Name: Manav Jawrani

Roll No.: 19

Subject: Advanced DevOps

Experiment No.: 4

Experiment 4.

Aim: To install Kube ctl and execute kubectl commands to manage the Kubernetes cluster and deploy your First Kubernetes Application.

Theory:

- What is Kube ctl?
  Kubectl is a Kubernetes command-line tool which allows us to run commands against Kubernetes clusters. we can use kube ctl to deploy applications, inspect and manage cluster resources and view logs.

- Features of kube ctl:

1. Communication between Nodes and control plane i.e. master node.
2. user name spaces
3. It allows to customize the outputs
4. It may invoke several API calls to API server to build a view for user

Here we had created our application using nginx web server

- What is nginx?
  NGINX is open source Software for web serving,

reverse proxying ,caching ,load balancing, media streaming and more. It started out as a web server designed for maximum performance and stability.

Deployed our application using kubernetes deployment

- What is kubernetes deployment?
A Kubernetes deployment tells Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can help to efficiently 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. Kubernetes deployments are completed using Kubectl command-line tool.

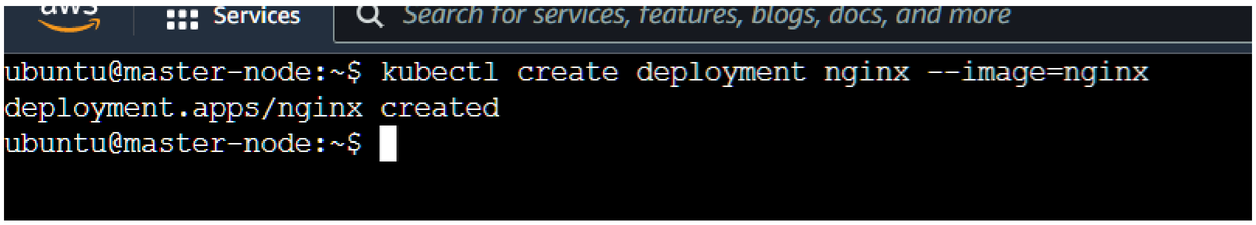
Implementation:

****Note: As we have created master and worker nodes and created a kubernetes cluster as well in our previous experiment 3 so will use that only.****

Running An Application on the Cluster

Step 1: As the cluster is up and running, we can deploy our nginx server on this cluster. Apply this deployment file using this command to create a deployment.

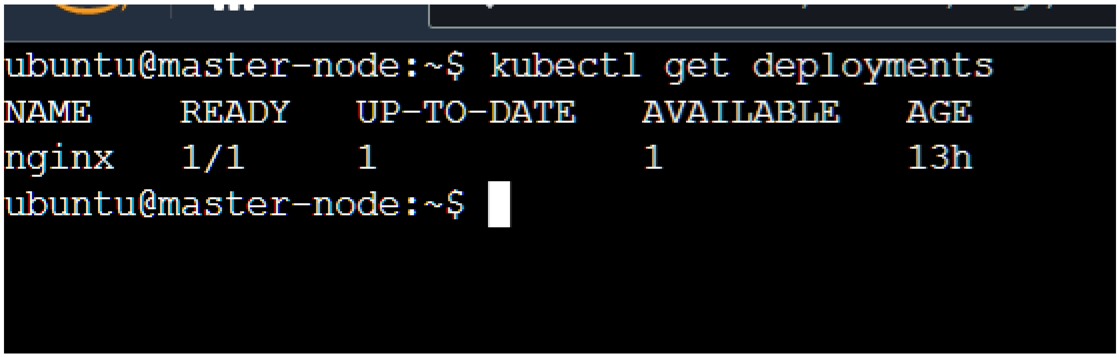
$kubectl create deployment nginx --image=nginx



```
ubuntu@master-node:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
ubuntu@master-node:~$
```

Step 2: Verify the deployment using the command:

