Lesson 08 Lesson-End Project

Deploying an App to the Kubernetes Cluster

Project Agenda: To deploy a Node.js application on the Kubernetes cluster

Description: You have created an application and want to containerize it. For efficient load balancing and auto-scaling of the containers, depending on the requirement, you decide to deploy your app on a Kubernetes cluster.

Tools required: Node.js, Docker, Kubernetes.

Prerequisites: You must have installed all the required tools and have a working Kubernetes cluster setup to proceed. In case you do not have it, please refer to Lesson 10 Demo 01 to install and set up a Kubernetes cluster.

Expected Deliverables: Create a Node.js application, docker image, and a Kubernetes deployment on the cluster

Steps to be followed:

1. Create a Node.js application
2. Create a Docker image of the application
3. Create a Kubernetes deployment using a **YAML** file
4. Verify the deployment of the app on the Kubernetes cluster

Step 1: Create a Node.js application

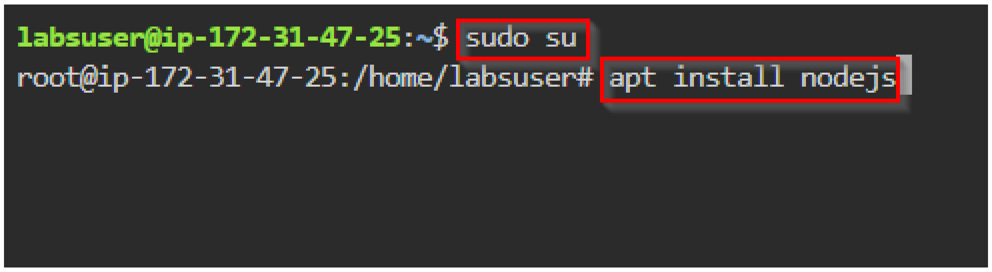
1.1 Install the Node.js and create a folder in your workspace by executing the commands given below:

sudo su

apt install nodejs

mkdir LEP10

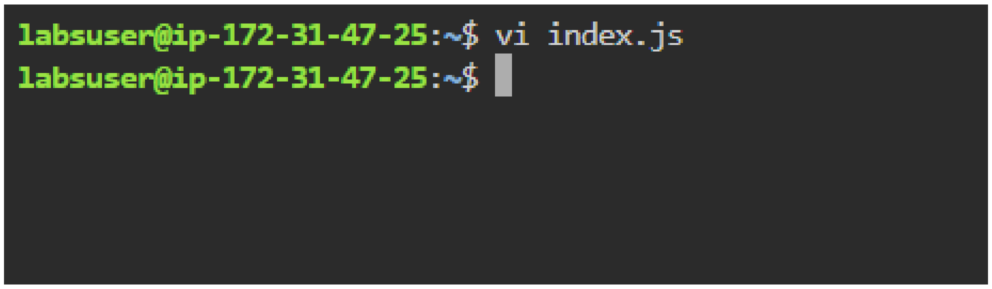
cd LEP10



```
labsuser@ip-172-31-47-25:~$ sudo su
root@ip-172-31-47-25:/home/labsuser# apt install nodejs
```

1.2 Create and open a file named index.js by using the command below:

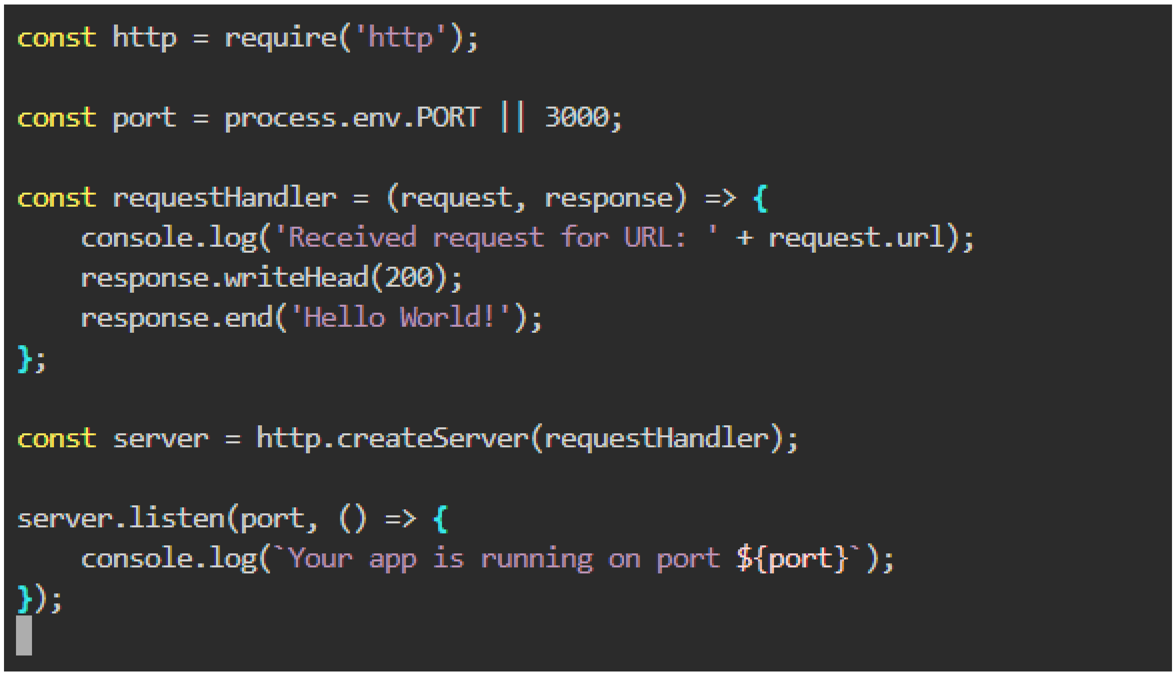
vi index.js



```
labsuser@ip-172-31-47-25:~$ vi index.js
labsuser@ip-172-31-47-25:~$
```

