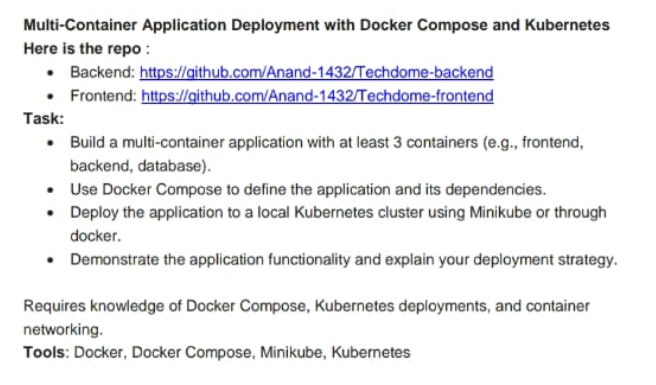
DevOps Assignment



Step 1: Clone the repositories

First I had Opened Mobaxterm and connect this to my EC2 server

Then I had placed git clone command:

git clone https://github.com/Anand-1432/Techdome-backend

git clone <https://github.com/Anand-1432/Techdome-frontend>

Step 2: Create a Dockerfile for Each Service

By using command ,

Touch dockerfile

For Backend: Navigate to the backend directory and create a Dockerfile.

cd Techdome-backend

Create a file named Dockerfile with the following content:

# Use the official Node.js image

FROM node:14

# Set the working directory

WORKDIR /app

# Copy package.json and install dependencies

COPY package\*.json ./

RUN npm install

# Copy the rest of the application code

COPY . .

# Expose the port

EXPOSE 5000

# Start the application

CMD ["npm", "start"]

For Frontend: Navigate to the frontend directory and create a Dockerfile.

Cd Techdome-frontend

Create a file named Dockerfile with the following content:

# Use the official Node.js image

FROM node:14

# Set the working directory

WORKDIR /app

# Copy package.json and install dependencies

COPY package\*.json ./

RUN npm install

# Copy the rest of the application code

COPY . .

# Expose the port

EXPOSE 3000

# Start the application

CMD ["npm", "start"]

Step 3: Create a Docker Compose File

create a docker-compose.yml file.

version: '3'

services:

backend:

build:

context: ./Techdome-backend

ports:

- "5000:5000"

networks:

- techdome-network

frontend:

build:

context: ./Techdome-frontend

ports:

- "3000:3000"

networks:

- techdome-network

depends\_on:

- backend

database:

image: postgres

environment:

POSTGRES\_DB: mydatabase

POSTGRES\_USER: user

POSTGRES\_PASSWORD: password

networks:

- techdome-network

networks:

techdome-network:

driver: bridge

Step 4: Build and Run with Docker Compose

In the same directory as your docker-compose.yml, run:

docker-compose up --build

This command builds and starts your containers.

Visit http://localhost:3000 to see the frontend.

Step 5: Prepare for Kubernetes Deployment

Stop the Docker Compose application with Ctrl + C.

Step 6: Set Up Minikube

minikube start

Step 7: Create Kubernetes Deployment Files

Create a directory for Kubernetes manifests:

mkdir k8s

cd k8s

Create deployment and service files:

1. Backend Deployment (backend-deployment.yml):

apiVersion: apps/v1

kind: Deployment

metadata:

name: backend

spec:

replicas: 1

selector:

matchLabels:

app: backend

template:

metadata:

labels:

app: backend

spec:

containers:

- name: backend

image: backend-image:latest # Use your built image name

ports:

- containerPort: 5000

---

apiVersion: v1

kind: Service

metadata:

name: backend

spec:

type: ClusterIP

ports:

- port: 5000

targetPort: 5000

selector:

app: backend

2. Frontend Deployment (frontend-deployment.yml):

apiVersion: apps/v1

kind: Deployment

metadata:

name: frontend

spec:

replicas: 1

selector:

matchLabels:

app: frontend

template:

metadata:

labels:

app: frontend

spec:

containers:

- name: frontend

image: frontend-image:latest # Use your built image name

ports:

- containerPort: 3000

---

apiVersion: v1

kind: Service

metadata:

name: frontend

spec:

type: LoadBalancer

ports:

- port: 3000

targetPort: 3000

selector:

app: frontend

3. Database Deployment (database-deployment.yml):

apiVersion: apps/v1

kind: Deployment

metadata:

name: database

spec:

replicas: 1

selector:

matchLabels:

app: database

template:

metadata:

labels:

app: database

spec:

containers:

- name: database

image: postgres

env:

- name: POSTGRES\_DB

value: mydatabase

- name: POSTGRES\_USER

value: user

- name: POSTGRES\_PASSWORD

value: password

ports:

- containerPort: 5432

---

apiVersion: v1

kind: Service

metadata:

name: database

spec:

type: ClusterIP

ports:

- port: 5432

targetPort: 5432

selector:

app: database

Step 8: Deploy to Kubernetes

Apply your deployment files:

kubectl apply -f backend-deployment.yml

kubectl apply -f frontend-deployment.yml

kubectl apply -f database-deployment.yml

Step 9: Access the Application

Get the Minikube IP:

minikube ip

You can access the frontend at http://<minikube-ip>:3000.

Step 10: Demonstrate Functionality

Check if the application is running smoothly, and connect the frontend to the backend as required. You can also check logs using:

kubectl logs <pod-name>

Deployment Strategy Explanation

1. Containerization: Each component (frontend, backend, database) runs in its own container, providing isolation and ease of scaling.

2. Docker Compose: Simplifies local development and testing of multiple interconnected services.

3. Kubernetes: Manages deployment, scaling, and networking in a production-like environment, ensuring resilience and availability.