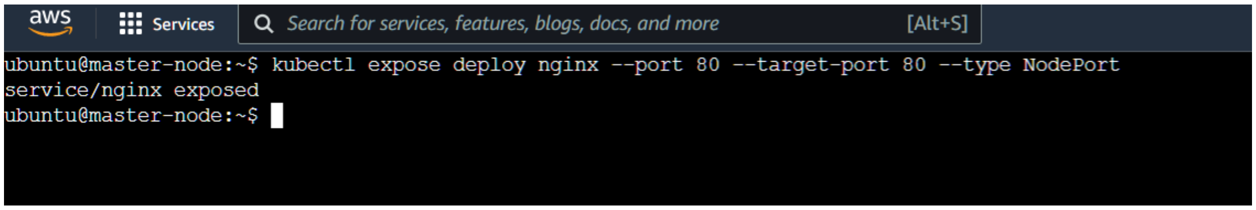
$kubectl get deployments



```
ubuntu@master-node:~$ kubectl get deployments
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx    1/1     1            1           13h
ubuntu@master-node:~$
```

Step 3: 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 with this service-type, Kubernetes will assign this service on ports on the **30000+** range.

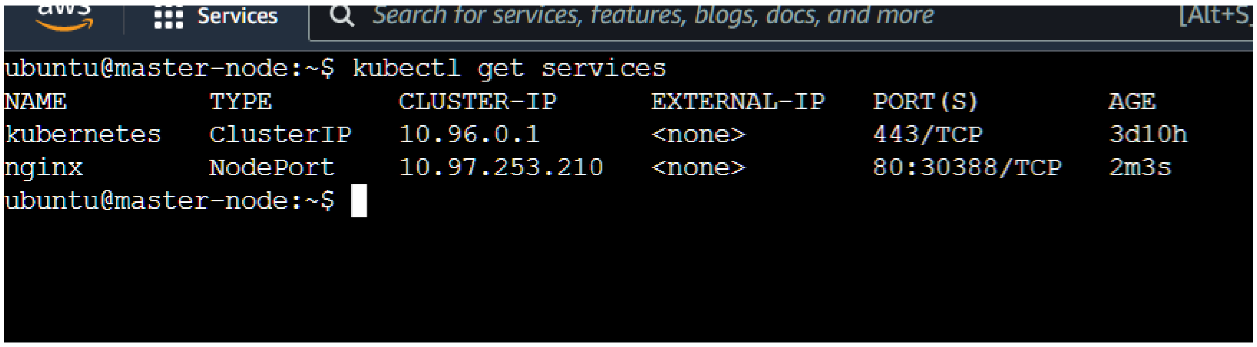
$kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort



```
ubuntu@master-node:~$ kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort
service/nginx exposed
ubuntu@master-node:~$
```

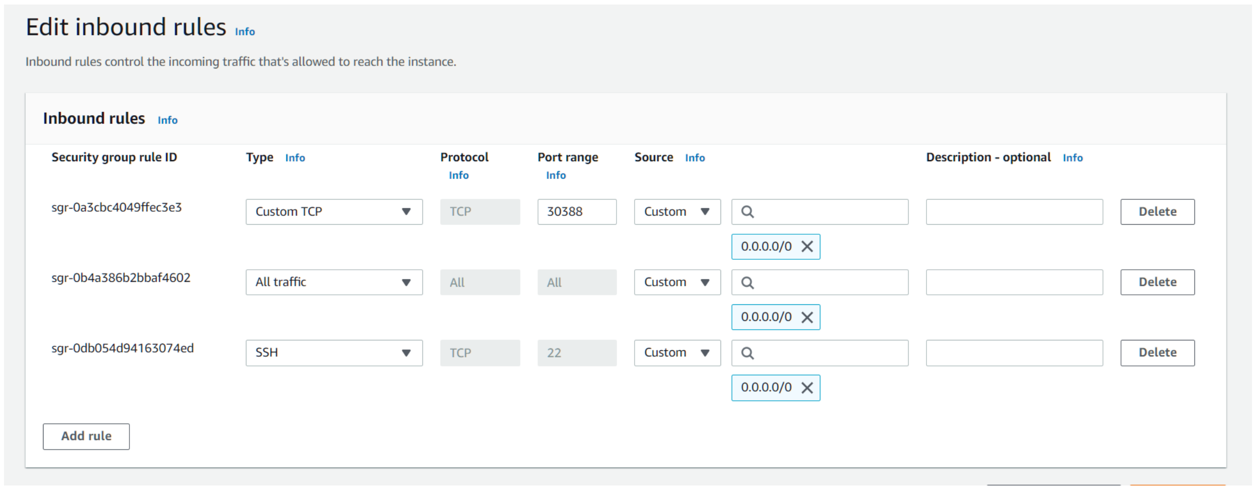
Step 4: Run this command to see a summary of the service and the ports exposed.

