project is to create a Continuous Integration/Continuous Deployment pipeline for a QR code generator and scanner application. The application will be developed using Python and deployed in Docker containers, which are managed using Docker Swarm. The project uses a variety of DevOps tools such as Azure, and AWS for building virtual machines, Terraform for infrastructure as code, Ansible for automation, Nagios for monitoring, and Jenkins for continuous integration and delivery. We will use Terraform to create the required infrastructure in Azure including the resources like virtual networks, subnets, security groups, and virtual machines

**INTRODUCTION**

This project focuses on implementing a Continuous Integration/Continuous Deployment (CI/CD) pipeline for a software development team. This project aims to automate the entire software development lifecycle, from code changes to deployment, while ensuring quality and reliability. The CI/CD pipeline consists of multiple stages, including building, testing, and deploying the application. We use modern tools and technologies such as Git, Jenkins, Docker, and Kubernetes to automate the pipeline in this project.

The CI/CD pipeline will automate the application's build and deployment processes, ensuring that the most recent changes are smoothly integrated, tested, and deployed. Jenkins will be used as the continuous integration tool, automatically initiating builds, testing, and deploying changes anytime they are uploaded to the code repository. The source code is committed to the version control system GIT, and updates are automatically grabbed by Jenkins before being tested and deployed as a Docker Swarm container. Terraform will be used to automate the provisioning of the cloud infrastructure that will host the docker swarm cluster, while Ansible will be used to manage the life cycle of the docker containers that will operate in the docker swarm cluster.  
 Clients will benefit from numerous aspects of the project, including faster and more reliable deployments, decreased human tasks, and increased application quality. As a result, implementing a CI/CD pipeline for the QR code generator and scanner application would increase the efficiency and reliability of the application development and deployment process dramatically. Another key feature that improves the end-user experience is high availability and fault tolerance.

**Pipeline Architecture:**

Diagram

Description automatically generated

**Methodology**

**Step 1: Create the application using Python**

The application is created using Python, and the code is stored in the version control system (GITHUB) repository. The source code is available in the below GitHub URL (mentioned below)

* Application is a web app that uses the Pywebio, flask framework
* It generates two types of codes, **QR codes, and Bar codes.** Codes are customizable too.
* It also allows users to download the generated codes

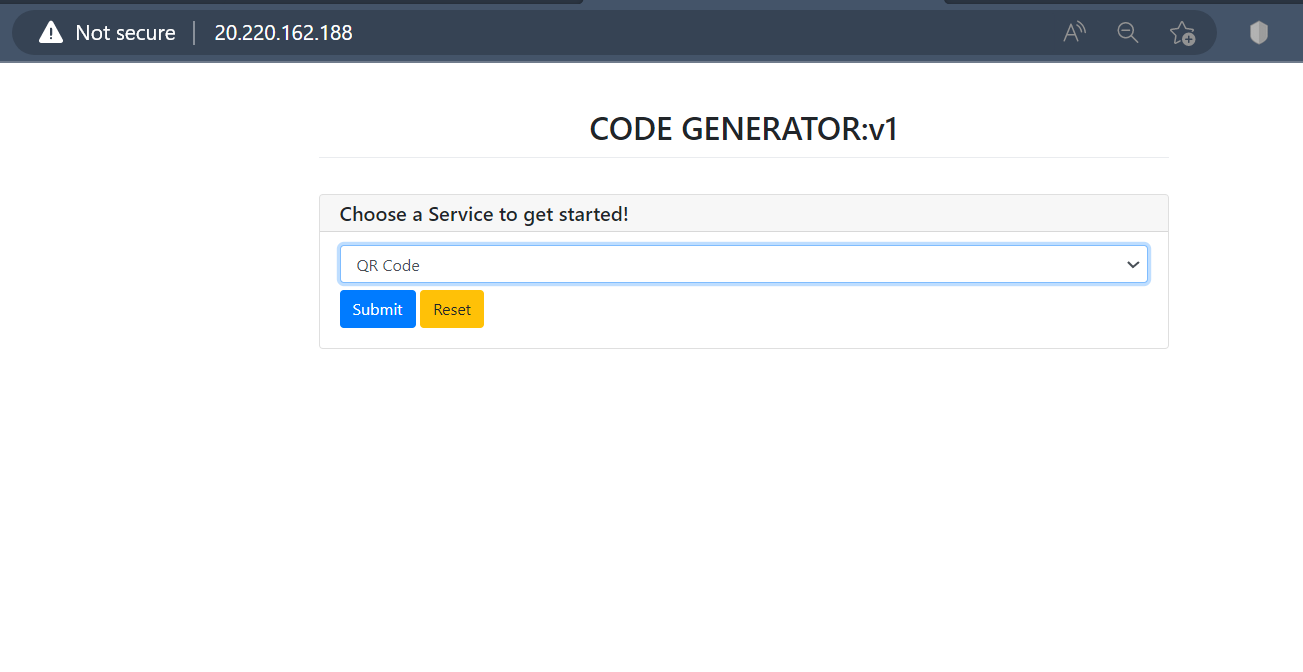
GitHub URL: <https://github.com/aswinks123/Devops-Project-Code-generator.git>

