**07. CodeBuild - Review Buildspec.yml**

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/11-DevOps-with-AWS-Developer-Tools>

**Review the buildspec.yml for CodeBuild & Environment Variables**

--- Code Build Introduction

1. Get a high-level overview about CodeBuild Service

--- Environment Variables for CodeBuild

REPOSITORY\_URI = 180789647333.dkr.ecr.us-east-1.amazonaws.com/eks-devops-nginx

EKS\_KUBECTL\_ROLE\_ARN = arn:aws:iam::180789647333:role/EksCodeBuildKubectlRole

EKS\_CLUSTER\_NAME = eksdemo1

--- **Review buildspec.yml**

version: 0.2

phases:

  install:

    commands:

      - echo "Install Phase - Nothing to do using latest Amazon Linux Docker Image for CodeBuild which has all AWS Tools - https://github.com/aws/aws-codebuild-docker-images/blob/master/al2/x86\_64/standard/3.0/Dockerfile"

  pre\_build:

      commands:

        # Docker Image Tag with Date Time & Code Buiild Resolved Source Version

        - TAG="$(date +%Y-%m-%d.%H.%M.%S).$(echo $CODEBUILD\_RESOLVED\_SOURCE\_VERSION | head -c 8)"

        # Update Image tag in our Kubernetes Deployment Manifest

        - echo "Update Image tag in kube-manifest..."

        - sed -i 's@CONTAINER\_IMAGE@'"$REPOSITORY\_URI:$TAG"'@' kube-manifests/01-DEVOPS-Nginx-Deployment.yml

        # Verify AWS CLI Version

        - echo "Verify AWS CLI Version..."

        - aws --version

        # Login to ECR Registry for docker to push the image to ECR Repository

        - echo "Login in to Amazon ECR..."

        - $(aws ecr get-login --no-include-email)

        # Update Kube config Home Directory

        - export KUBECONFIG=$HOME/.kube/config

  build:

    commands:

      # Build Docker Image

      - echo "Build started on `date`"

      - echo "Building the Docker image..."

      - docker build --tag $REPOSITORY\_URI:$TAG .

  post\_build:

    commands:

      # Push Docker Image to ECR Repository

      - echo "Build completed on `date`"

      - echo "Pushing the Docker image to ECR Repository"

      - docker push $REPOSITORY\_URI:$TAG

      - echo "Docker Image Push to ECR Completed -  $REPOSITORY\_URI:$TAG"

      # Extracting AWS Credential Information using STS Assume Role for kubectl

      - echo "Setting Environment Variables related to AWS CLI for Kube Config Setup"

      - CREDENTIALS=$(aws sts assume-role --role-arn $EKS\_KUBECTL\_ROLE\_ARN --role-session-name codebuild-kubectl --duration-seconds 900)

      - export AWS\_ACCESS\_KEY\_ID="$(echo ${CREDENTIALS} | jq -r '.Credentials.AccessKeyId')"

      - export AWS\_SECRET\_ACCESS\_KEY="$(echo ${CREDENTIALS} | jq -r '.Credentials.SecretAccessKey')"

      - export AWS\_SESSION\_TOKEN="$(echo ${CREDENTIALS} | jq -r '.Credentials.SessionToken')"

      - export AWS\_EXPIRATION=$(echo ${CREDENTIALS} | jq -r '.Credentials.Expiration')

      # Setup kubectl with our EKS Cluster

      - echo "Update Kube Config"

      - aws eks update-kubeconfig --name $EKS\_CLUSTER\_NAME

      # Apply changes to our Application using kubectl

      - echo "Apply changes to kube manifests"

      - kubectl apply -f kube-manifests/

      - echo "Completed applying changes to Kubernetes Objects"

      # Create Artifacts which we can use if we want to continue our pipeline for other stages

      - printf '[{"name":"01-DEVOPS-Nginx-Deployment.yml","imageUri":"%s"}]' $REPOSITORY\_URI:$TAG > build.json

      # Additional Commands to view your credentials

      #- echo "Credentials Value is..  ${CREDENTIALS}"

      #- echo "AWS\_ACCESS\_KEY\_ID...  ${AWS\_ACCESS\_KEY\_ID}"

      #- echo "AWS\_SECRET\_ACCESS\_KEY...  ${AWS\_SECRET\_ACCESS\_KEY}"

      #- echo "AWS\_SESSION\_TOKEN...  ${AWS\_SESSION\_TOKEN}"

      #- echo "AWS\_EXPIRATION...  $AWS\_EXPIRATION"

      #- echo "EKS\_CLUSTER\_NAME...  $EKS\_CLUSTER\_NAME"

artifacts:

  files:

    - build.json

    - kube-manifests/\*