$kubectl get services



```
ubuntu@master-node:~$ kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        3d10h
nginx        NodePort    10.97.253.210   <none>        80:30388/TCP   2m3s
ubuntu@master-node:~$
```

Step 5: Add the port which is displayed i.e. 30388 (in our case) in the inbound rules of the security group.



Edit inbound rules Info

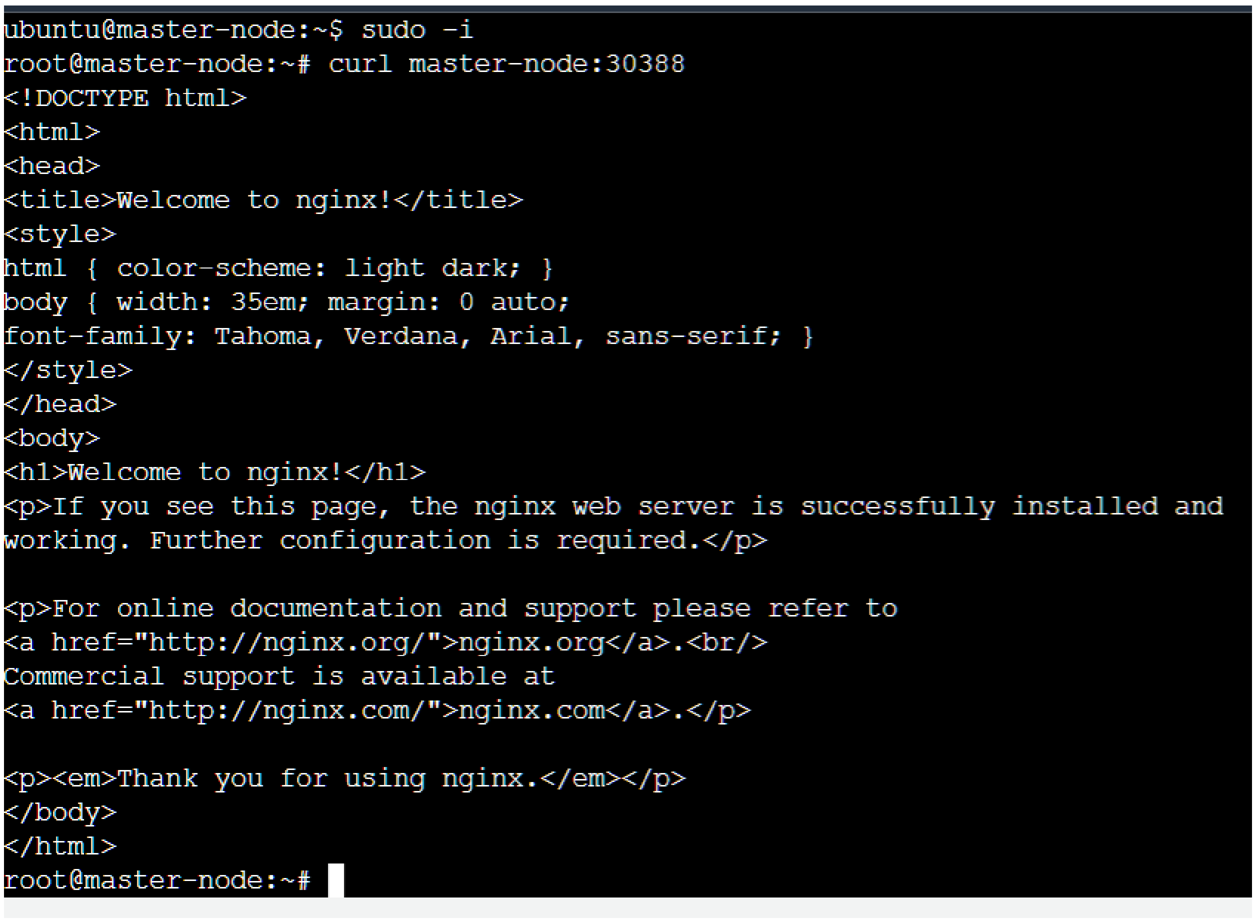
Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-0a3cbc4049ffec3e3	Custom TCP	TCP	30388	Custom	0.0.0.0/0		Delete
sgr-0b4a386b2bbaf4602	All traffic	All	All	Custom	0.0.0.0/0		Delete
sgr-0db054d94163074ed	SSH	TCP	22	Custom	0.0.0.0/0		Delete

