One region code and pipeline code  
**infra code**

AWSTemplateFormatVersion: '2010-09-09'

Description: EKS Infrastructure with VPC, Subnets, NAT Gateway, RDS, and DynamoDB

Parameters:

  VpcCidr:

    Type: String

    Default: 10.0.0.0/16

  PublicSubnet1Cidr:

    Type: String

    Default: 10.0.1.0/24

  PublicSubnet2Cidr:

    Type: String

    Default: 10.0.2.0/24

  PrivateSubnet1Cidr:

    Type: String

    Default: 10.0.3.0/24

  PrivateSubnet2Cidr:

    Type: String

    Default: 10.0.4.0/24

  EKSClusterName:

    Type: String

    Default: ecommerce-cluster

  EKSDesiredSize:

    Type: Number

    Default: 2

  EKSMaxSize:

    Type: Number

    Default: 3

  EKSMinSize:

    Type: Number

    Default: 1

  DBUsername:

    Type: String

    Default: admin

  DBPassword:

    Type: String

    NoEcho: true

    Default: admin1234

Resources:

  EKSClusterRole:

    Type: AWS::IAM::Role

    Properties:

      AssumeRolePolicyDocument:

        Version: '2012-10-17'

        Statement:

          - Effect: Allow

            Principal:

              Service: eks.amazonaws.com

            Action: sts:AssumeRole

      ManagedPolicyArns:

        - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy

        - arn:aws:iam::aws:policy/AmazonEKSVPCResourceController

  EKSNodeInstanceRole:

    Type: AWS::IAM::Role

    Properties:

      AssumeRolePolicyDocument:

        Version: '2012-10-17'

        Statement:

          - Effect: Allow

            Principal:

              Service: ec2.amazonaws.com

            Action: sts:AssumeRole

      ManagedPolicyArns:

        - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy

        - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly

        - arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy

  MyVPC:

    Type: AWS::EC2::VPC

    Properties:

      CidrBlock: !Ref VpcCidr

      EnableDnsSupport: true

      EnableDnsHostnames: true

  InternetGateway:

    Type: AWS::EC2::InternetGateway

  AttachGateway:

    Type: AWS::EC2::VPCGatewayAttachment

    Properties:

      VpcId: !Ref MyVPC

      InternetGatewayId: !Ref InternetGateway

  PublicSubnet1:

    Type: AWS::EC2::Subnet

    Properties:

      VpcId: !Ref MyVPC

      CidrBlock: !Ref PublicSubnet1Cidr

      AvailabilityZone: !Select [0, !GetAZs '']

      MapPublicIpOnLaunch: true

  PublicSubnet2:

    Type: AWS::EC2::Subnet

    Properties:

      VpcId: !Ref MyVPC

      CidrBlock: !Ref PublicSubnet2Cidr

      AvailabilityZone: !Select [1, !GetAZs '']

      MapPublicIpOnLaunch: true

  PrivateSubnet1:

    Type: AWS::EC2::Subnet

    Properties:

      VpcId: !Ref MyVPC

      CidrBlock: !Ref PrivateSubnet1Cidr

      AvailabilityZone: !Select [0, !GetAZs '']

  PrivateSubnet2:

    Type: AWS::EC2::Subnet

    Properties:

      VpcId: !Ref MyVPC

      CidrBlock: !Ref PrivateSubnet2Cidr

      AvailabilityZone: !Select [1, !GetAZs '']

  NatGatewayEIP:

    Type: AWS::EC2::EIP

    Properties:

      Domain: vpc

  NatGateway:

    Type: AWS::EC2::NatGateway

    Properties:

      AllocationId: !GetAtt NatGatewayEIP.AllocationId

      SubnetId: !Ref PublicSubnet1

  PublicRouteTable:

    Type: AWS::EC2::RouteTable

    Properties:

      VpcId: !Ref MyVPC

  PublicRoute:

    Type: AWS::EC2::Route

    Properties:

      RouteTableId: !Ref PublicRouteTable

      GatewayId: !Ref InternetGateway

      DestinationCidrBlock: 0.0.0.0/0

  PublicSubnet1RouteTableAssociation:

    Type: AWS::EC2::SubnetRouteTableAssociation

    Properties:

      SubnetId: !Ref PublicSubnet1

      RouteTableId: !Ref PublicRouteTable

  PublicSubnet2RouteTableAssociation:

    Type: AWS::EC2::SubnetRouteTableAssociation

    Properties:

      SubnetId: !Ref PublicSubnet2

      RouteTableId: !Ref PublicRouteTable

  PrivateRouteTable:

    Type: AWS::EC2::RouteTable

    Properties:

      VpcId: !Ref MyVPC

  PrivateRoute:

    Type: AWS::EC2::Route

    Properties:

      RouteTableId: !Ref PrivateRouteTable

      NatGatewayId: !Ref NatGateway

      DestinationCidrBlock: 0.0.0.0/0

  PrivateSubnet1RouteTableAssociation:

    Type: AWS::EC2::SubnetRouteTableAssociation

    Properties:

      SubnetId: !Ref PrivateSubnet1

      RouteTableId: !Ref PrivateRouteTable

  PrivateSubnet2RouteTableAssociation:

    Type: AWS::EC2::SubnetRouteTableAssociation

    Properties:

      SubnetId: !Ref PrivateSubnet2

      RouteTableId: !Ref PrivateRouteTable

  SecurityGroupEKSNodes:

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupDescription: Security group for EKS nodes

      VpcId: !Ref MyVPC

      SecurityGroupIngress:

        - IpProtocol: -1

          CidrIp: 0.0.0.0/0

  RdsSecurityGroup:

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupDescription: Allow MySQL access from EKS

      VpcId: !Ref MyVPC

      SecurityGroupIngress:

        - IpProtocol: tcp

          FromPort: 3306

          ToPort: 3306

          CidrIp: 10.0.0.0/16

  DBSubnetGroup:

    Type: AWS::RDS::DBSubnetGroup

    Properties:

      DBSubnetGroupDescription: ecommerce-db-subnet-group

      SubnetIds:

        - !Ref PrivateSubnet1

        - !Ref PrivateSubnet2

  RDSInstance:

    Type: AWS::RDS::DBInstance

    Properties:

      DBInstanceIdentifier: catalog-db

      DBInstanceClass: db.t3.micro

      Engine: mysql

      MasterUsername: !Ref DBUsername

      MasterUserPassword: !Ref DBPassword

      AllocatedStorage: 20

      DBSubnetGroupName: !Ref DBSubnetGroup

      VPCSecurityGroups:

        - !Ref RdsSecurityGroup

      PubliclyAccessible: false

      BackupRetentionPeriod: 1

      DBName: catalogdb

  EKSCluster:

    Type: AWS::EKS::Cluster

    Properties:

      Name: !Ref EKSClusterName

      RoleArn: !GetAtt EKSClusterRole.Arn

      ResourcesVpcConfig:

        SubnetIds:

          - !Ref PrivateSubnet1

          - !Ref PrivateSubnet2

        SecurityGroupIds:

          - !Ref SecurityGroupEKSNodes

  EKSNodeGroup:

    Type: AWS::EKS::Nodegroup

    Properties:

      ClusterName: !Ref EKSCluster

      NodeRole: !GetAtt EKSNodeInstanceRole.Arn

      Subnets:

        - !Ref PrivateSubnet1

        - !Ref PrivateSubnet2

      ScalingConfig:

        DesiredSize: !Ref EKSDesiredSize

        MaxSize: !Ref EKSMaxSize

        MinSize: !Ref EKSMinSize

      InstanceTypes:

        - t3.medium

      RemoteAccess:

        Ec2SshKey: bhargav1

      AmiType: AL2\_x86\_64

  CartDynamoDBTable:

    Type: AWS::DynamoDB::Table

    Properties:

      TableName: cart

      AttributeDefinitions:

        - AttributeName: id

          AttributeType: S

      KeySchema:

        - AttributeName: id

          KeyType: HASH

      ProvisionedThroughput:

        ReadCapacityUnits: 5

        WriteCapacityUnits: 5

Outputs:

  EKSClusterName:

    Description: EKS Cluster Name

    Value: !Ref EKSCluster

  RDSEndpoint:

    Description: RDS Endpoint

    Value: !GetAtt RDSInstance.Endpoint.Address

  DynamoDBTableName:

    Description: Cart DynamoDB Table

    Value: !Ref CartDynamoDBTable

Now first point go to iam -> user -> copy arn then got o cluster->access->add access  
then go to the terminal an type  
**aws eks update-kubeconfig --region us-east-1 --name MyEKSCluster**

Then kubectl get nodes -> kubectl get pods -> kubectl get svc

Then coming to pipeline first build the pipeline don’t forgot to add admin access role to the build role(it should be copied before)  
  
then in pipeline two changes should be done   
1)first give api and config  
2)copy arn of deployment and add to the access in the cluster

3)go to edit, edit deployment the replace source artifact  
  
**buildspec.yaml**

version: 0.2

phases:

install:

commands:

- echo Installing dependencies...

- export AWS\_SDK\_LOAD\_CONFIG=1

pre\_build:

commands:

- echo Logging in to Amazon ECR...

- ACCOUNT\_ID=423623849965

- REGION=us-east-1

- IMAGE\_TAG=v1.$CODEBUILD\_BUILD\_NUMBER

- aws ecr get-login-password --region $REGION | docker login --username AWS --password-stdin ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com

build:

commands:

- echo Building cart image...

- cd src/cart

- docker build -t cart:${IMAGE\_TAG} .

- docker tag cart:${IMAGE\_TAG} ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/cart:${IMAGE\_TAG}

- cd ../..

- echo Building catalog image...

- cd src/catalog

- docker build -t catalog:${IMAGE\_TAG} .

- docker tag catalog:${IMAGE\_TAG} ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/catalog:${IMAGE\_TAG}

- cd ../..

post\_build:

commands:

- echo Pushing cart image...

- docker push ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/cart:${IMAGE\_TAG}

- echo Pushing catalog image...

- docker push ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/catalog:${IMAGE\_TAG}

- echo Preparing Kubernetes deployment files...

