1. **Install Kubernetes Tools:** Make sure you have the **kubectl** command-line tool installed and configured to communicate with your Kubernetes cluster.
2. **Create Kubernetes Deployment and Service YAMLs:**

Create a file named **nodejs-deployment.yaml** with the following content for the Node.js application:

apiVersion: apps/v1

kind: Deployment

metadata:

name: nodejs-app

spec:

replicas: 1

selector:

matchLabels:

app: nodejs-app

template:

metadata:

labels:

app: nodejs-app

spec:

containers:

- name: nodejs-container

image: devopsedu/employee

ports:

- containerPort: 8888

Create another file named **mongo-deployment.yaml** for the MongoDB deployment:

yamlCopy code

apiVersion: apps/v1

kind: Deployment

metadata:

name: mongo

spec:

replicas: 1

selector:

matchLabels:

app: mongo

template:

metadata:

labels:

app: mongo

spec:

containers:

- name: mongo-container

image: mongo

ports:

- containerPort: 27017

Now, create a file named **mongo-service.yaml** for the MongoDB service:

yamlCopy code

apiVersion: v1

kind: Service

metadata:

name: mongo

spec:

selector:

app: mongo

ports:

- protocol: TCP

port: 27017

targetPort: 27017

1. **Apply the Configurations:**

Apply the configurations to your Kubernetes cluster using the following commands:

kubectl apply -f nodejs-deployment.yaml

kubectl apply -f mongo-deployment.yaml

kubectl apply -f mongo-service.yaml

1. **Access the Node.js Application:**

You can access the Node.js application through the Node.js service. You might need to expose the service if it's not externally accessible by default.

1. **Test the Application:**

After accessing the Node.js application, you can perform actions like adding an employee. Ensure that the employee data gets stored in the MongoDB database correctly.

1. **Verify MongoDB Connection:**

Within the Node.js application's code, make sure the MongoDB connection is set to **mongo:27017**, which is the service name and port you defined in the **mongo-service.yaml** file.