1.3 Add the below code in the index.js file:

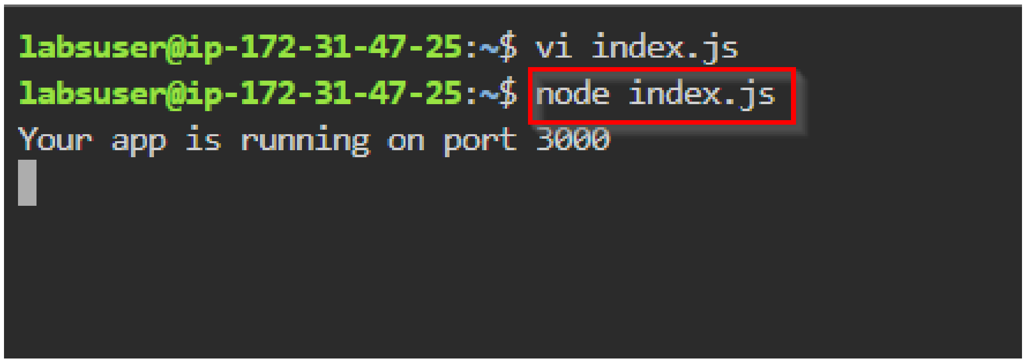
```
const http = require('http');

const port = process.env.PORT || 3000;

const requestHandler = (request, response) => {
   console.log('Received request for URL: ' + request.url);
   response.writeHead(200);
   response.end('Hello World!');
};

const server = http.createServer(requestHandler);

server.listen(port, () => {
   console.log(`Your app is running on port ${port}`);
});
```



```
const http = require('http');

const port = process.env.PORT || 3000;

const requestHandler = (request, response) => {
    console.log('Received request for URL: ' + request.url);
    response.writeHead(200);
    response.end('Hello World!');
};

const server = http.createServer(requestHandler);

server.listen(port, () => {
    console.log(`Your app is running on port ${port}`);
});
```

1.4 Run the app by using the command below:

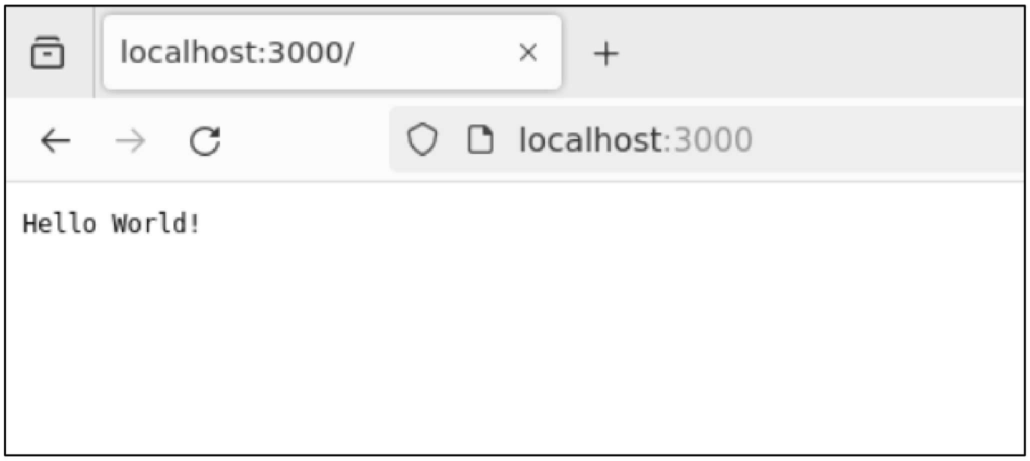
node index.js



```
labsuser@ip-172-31-47-25:~$ vi index.js
labsuser@ip-172-31-47-25:~$ node index.js
Your app is running on port 3000
```