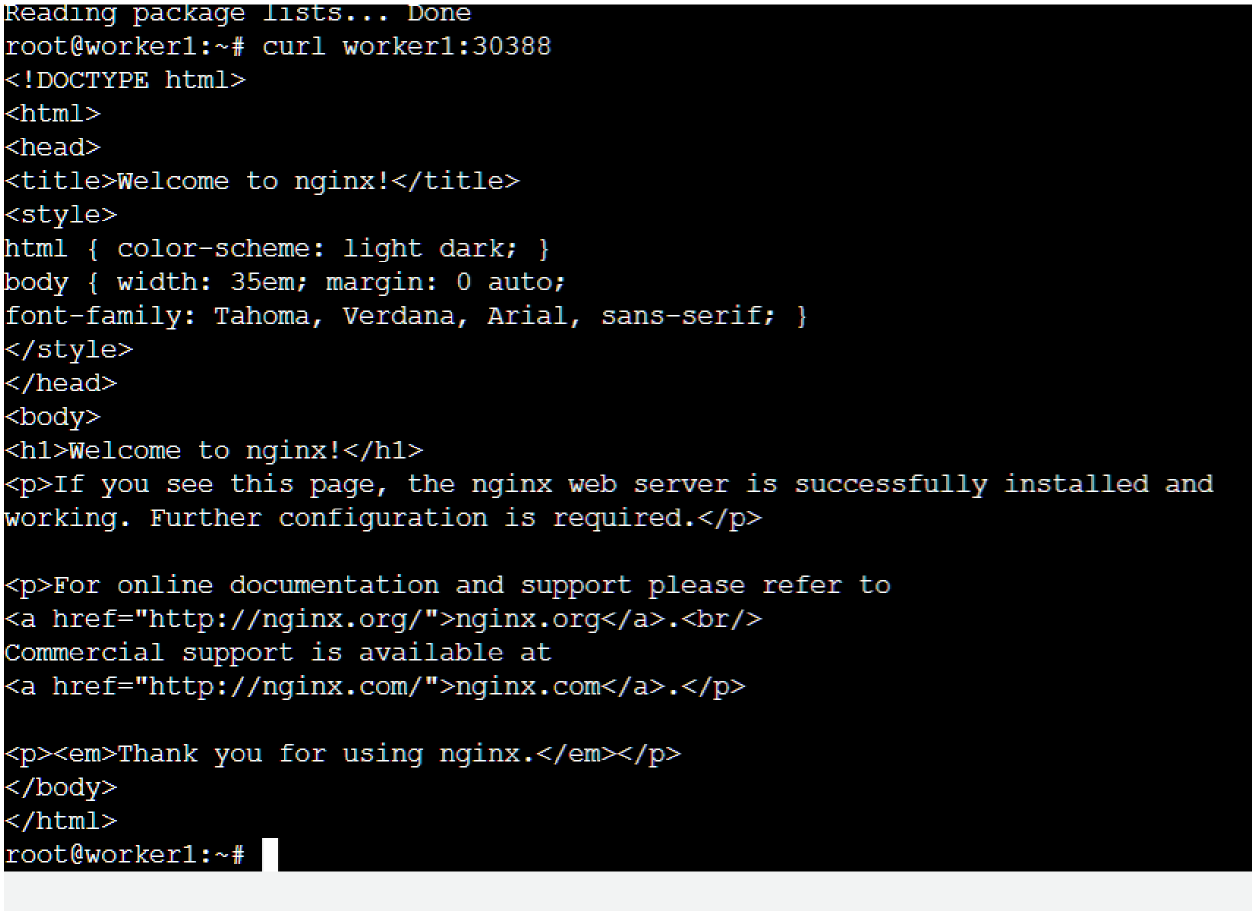
Step 6: Now you can verify that the Nginx page is reachable on all nodes using the curl command.

