

# Kubernetes

## Day3

### Lab 1

#### Requirements

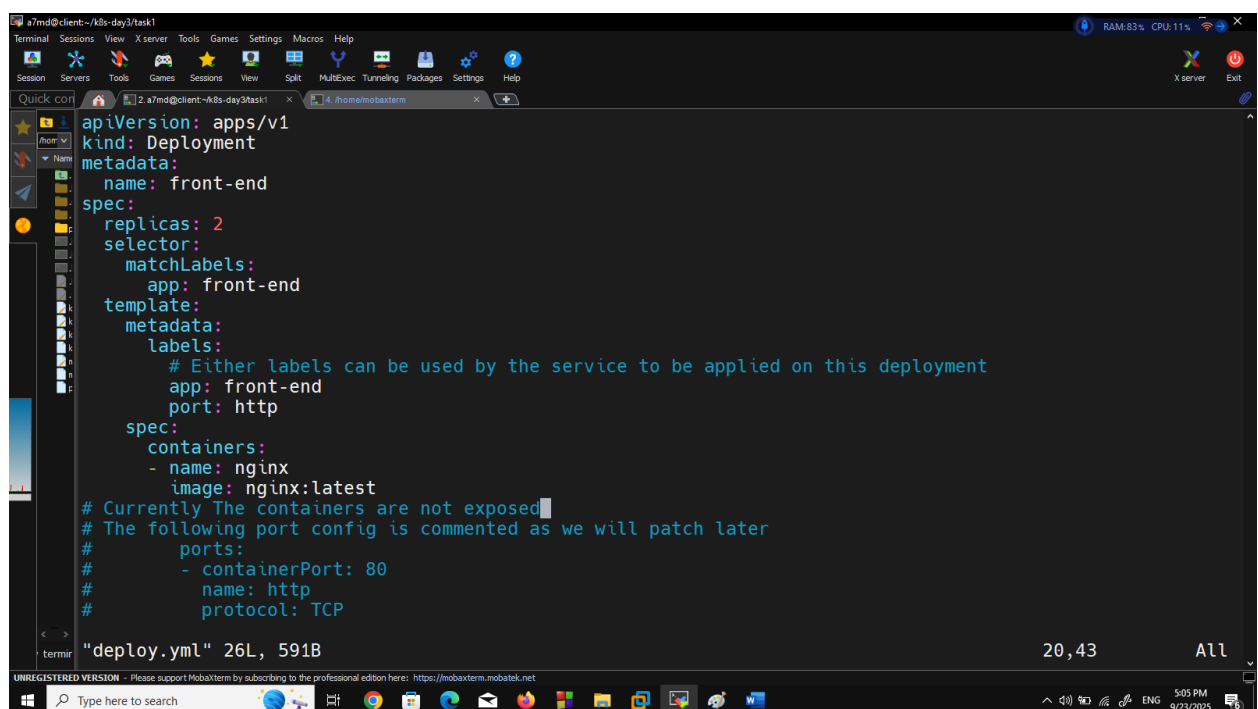
Reconfigure the existing deployment front-end and add a port specification named http exposing port 80/tcp of the existing container nginx.

Create a new service named front-end-svc exposing the container port http.

Configure the new service to also expose the individual Pods via a NodePort on the nodes on which they are scheduled.