1.5 Open the web browser in the master node, and paste this link http://localhost:3000/ to check the output:

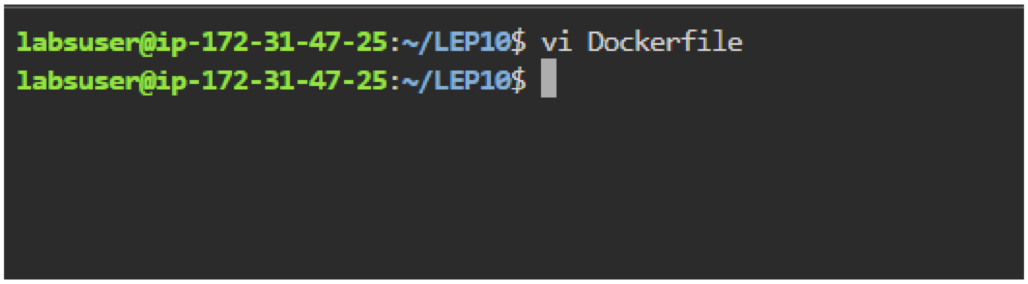
http://localhost:3000/



```
Hello World!
```

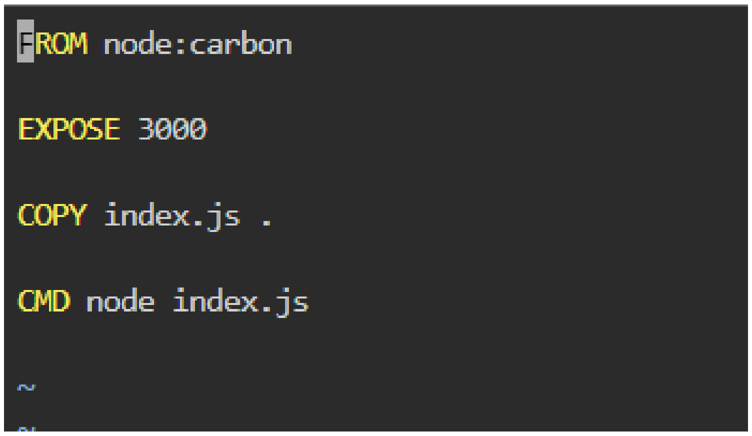
Step 2: Create a Docker image of the application

2.1 Create and open a file called **Dockerfile** by running the following command inside the LEP10 directory:
vi Dockerfile



```
labsuser@ip-172-31-47-25:~/LEP10$ vi Dockerfile
labsuser@ip-172-31-47-25:~/LEP10$
```

2.2 Add the following script in the Dockerfile:
FROM node:carbon
EXPOSE 3000
COPY index.js .
CMD node index.js

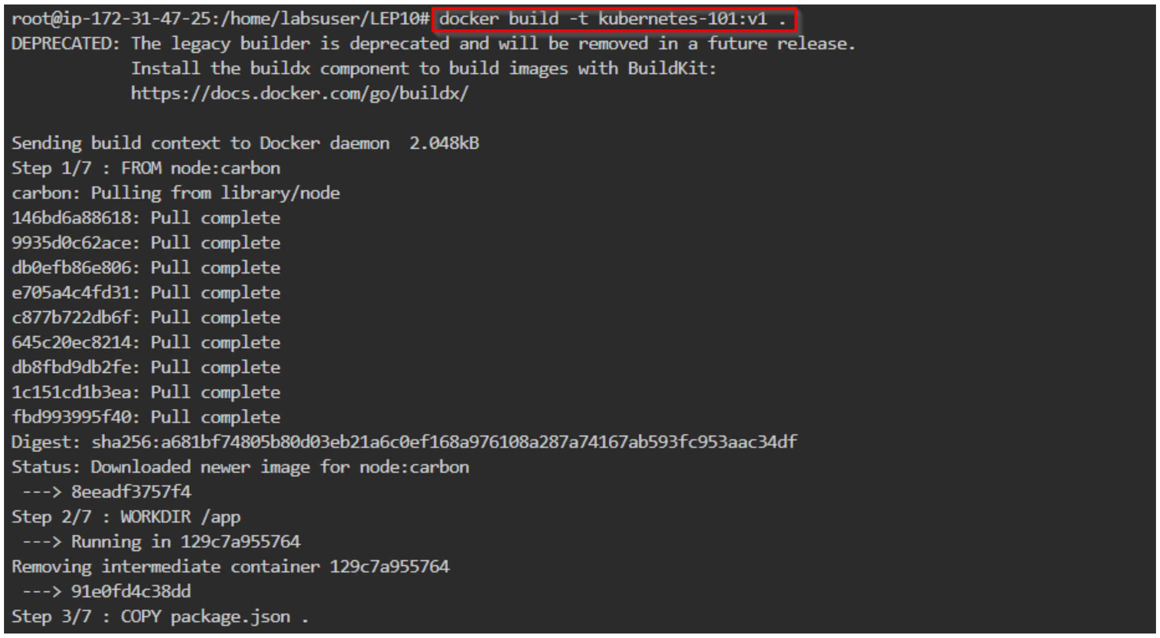


```
FROM node:carbon

EXPOSE 3000

COPY index.js .

CMD node index.js

~
```