Master



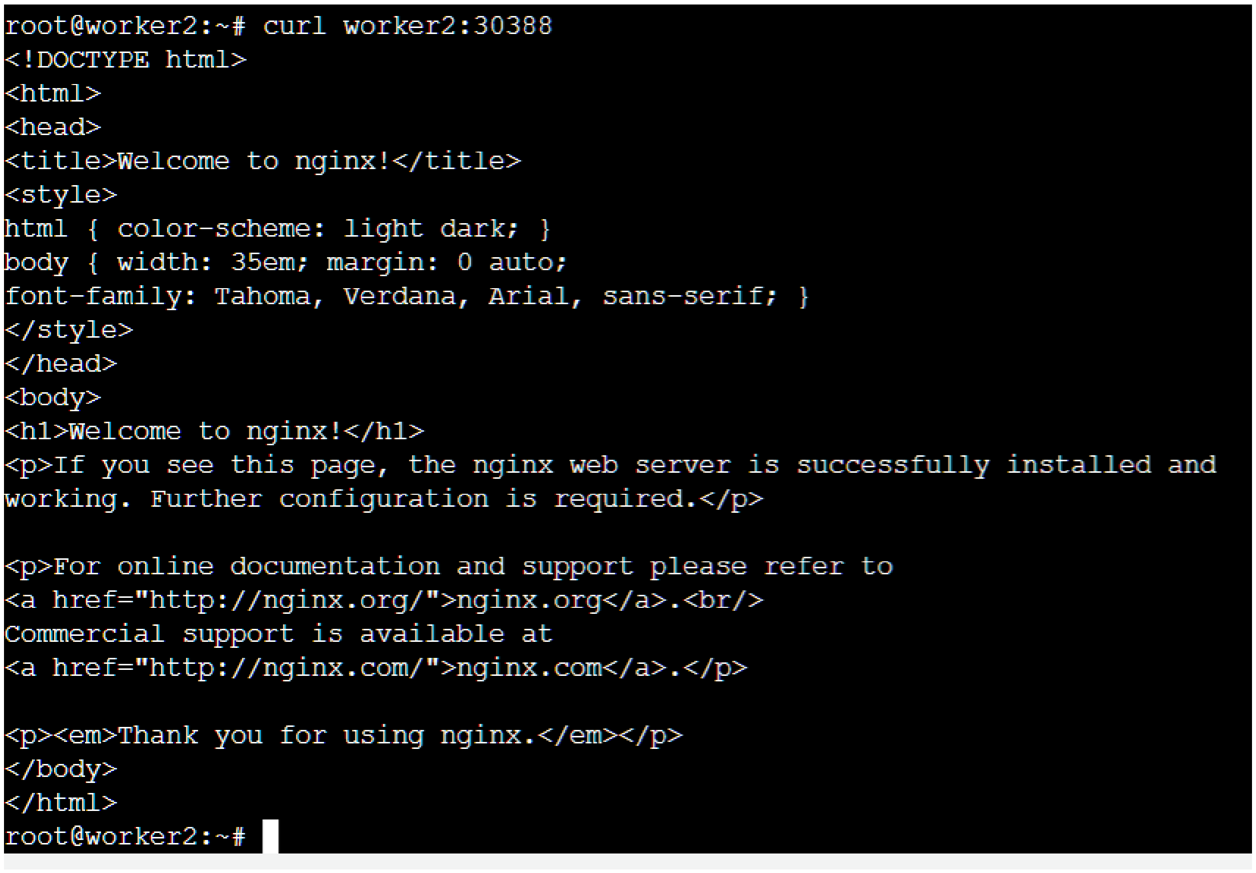
```
ubuntu@master-node:~$ sudo -i
root@master-node:~# curl master-node:30388
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@master-node:~#
```

