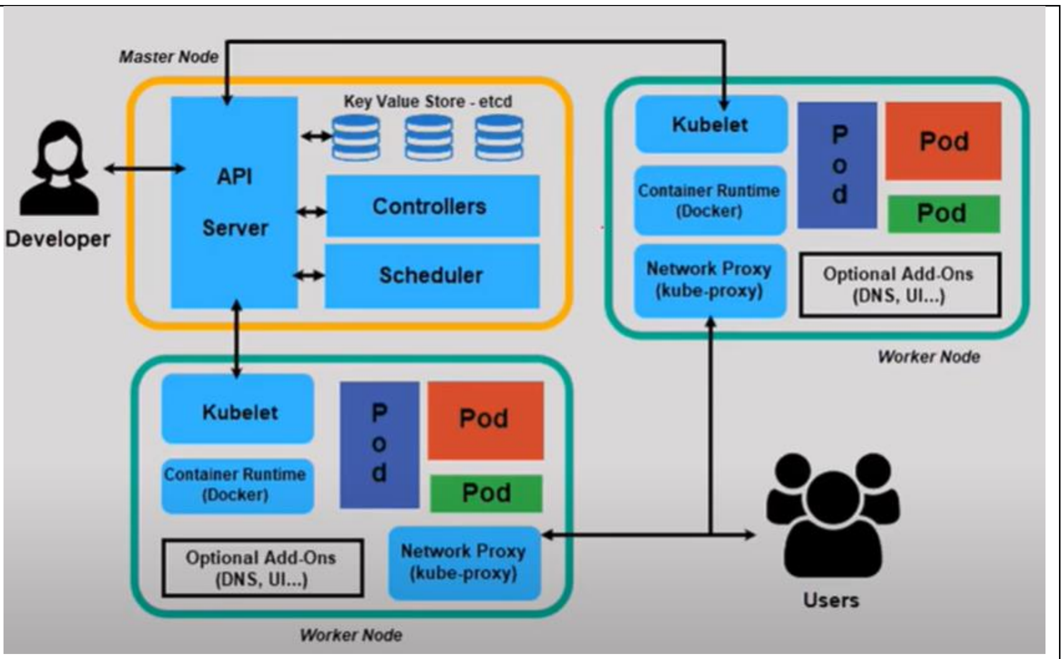
VIT Vidyalankar Institute of Technology Accredited A+ by NAAC	*DEPARTMENT OF INFORMATION TECHNOLOGY*

Semester	T.E. Semester V – Information Technology
Subject	Advance DevOps Lab
Subject Professor In-charge	Prof. Indu Anoop
Laboratory	(Leave blank for now)

Student Name	Rahul Chougule	
Roll Number	20101A0055	
Grade and Subject Teacher's Signature		

Experiment	4	
Problem Statement	To install kubectl and execute kubectl commands to manage the Kubernetes cluster and deploy your first Kubernetes Application.	
Resources / Apparatus Required	Hardware: Computer System (Internet Connectivity)	Software: Web Browser
Details	**Theory:** Kubernetes led by google is an open-source platform for managing container technologies such as Docker. Docker lets you create containers for a pre-configured image and application. *Kubernetes [Greek for "Pilot"] provides the next step, allowing you to balance loads between containers and run multiple containers across multiple systems.*	

	 Master Node Key Value Store - etcd API Server Controllers Scheduler Developer Kubelet Container Runtime (Docker) Network Proxy (kube-proxy) P o d Pod Pod Optional Add-Ons (DNS, UI...) Worker Node Kubelet Container Runtime (Docker) P o d Pod Pod Optional Add-Ons (DNS, UI...) Network Proxy (kube-proxy) Worker Node Users **Container:** Provides an isolated context in which an app together with it's environment (supporting structure eg: web server) can run. **Pods:** Represents a runnable unit usually consisting of a single container. [May contain more containers if containers are tightly coupled] Kubernetes connects the pod to the n/w and rest of the Kubernetes eco-system.
Code	Prerequisite: 2 AWS instance (virtual servers-ubuntu 20) one acting as Master Node and Other as Worker Node. Docker and Kubernetes installation done on both nodes. https://mobaxterm.mobatek.net/download.html Now that your cluster is verified successfully, let's schedule an example Nginx application on the cluster. **SECTION D: Running An Application on the Cluster** 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).

Step 1: Create deployment named nginx [on master]

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

```
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:

```
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:

```
kubectl get services
```

This will output text like 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 worker node switch to worker01

```
docker ps
```

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

SECTION E: Scale up replicas for a deployment

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

```
kubectl scale --current-replicas=1 --replicas=2 deployment/nginx
kubectl get pods
```

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

```
kubectl describe deployment/nginx
```

Output: give details about the service deployed

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

```
kubectl delete service nginx
```

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

```
kubectl get services
```

You will see the following output:
Output

```
NAME       TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
kubernetes ClusterIP 10.96.0.1    <none>       443/TCP   1d
```

Then delete the deployment:

```
kubectl delete deployment nginx
```

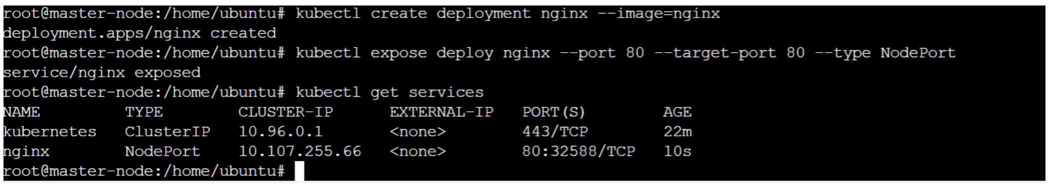
Run the following to confirm that this worked:

```
kubectl get deployments
```

Output
No resources found.