- mkdir -p k8s-out

- sed "s|IMAGE\_PLACEHOLDER|${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/cart:${IMAGE\_TAG}|g" k8s/cart-deployment.yaml > k8s-out/cart-deployment.yaml

- cp k8s/cart-service.yaml k8s-out/ || echo "No cart-service.yaml found"

- sed "s|IMAGE\_PLACEHOLDER|${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/catalog:${IMAGE\_TAG}|g" k8s/catalog-deployment.yaml > k8s-out/catalog-deployment.yaml

- cp k8s/catalog-service.yaml k8s-out/ || echo "No catalog-service.yaml found"

- cp k8s/ui-deployment.yaml k8s-out/

- cp k8s/ui-service.yaml k8s-out/

artifacts:

base-directory: k8s-out

files:

- cart-deployment.yaml

- catalog-deployment.yaml

- ui-deployment.yaml

- ui-service.yaml

**cart-deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: cart

spec:

replicas: 2

selector:

matchLabels:

app: cart

template:

metadata:

labels:

app: cart

spec:

containers:

- name: cart

image: 423623849965.dkr.ecr.us-east-1.amazonaws.com/cart:v1.10

ports:

- containerPort: 8080

env:

- name: AWS\_REGION

value: us-east-1

- name: DYNAMODB\_TABLE

value: cart

---

apiVersion: v1

kind: Service

metadata:

name: cart-svc

spec:

selector:

app: cart

ports:

- port: 80

targetPort: 8080

type: LoadBalancer

**catalog-deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: catalog

spec:

replicas: 2

selector:

matchLabels:

app: catalog

template:

metadata:

labels:

app: catalog

spec:

containers:

- name: catalog

image: 423623849965.dkr.ecr.us-east-1.amazonaws.com/catalog:v1.10

ports:

- containerPort: 8080

env:

- name: DB\_HOST

value: catalog-db.cb6eiuesce5w.us-east-1.rds.amazonaws.com

- name: DB\_PORT

value: "3306"

- name: DB\_USER

value: admin

- name: DB\_PASSWORD

value: admin1234

- name: DB\_NAME

value: catalog-db

---

apiVersion: v1

kind: Service

metadata:

name: catalog-svc

spec:

selector:

app: catalog

ports:

- port: 80

targetPort: 8080

type: LoadBalancer

**ui-deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: store-ui

spec:

replicas: 1

selector:

matchLabels:

app: store-ui

template:

metadata:

labels:

app: store-ui

spec:

containers:

- name: store-ui

image: public.ecr.aws/aws-containers/retail-store-sample-ui:1.0.0

ports:

- containerPort: 8080

**ui-service.yaml**

# k8s/ui-service.yaml

apiVersion: v1

kind: Service

metadata:

name: store-ui-svc

spec:

type: LoadBalancer

selector:

app: store-ui

ports:

- port: 80

targetPort: 8080

version: 0.2

phases:

install:

commands:

- echo Installing dependencies...

- export AWS\_SDK\_LOAD\_CONFIG=1

pre\_build:

commands:

- echo Logging in to Amazon ECR...

- ACCOUNT\_ID=423623849965

- REGION=us-east-1 # Change this when running in a different region

- IMAGE\_TAG=v1.$CODEBUILD\_BUILD\_NUMBER

- export DB\_HOST=catalog-db.cb6eiuesce5w.$REGION.rds.amazonaws.com

- aws ecr get-login-password --region $REGION | docker login --username AWS --password-stdin ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com

build:

commands:

- echo Building cart image...

- cd src/cart

- docker build -t cart:${IMAGE\_TAG} .

- docker tag cart:${IMAGE\_TAG} ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/cart:${IMAGE\_TAG}

- cd ../..

- echo Building catalog image...

- cd src/catalog

- docker build -t catalog:${IMAGE\_TAG} .

- docker tag catalog:${IMAGE\_TAG} ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/catalog:${IMAGE\_TAG}

- cd ../..

post\_build:

commands:

- echo Pushing cart image...

- docker push ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/cart:${IMAGE\_TAG}

- echo Pushing catalog image...

- docker push ${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/catalog:${IMAGE\_TAG}

- echo Preparing Kubernetes deployment files...

- mkdir -p k8s-out

# Replace placeholders in cart.yaml

- sed "s|IMAGE\_PLACEHOLDER\_CART|${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/cart:${IMAGE\_TAG}|g; s|REGION\_PLACEHOLDER|${REGION}|g" k8s/cart.yaml > k8s-out/cart.yaml

# Replace placeholders in catalog.yaml

- sed "s|IMAGE\_PLACEHOLDER\_CATALOG|${ACCOUNT\_ID}.dkr.ecr.${REGION}.amazonaws.com/catalog:${IMAGE\_TAG}|g; s|DB\_HOST\_PLACEHOLDER|${DB\_HOST}|g" k8s/catalog.yaml > k8s-out/catalog.yaml

# Copy UI file as-is

- cp k8s/ui.yaml k8s-out/

artifacts:

base-directory: k8s-out

files:

- cart.yaml

- catalog.yaml

- ui.yaml