Documentation & Reflection Report

1. Introduction

This project provisioned a Docker container having a sample application, and deployed this on cloud infrastructure using Terraform, managed the configuration using Ansible, and used Azure DevOps for the CI/CD pipeline. This involved the creation of pillars such as infrastructure and containerization of the application as well as the building and deployment of the pipeline.

2. Infrastructure Provisioning and Configuration

The cloud infrastructural requirements were set up using Terraform, launching a VPC, one public subnet, Internet Gateway and security groups. The Docker container was hosted on an EC2 instance which was installed for the purpose.

Challenges & Solutions: Reliable network access was ensured by establishing the correct security groups for communication. EC2 instance boot up process was used to automate Docker installation through a user data script.

3. Application Containerization

A Docker image was created using a Dockerfile, where the app was decomposed into its dependencies, environment variables and how the app should run.

Challenges & Solutions: Testing was an issue within the container. This was done by incorporating the right ENV and COPY statements in the Dockerfile as per the app's need.

4. Automation of Docker Container Deployment

Docker was installed on the instance with the help of Ansible and the application was deployed using Ansible so that it could provide an identical setup whether it is being deployed for the first time or for a second time.

Challenges & Solutions: OS configuration difference especially between Ubuntu and Amazon Linux was addressed through new condition check in the playbook to account for different packages.

5. CI/CD Pipeline Integration

CI/CD pipeline was integrated at Azure DevOps to begin as soon as there was update on the Github repository. It then used the Docker file to build a Docker container and pushed the image to docker, and lastly created an Azure Web App and deployed the program there.

Challenges & Solutions: The main issue was determining how authentications should work for Docker and Azure Web App. This was addressed by using service connections and storing the password for the virtual machine as a Pipeline Secret in Azure Pipelines.

6. Alternative Approaches & Suggested Improvements

•Kubernetes: Instead of EC2, Amazon EKS or Azure AKS could be used to have a better control of the containers.

•Serverless: It could be less complex to have Azure Functions or AWS Lambda to take care of the work.

• Automated Testing: The introduction of automated tests in the CI workflow means that the good quality code would be delivered to the CD pipeline.

Improvements:

• Error Handling: Some improvement could come from error handling and making notifications in the CI/CD pipeline appear and run all on their own as it would help in having more visibility in deployments.

•Documentation: Better documenting projects like README files would benefit the next developer who has to work on the project or spend less time onboarding.