Sample Source code:

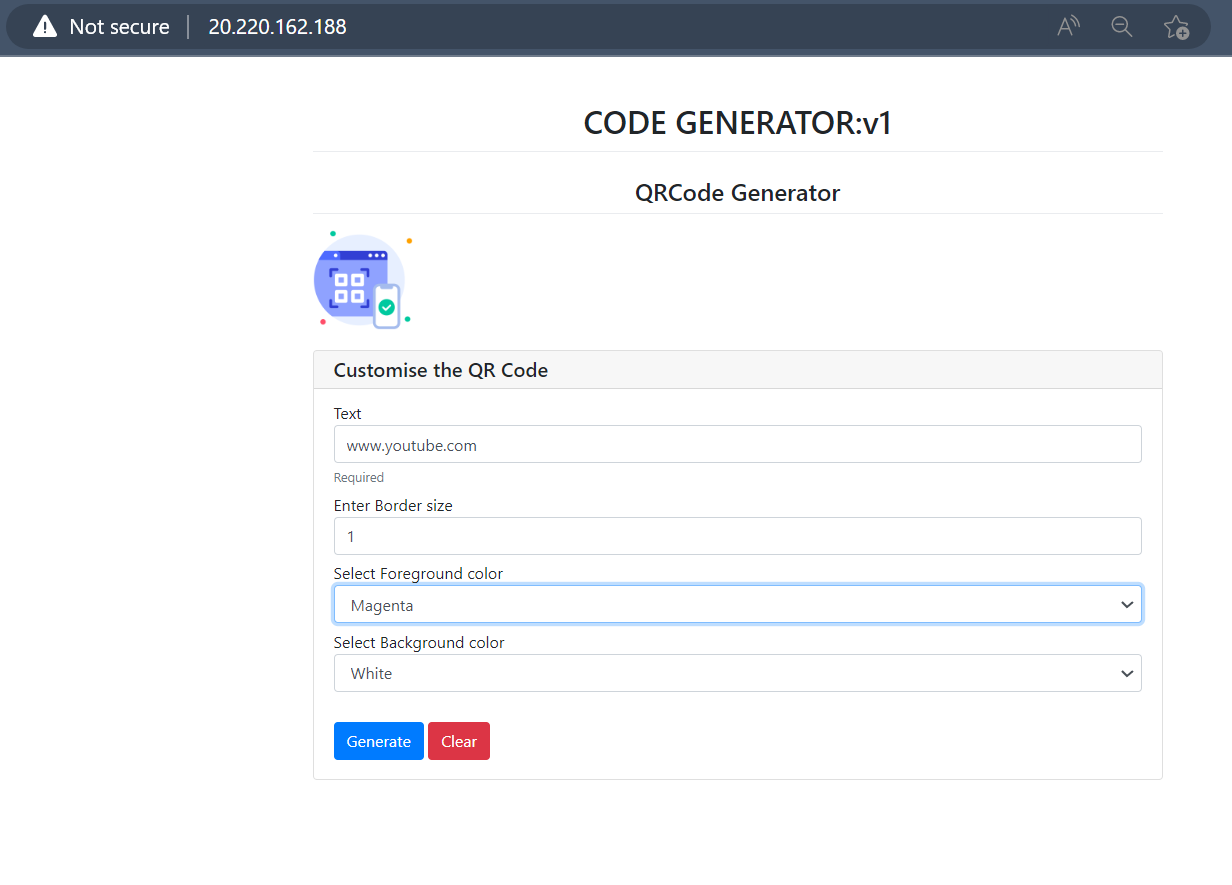
**Text

Description automatically generated**

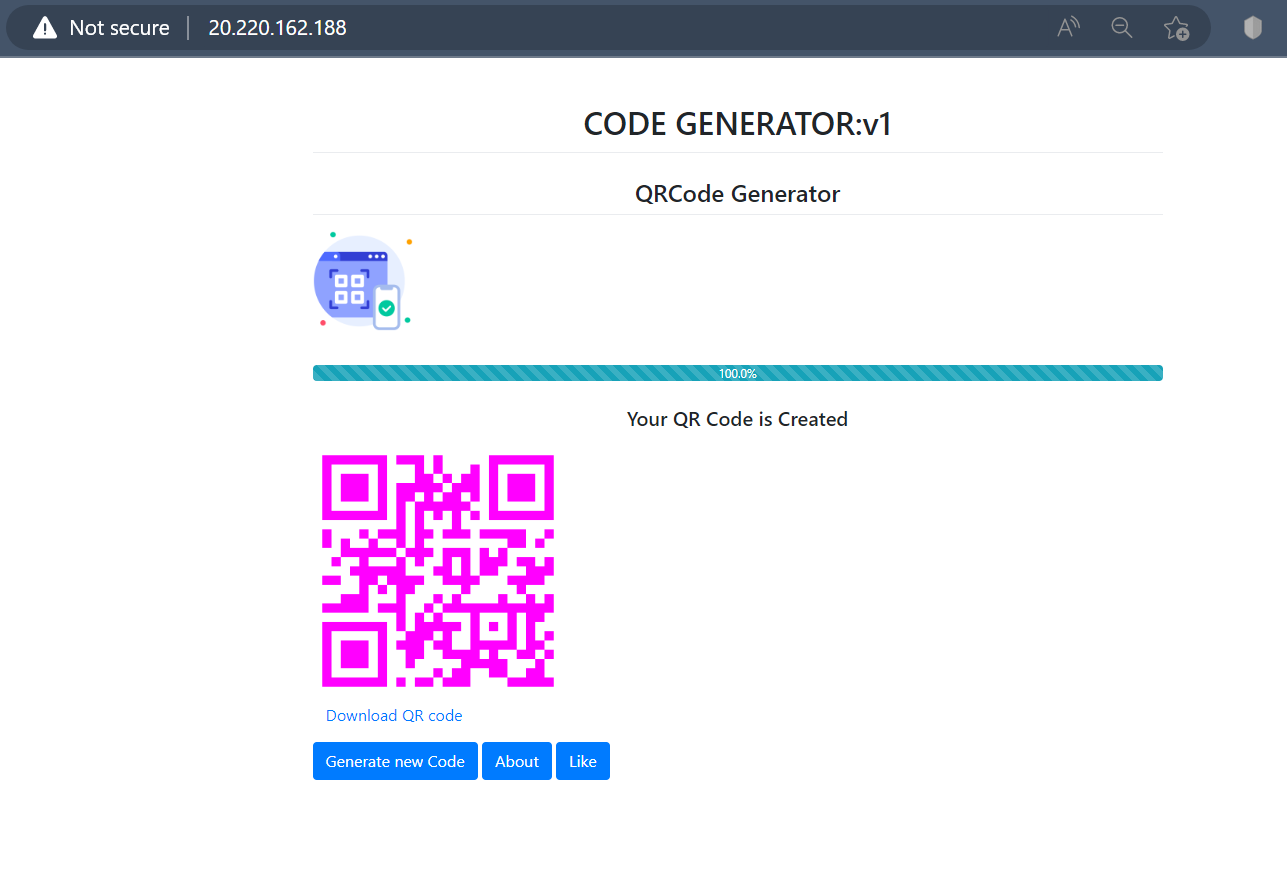
**Application User Interface**



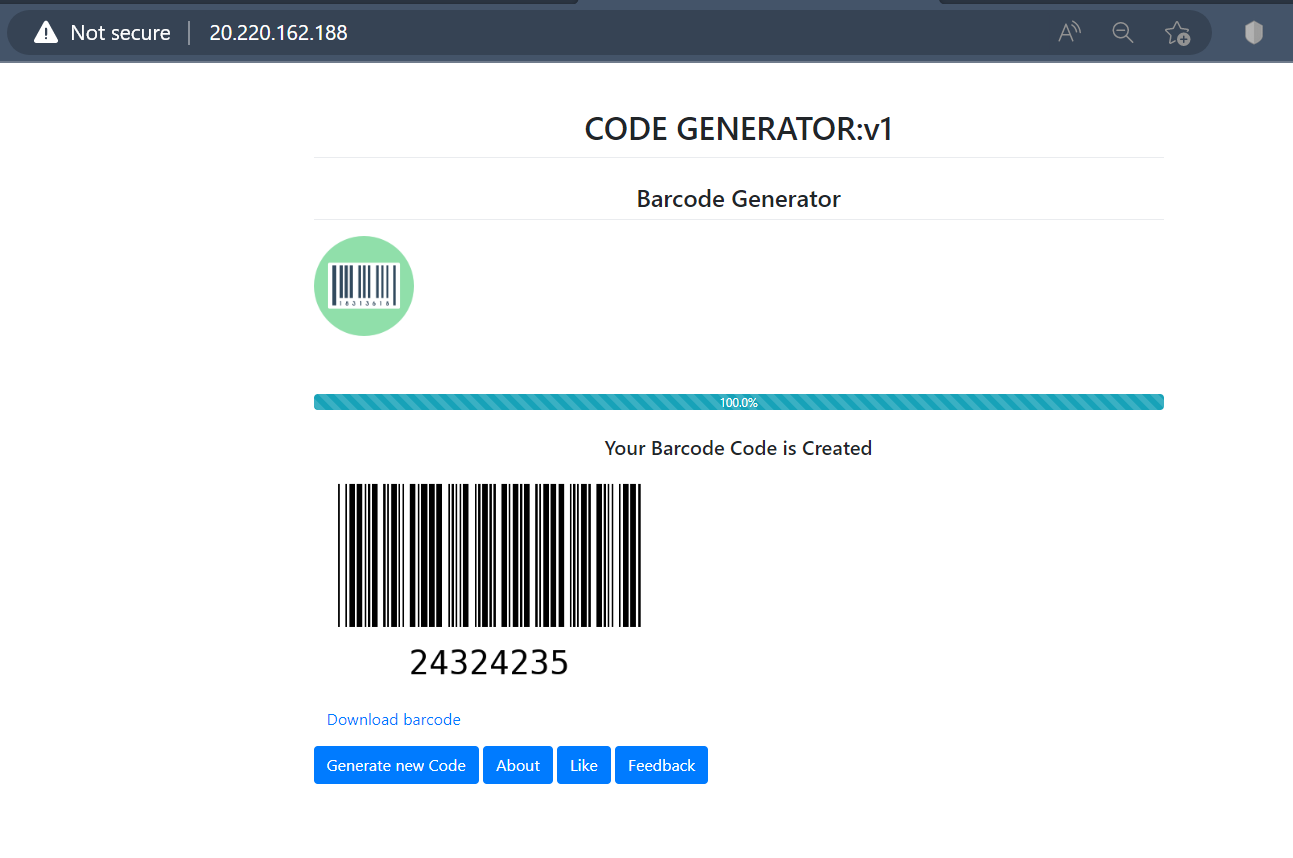
**Home page of webapp**



**Options for QR code**



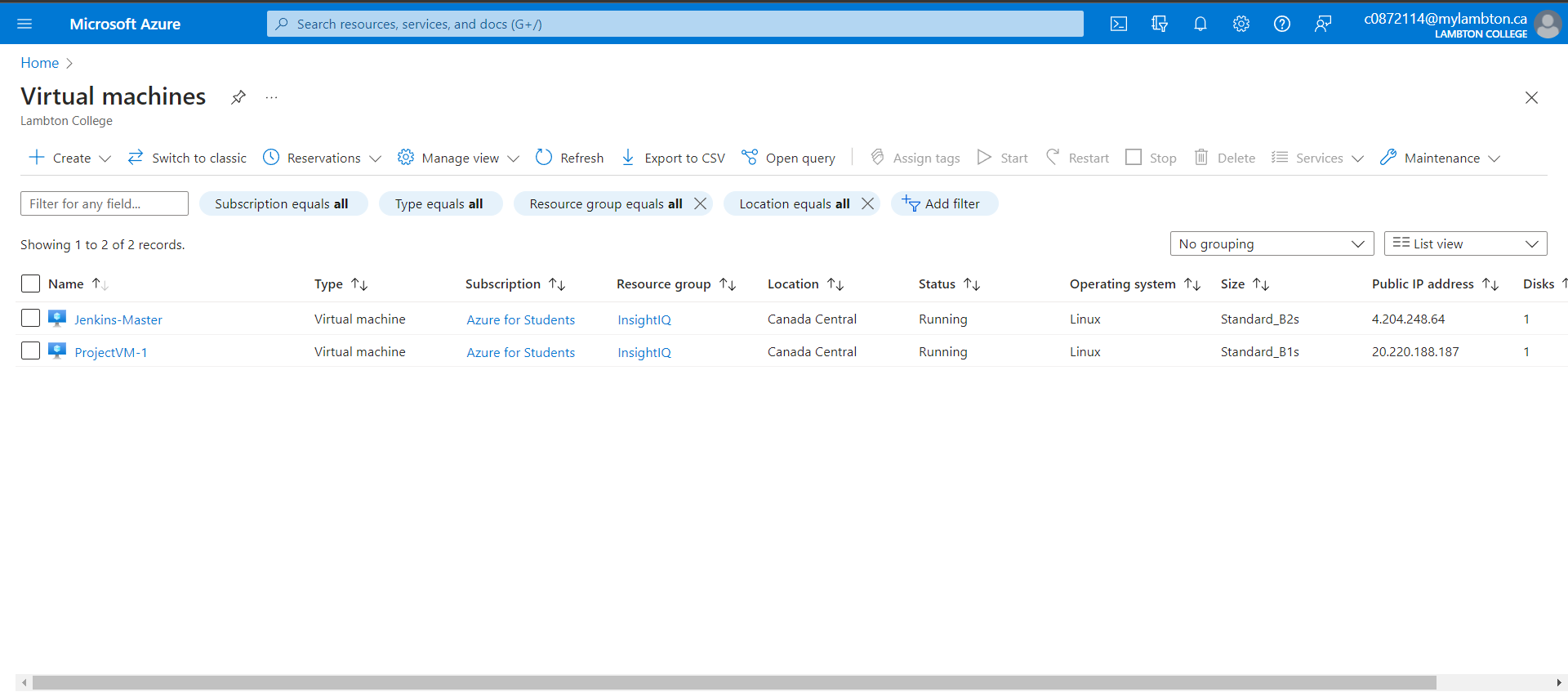
**Generated QR code**



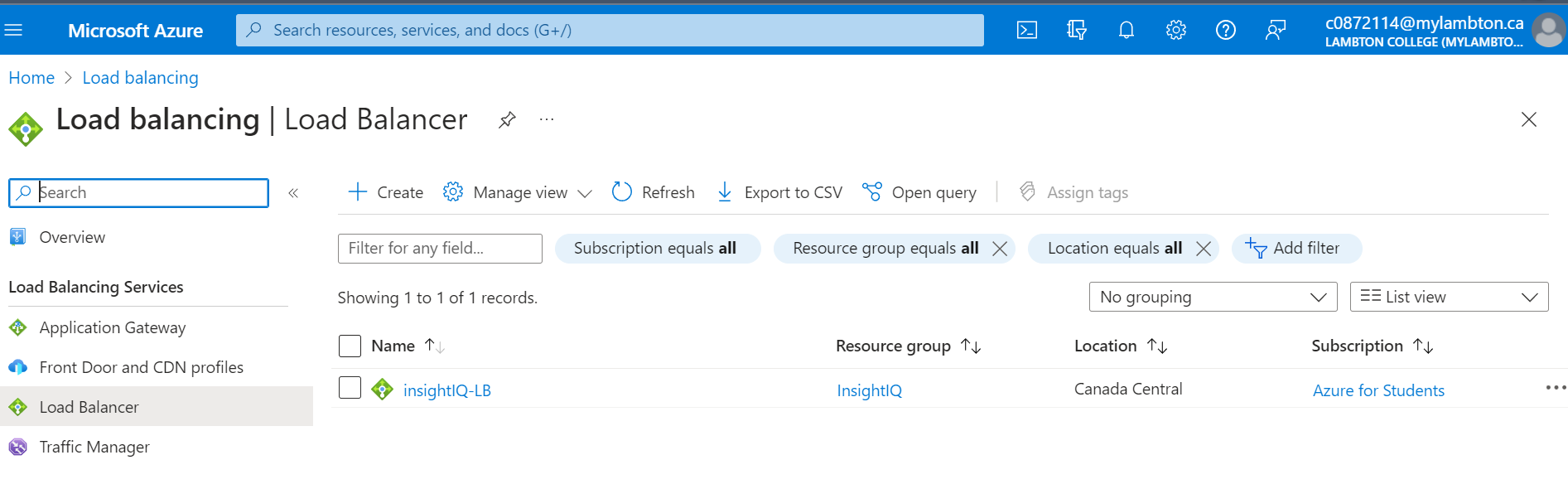
**Generated barcode**

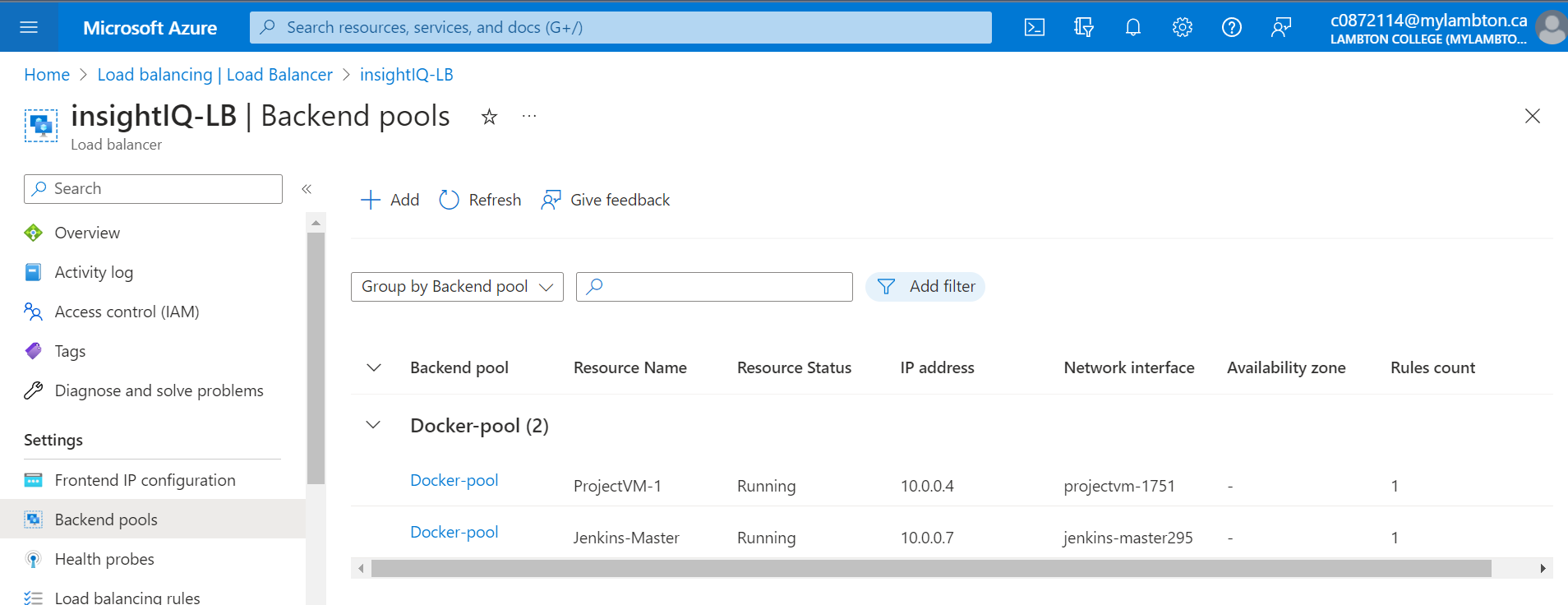
**Step 2: Create the Virtual machines and Load Balancer in Azure for the Jenkins server and Docker swarm.