#### Solutions

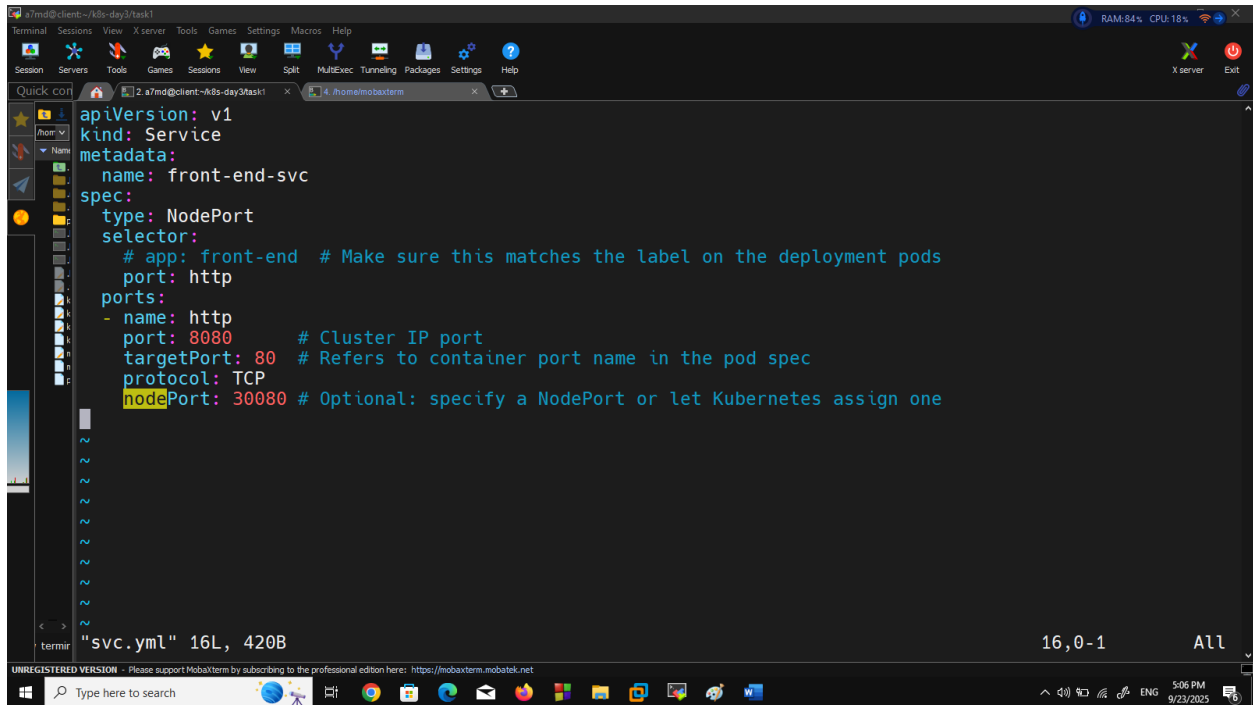
1- Yaml file for the deployment:



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: front-end
spec:
  replicas: 2
  selector:
    matchLabels:
      app: front-end
  template:
    metadata:
      labels:
        # Either labels can be used by the service to be applied on this deployment
        app: front-end
        port: http
    spec:
      containers:
        - name: nginx
          image: nginx:latest
# Currently The containers are not exposed
# The following port config is commented as we will patch later
#
#   ports:
#     - containerPort: 80
#       name: http
#       protocol: TCP
```

The screenshot shows a MobaXterm terminal window with a file explorer on the left displaying the contents of 'deploy.yml'. The terminal window title is 'a7md@client-~/k8s-day3/task1'. The status bar at the bottom indicates 'UNREGISTERED VERSION' and provides a link to the professional edition. The system tray shows the date and time as 9:05 PM on 9/23/2023.

## 2- Yaml file for the service that exposes the containers:



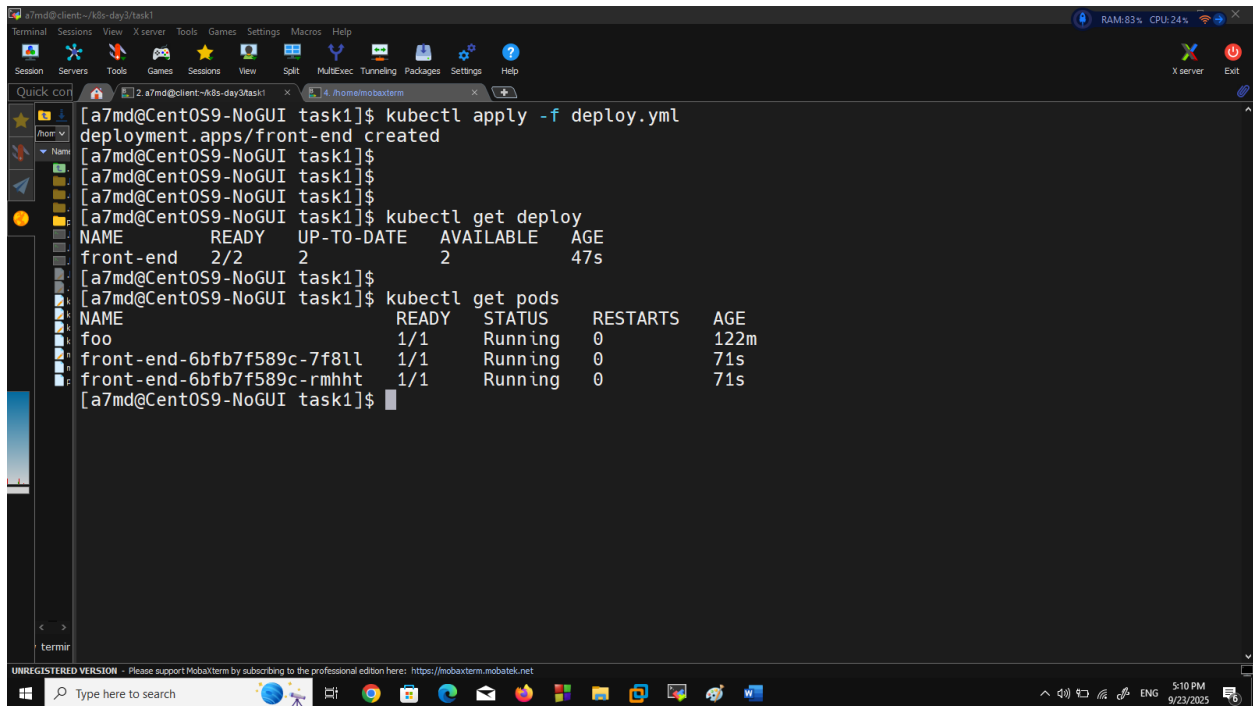