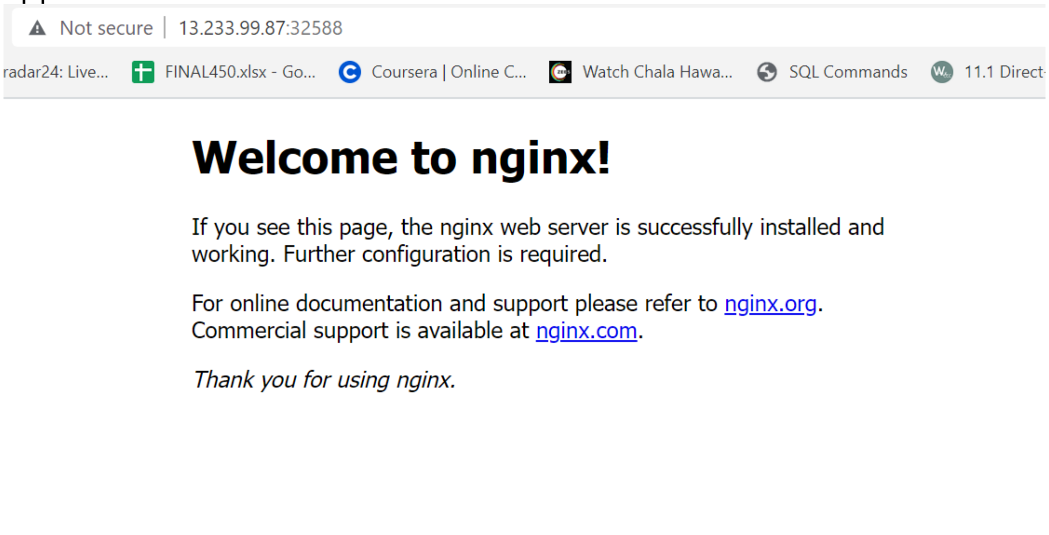
Output

On master node:



```
root@master-node:/home/ubuntu# kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
root@master-node:/home/ubuntu# kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort
service/nginx exposed
root@master-node:/home/ubuntu# kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        22m
nginx        NodePort    10.107.255.66   <none>        80:32588/TCP   10s
root@master-node:/home/ubuntu#
```

Access of worker node ip via browser to see successfully deployed application:



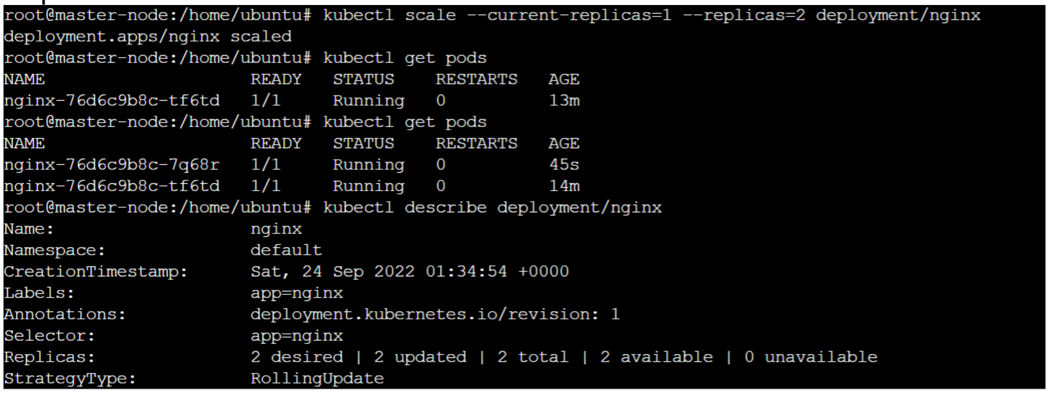
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

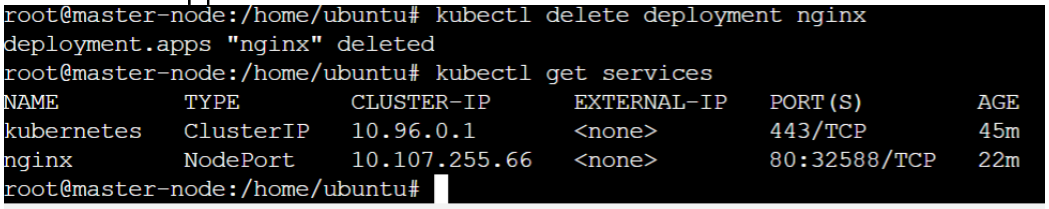
Thank you for using nginx.

Replication of Pods:



```
root@master-node:/home/ubuntu# kubectl scale --current-replicas=1 --replicas=2 deployment/nginx
deployment.apps/nginx scaled
root@master-node:/home/ubuntu# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-76d6c9b8c-tf6td    1/1     Running   0          13m
root@master-node:/home/ubuntu# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-76d6c9b8c-7q68r    1/1     Running   0          45s
nginx-76d6c9b8c-tf6td    1/1     Running   0          14m
root@master-node:/home/ubuntu# kubectl describe deployment/nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Sat, 24 Sep 2022 01:34:54 +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
```

Deletion of application:



```
root@master-node:/home/ubuntu# kubectl delete deployment nginx
deployment.apps "nginx" deleted
root@master-node:/home/ubuntu# kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        45m
nginx        NodePort    10.107.255.66   <none>        80:32588/TCP   22m
root@master-node:/home/ubuntu#
```

Conclusion

Executed kubectl commands to manage the Kubernetes cluster and deploy a nginx Application.