DevOps Engineer Home Assignment

Introduction

This assignment is designed to evaluate your knowledge and skills in DevOps practices, including infrastructure provisioning, containerization, CI/CD pipeline creation, and Kubernetes deployment. The goal is to assess how you approach and execute tasks in these areas while maintaining clarity and organization.

Please focus on delivering a functional, well-documented solution while adhering to the instructions below.

General Instructions

1. Create a **GitHub Repository** for this task and ensure all your code and configurations are stored there.
2. Use **Terraform** to manage the infrastructure in your chosen cloud provider (**AWS** or **Azure**).
3. Document your steps and decisions in a *`README.md`* file.
4. Your repository structure should be clear and organized, reflecting industry best practices.

Assignment Tasks

1. Infrastructure Provisioning

- Use Terraform to provision the following resources:
  - A **Kubernetes Service**:
    - Azure Kubernetes Service (AKS) or Amazon Elastic Kubernetes Service (EKS).
    - Single node pool (minimum setup).
  - A **Storage Account**:
    - Azure Storage Account or AWS S3 Bucket for managing the Terraform state file.
  - Configure the Kubernetes cluster to be accessible via a load balancer or external IP.

2. Basic Web Application

- Develop or clone a simple "Hello World" web application in your preferred programming language (e.g., Python, Node.js, Go).
- Include a health-check endpoint *(`/healthz`)* in the application.

3. Containerization
- Write a *`Dockerfile`* to containerize the web application.
- Build the Docker image and push it to a container registry:
  - Use Azure Container Registry (ACR) or AWS Elastic Container Registry (ECR).

4. CI/CD Pipeline for Application
- Create a **GitHub Actions pipeline** that automates:
  - Building the Docker image from the application source code.
  - Pushing the image to the container registry.
  - Deploying the application to the Kubernetes cluster using Kubernetes manifests or Helm charts.

5. Application Verification
- Verify that the application was successfully deployed using the CI/CD pipeline.
- Ensure the application is accessible via a public URL by performing a manual test.
- Document any issues or adjustments made during this step.

6. Documentation & Verification
- Document your setup and approach in a *`README.md`* file. Include:
  - Steps to provision the infrastructure.
  - Steps to run the CI/CD pipeline.
  - Verification steps to ensure the application is accessible.
- Provide any commands or scripts used in the process.

Bonus Tasks (Optional)
- Implement a **Horizontal Pod Autoscaler (HPA)** for the application and test its functionality.
- Use a benchmarking tool (e.g., Apache Bench) to simulate load and validate the HPA setup.

Evaluation Criteria
Your submission will be evaluated based on:
1. **Completeness**: Does the solution meet the required tasks?
2. **Functionality**: Does the infrastructure deploy correctly, and is the application accessible?
3. **Code Quality**: Is the code organized, reusable, and easy to understand?
4. **Documentation**: Is the `README.md` file comprehensive and clear?
5. **Best Practices**: Are industry standards followed for Terraform, Docker, Kubernetes, and CI/CD pipelines?

Time Estimate

The assignment is designed to take approximately **8 hours** of hands-on work.

Submission

1. Share the link to your GitHub repository containing all the code, configuration, and documentation.
2. Ensure the repository is public or provide access permissions for review.

Thank you for your efforts, and we look forward to reviewing your submission!