

# Module 9: Container Orchestration using Kubernetes Part-II

---

## Case Study – Solution

edureka!

**edureka!**

© Brain4ce Education Solutions Pvt. Ltd.

## Problem Statement

You are working as a Devops Administrator. You've been tasked to deploy a multi-tier application on Kubernetes Cluster. The application is a NodeJS application available on Docker Hub with the following name:

*devopsedu/employee*

This NodeJS application works with a mongo database. MongoDB image is available on DockerHub with the following name:

*mongo*

You are required to deploy this application on Kubernetes:

- NodeJS is available on port 8888 in the container and will be reaching out to port 27017 for mongo database connection
- MongoDB will be accepting connections on port 27017

You must deploy this application using the CLI.

Once your application is up and running, ensure you can add an employee from the NodeJS application and verify by going to Get Employee page and retrieving your input.

**Hint:** Name the Mongo DB Service and deployment, specifically as "mongo".

## Solution

- **Step 1:** Ensure, all your pods are up and running before starting this project.

```
edureka@kmaster:~$ kubectl get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	calico-node-5894z	2/2	Running	10	2h
kube-system	calico-node-mjw6w	2/2	Running	0	25m
kube-system	etcd-kmaster	1/1	Running	6	2h
kube-system	kube-apiserver-kmaster	1/1	Running	7	2h
kube-system	kube-controller-manager-kmaster	1/1	Running	6	2h
kube-system	kube-dns-86f4d74b45-d8fl7	3/3	Running	15	2h
kube-system	kube-proxy-28d8l	1/1	Running	5	2h
kube-system	kube-proxy-lxc7j	1/1	Running	0	25m
kube-system	kube-scheduler-kmaster	1/1	Running	6	2h
kube-system	kubernetes-dashboard-7d5dcd6d9-q9plq	1/1	Running	5	2h

- **Step 2:** Build a deployment-backend.yaml for your Mongo DB, with the following code.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: mongo
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: mongo
    spec:
      containers:
        - name: mongo
          image: mongo
          ports:
            - containerPort: 27017
```

- **Step 2:** Run the following code in the terminal:

```
kubectl create -f deploy-backend.yaml
```

```
edureka@kmaster:~$ kubectl create -f deploy-backend.yaml
deployment.extensions "mongo" created
```

- **Step 3:** Next, we will create a service for this deployment, run the following command:

```
kubectl create service clusterip mongo --tcp=27017:27017
```

```
edureka@kmaster:~$ kubectl create service clusterip mongo --tcp=27017:27017
service "mongo" created
```

- **Step 2:** Build a deployment-frontend.yaml for your front-end website, with the following code.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: front-end
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: front-end
    spec:
      containers:
        - name: front-end
          image: devopsedu/employee
          ports:
            - containerPort: 8888
```

- **Step 3:** Run the following command on the terminal:

```
kubectl create -f deploy-frontend.yaml
edureka@kmaster:~$
edureka@kmaster:~$ kubectl create -f deployment-frontend.yaml
deployment.extensions "front-end" created
```

- **Step 4:** Next, we will create a service for this deployment, run the following command:

```
kubectl create service nodeport front-end --tcp=80:8888
```

```
edureka@kmaster:~$ kubectl create service nodeport front-end --tcp=80:8888
service "front-end" created
```

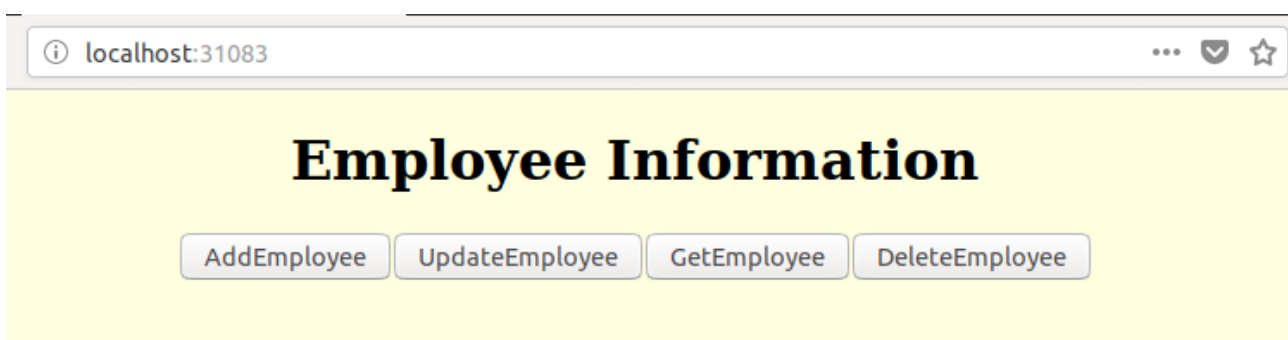
- **Step 5:** Enter the following command, to get the port number, where your front end will run.

```
kubectl get svc front-end
```

```
edureka@kmaster:~$ kubectl get svc front-end
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
front-end    NodePort    10.98.150.188 <none>        80:31083/TCP     6m
```

- **Step 6:** Copy this port number, and execute on the browser. You will be able to see the website.

```
http://localhost:<port-number>
```



➤ **Step 7:** Next, let's check our website's connectivity with the database

## Add New Employee

First Name :	<input type="text" value="edureka"/>
Last Name :	<input type="text" value="edureka"/>
ID :	<input type="text" value="1"/>
Department :	<input type="text" value="Content"/>
Designation :	<input type="text" value="Content"/>
Salary :	<input type="text" value="12345678"/>

Added employee with id 1 into the employee collection.

Our website has successfully connected to the database!

# edureka!