Worker 1



```
Reading package lists... Done
root@worker1:~# curl worker1:30388
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@worker1:~#
```

Worker 2

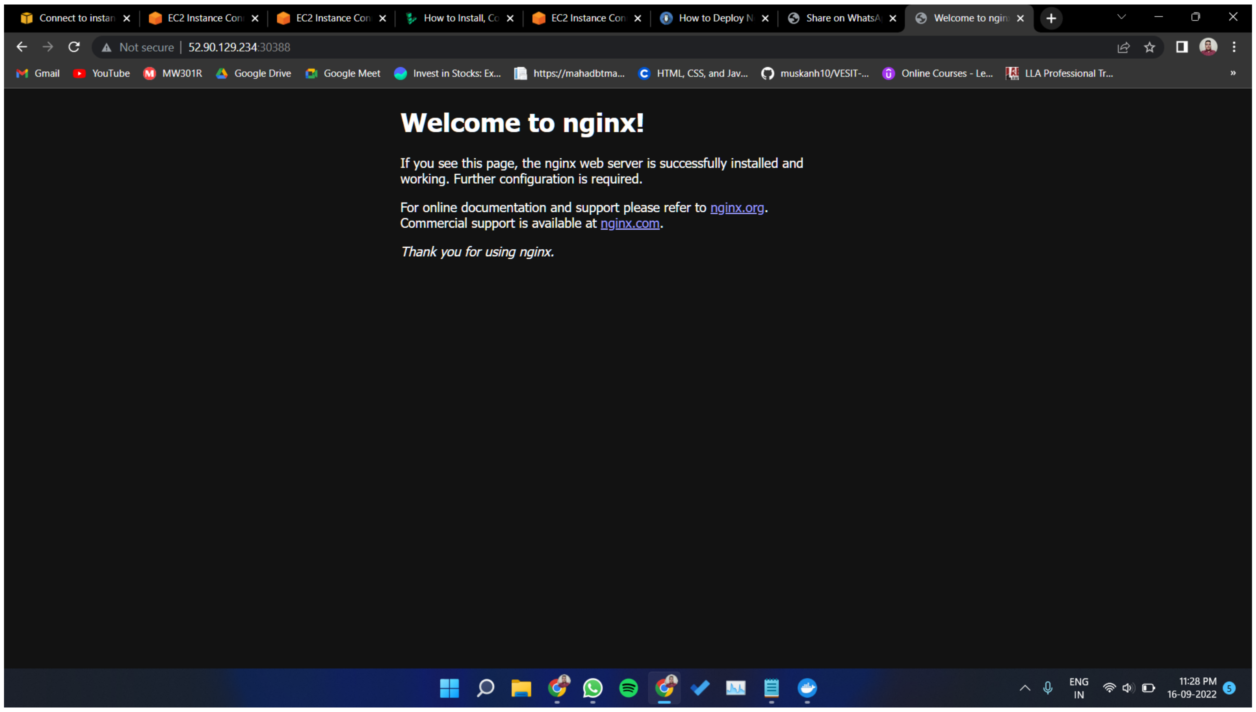


```
root@worker2:~# curl worker2:30388
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@worker2:~#
```

As you can see, the “**WELCOME TO NGINX!**” page can be reached.