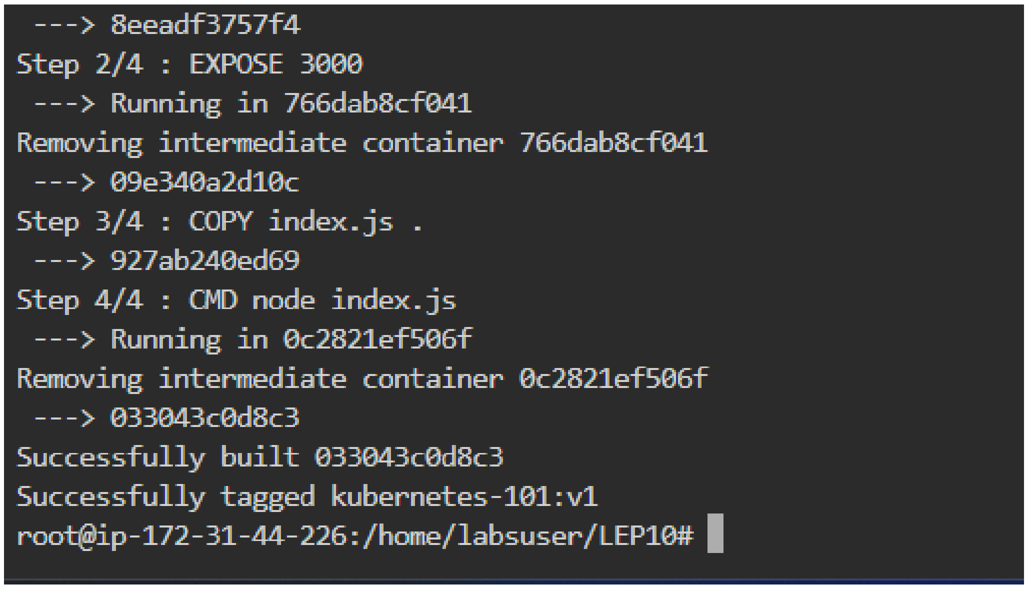
Note: This image extends the official Node.js image, exposes port 3000, copies the index.js file to the image, and starts the Node.js server

2.3 Build the Docker image using the command below:

docker build -t kubernetes-101:v1 .



```
root@ip-172-31-47-25:/home/labsuser/LEP10# docker build -t kubernetes-101:v1 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  2.048kB
Step 1/7 : FROM node:carbon
carbon: Pulling from library/node
146bd6a88618: Pull complete
9935d0c62ace: Pull complete
db0efb86e806: Pull complete
e705a4c4fd31: Pull complete
c877b722db6f: Pull complete
645c20ec8214: Pull complete
db8fbd9db2fe: Pull complete
1c151cd1b3ea: Pull complete
fbd993995f40: Pull complete
Digest: sha256:a681bf74805b80d03eb21a6c0ef168a976108a287a74167ab593fc953aac34df
Status: Downloaded newer image for node:carbon
 ---> 8eeadf3757f4
Step 2/7 : WORKDIR /app
 ---> Running in 129c7a955764
Removing intermediate container 129c7a955764
 ---> 91e0fd4c38dd
Step 3/7 : COPY package.json .
```

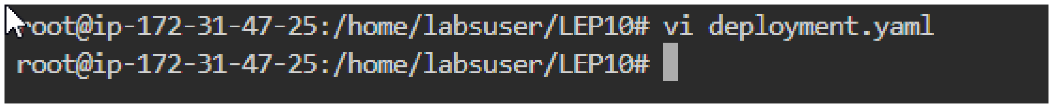


```
 ---> 8eeadf3757f4
Step 2/4 : EXPOSE 3000
 ---> Running in 766dab8cf041
Removing intermediate container 766dab8cf041
 ---> 09e340a2d10c
Step 3/4 : COPY index.js .
 ---> 927ab240ed69
Step 4/4 : CMD node index.js
 ---> Running in 0c2821ef506f
Removing intermediate container 0c2821ef506f
 ---> 033043c0d8c3
Successfully built 033043c0d8c3
Successfully tagged kubernetes-101:v1
root@ip-172-31-44-226:/home/labsuser/LEP10#
```

