**I. Create a Dockerfile**

* **Action**: Define the environment and application dependencies in a Dockerfile.
* **Command**: Ensure the Dockerfile is properly structured with all necessary instructions (e.g., FROM, COPY, RUN, CMD).

**II. Build the Docker Image**

* **Action**: Build the Docker image locally.
* **Command**:

docker build -t <image-name>:<tag> <path-to-dockerfile>

* **Example**:

docker build -t my-app:latest .

**III. Create a Repository in ECR**

* **Action**: Create an AWS Elastic Container Registry (ECR) repository to store the Docker image.
* **Command**:

aws ecr create-repository --repository-name <your-repository-name> --region <your-region>

* **Example**:

aws ecr create-repository --repository-name my-app --region eu-west-1

**IV. Push the Image into the Repository**

* **Action**: Push the locally built Docker image to the ECR repository.
* **Steps**:
  1. **Authenticate Docker with ECR**:

aws ecr get-login-password --region <your-region> | docker login --username AWS --password-stdin <account-id>.dkr.ecr.<region>.amazonaws.com

* 1. **Tag the image for ECR**:

docker tag <image-name>:<tag> <account-id>.dkr.ecr.<region>.amazonaws.com/<your-repository-name>:<tag>

* 1. **Push the image**:

docker push <account-id>.dkr.ecr.<region>.amazonaws.com/<your-repository-name>:<tag>

# TopSurvey Application Deployment Guide

## Prerequisites

* AWS CLI configured
* Terraform installed
* Docker installed
* kubectl installed
* Helm installed
* AWS account access
* Sufficient permissions for:
  + IAM management
  + EKS cluster creation
  + ECR repository management

## 1. Initial Setup

### 1.1 Build Terraform S3 Backend

# Create S3 backend for remote state storage

# Note: This step typically involves creating an S3 bucket and configuring backend in Terraform

## 2. Infrastructure Deployment

### 2.1 Deploy Terraform Infrastructure

# Navigate to your Terraform directory

terraform init

terraform plan

terraform apply

**Infrastructure Components Created:**

* VPC
* EKS Cluster
* RDS Database
* CloudWatch Log Groups
* SNS Topics
* IAM Roles

## 3. Docker Setup

### 3.1 Create Dockerfile

# Ensure Dockerfile is prepared for backend application

### 3.2 Build Docker Image

docker build -t topsurvey-backend:v1 .

### 3.3 Create ECR Repository

aws ecr create-repository --repository-name top-surveys --region eu-west-1

### 3.4 Authenticate Docker with ECR

aws ecr get-login-password --region eu-west-1 | docker login --username AWS --password-stdin 381491868231.dkr.ecr.eu-west-1.amazonaws.com

### 3.5 Tag and Push Docker Image

docker tag topsurvey-backend:v1 381491868231.dkr.ecr.eu-west-1.amazonaws.com/top-surveys:v1

docker push 381491868231.dkr.ecr.eu-west-1.amazonaws.com/top-surveys:v1

## 4. Kubernetes Setup

### 4.1 Configure kubectl (main directory)

aws eks update-kubeconfig --name topsurvey-dev-eks-cluster --region eu-west-1

### 4.2 Verify EKS Cluster

kubectl get nodes

## 5. Kubernetes Add-ons Installation

1. Install Helm Repositories

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

helm repo add external-secrets https://charts.external-secrets.io

helm repo update

1. Install AWS Load Balancer Controller

kubectl apply -f alb-controller.yaml

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

--namespace kube-system \

--set clusterName=topsurvey-dev-eks-cluster \

--set serviceAccount.create=false \

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

1. Install External Secrets Operator

helm install external-secrets external-secrets/external-secrets \

--namespace external-secrets \

--create-namespace \

--set webhook.create=true

1. Apply Kubernetes Manifests in Order

kubectl apply -f namespace.yaml

kubectl apply -f service-account.yaml

kubectl apply -f secretstore.yaml

kubectl apply -f secrets.yaml

kubectl apply -f alb-controller-token.yaml

# Deploy application

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

kubectl apply -f ingress.yaml

1. Verify Deployments

# Verify secrets

kubectl get secrets -n surveys

# Check deployments

kubectl get deployments -n surveys

### 6.2 Verify Ingress

kubectl get ingress -n surveys

kubectl describe ingress surveys-alb-dev -n surveys

## 7. Monitoring Setup

### 7.1 Install CloudWatch Container Insights

cd monitoring/

kubectl apply -f amazon-cloudwatch-namespace.yaml

kubectl apply -f cluster-info-configmap.yaml

chmod +x install-container-insights.sh

./install-container-insights.sh

### 7.2 Verify Monitoring

kubectl get pods -n amazon-cloudwatch

## 8. DNS and Frontend Setup

### 8.1 Get ALB DNS Name

kubectl get ingress -n surveys -o jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}'

### 8.2 Update Route53 Record

* Manually update the Route53 record for api.cloudspace-consulting.com with the ALB DNS name

### 8.3 Build and Deploy Frontend

cd frontend

npm install

npm run build

# Sync build files to S3

aws s3 sync build/ s3://topsurvey-dev-topsurvey-st/

## Troubleshooting

* Ensure all IAM roles and policies have correct permissions
* Check network security groups and VPC configurations
* Verify AWS CLI and kubectl configurations
* Review Terraform state and resource dependencies

## Post-Deployment Checks

1. Verify all Kubernetes resources are running
2. Check application logs
3. Test application endpoints
4. Monitor CloudWatch for any issues

This workflow now properly accounts for your existing manifest files and follows a logical sequence to ensure all dependencies are in place before dependent resources are created.

Update ingress file with the alb certificate

Manifests files: backend > deployment = change image with the ecr uri

Explicar que uso ssm

Your manifest files serve distinct purposes:

1. **Core application resources**:
   * deployment.yaml - Your application pods
   * service.yaml - Your application network service
   * ingress.yaml - External access to your application
2. **Infrastructure resources**:
   * namespace.yaml - Defines the isolated environment
   * alb-controller.yaml - ServiceAccount for ALB controller
   * service-account.yaml - ServiceAccount for External Secrets
3. **Secret management resources**:
   * secretstore.yaml - Configures connection to AWS Secrets Manager
   * secrets.yaml - Defines which external secrets to fetch
4. Manually add the permissions for testing:

bash

Copy

aws iam put-role-policy --role-name aws-load-balancer-controller --policy-name manual-elb-permissions --policy-document '{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"elasticloadbalancing:DescribeListenerAttributes",

"elasticloadbalancing:DescribeLoadBalancerAttributes",

"elasticloadbalancing:ModifyListener",

"elasticloadbalancing:ModifyLoadBalancerAttributes"

],

"Resource": "\*"

}

]

}'

1. Check if the Terraform apply completed successfully:

bash

Copy

terraform state show aws\_iam\_policy.load\_balancer\_controller\_additional

terraform state show aws\_iam\_role\_policy\_attachment.load\_balancer\_controller\_additional

1. Restart the controller again after making sure the policy is attached:

bash

Copy

kubectl rollout restart deployment -n kube-system aws-load-balancer-controller

Continue monitoring the ingress creation:

bash

Copy

kubectl get ingress -n surveys -w