Name : Abhishek Chaurasiya **Class : D15B** **Roll no :**09

Adv Devops Exp 4

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=ngi**

```
ubuntu@ip-172-31-95-40:~$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   16m
ubuntu@ip-172-31-95-40:~$
```

```
ubuntu@ip-172-31-95-40:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
ubuntu@ip-172-31-95-40:~$
```

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

```
ubuntu@ip-172-31-95-40:~$ kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort
service/nginx exposed
ubuntu@ip-172-31-95-40:~$
```

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:~$ kubectl get service

```
ubuntu@ip-172-31-95-40:~$ kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        20m
nginx        NodePort    10.100.213.17   <none>        80:30889/TCP   35s
ubuntu@ip-172-31-95-40:~$
```

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.

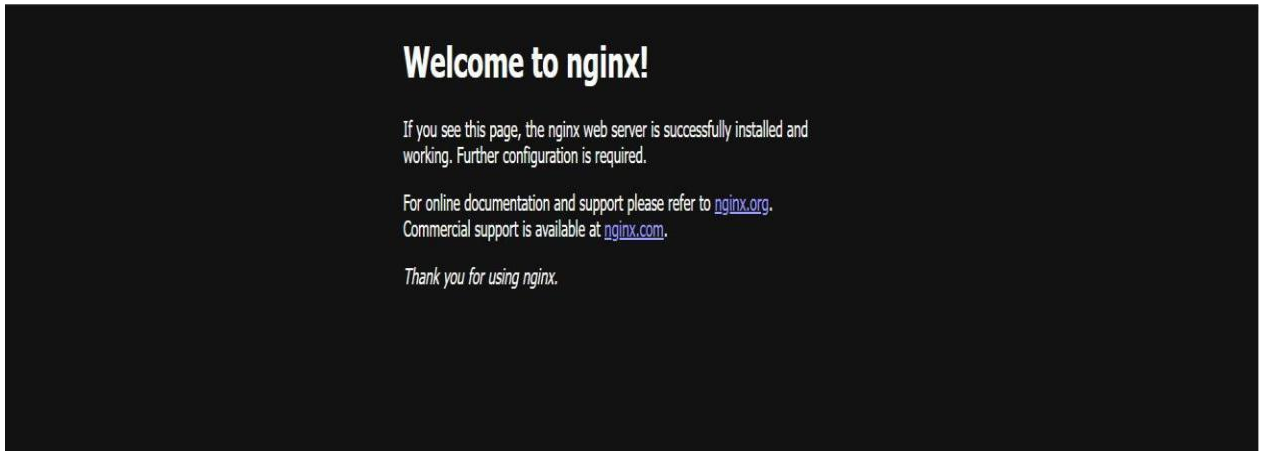
To test that everything is working, visit

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



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/ngin

```
ubuntu@ip-172-31-95-40:~$ kubectl scale --current-replicas=1 --replicas=2 deployment/nginx
deployment.apps/nginx scaled
```

kubernetes-master:~$ kubectl get pods

```
ubuntu@ip-172-31-95-40:~$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-76d6c9b8c-vjclg    1/1     Running   0          62s
nginx-76d6c9b8c-w7mk8    1/1     Running   0          16m
ubuntu@ip-172-31-95-40:~$
```

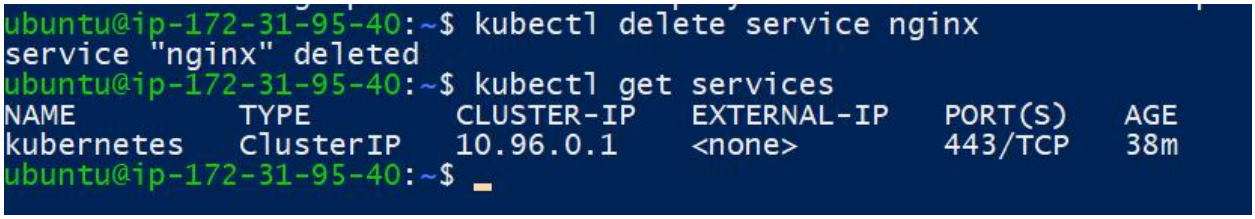
kubernetes-master:~$ kubectl describe deployment/nginx

```
ubuntu@ip-172-31-95-40:~$ kubectl describe deployment/nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Tue, 30 Aug 2022 11:38:31 +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
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  16m   deployment-controller  Scaled up replica set nginx-76d6c9b8c to 1
  Normal  ScalingReplicaSet  95s   deployment-controller  Scaled up replica set nginx-76d6c9b8c to 2 from 1
ubuntu@ip-172-31-95-40:~$
```

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

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

kubernetes-master:~$ kubectl delete service nginx



Then delete the deployment:

kubernetes-master:~$ kubectl delete deployment nginx Run the following to confirm that this worked: **kubernetes-master:~$ kubectl get deployments**

```
ubuntu@ip-172-31-95-40:~$ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   2/2     2            2           19m
ubuntu@ip-172-31-95-40:~$ kubectl delete deployment nginx
deployment.apps "nginx" deleted
ubuntu@ip-172-31-95-40:~$ kubectl get deployments
No resources found in default namespace.
ubuntu@ip-172-31-95-40:~$
```

Remove a node from Kubernetes

On Master Node

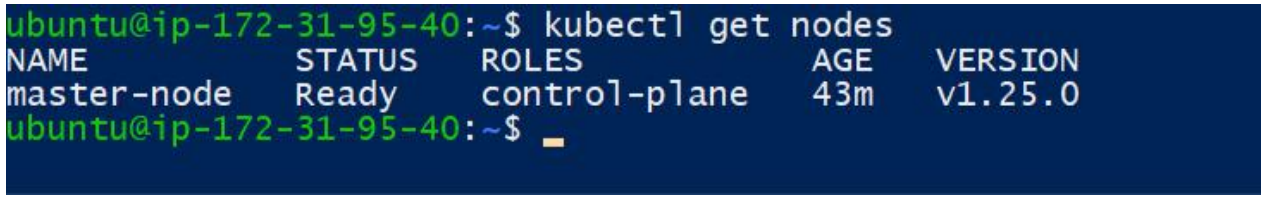
Find the node : kubernetes-master:~$ kubectl get nodes

Drain the node : kubernetes-master:~$ kubectl drain nodetoberemoved

Delete Node : kubernetes-master:~$ kubectl delete node nodetoberemoved

```
ubuntu@ip-172-31-95-40:~$ kubectl drain worker-01
node/worker-01 already cordoned
error: unable to drain node "worker-01" due to error:cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore): kube-flannel/kube-flannel-ds-nj
pp6, kube-system/kube-proxy-l14bq, continuing command...
There are pending nodes to be drained:
 worker-01
cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore): kube-flannel/kube-flannel-ds-njpp6, kube-system/kube-proxy-l14bq
ubuntu@ip-172-31-95-40:~$ kubectl delete node worker-01
node "worker-01" deleted
ubuntu@ip-172-31-95-40:~$ kubectl delete node worker-02
node "worker-02" deleted
ubuntu@ip-172-31-95-40:~$
```

kubernetes-master:~$ kubectl get nodes



All Worker Nodes got deleted.