Note: If installation is not possible, utilize the **sudo su** command

Step 3: Create a Kubernetes deployment using a YAML file

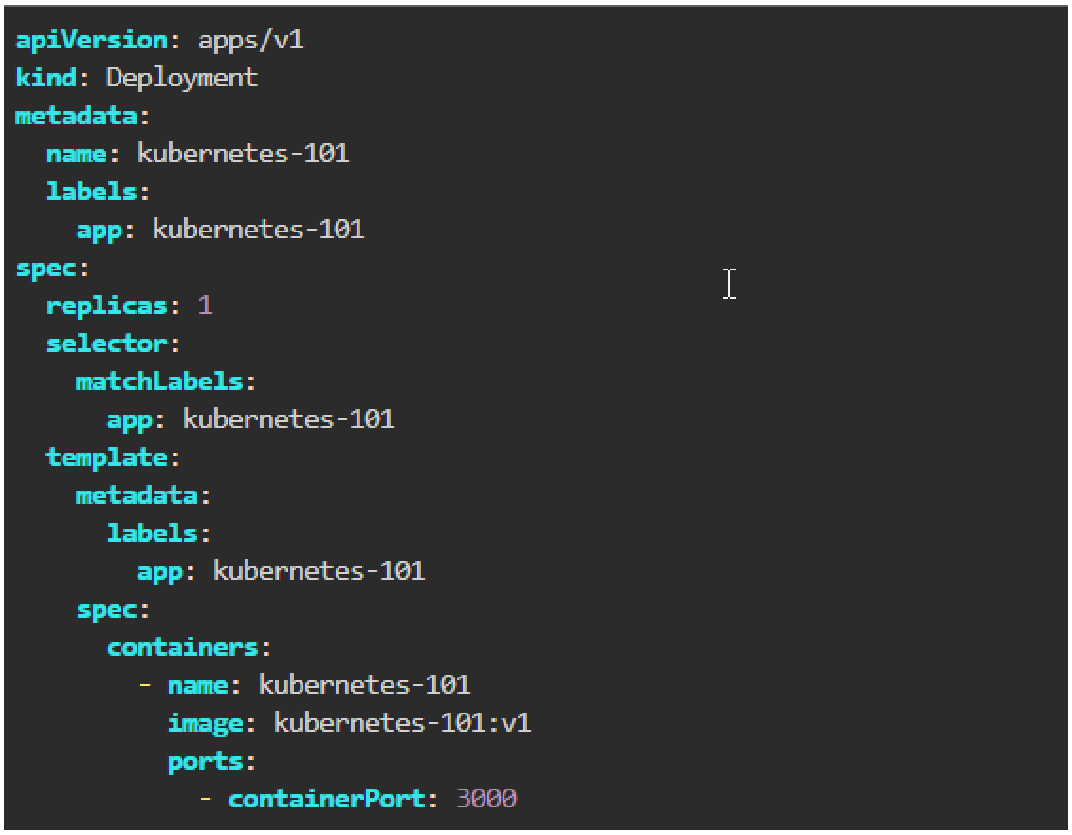
3.1 Create a file named **deployment** using the command below:
vi deployment.yaml



```
root@ip-172-31-47-25:/home/labsuser/LEP10# vi deployment.yaml
root@ip-172-31-47-25:/home/labsuser/LEP10#
```

3.2 Add the following code in the **deployment.yaml** configuration file:

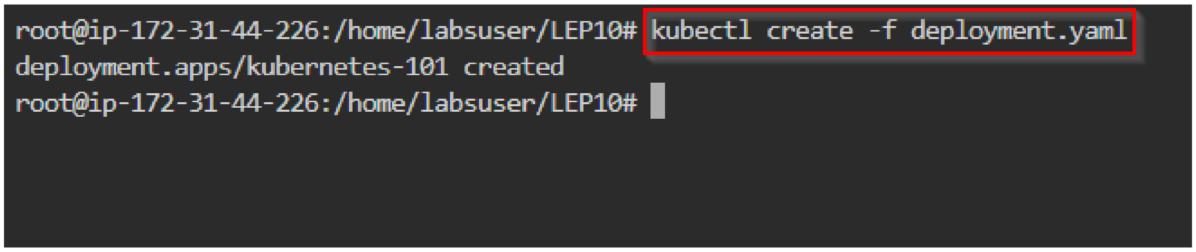
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kubernetes-101
  labels:
    app: kubernetes-101
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kubernetes-101
  template:
    metadata:
      labels:
        app: kubernetes-101
    spec:
      containers:
        - name: kubernetes-101
          image: kubernetes-101:v1
          ports:
            - containerPort: 3000
```



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kubernetes-101
  labels:
    app: kubernetes-101
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kubernetes-101
  template:
    metadata:
      labels:
        app: kubernetes-101
    spec:
      containers:
        - name: kubernetes-101
          image: kubernetes-101:v1
          ports:
            - containerPort: 3000
```

