Name- Miki I. Lalwani D15B/37

Advance DevOps lab

Experiment 4

Aim-
To install kubectl and execute kubectl commands to manage the kubernetes cluster and deploy your first kubernetes application.

Theory-

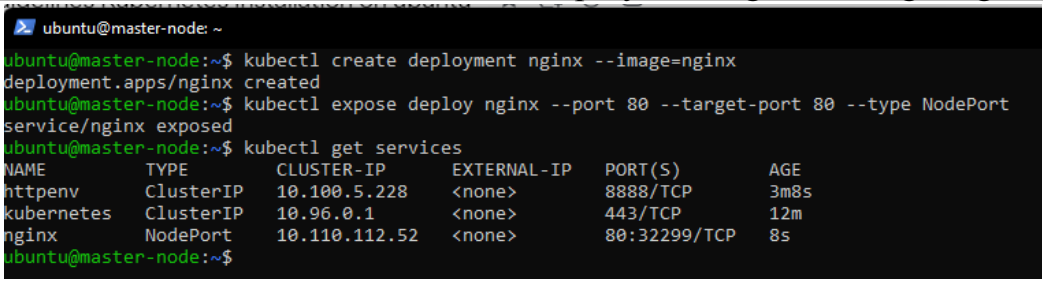
Docker is an open platform for developing, shipping and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker you can manage your infrastructure in the same ways you manage your applications. By taking advantage of docker's methodologies for shipping, testing and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

The kubernetes command-line tool, kubectl, allows you to run commands against kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources and view logs.

You can now deploy any containerized application to your cluster. To keep things familiar, let's deploy Nginx using *Deployments* and *Services* to see how this application can be deployed to the cluster. You can use the commands below for other containerized applications as well, provided you change the Docker image name and any relevant flags (such as ports and volumes).

Still within the master node, execute the following command to create a deployment named nginx:

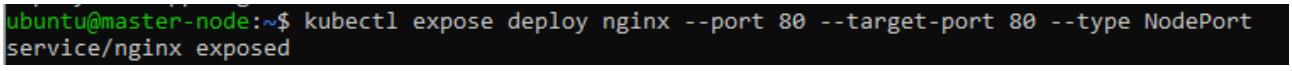
kubernetes-master:~$kubectl create deployment nginx --image=nginx



A deployment is a type of Kubernetes object that ensures there's always a specified number of pods running based on a defined template, even if the pod crashes during the cluster's lifetime. The above deployment will create a pod with one container from the Docker registry's Nginx Docker Image.

Next, run the following command to create a service named nginx that will expose the app publicly. It will do so through a *NodePort*, a scheme that will make the pod accessible through an arbitrary port opened on each node of the cluster:

kubernetes-master:~$kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort



Services are another type of Kubernetes object that expose cluster internal services to clients, both internal and external. They are also capable of load balancing requests to multiple pods, and are an integral component in Kubernetes, frequently interacting with other components.

Run the following command:

kubernetes-master:~$kubctl get services

```
ubuntu@master-node:~$ kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
httpenv      ClusterIP   10.100.5.228    <none>        8888/TCP       3m8s
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        12m
nginx        NodePort    10.110.112.52   <none>        80:32299/TCP   8s
```

This will output text similar to the following:

Output

```
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          1d
nginx        NodePort    10.109.228.209  <none>        80:nginx_port/TCP 40m
```

From the third line of the above output, you can retrieve the port that Nginx is running on. Kubernetes will assign a random port that is greater than 30000 automatically, while ensuring that the port is not already bound by another service.

Note: if you're running your setup on ec2 ensure the nginx_port is open under the inbound rules in the security groups.

To test that everything is working, visit

http://worker_1_ip:nginx_port or

http://worker_2_ip:nginx_port

through a browser on your local machine. You will see Nginx's familiar welcome

page. To see the deployed container on the worker node switch to worker01

on-slave#docker ps

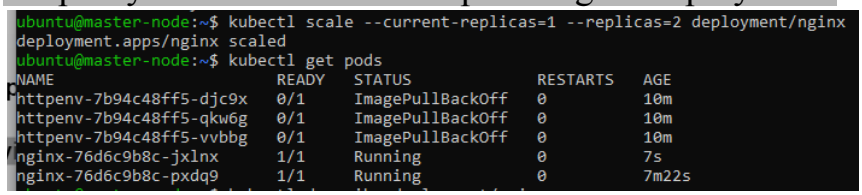
Output: you will see the container for nginx image running.

If you want to scale up the replicas for a deployment (nginx in our case) the use the following command:

kubernetes-master:~$kubectl scale --current-replicas=1 --replicas=2 deployment/nginx

kubernetes-master:~$kubectl get pods

Output: you will see 2/2 as output in nginx deployment.



kubernetes-master:~$kubectl describe deployment/nginx

Output: give details about the service deployed

```
ubuntu@master-node: ~
ubuntu@master-node:~$ kubectl describe deployment/nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Tue, 20 Sep 2022 10:37:35 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Progressing    True    NewReplicaSetAvailable
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-76d6c9b8c (2/2 replicas created)
Events:
  Type    Reason             Age    From                    Message
```

If you would like to remove the Nginx application, first delete the nginx service from the master node:

kubernetes-master:~$kubectl delete service nginx

```
ubuntu@master-node:~$ kubectl delete service nginx
service "nginx" deleted
```

Run the following to ensure that the service has been deleted:

kubernetes-master:~$kubectl get services

You will see the following output:

```
ubuntu@master-node:~$ kubectl get services
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
httpenv      ClusterIP   10.100.5.228   <none>        8888/TCP   10m
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    20m
```

Output

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 1d

Then delete the deployment:

kubernetes-master:~$kubectl delete deployment nginx

```
ubuntu@master-node:~$ kubectl delete deployment nginx
deployment.apps "nginx" deleted
```

Run the following to confirm that this worked:

kubernetes-master:~$kubectl get deployments

```
ubuntu@master-node:~$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
httpenv   0/3     3            0           11m
ubuntu@master-node:~$
```

Output

No resources found.

Conclusion-

Thus we successfully installed and executed kubectl.