DEVOPS ENGINEER TEST

Test 1: Multi-Language Microservice Deployment

Objective: Create a microservice architecture using Node.js, Python, and Go. Each service should perform a simple task (e.g., string manipulation, arithmetic operation, data retrieval) and communicate with the other services.

Tasks:

1. **Node.js Service:** Create a RESTful API service in Node.js that accepts requests and forwards them to the Python and Go services.
2. **Python Service:** Develop a Python service that performs a simple data processing task (e.g., sorting a list of numbers received from the Node.js service).
3. **Go Service:** Implement a Go service that logs the requests received and responses sent, and returns a confirmation back to the Node.js service.

Additional Requirements:

- Containerize each service using Docker.
- Write a Docker Compose file to orchestrate the multi-service deployment.

Sample Input/Output:

- Input to Node.js service: A list of numbers.
- Output from Python service: Sorted list.
- Log from Go service: Timestamped record of requests and responses.

Evaluation Criteria:

- Correct implementation of services in Node.js, Python, and Go.
- Successful containerization and orchestration of services.
- Effective communication between services.

Test 2: Infrastructure as Code for CI/CD Pipeline

Objective: Set up a CI/CD pipeline for a simple application (use any of the three languages: Node.js, Python, Go) using Terraform and a cloud provider of your choice (AWS, Azure, GCP).

Tasks:

1. **Application Setup:** Create a basic application in Node.js/Python/Go.
2. **Terraform Configuration:** Write Terraform scripts to provision necessary cloud resources for the application (e.g., compute instances, load balancers).
3. **CI/CD Pipeline Setup:** Configure a CI/CD pipeline using Jenkins/GitHub Actions/CircleCI for automated testing and deployment of the application.

Additional Requirements:

- Ensure the application is deployed to a scalable and secure cloud environment.
- Document the steps and choices made in the configuration.

Sample Test Cases:

- Terraform script successfully creates and configures cloud resources.
- The CI/CD pipeline automatically triggers on code push, runs tests, and deploys the application.
- The application is accessible and functional post-deployment.

Evaluation Criteria:

- Correct application setup and functioning.
- Effective use of Terraform for infrastructure provisioning.
- Successful configuration and execution of the CI/CD pipeline.

Submission Guidelines:

Submit your code as a ZIP file. Ensure the zip file includes a README with instructions on how to run your application, how you have used each library in the task, and their purpose.

Note: Don’t worry about copyright. If you are not selected, this task is yours to showcase on your GitHub forever.