# Prerequisites

Before we start, make sure you have the following:

* **Docker Knowledge**: Familiarity with creating Dockerfiles and managing images.
* **Kubernetes Basics**: Understanding of pods, services, and deployments.
* **AWS Account**: Required for setting up EKS.
* **Command Line Tools**: Ensure Docker, kubectl, eksctl, helm, and AWS CLI are installed and configured.

**CREATE A CLUSTER**

eksctl create cluster \

-n mern-stack \

--nodegroup-name mern-nodegroup \

--region eu-west-1 \

--node-type t3.medium \

--nodes 2 \

--with-oidc \

**This will create:**

* 1 VPC
* Internet gateway
* 4 Subnets (2 public and 2 private)
* Elastic IP address
* Route tables
* NAT Gateway
* Security Groups
* EKS Cluster
* Node group with 2 nodes
* 2 EC2 instances

**install the AWS Load Balancer Controller**

After the cluster is created, install the AWS Load Balancer Controller using Helm. This is required to use the ALB ingress controller mentioned in your ingress manifest.

Here's how to install the AWS Load Balancer Controller:

1. First, create an IAM policy for the controller:

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

1. Create an IAM role and service account for the controller:

eksctl create iamserviceaccount \

--cluster=mern-stack-cluster \

--namespace=kube-system \

--name=aws-load-balancer-controller \

--attach-policy-arn=arn:aws:iam::<aws acc id>:policy/AWSLoadBalancerControllerIAMPolicy \

--approve

Replace <AWS\_ACCOUNT\_ID> with your actual AWS account ID.

1. Install the AWS Load Balancer Controller using Helm:

# Add the Helm repository

helm repo add eks https://aws.github.io/eks-charts

helm repo update

# Install the controller

helm install aws-load-balancer-controller eks/aws-load-balancer-controller \

-n kube-system \

--set clusterName=mern-stack-cluster \

--set serviceAccount.create=false \

--set serviceAccount.name=aws-load-balancer-controller

After completing these steps, the AWS Load Balancer Controller will be running in your cluster and will watch for Ingress resources with the appropriate annotations. When you deploy your application's Ingress resource, the controller will automatically provision an ALB for you.

create the Route 53 record for your domain to point to the AWS Load Balancer.

 Open AWS Route 53 Console

 Navigate to your hosted zone for cloudspace-consulting.com

 Create a new Record:

* Record type: A Record (Alias)
* Route traffic to: Alias to Application Load Balancer
* Choose your region (eu-west-1)
* Select the Load Balancer: k8s-mern-mernalbd-dedc86e6f3-1055444736.eu-west-1.elb.amazonaws.com

Put image here

**To be run on: PROJETS-AWS-EKS/EKS-MERN-Project/DevOps/docker**

**Database Dockerfile**

**Commands:**

docker build -t <your\_dockerhub\_username>/mysql-image:<tag> -f Dockerfile-Database .

docker push <your\_dockerhub\_username>/mysql-image:<tag>

# Backend Dockerfile

docker build -t <your\_dockerhub\_username>/backend-image:<tag> -f Dockerfile-Backend .

docker push <your\_dockerhub\_username>/backend-image:<tag>

# Frontend Dockerfile

docker build -t <your\_dockerhub\_username>/frontend-image:<tag> -f Dockerfile-Backend .

docker push <your\_dockerhub\_username>/frontend-image:<tag>

**TO BE RUN ON MAIN DIRECTORY: EKS-MERN-PROJECT**

First deploy the namespace (since other resources depend on it):

kubectl apply -f DevOps/Kubernetes-Manifests/Database/namespace.yaml

Then deploy the Database components (PV, PVC, and secrets needed before deployments):

kubectl apply -f DevOps/Kubernetes-Manifests/Database/

Next deploy the Backend:

kubectl apply -f DevOps/Kubernetes-Manifests/Backend/

Then deploy the Frontend:

kubectl apply -f DevOps/Kubernetes-Manifests/Frontend/

Finally, deploy the Ingress:

kubectl apply -f DevOps/Kubernetes-Manifests/ingress.yaml

check if the pods are running: kubectl get pods -n <namespace>

check if all three services are present in your Kubernetes cluster: kubectl get services --all-namespaces

check if Ingress is reconciled successfully in Kubernetes: kubectl get ingress -n <namespace>

run a command to check all the resources in our mern namespace: kubectl get all -n mern

**eksctl delete cluster -n** mern-stack-cluster

delete route53 record

delete load balancer???

**create the diagram with dockerhub and not ecr**