The screenshot shows a MobaXterm terminal window with a dark theme. The terminal displays a Kubernetes service YAML configuration. The configuration includes the API version, kind, metadata (name: front-end-svc), and a spec section. The spec defines a NodePort type service with a selector for 'app: front-end' and a port named 'http'. It also lists the ports, including a NodePort of 30080. Comments explain the purpose of each field, such as matching the deployment pods and referring to the container port name. The terminal status bar at the bottom indicates the file size is 16L, 420B and the session is 16,0-1. The Windows taskbar is visible at the bottom of the screen.

```
apiVersion: v1
kind: Service
metadata:
  name: front-end-svc
spec:
  type: NodePort
  selector:
    # app: front-end # Make sure this matches the label on the deployment pods
    port: http
  ports:
    - name: http
      port: 8080 # Cluster IP port
      targetPort: 80 # Refers to container port name in the pod spec
      protocol: TCP
      nodePort: 30080 # Optional: specify a NodePort or let Kubernetes assign one
```

"svc.yml" 16L, 420B 16,0-1 All

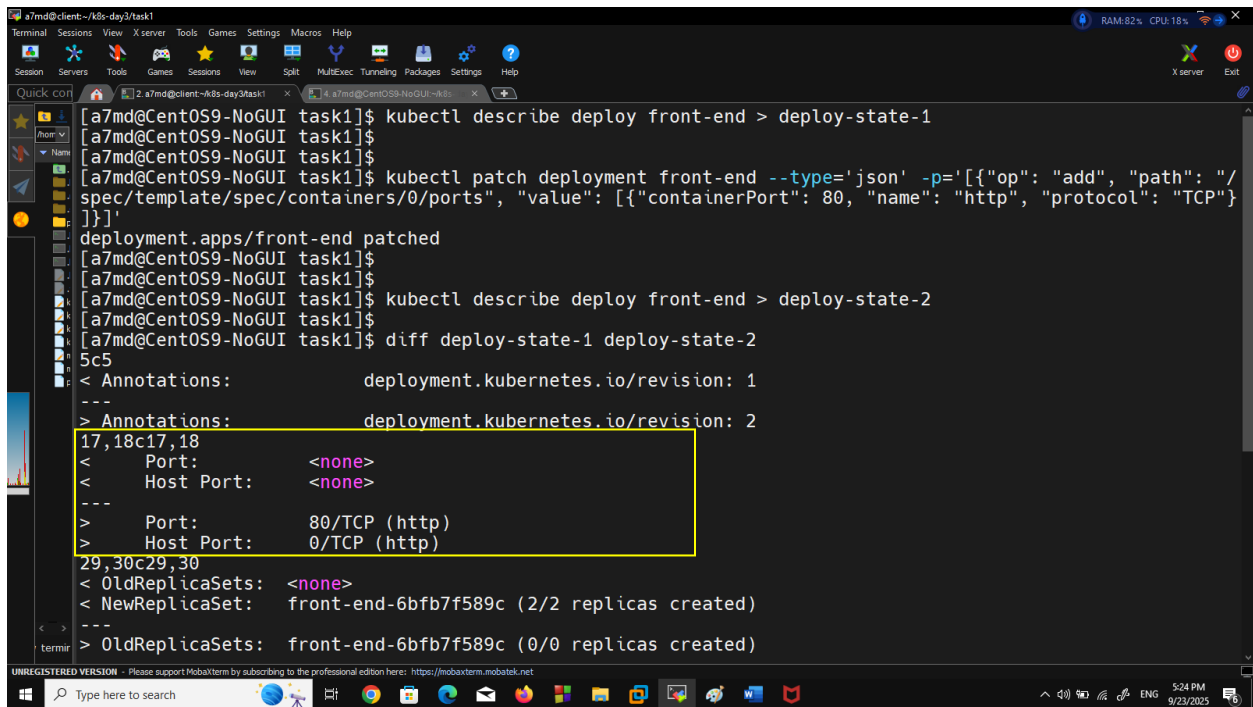
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

### 3- Patching the port configuration to the container's configuration:



```
[a7md@CentOS9-NoGUI task1]$ kubectl apply -f deploy.yml
deployment.apps/front-end created
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ kubectl get deploy
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
front-end  2/2     2             2           47s
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
foo                                  1/1     Running   0           122m
front-end-6bfb7f589c-7f8ll          1/1     Running   0           71s
front-end-6bfb7f589c-rmhht          1/1     Running   0           71s
[a7md@CentOS9-NoGUI task1]$
```

Figure 1: Start the deploy



```
[a7md@CentOS9-NoGUI task1]$ kubectl describe deploy front-end > deploy-state-1
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ kubectl patch deployment front-end --type=json -p='[{"op": "add", "path": "/spec/template/spec/containers/0/ports", "value": [{"containerPort": 80, "name": "http", "protocol": "TCP"}]}'
deployment.apps/front-end patched
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ kubectl describe deploy front-end > deploy-state-2
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ diff deploy-state-1 deploy-state-2
5c5
< Annotations:          deployment.kubernetes.io/revision: 1
---
> Annotations:          deployment.kubernetes.io/revision: 2
17,18c17,18
<   Port:               <none>
<   Host Port:          <none>
---
>   Port:               80/TCP (http)
>   Host Port:          0/TCP (http)
29,30c29,30
< OldReplicaSets:      <none>
< NewReplicaSet:       front-end-6bfb7f589c (2/2 replicas created)
---
> OldReplicaSets:      front-end-6bfb7f589c (0/0 replicas created)
```

Figure 2: Inspecting the change in the port configurations of the deployment before and after patching

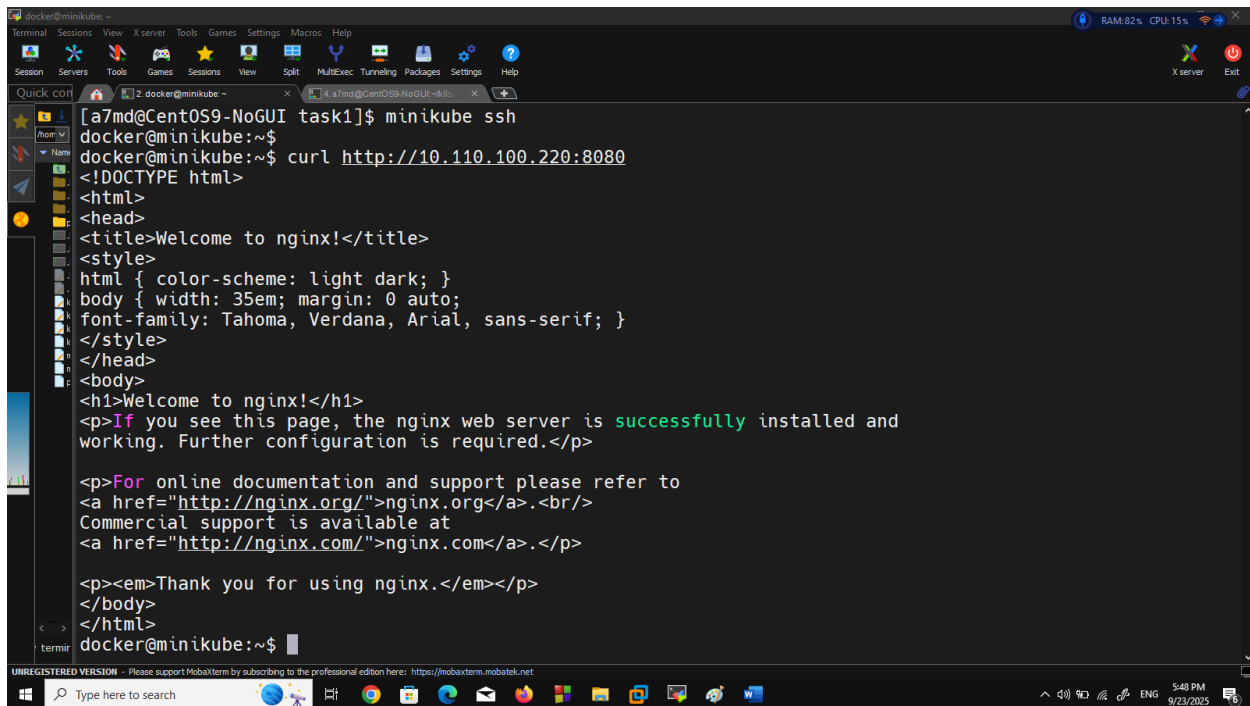
```
a7md@client-~k8s-day3/task1
Terminal Sessions View X server Tools Games Settings Macros Help
[Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help]
Quick con
[a7md@CentOS9-NoGUI task1]$ tail deploy-state-2
OldReplicaSets: front-end-6bfb7f589c (0/0 replicas created)
NewReplicaSet: front-end-75ffbb6998 (2/2 replicas created)
Events:
  Type      Reason          Age    From                      Message
  ----      -
  Normal    ScalingReplicaSet 6m53s  deployment-controller     Scaled up replica set front-end-6bfb7f589c from 0 to 2
  Normal    ScalingReplicaSet 35s    deployment-controller     Scaled up replica set front-end-75ffbb6998 from 0 to 1
  Normal    ScalingReplicaSet 29s    deployment-controller     Scaled down replica set front-end-6bfb7f589c from 2 to 1
  Normal    ScalingReplicaSet 29s    deployment-controller     Scaled up replica set front-end-75ffbb6998 from 1 to 2
  Normal    ScalingReplicaSet 22s    deployment-controller     Scaled down replica set front-end-6bfb7f589c from 2 to 0
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
foo                                  1/1     Running   0           138m
front-end-75ffbb6998-j2q4f          1/1     Running   0           3m1s
front-end-75ffbb6998-sdrnj          1/1     Running   0           3m7s
[a7md@CentOS9-NoGUI task1]$
```

Figure 3: Notice also that the deployment has created new pods with the new port configurations, instead of the old ones

#### 4- Applying and running the service:

```
a7md@client-~k8s-day3/task1
Terminal Sessions View X server Tools Games Settings Macros Help
[Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help]
Quick con
[a7md@CentOS9-NoGUI task1]$ kubectl apply -f svc.yml
service/front-end-svc created
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ kubectl get svc
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
front-end-svc  NodePort    10.110.100.220  <none>       8080:30080/TCP   20s
kubernetes   ClusterIP   10.96.0.1       <none>       443/TCP          46h
[a7md@CentOS9-NoGUI task1]$
```

## 5- Accessing the web service from the cluster:

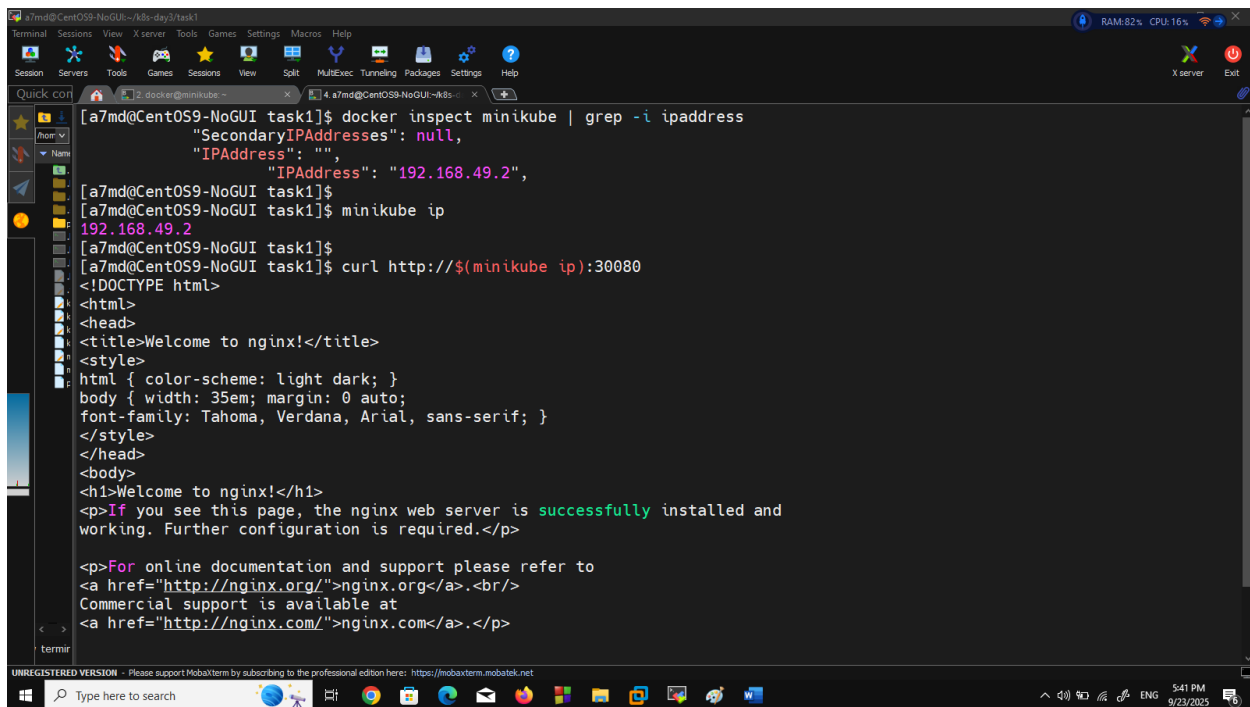


