1. **Build Dockerfile for ecom-app locally and push to DockerHub**

# Use the official PHP 7.4 Apache base image

FROM php:7.4-apache

# Install mysqli extension for PHP

RUN docker-php-ext-install mysqli

# Set environment variables for database connection

ENV DB\_HOST=mysql-service

ENV DB\_PORT=3306

ENV DB\_USER=ecomuser

ENV DB\_PASSWORD=ecompassword

ENV DB\_NAME=ecomdb

# Copy the application source code to /var/www/html/

COPY ./learning-app-ecommerce /var/www/html/

# Update database connection strings to point to a Kubernetes service named mysql-service

# Replace the following placeholders with your actual database connection details

RUN sed -i 's/localhost/mysql-service/g' /var/www/html/index.php

# Set the working directory

WORKDIR /var/www/html/

# Expose port 80 for Apache

EXPOSE 80

# Start Apache

CMD ["apache2-foreground"]

**Create website deployment using image pushed to DockerHub** (website-deployment.yaml)

apiVersion: apps/v1

kind: Deployment

metadata:

  creationTimestamp: null

  labels:

    app: ecom-app

  name: ecom-app

spec:

  replicas: 1

  selector:

    matchLabels:

      app: ecom-app

  strategy: {}

  template:

    metadata:

      creationTimestamp: null

      labels:

        app: ecom-app

    spec:

      containers:

      - image: gforsack/ecom-web:v2

        name: ecom-app

        envFrom:

        - configMapRef:

            name: db-credentials

        resources: {}

        imagePullPolicy: Always

status: {}

---

apiVersion: v1

data:

  DB\_HOST: mysql-service

  DB\_NAME: ecomdb

  DB\_PASSWORD: ecompassword

  DB\_PORT: "3306"

  DB\_USER: ecomuser

kind: ConfigMap

metadata:

  name: db-credentials

---

apiVersion: v1

kind: Service

metadata:

  creationTimestamp: null

  labels:

    app: ecom-app

  name: website-service

spec:

  ports:

  - port: 80

    protocol: TCP

    targetPort: 80

  selector:

    app: ecom-app

  type: LoadBalancer

status:

  loadBalancer: {}

1. **Database containerization** (mariadb-deployment.yaml)

apiVersion: apps/v1

kind: Deployment

metadata:

  name: mariadb-deployment

spec:

  replicas: 1

  selector:

    matchLabels:

      app: mariadb

  template:

    metadata:

      labels:

        app: mariadb

    spec:

      containers:

      - name: mariadb

        image: mariadb:latest

        env:

        - name: MYSQL\_ROOT\_PASSWORD

          valueFrom:

            secretKeyRef:

              name: mariadb-root-password

              key: root-password

        ports:

        - containerPort: 3306

        volumeMounts:

        - name: mariadb-init-script

          mountPath: /docker-entrypoint-initdb.d

      volumes:

      - name: mariadb-init-script

        configMap:

          name: mariadb-init-script

---

apiVersion: v1

kind: Secret

metadata:

  name: mariadb-root-password

type: Opaque

data:

  root-password: ZWNvbXBhc3N3b3Jk #DB root password passed as secret for ‘security’

---

apiVersion: v1

kind: ConfigMap

metadata:

  name: mariadb-init-script

data:

  db-load-script.sql: |

    -- Create the database and user

    CREATE DATABASE IF NOT EXISTS ecomdb;

    GRANT ALL ON ecomdb.\* TO 'ecomuser'@'%' IDENTIFIED BY 'ecompassword';

    FLUSH PRIVILEGES;

    -- Use the database

    USE ecomdb;

    -- Create table

    CREATE TABLE IF NOT EXISTS products (

      id mediumint(8) unsigned NOT NULL auto\_increment,

      Name varchar(255) default NULL,

      Price varchar(255) default NULL,

      ImageUrl varchar(255) default NULL,

      PRIMARY KEY (id)

    ) AUTO\_INCREMENT = 1;

    -- Insert initial data

    INSERT INTO products (Name, Price, ImageUrl)

    VALUES

      ("Laptop", "100", "c-1.png"),

      ("Drone", "200", "c-2.png"),

      ("VR", "300", "c-3.png"),

      ("Tablet", "50", "c-5.png"),

      ("Watch", "90", "c-6.png"),

      ("Phone Covers", "20", "c-7.png"),

      ("Phone", "80", "c-8.png"),

      ("Laptop", "150", "c-4.png");

---

apiVersion: v1

kind: Service

metadata:

  name: mysql-service

spec:

  selector:

    app: mariadb

  ports:

  - protocol: TCP

    port: 3306

    targetPort: 3306

1. **Set Up EKS Cluster**
2. **Deploy Your Website to Kubernetes**

Navigate to the directory of your website-deployment.yaml file and run kubectl create -f .

1. **Expose Your Website**

This can be done in 2 ways – imperatively or declaratively.

kubectl expose deployment ecom-app --port=80 --target-port=80 --type=LoadBalancer

apiVersion: v1

kind: Service

metadata:

  creationTimestamp: null

  labels:

    app: ecom-app

  name: website-service

spec:

  ports:

  - port: 80

    protocol: TCP

    targetPort: 80

  selector:

    app: ecom-app

  type: LoadBalancer

status:

  loadBalancer: {}

1. **Scale Your Application**
2. **Perform a Rolling Update**
3. **Roll Back a Deployment**
4. **Autoscale Your Application**
5. **Implement Liveness and Readiness Probe**
6. **Utilize ConfigMaps and Secrets**
7. **Document Your Process**
8. **Package Everything in Helm**
9. **Implement Storage**
10. **Implement Basic CI/CD Pipeline**