Step 7: 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.

Worker 1: http://52.90.129.234:30388



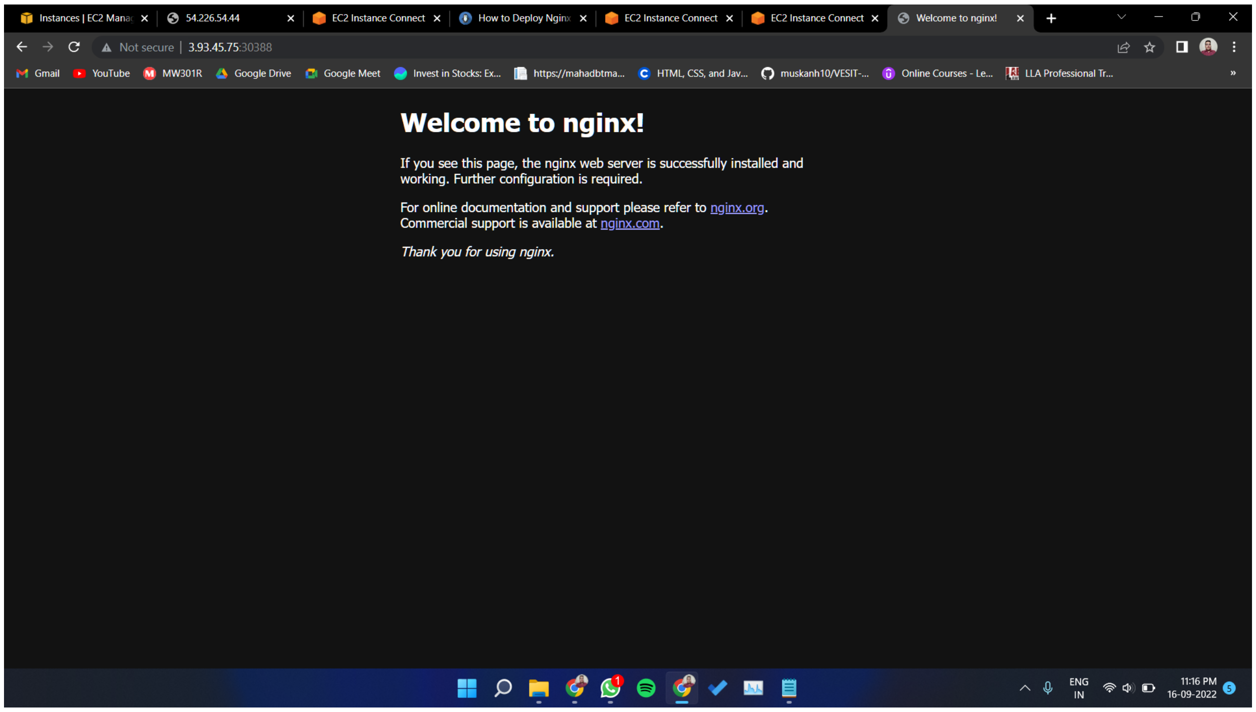
Welcome to nginx!

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.

Worker 2: http://3.93.45.75:30388



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.

****Note: IPs of the machines will change again and again as soon as you stop the instance and start it again.****

Conclusion :

In this experiment, we have learned how to deploy an application on a Kubernetes cluster using the Kubectl command line and Kubernetes deployment, and how to access the application deployment via the cluster's worker nodes.