**

Two virtual machines are created in the Azure cloud platform named Jenkins master server and Docker. The Jenkins server is used for automation purposes, and Docker is used for application deployment.

****

A standard Azure load balancer is created to distribute the traffic between the docker nodes and act as a front-end IP for users to access the web application





**Step 3: Pull the Source code from the GitHub repository to the Jenkins server using pipelines.**

Jenkins is an open-source automation server used for building, testing, and deploying software. It provides a wide range of plugins and integrations with other tools, allowing it to be customized to meet the needs. We used the Jenkins tool to automate the entire software development lifecycle, from building and testing code to deploying in Docker environments.

When any changes are made to the source code stored in the GitHub repository Jenkins will automatically pull the source code using the Git Webhook feature and initiate the Jenkins pipeline.

**The Deployment stages in Jenkins:**

* Stage-1: Declarative: Checkout SCM

If any new changes are pushed to the repository Jenkins triggers the job

* Stage-2: Clone GitHub Repository

In this stage cloning the Git Hub repository and updating the app.py file.

* Stage-3: Copy source code to Docker swarm

The updated source code is then pushed to Docker Swarm

* Stage-4: Build & Push the new Image to the Docker hub

Docker will build a new image using the updated code and push this image to Docker Hub

* Stage-5: Deploying new Service in Docker Swarm

A new service is deployed using the image and updated changes are reflected in the application.

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

**Step 4: Deployment of the application using Docker Swarm**

Docker is a software platform that allows developers to create, deploy, and run applications in containers easily. Containers are lightweight, portable, and self-contained environments that can run an application and all its dependencies. The Updated source code is pulled to docker from the Jenkins server.

