Task: Automate Deployment with Kubernetes

Scenario:

Your development team has just finished building a new microservices-based application, and

you need to deploy it to a Kubernetes cluster. The application consists of three microservices:

frontend, backend, and database. Each microservice is packaged as a Docker container, and

the team has provided the necessary Kubernetes deployment and service configurations.

Test Case 1: Successful Deployment

Scenario: Execute the deployment script or process.

Expected Result: All microservices (frontend, backend, database) are successfully deployed to

the Kubernetes cluster without any errors. Verify that the pods are running and services are

accessible.

Test Case 2: Rolling Deployment Validation

Scenario: Initiate a new release using the deployment script or process.

Expected Result: The backend microservice should undergo a rolling deployment, updating the

replicas gradually without causing downtime. Verify that the application remains accessible and

responsive throughout the deployment.

Test Case 3: Ingress Configuration

Scenario: Access the application externally through the configured Ingress resource.

Expected Result: Access the frontend microservice using the external URL specified in the

Ingress resource. Confirm that the application is reachable and functions correctly through the

Ingress.

To perform Test Case 1: Successful Deployment, follow these steps:

* Prerequisites:
  + Ensure you have a running Kubernetes cluster. You can use tools like Minikube for local development or a cloud provider for a production environment.
  + Make sure kubectl is installed and configured to point to your Kubernetes cluster.

Step 2: Dockerize Microservices

Make sure each microservice (frontend, backend, and database) is packaged as a Docker container. Create a Dockerfile for each microservice.

1.FROM node:14 AS frontend

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 80

CMD ["npm", "start"]

2.FROM node:14 AS backend

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]

3.FROM postgres:latest AS database

Step 3: Kubernetes Deployment YAMLs

Create Kubernetes Deployment and Service YAML files for each microservice. Here is an example structure:

frontend-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: frontend-deployment

spec:

replicas: 3

selector:

matchLabels:

app: frontend

template:

metadata:

labels:

app: frontend

spec:

containers:

- name: frontend

image: reactapp

ports:

- containerPort: 80

2.backend-deployment.yaml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: backend-deployment

spec:

replicas: 3

selector:

matchLabels:

app: backend

template:

metadata:

labels:

app: backend

spec:

containers:

- name: backend

image: nodejs

ports:

- containerPort: 3000

3.database-deployment.yaml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: database-deployment

spec:

replicas: 1

selector:

matchLabels:

app: database

template:

metadata:

labels:

app: database

spec:

containers:

- name: database

image: postgres:latest

ports:

- containerPort: 5432

Step 4: Service YAMLs

Create Service YAML files for each microservice:

* frontend-service.yaml:

apiVersion: v1

kind: Service

metadata:

name: frontend-service

spec:

selector:

app: frontend

ports:

- protocol: TCP

port: 80

targetPort: 80

2.backend-service.yaml:

apiVersion: v1

kind: Service

metadata:

name: backend-service

spec:

selector:

app: backend

ports:

- protocol: TCP

port: 3000

targetPort: 3000

3.database-service.yaml:

apiVersion: v1

kind: Service

metadata:

name: database-service

spec:

selector:

app: database

ports:

- protocol: TCP

port: 5432

targetPort: 5432

Step 5: Ingress YAML

Create an Ingress YAML file:

ingress.yaml:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: my-ingress

spec:

rules:

- host: your-domain.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: frontend-service

port:

number: 80

Step 6: Deployment Script

Create a deployment script (e.g., deploy.sh) that applies all the YAML files to the Kubernetes cluster:

#!/bin/bash

kubectl apply -f frontend-deployment.yaml

kubectl apply -f backend-deployment.yaml

kubectl apply -f database-deployment.yaml

kubectl apply -f frontend-service.yaml

kubectl apply -f backend-service.yaml

kubectl apply -f database-service.yaml

kubectl apply -f ingress.yaml

To achieve the task of automating the deployment with Kubernetes and meeting the specified test cases, you can follow the steps outlined below. This is a simplified guide, and you may need to adapt it based on your specific requirements and configurations.

**Step 7: Rolling Deployment**

To achieve rolling deployments, update the Docker images in your Deployment YAMLs and run the deployment script again.

Step 8: Testing

Perform the test cases mentioned:

Test Case 1: Successful Deployment

* Run the deployment script and verify that all microservices are deployed without errors.

Verification:

* Use kubectl get pods to check that pods for frontend, backend, and database are in the "Running" state.

kubectl get pods

* Use kubectl get services to check that services for frontend, backend, and database are in the "Running" state and have ClusterIPs assigned.

kubectl get services

* kubeOptionally, you can test the accessibility of services by using kubectl port-forward or by exposing services externally if needed.

kubectl port-forward service/frontend-service 8080:80

Open a web browser and navigate to http://localhost:8080 to see if the frontend is accessible.

Test Case 2: Rolling Deployment Validation

* Update the backend microservice image in its Deployment YAML, run the deployment script, and verify that the application remains accessible and responsive throughout the deployment.

Test Case 3: Ingress Configuration

* Access the application using the external URL specified in the Ingress resource. Confirm that the frontend microservice is reachable and functions correctly.