Complex Terraform Configuration with Helm and Istio for DigitalOcean Kubernetes

Grok

June 1, 2025

1 Introduction

This document provides Terraform configurations to provision a DigitalOcean Kubernetes (DOKS) cluster with VPC, Load Balancer, and Spaces for state storage. The three-service application (React front-end, Node.js/Express backend, MongoDB database) is deployed using a Helm chart adapted from existing Kubernetes manifests, with Istio for service mesh features (mTLS, traffic routing, observability with Prometheus and Grafana). Comments explain adaptability to requirement changes.

2 Prerequisites

- DigitalOcean account with a Personal Access Token (PAT).
- Terraform installed (`https://www.terraform.io/downloads.html`).
- Helm installed (`https://helm.sh/docs/intro/install/`).
- `kubectl` configured for Kubernetes access.
- Docker images pushed to a registry (e.g., Docker Hub).
- Existing Kubernetes manifests to adapt into a Helm chart.

3 Terraform Configuration

Provisioning DOKS cluster, VPC, Load Balancer, and Spaces.

Listing 1: main.tf

```
# Configure DigitalOcean provider
provider "digitalocean" {
  token = var.do_token
}

# Configure Terraform state storage in DigitalOcean Spaces
terraform {
  backend "s3" {
    endpoint = "https://nyc3.digitaloceanspaces.com" # Update for your region
```

```
    key = "terraform/state/terraform.tfstate"
    bucket = "app-state-bucket" # Update with your Spaces bucket name
    access_key = var.spaces_access_key
    secret_key = var.spaces_secret_key
    skip_credentials_validation = true
    skip_metadata_api_check = true
    skip_region_validation = true
  }
}

# Create VPC for network isolation
resource "digitalocean_vpc" "app_vpc" {
  name = "app-vpc" # Change for naming convention
  region = "nyc1" # Change to other regions (e.g., sfo3, lon1)
}

# Create DOKS cluster
resource "digitalocean_kubernetes_cluster" "app_cluster" {
  name = "app-cluster" # Change for naming convention
  region = "nyc1" # Align with VPC region
  version = "1.28" # Update to latest supported Kubernetes version
  vpc_uuid = digitalocean_vpc.app_vpc.id

  node_pool {
    name = "worker-pool" # Change for naming convention
    size = "s-4vcpu-8gb" # Adjust for more resources (e.g., s-8vcpu-16gb)
    node_count = 3 # Adjust for scaling needs
    auto_scale = true
    min_nodes = 2
    max_nodes = 5 # Adjust for autoscaling limits
  }
}

# Create DigitalOcean Load Balancer
resource "digitalocean_loadbalancer" "app_lb" {
  name = "app-lb" # Change for naming convention
  region = "nyc1" # Align with cluster region
  vpc_uuid = digitalocean_vpc.app_vpc.id
  droplet_tag = "k8s:${digitalocean_kubernetes_cluster.app_cluster.id}"

  forwarding_rule {
    entry_port = 80
    entry_protocol = "http"
    target_port = 80 # Align with Istio gateway port
    target_protocol = "http"
  }

  healthcheck {
    port = 80
    protocol = "http"
    path = "/" # Update if health check endpoint changes
```

```
  }
}

# Output kubeconfig
output "kubeconfig" {
  value = digitalocean_kubernetes_cluster.app_cluster.kube_config[0].
      raw_config
  sensitive = true
}

# Output load balancer IP
output "loadbalancer_ip" {
  value = digitalocean_loadbalancer.app_lb.ip
}
```

Listing 2: variables.tf

```
variable "do_token" {
  description = "DigitalOcean API token"
  type = string
  sensitive = true
}

variable "spaces_access_key" {
  description = "DigitalOcean Spaces access key"
  type = string
  sensitive = true
}

variable "spaces_secret_key" {
  description = "DigitalOcean Spaces secret key"
  type = string
  sensitive = true
}
```

Listing 3: terraform.tfvars

```
# Update with your credentials
do_token = "your_digitalocean_token"
spaces_access_key = "your_spaces_access_key"
spaces_secret_key = "your_spaces_secret_key"
```

4 Helm Chart with Adapted Manifests

The existing manifests are adapted into a Helm chart for parameterized deployment.

Listing 4: Chart Directory Structure

```
mkdir -p my-app/templates
touch my-app/Chart.yaml my-app/values.yaml
touch my-app/templates/secret.yaml my-app/templates/frontend-deployment.yaml
```

```
touch my-app/templates/backend-deployment.yaml my-app/templates/database-deployment.yaml
touch my-app/templates/gateway.yaml my-app/templates/virtualservice.yaml
```

Listing 5: my-app/Chart.yaml

```
apiVersion: v2
name: my-app
description: Helm chart for three-service application
type: application
version: 0.1.0
appVersion: "1.0"
```

Listing 6: my-app/values.yaml

```
frontend:
  image: your_dockerhub_username/frontend:latest # Update with your Docker Hub image
  replicas: 2 # Adjust for scaling
  port: 3000 # Change if frontend framework changes (e.g., Angular: 4200)
backend:
  image: your_dockerhub_username/backend:latest # Update with your Docker Hub image
  replicas: 2 # Adjust for scaling
  port: 5000 # Change if backend framework changes (e.g., FastAPI: 8000)
mongodb:
  image: mongo:7.0 # Change to different DB if needed
  replicas: 1 # Typically single replica for MongoDB
  port: 27017
  storage: 10Gi # Adjust storage size
  username: mongo_user # Update for security
  password: mongo_password # Update for security
istio:
  gateway:
    host: app.example.com # Update with your domain or LoadBalancer IP
```

