**Lesson 10 Lesson-End Project**

**Deploy an App to the Kubernetes Cluster**



**Steps to be followed:**

1. Creating a Node.js application
2. Creating a Docker image of the application
3. Creating a Kubernetes deployment using a yaml file
4. Verifying the deployment of the app on the Kubernetes cluster

**Step 1: Creating a Node.js application**

1. Create a folder in your workspace by executing the below commands:  
     
   ***sudo su***

***apt install nodejs***

***mkdir LEP10***

***cd LEP10***

1. Create a file named index.js

***vi index.js***

1. Add the following code inside the index.js which will setup a simple server that can be used to test the deployment

***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!');***

***};***

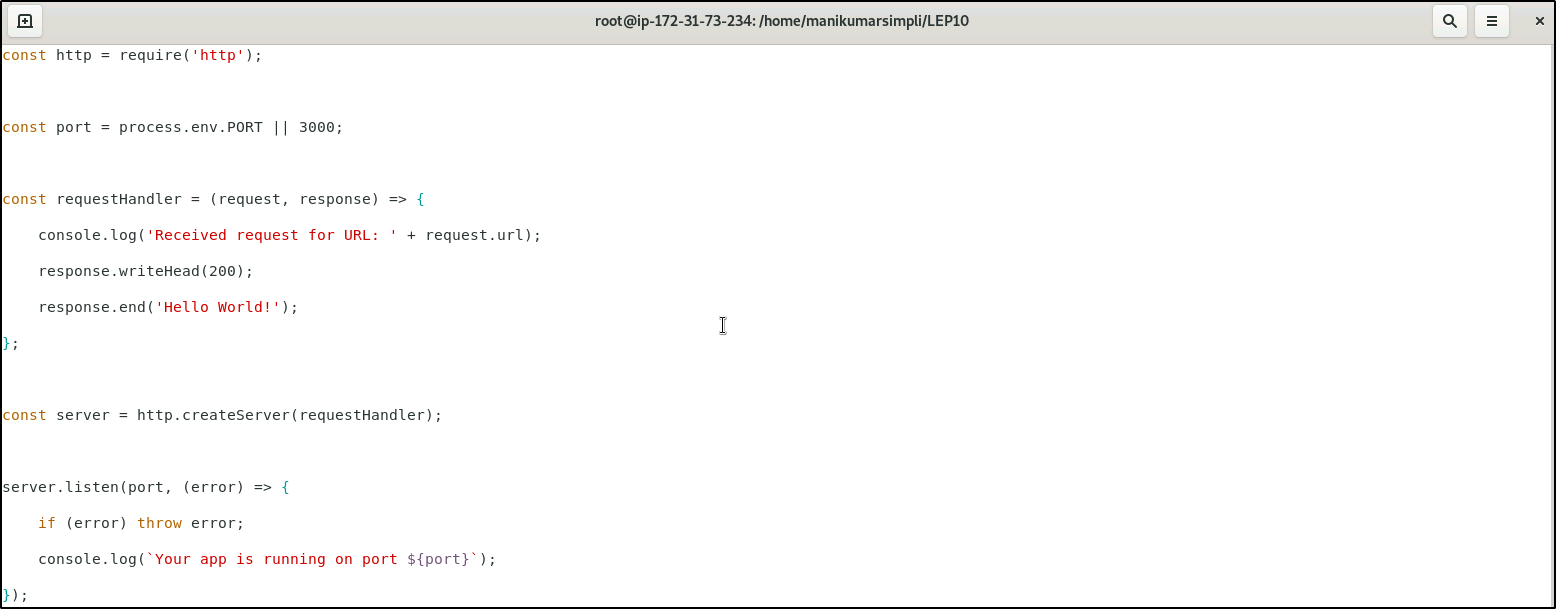
***const server = http.createServer(requestHandler);***

***server.listen(port, (error) => {***

***if (error) throw error;***

***console.log(`Your app is running on port ${port}`);***

***});***

****

1. Run the app to make sure it works fine

***node index.js***

****

1. From your browser, on navigating to <http://localhost:3000> you must be able to see *Hello World*

**

**Step 2: Creating a Docker image of the application**

1. Create a file called Dockerfile by running the following command inside the LEP10 directory:

***vi Dockerfile***

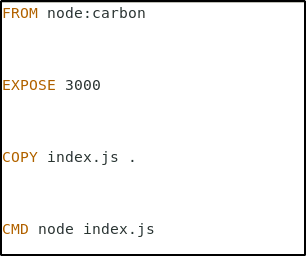
1. Add the following content in the file:

***FROM node:carbon***

***EXPOSE 3000***

***COPY index.js .***

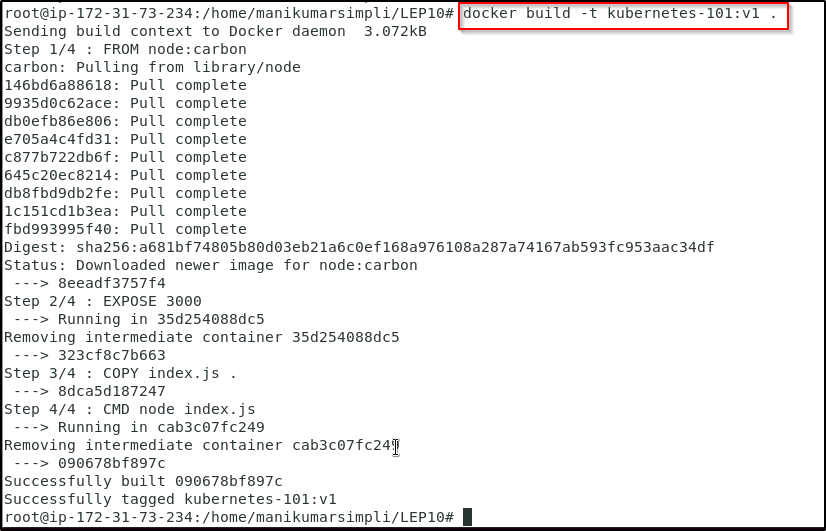
***CMD node index.js***

******

1. This image extends the official Node.js image, exposes port 3000, copies index.js file to the image and starts the NodeJs server.
2. Build the Docker image using the below command

***docker build -t kubernetes-101:v1 .***

1. The command above will build our image tagged kubernetes-101:v1 using our Dockerfile.

****

**Step 3: Creating a Kubernetes deployment using a yaml file**

1. Create a file called deployment.yaml

***vi deployment.yaml***

1. 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:***

***- name: http***

***containerPort: 3000***

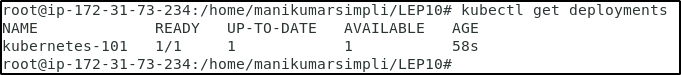
1. Run the following command to create a deployment