* Docker swarm is a container orchestration platform consisting of a cluster of Docker Nodes (Master and Worker)
* It helps in the high availability and fault tolerance of containers
* We have 2 nodes, one Master and another Worker
* Master node controls the cluster operations.
* Cluster is built on top of Azure Virtual machines

Once the image is built by the docker, it will be pushed and stored in Docker Hub.

Graphical user interface, text, application, email

Description automatically generated

This image is pulled by the docker swarm. Using this image, we created a docker swarm service with two replicas running on two different nodes for high availability and fault tolerance. The service is listening on port 80, and we have deployed an Azure load balancer that will distribute the traffic between the two docker servers.

**Graphical user interface, application, website

Description automatically generated**

Docker swarm running service (insightiq) with 2 replicas.

**Text

Description automatically generated**

Containers are distributed across 2 nodes

**Text

Description automatically generated**

Details of Nodes participating in the cluster

Docker swarm service: A service defines the tasks to execute on the manager or worker nodes. Service can run across multiple nodes and support advanced features like replication, high availability, automated container migration, etc.

**Infrastructure as Code (Terraform)**

* We used Terraform as an Infrastructure as a code tool to deploy our virtual machines.
* Resources are created using the Azure provider
* Following resources are created:
* Resource group, VM, security group, Virtual network
* network interface, Public IP.

**Text

Description automatically generated**

**Configuration Management (Ansible)**

* We used Ansible as a configuration management tool to automate the container lifecycle.
* By using the Ansible playbook, we did the following tasks
* Copy the code to the remote docker server
* Destroy the previously running containers before deploying new containers
* Build, and push the docker image to the docker hub

**Text

Description automatically generated**

**Step 5: Monitoring the VMs using Nagios**

**Overview of Nagios:**

Monitoring is an essential aspect of the DevOps pipeline. It helps in ensuring that the application infrastructure and its components are running smoothly and efficiently. One of the most significant advantages of monitoring is its ability to identify issues before they become critical, reducing the downtime of the application.

Graphical user interface, text, application, email

Description automatically generated

Nagios is an open-source monitoring tool that provides real-time monitoring of the application infrastructure and its components. In our project, we have opted to use the AWS cloud service to provide High Availability (HA) for the monitoring part. This ensures that the monitoring system is always available, even in the case of failure of Azure service where the remote hosts are deployed. AWS is a reliable cloud service provider that offers a range of tools and services to ensure high availability, scalability, and security. To deploy Nagios, we have used the Apache web server. The Apache web server is a widely used web server known for its stability and security. Nagios can be deployed on the Apache web server to provide a robust and scalable monitoring system.

**Nagios Configuration:**

**Nagios Configuration:**  
 Nagios configuration is a crucial step in setting up an effective monitoring solution for any application infrastructure. Nagios will be used to monitor the virtual machine running the applications. This will include defining checks, notifications, and escalations to ensure the smooth operation of the application. As part of the Nagios configuration, device information such as hostname, IP address, and other details will be marked. This will help in identifying the virtual machine quickly and taking appropriate action in case of any issues. In addition, the configuration of Nagios will include setting up a dashboard for real-time monitoring and alerts for critical issues. The dashboard will provide a comprehensive view of the health and status of the virtual machine and the services it is running. This will help in identifying issues early on and taking corrective action before they impact the application’s performance or availability.

**Nagios Dashboard:**

The Nagios dashboard is a critical component of the Nagios monitoring system that provides a comprehensive view of the health and status of the virtual machine, network, and services of the application. It provides a centralized location for monitoring various components and services, making it easier to manage and troubleshoot issues. The right side of the Nagios dashboard interface provides various analysis tools and services that can be used to judge the availability and performance of virtual machines. These tools and services include graphs, charts, and tables that display metrics such as CPU usage, memory usage, and network bandwidth.

Graphical user interface, website

Description automatically generated

Graphical user interface, table

Description automatically generated with medium confidence

**Nagios Plugins:** Nagios plugins are essential components of the Nagios monitoring system that allow you to monitor various components and services of remote host machines. One of the most commonly used Nagios plugins is the NRPE (Nagios Remote Plugin Executor) plugin, which enables you to execute plugins on remote hosts. This plugin allows you to monitor various aspects of the application, such as CPU usage, disk space, and memory usage. By monitoring these critical resources, you can identify potential issues before they cause significant downtime or failure of the application.

The metrics monitored by Nagios plugins are customizable, allowing you to configure them to meet your specific monitoring requirements. This provides a significant advantage in terms of providing a comprehensive view of the application's health and performance, which is vital for identifying and resolving potential issues. The NRPE daemon runs on the remote host, and the check\_nrpe plugin runs on the Nagios server. The check\_nrpe plugin sends requests to the NRPE daemon on the remote host to execute the desired plugin and return the results. This communication between the Nagios server and the NRPE daemon enables the monitoring of a wide range of metrics, which is essential for ensuring the application's health and performance.