3.3 Run the following command to create a deployment:

kubectl create -f deployment.yaml



```
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl create -f deployment.yaml
deployment.apps/kubernetes-101 created
root@ip-172-31-44-226:/home/labsuser/LEP10#
```

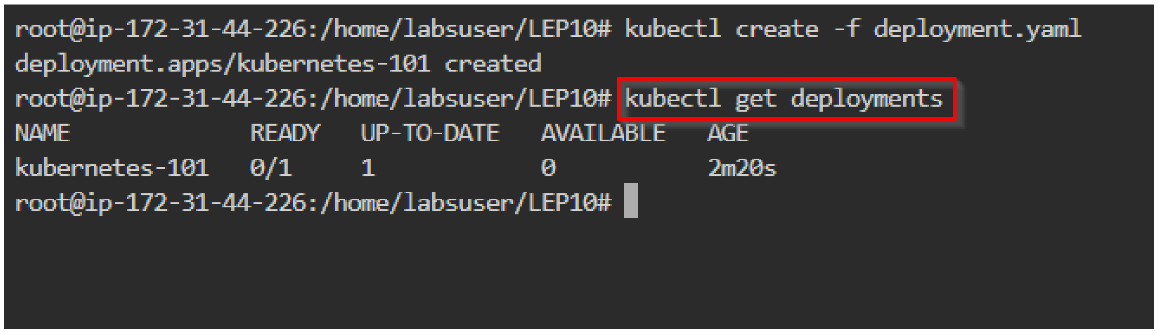
Note: If an error occurs, refer to the lesson 08 demo 01 to know how to install kubectl