```
[a7md@CentOS9-NoGUI task1]$ minikube ssh
docker@minikube:~$
docker@minikube:~$ curl http://10.110.100.220:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</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>
docker@minikube:~$
```

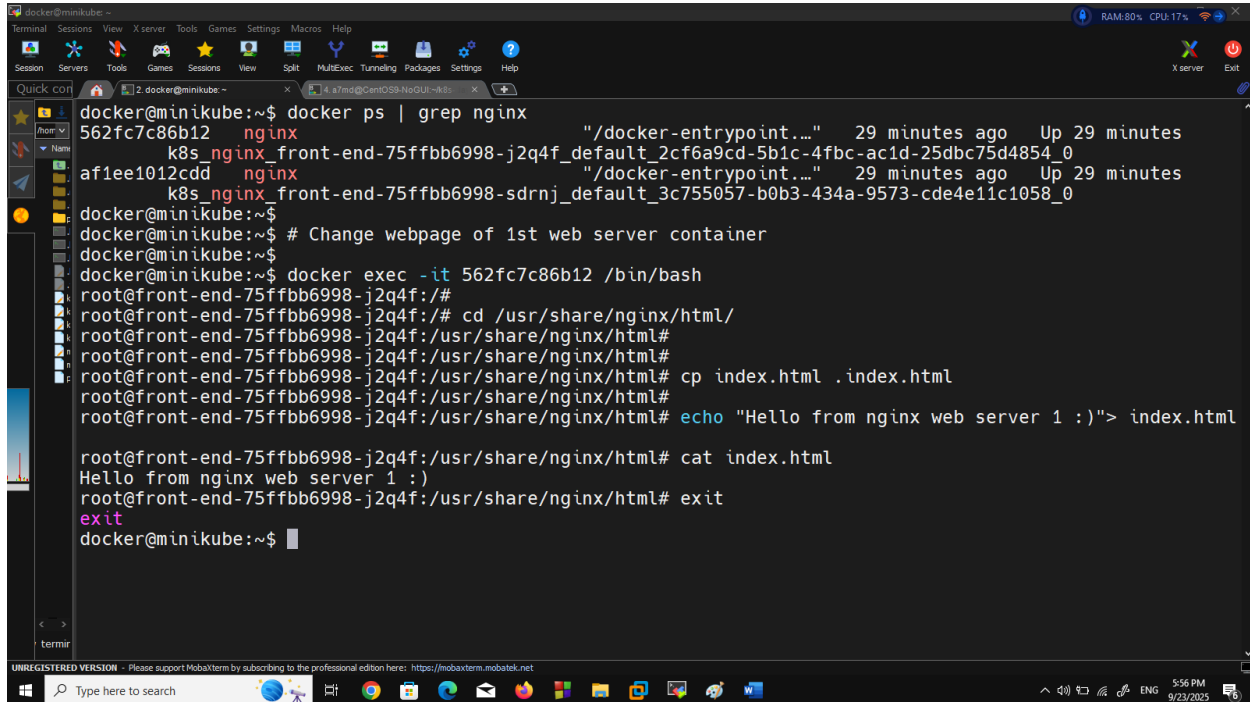
## 6- Accessing the web service from the host machine:



```
[a7md@CentOS9-NoGUI task1]$ docker inspect minikube | grep -i ipaddress
    "SecondaryIPAddresses": null,
    "IPAddress": "",
    "IPAddress": "192.168.49.2",
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ minikube ip
192.168.49.2
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ curl http://$(minikube ip):30080
<!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>
```

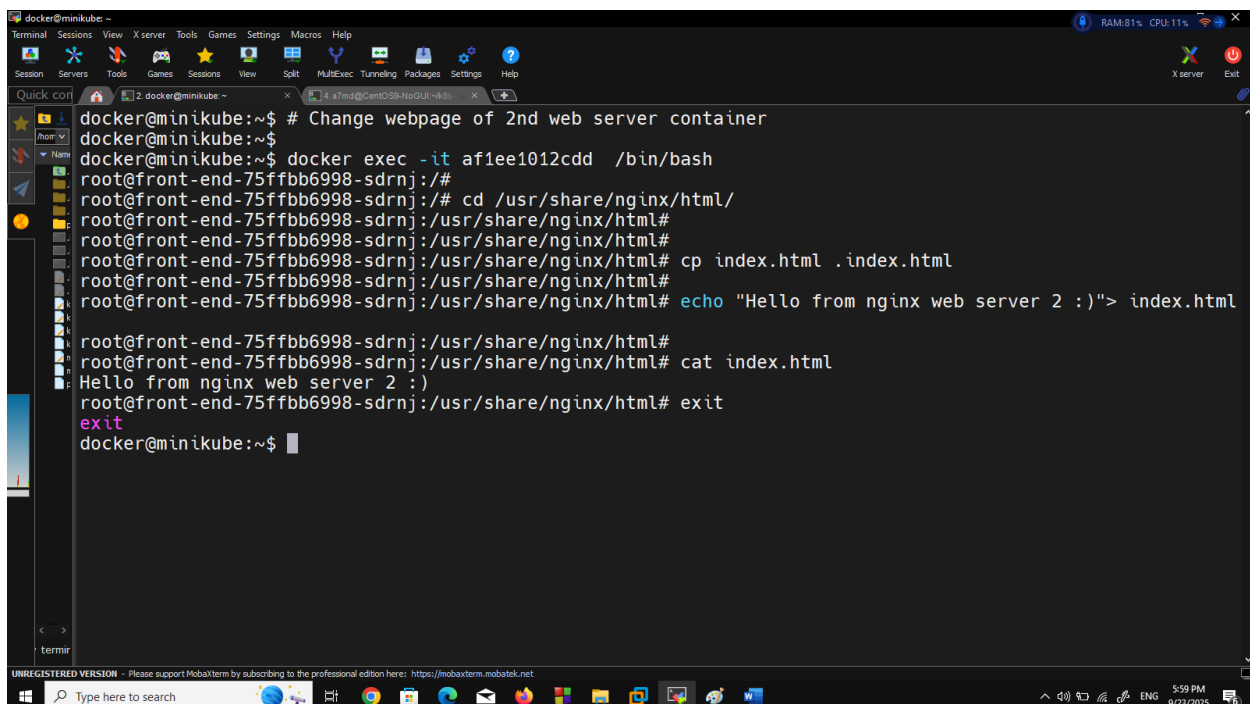
## 7- Which web server relies to the service ?