**Nagios Reporting:**

Nagios reporting is a powerful feature that provides comprehensive insights into the performance and trends of the monitored infrastructure. Reports can be generated for each remote host as well as the local host. These reports provide detailed information about the performance metrics, including the uptime, availability, and response time of the monitored services. The reports also help in identifying issues and improving the efficiency and reliability of the application by highlighting the areas that require attention. The data is presented in an easy-to-understand format, making it easier for the DevOps team to take corrective actions.

Graphical user interface, text

Description automatically generated

**Nagios Notifications:**  
 Nagios has a powerful notification system that can be used to alert DevOps teams about any issues detected during monitoring. This feature plays a crucial role in reducing the downtime of the application by notifying the responsible team as soon as an issue is detected.

Graphical user interface, text, application

Description automatically generated

The customizable email templates of Nagios can be configured to include detailed information about the problem, its severity, and its potential impact on the application or infrastructure. This helps DevOps teams quickly identify the issue and take necessary action to resolve it. Nagios allows you to configure email notifications to be sent to specific individuals or groups based on their roles or responsibility. This feature ensures that the right person or team is notified about the issue, reducing the response time and improving the overall efficiency of the monitoring system.

**Step 6: Elaborated visualization of the VMs performance using DataDog.**

Datadog is a cloud monitoring tool that provides real-time insights into the performance of applications, infrastructure, and networks. In this technical report, I will document the steps I have taken to install and configure the Datadog agents and create a dashboard for monitoring a web application.

* Installation and Configuration of Datadog Agents:
* Sign up for a Datadog account: First, We have to sign up for a Datadog account by visiting their website and creating an account.
* Install the Datadog Agent on the server: we have installed the Datadog Agent on the server

Graphical user interface, text, application, email

Description automatically generated

Start the Datadog Agent: After configuring the Datadog Agent, start it using the following command: *“sudo systemctl start datadog-agent”*

* Log in to Datadog: After starting the Datadog Agent.
* Create a new dashboard: In the Datadog dashboard, create a new dashboard by clicking on the “New Dashboard” button.

Graphical user interface, application

Description automatically generated

Add widgets to the dashboard: I added various widgets to the dashboard to monitor my web application’s performance. For example, I added a widget to monitor the system CPU and system RAM and SWAP.

A screenshot of a computer

Description automatically generated

**Conclusion**

In conclusion, the Continuous Integration/Continuous Deployment (CI/CD) approach is an essential element of modern software development that helps streamline the software development process by automating various stages of the software development lifecycle. This project demonstrates the practical application of CI/CD by creating a pipeline for a QR code generator and scanner application using Python and Docker containers managed by Docker Swarm. The project leverages a variety of DevOps tools such as Azure, AWS, Terraform, Ansible, Nagios, and Jenkins to enable continuous integration and delivery. With Terraform, the required infrastructure in Azure, including virtual networks, subnets, security groups, and virtual machines, can be created seamlessly. This project provides a clear example of how DevOps tools can be used to simplify the software development process and improve the efficiency and reliability of software delivery. For monitoring, the Nagios is an essential tool in the DevOps pipeline that helps in identifying issues, reducing downtime, and improving the efficiency and reliability of the remote hosts. Nagios configuration, dashboard, plugins, reporting, and notifications provide a comprehensive view of the VM’s health and performance.

**References**

1. *PyWebIO - Build full stack web app with Python*. (n.d.).

<https://www.pyweb.io/>

2. *Swarm mode overview*. (2023, February 9). Docker Documentation.

<https://docs.docker.com/engine/swarm/>

3. *Table Of Contents · Nagios Core Documentation*. (n.d.).

[https://as-sets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/toc.html](https://as-sets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/toc.html%20)

4. *Getting Started with the Agent*. (n.d.). Datadog Infrastructure and Application Monitoring.

<https://docs.datadoghq.com/getting_started/agent/>

5. Trapani, K. (2023). How to Integrate Jenkins with GitHub. *Cprime*.

<https://www.cprime.com/resources/blog/how-to-integrate-jenkins-github/>

**Appendix**

**Links:**

Source code link: <https://github.com/insightiq/insightiq-devops>

Docker hub repository: [https://hub.docker.com/r/1365890/devops\_projects](https://hub.docker.com/r/1365890/devops_projects%20)