3.4 Run the below commands to check the deployment that has been created and to know the details of the same:

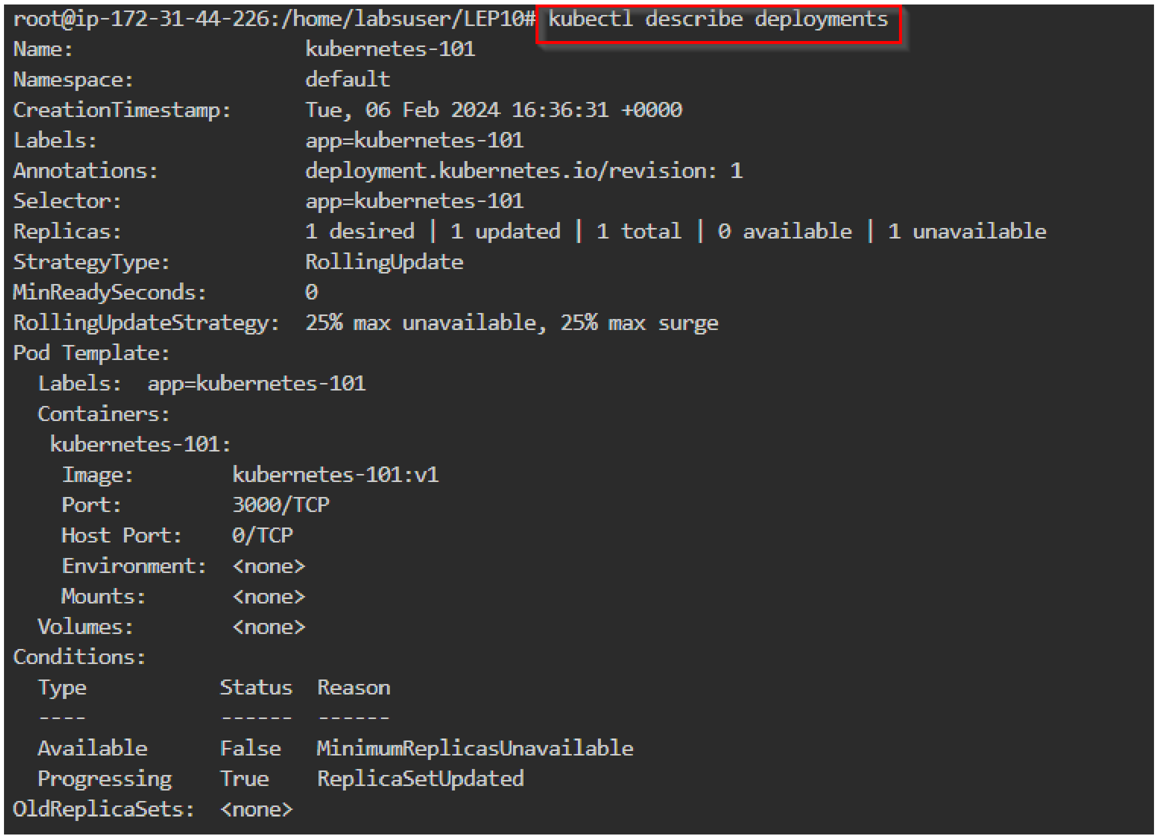
kubectl get deployments

kubectl describe deployments

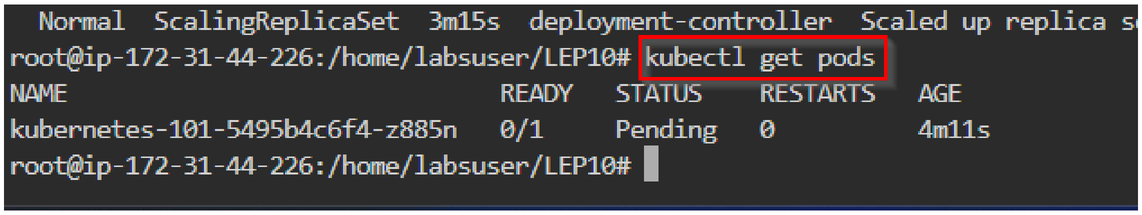
kubectl get pods



```
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl create -f deployment.yaml
deployment.apps/kubernetes-101 created
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-101   0/1     1            0           2m20s
root@ip-172-31-44-226:/home/labsuser/LEP10#
```



```
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl describe deployments
Name:                   kubernetes-101
Namespace:              default
CreationTimestamp:      Tue, 06 Feb 2024 16:36:31 +0000
Labels:                 app=kubernetes-101
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=kubernetes-101
Replicas:               1 desired | 1 updated | 1 total | 0 available | 1 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=kubernetes-101
  Containers:
   kubernetes-101:
    Image:        kubernetes-101:v1
    Port:         3000/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      False   MinimumReplicasUnavailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:  <none>
```

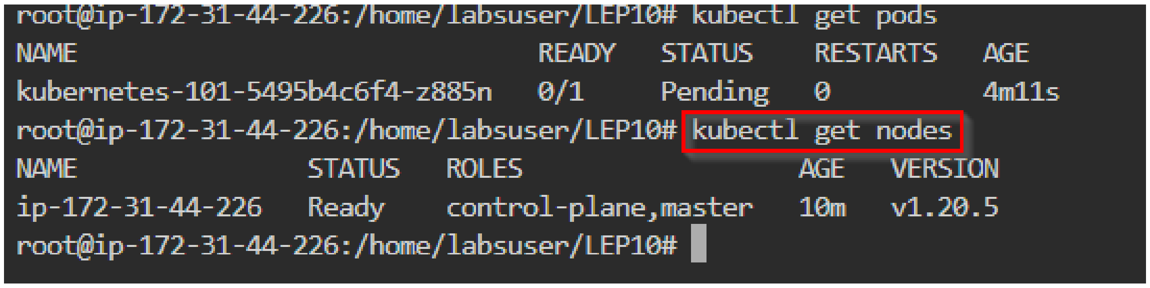


```
  Normal  ScalingReplicaSet  3m15s  deployment-controller  Scaled up replica s
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
kubernetes-101-5495b4c6f4-z885n   0/1     Pending   0          4m11s
root@ip-172-31-44-226:/home/labsuser/LEP10#
```

Note: If the pod is in pending state, we need to change the pod status from a **pending** to **running** state. Execute the below commands.

3.5 Use the below command to get the node status:

kubectl get nodes

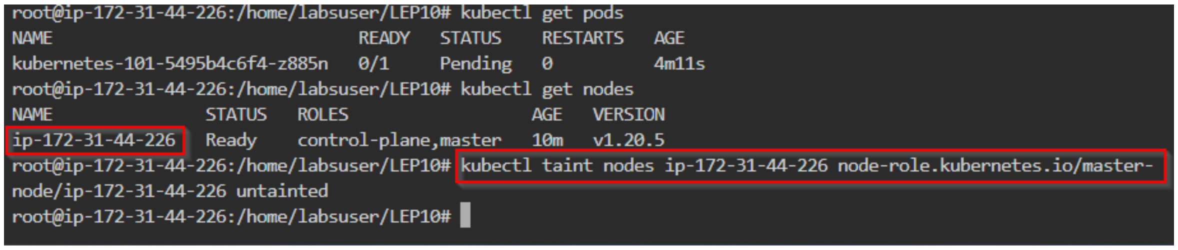


```
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
kubernetes-101-5495b4c6f4-z885n   0/1     Pending   0          4m11s
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl get nodes
NAME               STATUS   ROLES                  AGE   VERSION
ip-172-31-44-226   Ready    control-plane,master   10m   v1.20.5
root@ip-172-31-44-226:/home/labsuser/LEP10#
```