```
docker@minikube:~$ docker ps | grep nginx
562fc7c86b12      nginx            "/docker-entrypoint..." 29 minutes ago Up 29 minutes
k8s_nginx_front-end-75ffbb6998-j2q4f_default_2cf6a9cd-5b1c-4fbc-ac1d-25dbc75d4854_0
af1ee1012cdd      nginx            "/docker-entrypoint..." 29 minutes ago Up 29 minutes
k8s_nginx_front-end-75ffbb6998-sdrnj_default_3c755057-b0b3-434a-9573-cde4e11c1058_0

docker@minikube:~$ # Change webpage of 1st web server container
docker@minikube:~$ docker exec -it 562fc7c86b12 /bin/bash
root@front-end-75ffbb6998-j2q4f:/#
root@front-end-75ffbb6998-j2q4f:/# cd /usr/share/nginx/html/
root@front-end-75ffbb6998-j2q4f:/usr/share/nginx/html#
root@front-end-75ffbb6998-j2q4f:/usr/share/nginx/html#
root@front-end-75ffbb6998-j2q4f:/usr/share/nginx/html# cp index.html .index.html
root@front-end-75ffbb6998-j2q4f:/usr/share/nginx/html# echo "Hello from nginx web server 1 :)"> index.html
root@front-end-75ffbb6998-j2q4f:/usr/share/nginx/html# cat index.html
Hello from nginx web server 1 :)
root@front-end-75ffbb6998-j2q4f:/usr/share/nginx/html# exit
exit
docker@minikube:~$
```

Figure 4: Change the webpage on 1st nginx container



```
docker@minikube:~$ # Change webpage of 2nd web server container
docker@minikube:~$ docker exec -it af1ee1012cdd /bin/bash
root@front-end-75ffbb6998-sdrnj:/#
root@front-end-75ffbb6998-sdrnj:/# cd /usr/share/nginx/html/
root@front-end-75ffbb6998-sdrnj:/usr/share/nginx/html#
root@front-end-75ffbb6998-sdrnj:/usr/share/nginx/html#
root@front-end-75ffbb6998-sdrnj:/usr/share/nginx/html# cp index.html .index.html
root@front-end-75ffbb6998-sdrnj:/usr/share/nginx/html# echo "Hello from nginx web server 2 :)"> index.html
root@front-end-75ffbb6998-sdrnj:/usr/share/nginx/html#
root@front-end-75ffbb6998-sdrnj:/usr/share/nginx/html# cat index.html
Hello from nginx web server 2 :)
root@front-end-75ffbb6998-sdrnj:/usr/share/nginx/html# exit
exit
docker@minikube:~$
```

Figure 5: Change the webpage on 2nd nginx container

The screenshot shows a MobaXterm terminal window with the following content:

```
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 1 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 1 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 1 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 2 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 2 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 2 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 2 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 1 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 1 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 2 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 2 :)
docker@minikube:~$ curl http://10.110.100.220:8080
Hello from nginx web server 1 :)
docker@minikube:~$ curl http://10.110.100.220:8080
```

The terminal window has a menu bar (Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, Help) and a toolbar. The status bar at the bottom shows "UNREGISTERED VERSION" and a link to the professional edition. The Windows taskbar is visible at the bottom with the search bar and various application icons.

Figure 6: Notice how load-balancing is performed on accessing the deployment pods