**Title:** Deploying a Django Application on Amazon EKS with ALB Ingress and HPA

## 1. Overview

This document outlines the setup and deployment of a Django 5.2.4 application on **Amazon Elastic Kubernetes Service (EKS)** using:

* AWS Load Balancer Controller (ALB Ingress)
* Horizontal Pod Autoscaler (HPA)
* ConfigMap and Secrets for environment configuration
* Resource requests and limits for pods

## 2. Architecture Diagram

┌──────────▼───────────┐  
 | ALB Ingress | (AWS ALB)  
 └──────────┬───────────┘  
 │  
 ┌────────▼────────┐  
 | EKS Cluster | (Assessment-cluster)  
 └────────┬────────┘  
 ┌────────────┬───┴───────┬────────────┐  
 ▼ ▼ ▼ ▼  
 Deployment Service HPA ConfigMap/Secrets  
 │ │   
 Django App NodePort Auto-scales Env Injection  
 Pod to ALB Pods

## 3. Setup Instructions

### ✅ Prerequisites

* AWS CLI and eksctl installed
* IAM permissions
* Helm installed
* EKS cluster running (Assessment-cluster)

### 🔧 Step-by-Step Setup

#### Step 1: Associate IAM OIDC Provider

eksctl utils associate-iam-oidc-provider --cluster Assessment-cluster --approve

#### Step 2: Create IAM Policy

curl -o iam-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/main/docs/install/iam\_policy.json  
aws iam create-policy \  
 --policy-name AWSLoadBalancerControllerIAMPolicy \  
 --policy-document file://iam-policy.json

#### Step 3: Create IAM ServiceAccount

eksctl create iamserviceaccount \  
 --cluster Assessment-cluster \  
 --region ap-south-1 \  
 --namespace kube-system \  
 --name aws-load-balancer-controller \  
 --attach-policy-arn arn:aws:iam::<ACCOUNT\_ID>:policy/AWSLoadBalancerControllerIAMPolicy \  
 --approve

#### Step 4: Install Load Balancer Controller via Helm

helm repo add eks https://aws.github.io/eks-charts  
helm repo update  
  
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \  
 -n kube-system \  
 --set clusterName=Assessment-cluster \  
 --set serviceAccount.create=false \  
 --set serviceAccount.name=aws-load-balancer-controller \  
 --set region=ap-south-1 \  
 --set vpcId=vpc-0b471a8b999e1793d

## 4. Kubernetes Manifests

All files are placed under /k8s folder.

### deployment.yaml

* Includes resource requests & limits
* Liveness and readiness probes

### service.yaml

* Type: NodePort
* Annotations for ALB

### configmap.yaml

* Includes environment variables like DEBUG, ALLOWED\_HOSTS

### secrets.yaml

* Holds sensitive vars like DJANGO\_SECRET\_KEY

### ingress.yaml

* Uses ALB Ingress with alb ingressClassName
* Exposes service publicly

### hpa.yaml

* Sets min/max pods
* CPU-based autoscaling (target 60%)

## 5. Deployment Process

kubectl apply -f k8s/namespace.yaml  
kubectl apply -f k8s/configmap.yaml  
kubectl apply -f k8s/secrets.yaml  
kubectl apply -f k8s/deployment.yaml  
kubectl apply -f k8s/service.yaml  
kubectl apply -f k8s/hpa.yaml  
kubectl apply -f k8s/ingress.yaml

Get ALB DNS:

kubectl get ingress -n prod

## 6. Result

* Django app accessible via ALB:

http://k8s-prod-<auto-generated>.ap-south-1.elb.amazonaws.com

* HPA working: auto-scales when CPU > 60%
* Service stable, domain can be pointed via Route53

## 7. Optional Next Steps

* Setup HTTPS with ACM
* Use RDS PostgreSQL instead of SQLite
* Add GitHub Actions CI/CD pipeline