Note: Copy the node name and execute the below command using the copied node name

3.6 Use the following command to taint nodes from the master node:

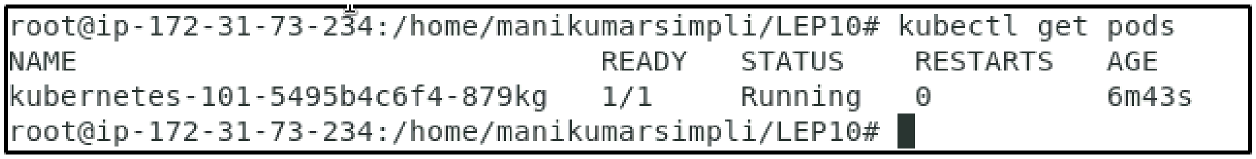
kubectl taint nodes <node name> node-role.kubernetes.io/master-



```
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
kubernetes-101-5495b4c6f4-z885n   0/1     Pending   0          4m11s
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl get nodes
NAME               STATUS   ROLES                  AGE   VERSION
ip-172-31-44-226   Ready    control-plane,master   10m   v1.20.5
root@ip-172-31-44-226:/home/labsuser/LEP10# kubectl taint nodes ip-172-31-44-226 node-role.kubernetes.io/master-
node/ip-172-31-44-226 untainted
root@ip-172-31-44-226:/home/labsuser/LEP10#
```

Note: Use the <node name> as your NAME of your nodes

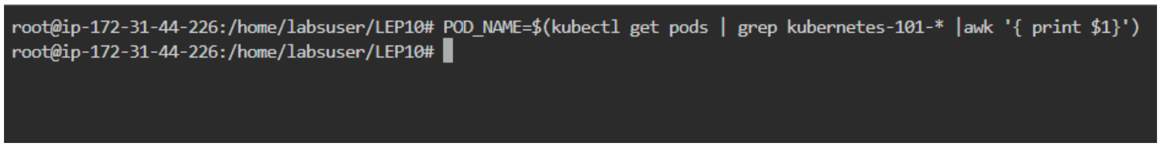
Run **kubectl get pods** to check if the pod is running, as shown above.



```
root@ip-172-31-73-234:/home/manikumarsimpli/LEP10# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
kubernetes-101-5495b4c6f4-879kg   1/1     Running   0          6m43s
root@ip-172-31-73-234:/home/manikumarsimpli/LEP10# 
```

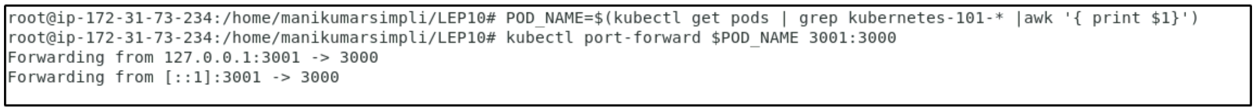
Step 4: Verify the deployment of the app on the Kubernetes cluster

4.1 Verify the deployment via Kubernetes port forwarding. Run the following command on the terminal to get the pod name:
POD_NAME=$(kubectl get pods | grep kubernetes-101-* |awk '{ print $1}')



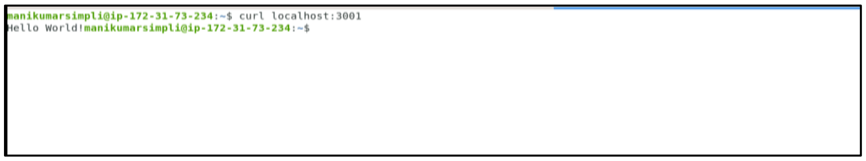
```
root@ip-172-31-44-226:/home/labsuser/LEP10# POD_NAME=$(kubectl get pods | grep kubernetes-101-* |awk '{ print $1}')
root@ip-172-31-44-226:/home/labsuser/LEP10# 
```

4.2 Run the following command to forward the pod name:
kubectl port-forward $POD_NAME 3001:3000



```
root@ip-172-31-73-234:/home/manikumarsimpli/LEP10# POD_NAME=$(kubectl get pods | grep kubernetes-101-* |awk '{ print $1}')
root@ip-172-31-73-234:/home/manikumarsimpli/LEP10# kubectl port-forward $POD_NAME 3001:3000
Forwarding from 127.0.0.1:3001 -> 3000
Forwarding from [::1]:3001 -> 3000
```

4.3 Open another terminal without closing the current one to access the deployed application on the second terminal using the command:
curl localhost:3001



```
manikumarsimpli@ip-172-31-73-234:~$ curl localhost:3001
Hello World!manikumarsimpli@ip-172-31-73-234:~$
```

By following these steps, you have successfully deployed the application on the Kubernetes cluster.