Listing 7: my-app/templates/secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-credentials
type: Opaque
data:
  mongo-user: {{ .Values.mongodb.username | b64enc }} # Update if credentials change
  mongo-password: {{ .Values.mongodb.password | b64enc }}
```

Listing 8: my-app/templates/frontend-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: frontend
  labels:
    app: frontend
spec:
  replicas: {{ .Values.frontend.replicas }}
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: {{ .Values.frontend.image }}
        ports:
        - containerPort: {{ .Values.frontend.port }}
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
  - port: 80
    targetPort: {{ .Values.frontend.port }}
  type: ClusterIP
```

Listing 9: my-app/templates/backend-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
  labels:
    app: backend
spec:
  replicas: {{ .Values.backend.replicas }}
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: backend
        image: {{ .Values.backend.image }}
```

```
        ports:
        - containerPort: {{ .Values.backend.port }}
        env:
        - name: MONGO_URI
          value: "mongodb://{{ .Values.mongodb.username }}:{{ .Values.mongodb.
              password }}@mongodb-service:{{ .Values.mongodb.port }}/mydb"
---
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend
  ports:
  - port: 80
    targetPort: {{ .Values.backend.port }}
  type: ClusterIP
```

Listing 10: my-app/templates/database-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb
  labels:
    app: mongodb
spec:
  replicas: {{ .Values.mongodb.replicas }}
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb
        image: {{ .Values.mongodb.image }}
        ports:
        - containerPort: {{ .Values.mongodb.port }}
        env:
        - name: MONGO_INITDB_ROOT_USERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-credentials
              key: mongo-user
        - name: MONGO_INITDB_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-credentials
```

```
            key: mongo-password
        volumeMounts:
        - name: mongo-data
          mountPath: /data/db
      volumes:
      - name: mongo-data
        persistentVolumeClaim:
          claimName: mongo-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  ports:
  - port: {{ .Values.mongodb.port }}
    targetPort: {{ .Values.mongodb.port }}
  type: ClusterIP
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: {{ .Values.mongodb.storage }}
```

Listing 11: my-app/templates/gateway.yaml

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: app-gateway
spec:
  selector:
    istio: ingressgateway # Change if gateway name differs
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "{{ .Values.istio.gateway.host }}" # Update with your domain or
        LoadBalancer IP
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
```

```
metadata:
  name: app-virtualservice
spec:
  hosts:
  - "{{ .Values.istio.gateway.host }}"
  gateways:
  - app-gateway
  http:
  - route:
    - destination:
        host: frontend-service
        port:
          number: 80
```

5 How to Run

1. Save Terraform files (`main.tf`, `variables.tf`, `terraform.tfvars`) in a directory.
2. Create Helm chart directory (`my-app/`) and save the adapted manifests.
3. Initialize Terraform: `terraform init`.
4. Apply Terraform configuration: `terraform apply`.
5. Save kubeconfig: `terraform output -raw kubeconfig > kubeconfig.yaml`.
6. Configure `kubectl`: `export KUBECONFIG=./kubeconfig.yaml`.
7. Install Istio:

```
helm repo add istio https://istio-release.storage.googleapis.com/charts
helm repo update
kubectl create namespace istio-system
helm install istio-base istio/base -n istio-system --wait
helm install istiod istio/istiod -n istio-system --wait --set profile=
    demo
helm install istio-ingress istio/gateway -n istio-system --wait
kubectl label namespace default istio-injection=enabled
```

8. Install observability addons:

```
helm repo add prometheus-community https://prometheus-community.github.io
    /helm-charts
helm repo update
helm install prometheus prometheus-community/prometheus -n istio-system
    --wait
helm install grafana grafana/grafana -n istio-system --wait
```

9. Deploy application: `helm install my-app ./my-app -n default`.
10. Get LoadBalancer IP: `terraform output loadbalancer`$_{ip}$.*Accesstheapplicationat*`http://<Loac`
11. Monitor via Grafana: `kubectl port-forward svc/grafana -n istio-system 3000:3000`, then visit `http://localhost:3000`.

12. Destroy resources: `terraform destroy; helm uninstall my-app -n default; helm uninstall istio-base istiod istio-ingress prometheus grafana -n istio-syste`

6 Adapting to Requirement Changes

- **Change Region or Node Size**: Update `region` or `size` in `main.tf`.
- **Change Frontend Framework**: Update `values.yaml` and `frontend-deployment.yaml` for image and port.
- **Change Backend Framework**: Update `values.yaml` and `backend-deployment.yaml` for image and port.
- **Change Database**: Update `values.yaml` and `database-deployment.yaml` for image and volume paths.

- **Scaling**: Adjust $\texttt{node}_{c}ountor\texttt{min}_{n}odes/max_{n}odesin\texttt{main.tf}, or\texttt{replicas}in\texttt{values.yaml}.\textbf{\texttt{Change Doma}}$