Simple Terraform Configuration with Existing Kubernetes Manifests for DigitalOcean

Grok

June 1, 2025

1 Introduction

This document provides Terraform configurations to provision a DigitalOcean Kubernetes (DOKS) cluster and deploy a three-service application (React front-end, Node.js/Express backend, MongoDB database) by reusing existing Kubernetes manifests (`secret.yaml`, `frontend-deployment.yaml`, `backend-deployment.yaml`, `database-deployment.yaml`) with `kubectl`. The setup is minimal, suitable for development or small-scale deployments. Comments explain adaptability to requirement changes.

2 Prerequisites

- DigitalOcean account with a Personal Access Token (PAT).
- Terraform installed (`https://www.terraform.io/downloads.html`).
- `kubectl` configured for Kubernetes access.
- Docker images for front-end and backend pushed to a registry (e.g., Docker Hub).
- Existing Kubernetes manifests: `secret.yaml`, `frontend-deployment.yaml`, `backend-deploymen` `database-deployment.yaml`.

3 Terraform Configuration

Defining the DOKS cluster.

Listing 1: main.tf

```
# Configure DigitalOcean provider; update if using another cloud provider
provider "digitalocean" {
  token = var.do_token # Store token in a secure variable or environment
      variable
}

# Define DOKS cluster
resource "digitalocean_kubernetes_cluster" "app_cluster" {
  name = "app-cluster" # Change for different naming convention
```

```
  region = "nyc1" # Change to other DigitalOcean regions (e.g., sfo3, lon1)
  version = "1.28" # Update to latest supported Kubernetes version

  node_pool {
    name = "worker-pool" # Change for naming convention
    size = "s-2vcpu-4gb" # Adjust droplet size (e.g., s-4vcpu-8gb for more resources)
    node_count = 2 # Adjust for scaling needs
  }
}

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

Listing 2: variables.tf

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

Listing 3: terraform.tfvars

```
# Update with your DigitalOcean PAT
do_token = "your_digitalocean_token"
```

4 Reused Kubernetes Manifests

The following manifests are reused from the previous simple Kubernetes setup.

Listing 4: secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-credentials # Change if naming convention changes
type: Opaque
data:
  mongo-user: bW9uZ29fdXNlcg== # Base64-encoded 'mongo_user'; update if credentials change
  mongo-password: bW9uZ29fcGFzc3dvcmQ= # Base64-encoded 'mongo_password'; update if credentials change
```

Listing 5: frontend-deployment.yaml

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

```
metadata:
  name: frontend # Change if naming convention changes
spec:
  replicas: 2 # Adjust for scaling needs
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend # Change if label changes
    spec:
      containers:
      - name: frontend
        image: your_dockerhub_username/frontend:latest # Update with your Docker Hub image
        ports:
        - containerPort: 3000 # Change if frontend framework uses different port (e.g., Angular: 4200)
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service # Change if naming convention changes
spec:
  selector:
    app: frontend
  ports:
  - port: 80
    targetPort: 3000 # Align with containerPort
  type: LoadBalancer # Change to ClusterIP or NodePort for different access needs
```

Listing 6: backend-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend # Change if naming convention changes
spec:
  replicas: 2 # Adjust for scaling needs
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend # Change if label changes
    spec:
      containers:
      - name: backend
        image: your_dockerhub_username/backend:latest # Update with your Docker
```

```
            Hub image
        ports:
        - containerPort: 5000 # Change if backend framework uses different port
            (e.g., FastAPI: 8000)
        env:
        - name: MONGO_URI
          value: "mongodb://mongo_user:mongo_password@mongodb-service:27017/
              mydb" # Update if credentials or DB name change
---
apiVersion: v1
kind: Service
metadata:
  name: backend-service # Change if naming convention changes
spec:
  selector:
    app: backend
  ports:
  - port: 80
    targetPort: 5000 # Align with containerPort
  type: ClusterIP # Change to LoadBalancer if external access needed
```

Listing 7: database-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb # Change if naming convention changes
spec:
  replicas: 1 # Adjust for scaling (MongoDB typically single replica)
  selector behoefte:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb # Change if label changes
    spec:
      containers:
      - name: mongodb
        image: mongo:7.0 # Change to different DB (e.g., postgres:16) if needed
        ports:
        - containerPort: 27017 # Change if DB uses different port
        env:
        - name: MONGO_INITDB_ROOT_USERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-credentials # Align with Secret name
              key: mongo-user
        - name: MONGO_INITDB_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-credentials
```

```
            key: mongo-password
        volumeMounts:
        - name: mongo-data
          mountPath: /data/db # Change if DB uses different data path
      volumes:
      - name: mongo-data
        persistentVolumeClaim:
          claimName: mongo-pvc # Align with PVC name
---
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service # Change if naming convention changes
spec:
  selector:
    app: mongodb
  ports:
  - port: 27017
    targetPort: 27017 # Align with containerPort
  type: ClusterIP # Change if external access needed
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc # Change if naming convention changes
spec:
  accessModes:
    - ReadWriteOnce # Adjust based on storage needs
  resources:
    requests:
      storage: 10Gi # Adjust size as needed
```

5 How to Run

1. Save Terraform files (`main.tf`, `variables.tf`, `terraform.tfvars`) in a directory.
2. Save the existing Kubernetes manifests (`secret.yaml`, `frontend-deployment.yaml`, `backend-deployment.yaml`, `database-deployment.yaml`).
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. Apply manifests: `kubectl apply -f secret.yaml -f frontend-deployment.yaml -f backend-deployment.yaml -f database-deployment.yaml`.
8. Verify pods: `kubectl get pods`.
9. Get LoadBalancer IP: `kubectl get svc frontend-service`.

10. Access the application at `http://<LoadBalancer-IP>`.
11. Destroy resources: `terraform destroy`.

6 Adapting to Requirement Changes

- **Change Region or Node Size**: Update `region` or `size` in `main.tf`.
- **Change Frontend Framework**: Update `frontend-deployment.yaml` image and port (e.g., Angular uses port 4200).
- **Change Backend Framework**: Update `backend-deployment.yaml` image and port (e.g., FastAPI uses port 8000).
- **Change Database**: Update `database-deployment.yaml` image and volume paths (e.g., `postgres:16`).

- **Scaling**: Adjust `node`$_{c}ountin$`main.tf`or`replicas`inmanifests`.**`External Access`** : $Change$`Service`typ