***kubectl create -f deployment.yaml***

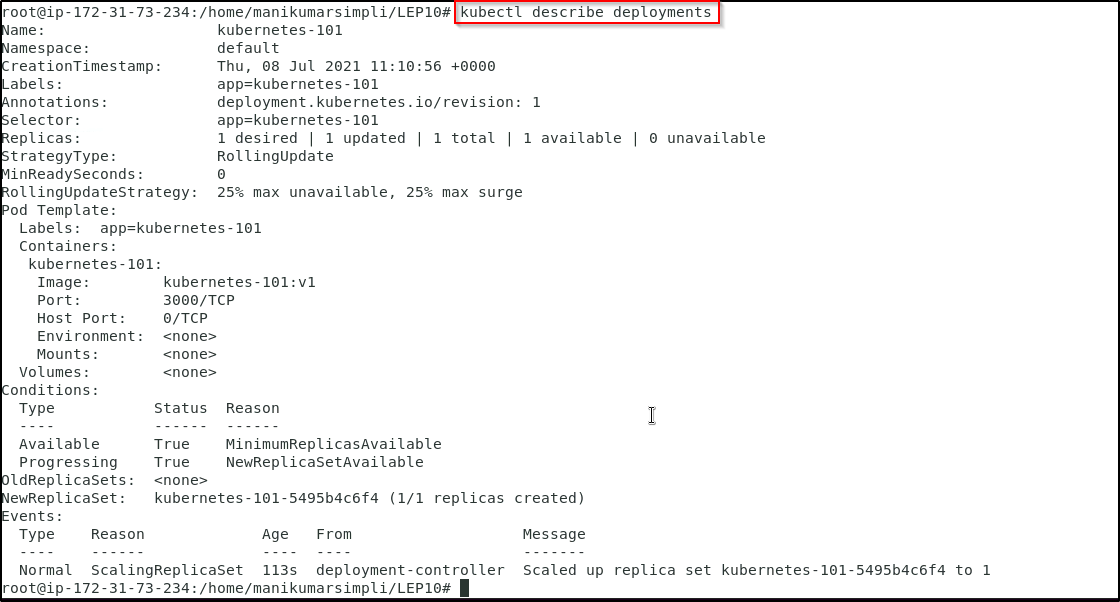
****

1. Run the below commands to check the deployment that has been created and also to know the details of the same

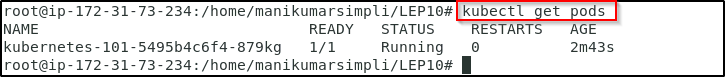
***kubectl get deployments***

******

***kubectl describe deployments***

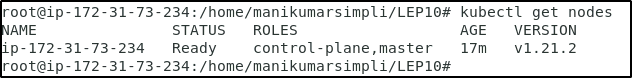
******

***kubectl get pods***

******

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

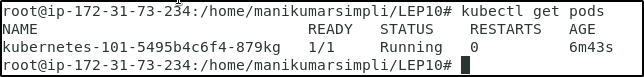
***kubectl get nodes***

******

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

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

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



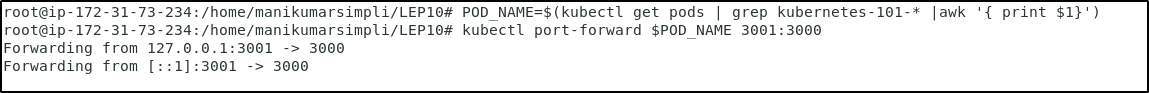
**Step 4: Verifying the deployment of the app on the Kubernetes cluster**

1. We can 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}')***

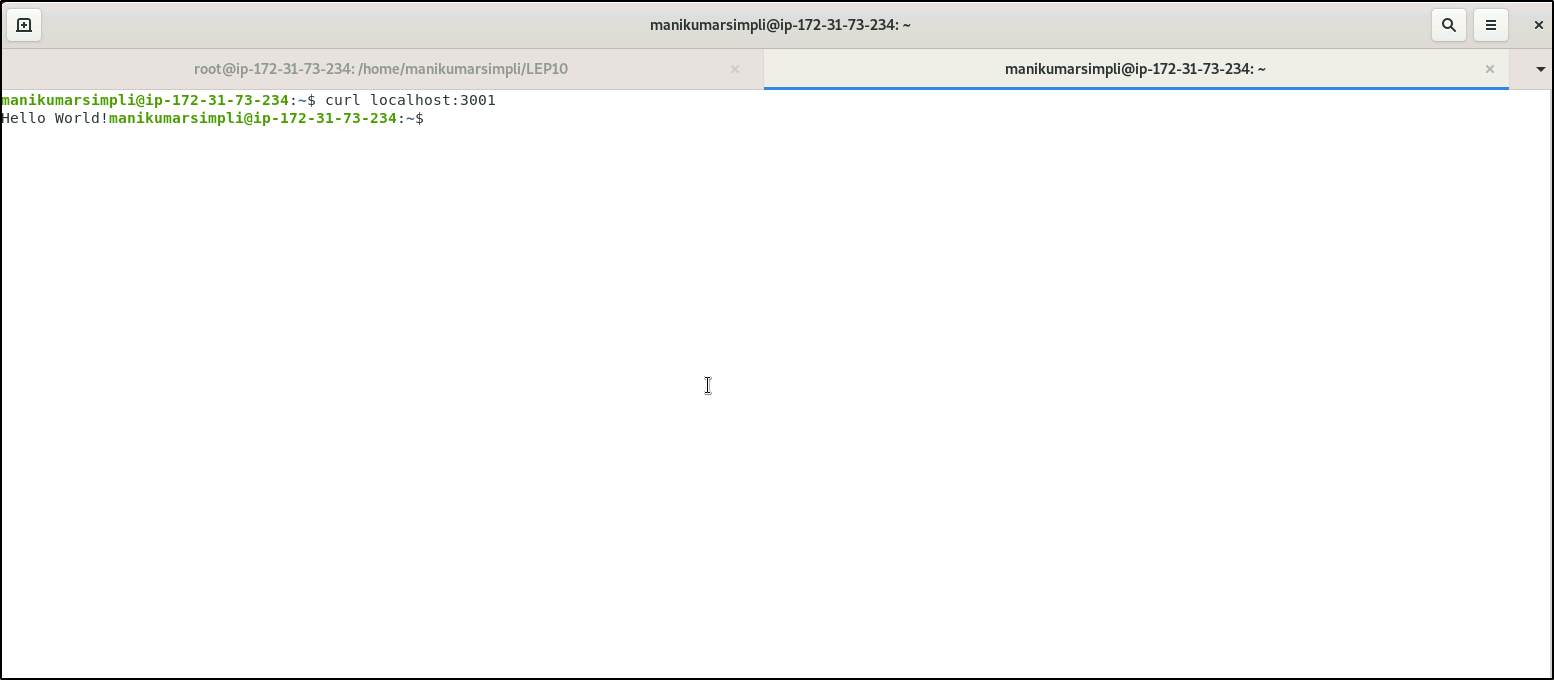
1. In order to port forward pod’s port to localhost run the following command:

***kubectl port-forward $POD\_NAME 3001:3000***

******

1. Open another terminal without closing the current one. We should now be able to access the deployed application on the second terminal using the command:

***curl localhost:3001***

******

1. This confirms that we have successfully deployed the application on the Kubernetes cluster.