Deploying an EKS Cluster with Terraform and Deploying a BlogApp using custom Helm and Nginx Ingress Controller

1. Create a Terraform `main.tf` File for Creating EKS Cluster

```
provider "aws" {
  region = "ap-south-1"
}

# Fetch default VPC
data "aws_vpc" "default" {
  default = true
}

# Fetch default subnets
data "aws_subnets" "default" {
  filter {
    name   = "vpc-id"
    values = [data.aws_vpc.default.id]
  }
}

# Security Group for EKS
resource "aws_security_group" "eks_sg" {
  vpc_id = data.aws_vpc.default.id

  ingress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "eks-security-group"
  }
}

# IAM Role for EKS Cluster
resource "aws_iam_role" "eks_role" {
  name = "eks-cluster-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Action = "sts:AssumeRole"
      Effect = "Allow"
      Principal = {
        Service = "eks.amazonaws.com"
      }
    }]
  })
}

# Attach necessary IAM policies to EKS Role
resource "aws_iam_role_policy_attachment" "eks_policy_attach" {
```

```
  role       = aws_iam_role.eks_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
}

# Create EKS Cluster
resource "aws_eks_cluster" "eks_cluster" {
  name     = "my-eks-cluster"
  role_arn = aws_iam_role.eks_role.arn

  vpc_config {
    subnet_ids         = data.aws_subnets.default.ids
    security_group_ids = [aws_security_group.eks_sg.id]
  }
}

# Output EKS Cluster Details
output "cluster_name" {
  value = aws_eks_cluster.eks_cluster.name
}
```

Run the following Terraform commands:

```
terraform init
terraform validate
terraform plan
terraform apply -auto-approve
```

Configure Kubernetes Access:

```
aws eks update-kubeconfig --region ap-south-1 --name my-eks-cluster
```

Verify Cluster Details:

```
kubectl get nodes -A
kubectl get all -A
```

2. Create a Custom Helm Chart for Deploying BlogApp

Create Helm Chart:

```
helm create blogapp-chart
```

Directory Structure:

```
blogapp-chart/
  ├── templates/
  │   ├── deployment.yaml
  │   ├── service.yaml
  ├── values.yaml
  ├── Chart.yaml
```

Define Custom Deployment (`templates/deployment.yaml`):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}
  namespace: {{ .Values.namespace }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: {{ .Values.appName }}
  template:
    metadata:
      labels:
        app: {{ .Values.appName }}
    spec:
      containers:
      - name: {{ .Values.appName }}
        image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
        ports:
        - containerPort: {{ .Values.service.port }}
```

Define Service (`templates/service.yaml`):

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-svc
  namespace: {{ .Values.namespace }}
spec:
  selector:
    app: {{ .Values.appName }}
  ports:
    - protocol: TCP
      port: {{ .Values.service.port }}
      targetPort: {{ .Values.service.targetPort }}
```

Define Values (`values.yaml`):

```
appName: blogapp
namespace: webapps
replicaCount: 1

image:
  repository: <your-repository>
  tag: "latest"

service:
  port: 8080
  targetPort: 8080
```

Install the Helm Chart:

```
helm install blogapp ./blogapp-chart -n webapps
kubectl get all -n webapps
```

3. Expose BlogApp Externally using Nginx Ingress Controller

Install Nginx Ingress Controller:

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update
helm install nginx-ingress ingress-nginx/ingress-nginx --namespace ingress-nginx --create-namespace
```

Check Ingress Controller Status:

```
kubectl get all -n ingress-nginx
```

Create an Ingress Resource (`templates/ingress.yaml`):

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: blogapp-ingress
  namespace: webapps
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: test.example.com
    http:
      paths:
      - path: /blog
        pathType: Prefix
        backend:
          service:
            name: blogapp-svc
            port:
              number: 8080
```

Apply Ingress Configuration:

```
kubectl apply -f ingress.yaml
```

Verify Ingress:

```
kubectl get ingress -n webapps
```

Get External IP Address:

```
nslookup <aws-eks-endpoint>
```

Test Locally:

Edit `/etc/hosts` (Linux/macOS) or `C:\Windows\System32\drivers\etc\hosts` (Windows):

```
<INGRESS-EXTERNAL-IP> test.example.com
```

Now your **BlogApp** is accessible externally!