

EXPERIMENT - 4

AIM: To install kubectl and execute kubectl commands to manage the Kubernetes cluster and deploy your first Kubernetes application.

THEORY:

Originally developed by Google, Kubernetes is an open source container orchestration platform designed to automate the deployment, scaling and management of containerized applications.

In fact, Kubernetes has established itself as the de facto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation backed by key players like Google, AWS, Microsoft, IBM, Cisco and Red Hat.

Kubernetes Deployment: - It is used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployment can scale the number of replica pods, enable the rollout of updated code in a controlled manner or roll back to an earlier deployment version if necessary.

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
```

```
ubuntu@ip-172-31-82-86:~$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes    ClusterIP     10.96.0.1     <none>         443/TCP    24m
```

```
ubuntu@ip-172-31-82-86:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
```

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:

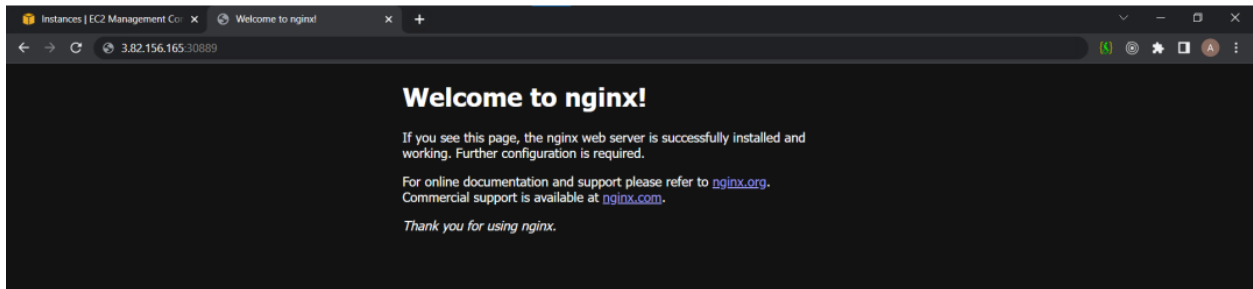
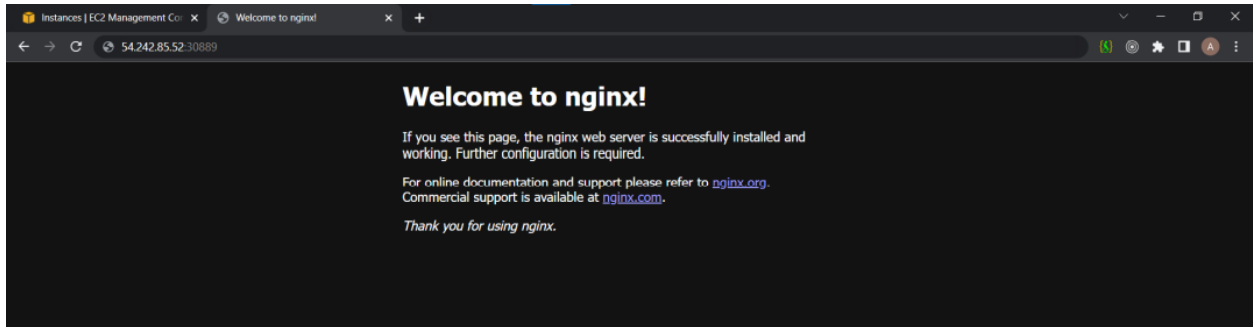
```
kubernetes-master:~$ kubectl expose deploy nginx --port 80
--target-port 80 --type NodePort
```

```
ubuntu@ip-172-31-82-86:~$ kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort
service/nginx exposed
ubuntu@ip-172-31-82-86:~$
```

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 services**

```
ubuntu@ip-172-31-82-86:~$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes    ClusterIP     10.96.0.1     <none>         443/TCP    26m
nginx         NodePort      10.107.160.28 <none>         80:31810/TCP 71s
ubuntu@ip-172-31-82-86:~$
```

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) then use the following Command:

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

```
ubuntu@ip-172-31-82-86:~$ kubectl scale --current-replicas=1 --replicas=2 deployment/nginx
deployment.apps/nginx scaled
```

```
kubernetes-master:~$ kubectl get pods
```

```
ubuntu@ip-172-31-82-86:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-76d6c9b8c-ksws8              1/1     Running   0           7m31s
nginx-76d6c9b8c-pbs8d              1/1     Running   0           57s
ubuntu@ip-172-31-82-86:~$
```

```
kubernetes-master:~$ kubectl describe deployment/nginx
```

```

ubuntu@ip-172-31-82-86:~$ kubectl describe deployment/nginx
Name: nginx
Namespace: default
CreationTimestamp: Sun, 18 Sep 2022 08:57:30 +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  8m46s         deployment-controller Scaled up replica set nginx-76d6c9b8c to 1
    Normal          ScalingReplicaSet  2m12s         deployment-controller Scaled up replica set nginx-76d6c9b8c to 2 from 1
ubuntu@ip-172-31-82-86:~$

```

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
```

```

ubuntu@ip-172-31-82-86:~$ kubectl delete service nginx
service "nginx" deleted

```

```

ubuntu@ip-172-31-82-86:~$ kubectl get services
NAME         TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP     10.96.0.1    <none>        443/TCP    35m

```

Then delete the deployment:

```
kubernetes-master:~$ kubectl delete deployment nginx
```

```

ubuntu@ip-172-31-82-86:~$ kubectl delete deployment nginx
deployment.apps "nginx" deleted

```

Run the following to confirm that this worked:

```
kubernetes-master:~$ kubectl get deployments
```

```

ubuntu@ip-172-31-82-86:~$ kubectl get deployments
No resources found in default namespace.

```

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**

kubernetes-master:~\$ kubectl get nodes

```
ubuntu@ip-172-31-82-86:~$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
master-node   Ready     control-plane   37m   v1.25.1
worker-01     Ready     <none>         21m   v1.25.1
worker-02     Ready     <none>         19m   v1.25.1
ubuntu@ip-172-31-82-86:~$ kubectl drain worker-01
node/worker-01 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-n8sqv, kube-system/kube-proxy-vmjxz, 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-n8sqv, kube-system/kube-proxy-vmjxz
ubuntu@ip-172-31-82-86:~$ kubectl drain worker-02
node/worker-02 cordoned
error: unable to drain node "worker-02" due to error:cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore): kube-flannel/kube-flannel-ds-l2gsm, kube-system/kube-proxy-k9wl8, continuing command...
There are pending nodes to be drained:
  worker-02
cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore): kube-flannel/kube-flannel-ds-l2gsm, kube-system/kube-proxy-k9wl8
ubuntu@ip-172-31-82-86:~$ kubectl delete node worker-01
node "worker-01" deleted
ubuntu@ip-172-31-82-86:~$ kubectl delete node worker-02
node "worker-02" deleted
ubuntu@ip-172-31-82-86:~$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
master-node   Ready     control-plane   38m   v1.25.1
```

All Worker Nodes got deleted

Conclusion:

In this experiment, we learned how to install kubectl and execute kubectl commands to manage the Kubernetes cluster and deploy Kubernetes application.