DevOps Assignment- 7 Solutions

Requirement :

The sample application is developed using Go. Our development team would like to deliver this application to Production. As a DevOps engineer, you are responsible to complete the tasks by following these key areas: High Availability, Scalability, Security.

Overview :

This project automates the deployment of a Go-based application using Docker, Kubernetes, and ArgoCD while ensuring high availability, scalability, and security. The pipeline is fully automated using GitOps practices, leveraging Terraform for infrastructure management and ArgoCD for deployment.

Prerequisites :

1. Google Cloud Platform (GCP) account with sufficient permissions to create resources (IAM roles, GKE, etc.).
2. Docker installed for building and pushing the Docker image.
3. Terraform installed for provisioning the GKE cluster.
4. Kustomize for managing Kubernetes manifests.
5. ArgoCD installed for GitOps deployment.
6. GitHub Actions for CI/CD pipeline (or alternative CI/CD tools like GitLab, Jenkins, etc.).

Setup Instructions :

1. Dockerfile:

   Create a Dockerfile to build the Go application. This will be used to build and deploy the application inside a container.

```
Stage 1: Build the Go application
FROM golang:1.21 AS builder

WORKDIR /app

#Copy the source code
COPY . .

#Download dependencies
RUN go mod tidy

#Build the application
RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o app

Stage 2: Create a minimal runtime image
FROM alpine:latest

WORKDIR /root/

#Copy the compiled binary from the builder stage
COPY --from=builder /app/app .

#Set the command to run the application
CMD ["./app"]
```

2. Build & Push Docker Image

To build and push the Docker image to Docker Hub, run the following commands:

1. bash
2. Log in to Docker Hub
3. docker login
4. Build the Docker image
5. docker build -t <your-dockerhub-username>/go-app:latest .
6. Push the image to Docker Hub
7. docker push <your-dockerhub-username>/go-app:latest
8. Docker Hub URL:
   `https://hub.docker.com/r/<your-dockerhub-username>/go-app`

3. Kustomize Manifest

Use Kustomize to define a flexible and reusable Kubernetes manifest for deploying the Go application.

Directory Structure:

```
kustomize/
├── base/
│   ├── deployment.yaml
│   ├── service.yaml
│   ├── kustomization.yaml
├── overlays/dev/
│   ├── kustomization.yaml
├── overlays/prod/
│   ├── kustomization.yaml
```

`base/deployment.yaml`

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: go-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: go-app
  template:
    metadata:
      labels:
        app: go-app
    spec:
      containers:
        - name: go-app
          image: <your-dockerhub-username>/go-app:latest
          ports:
            - containerPort: 8080
```

`base/service.yaml`

```yaml
apiVersion: v1
kind: Service
metadata:
  name: go-app-service
```

```
spec:
  selector:
    app: go-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

4. GKE Cluster Setup with Terraform

`main.tf` (Provision GKE Cluster)

```
hcl
provider "google" {
  project = var.project_id
  region  = var.region
}

resource "google_container_cluster" "primary" {
  name               = "gke-cluster"
  location           = var.region
  initial_node_count = 3

  node_config {
    machine_type = "e2-medium"
    oauth_scopes = [
      "https://www.googleapis.com/auth/cloud-platform"
    ]
  }
}
```

Run the following commands to initialize and apply Terraform:

```
bash
terraform init
terraform apply -var="project_id=<your-gcp-project-id>"
```

5. ArgoCD Deployment

ArgoCD Application Manifest:

Create an ArgoCD application manifest to deploy the Go application using GitOps.

```
yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: go-app
  namespace: argocd
spec:
  destination:
    namespace: default
    server: https://kubernetes.default.svc
  source:
    repoURL: "https://github.com/<your-repo>/kustomize-config"
    targetRevision: main
    path: overlays/prod
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

Deploy it with the following command:

```
bash
kubectl apply -f argocd-application.yaml
```

6. CI/CD Pipeline

GitHub Actions Workflow

Create a `.github/workflows/cicd-pipeline.yml` file to automate building and deploying the Go application.

```
yaml
        name: CI/CD Pipeline
```

```
on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Set up Go
      uses: actions/setup-go@v3
      with:
        go-version: 1.21

    - name: Build the application
      run: go build -o app

    - name: Login to Docker Hub
      run: echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{
secrets.DOCKER_USERNAME }}" --password-stdin

    - name: Build and push Docker image
      run: |
        docker build -t <your-dockerhub-username>/go-app:latest .
        docker push <your-dockerhub-username>/go-app:latest

  deploy:
    runs-on: ubuntu-latest
    needs: build

    steps:
    - name: Checkout repository
      uses: actions/checkout@v3

    - name: Update Kustomize Manifest
      run: |
        sed -i "s|newTag: .*|newTag: latest|g"
kustomize/overlays/prod/kustomization.yaml
        git config --global user.email "github-actions@github.com"
        git config --global user.name "GitHub Actions"
```

```
        git commit -am "Update image tag"
        git push

    - name: Sync ArgoCD
      run: |
        kubectl apply -f argocd-application.yaml
```

7. Deployment:

    1. Set up a GKE cluster using Terraform.
    2. Deploy the Go application with ArgoCD using GitOps.
    3. Automate the build and deployment process using GitHub Actions.