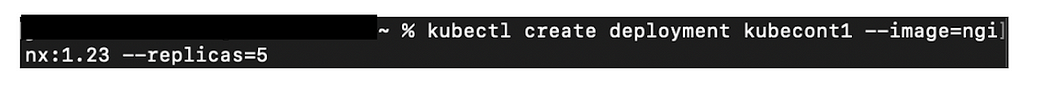
# **How to Deploy Kubernetes pods with a custom HTML on a NGINX web server**

1. Create two deployments. The first deployment contains 5 pods all running a version of NGINX. (I created 2 deployments (kubecont1 & kubecont2), the 2nd deployment kubecont2 is for another portion that may be posted later. Just ignore the 2nd deployment kubecont1 when you see it below. This example is only for 1 deployment kubecont1)
2. Create a service to expose your deployment from the internet.
3. Create a ConfigMap that points to a custom index.html page that contains the line “This web page is housed on a Pod running NGINX”.
4. Ensure you can reach the service from your web browser.

**Step 1.**

Create two deployments. The first deployment contains 5 pods all running a version of NGINX.

#create the deployment  
kubectl create deployment <name the container> --image=nginx:1.23 --replicas=5

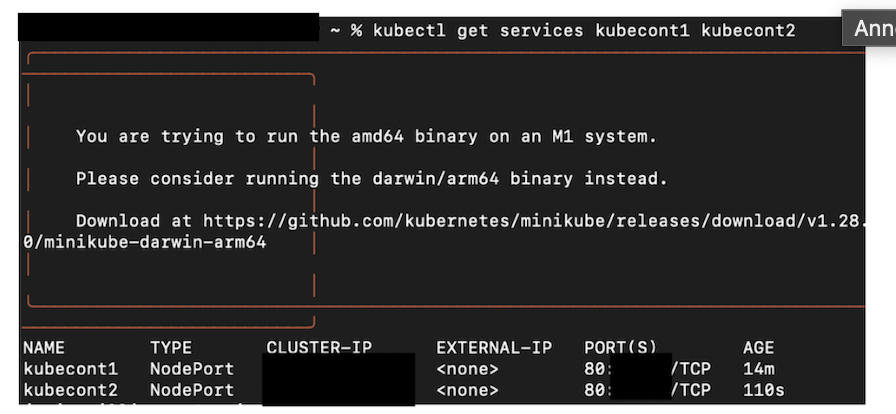




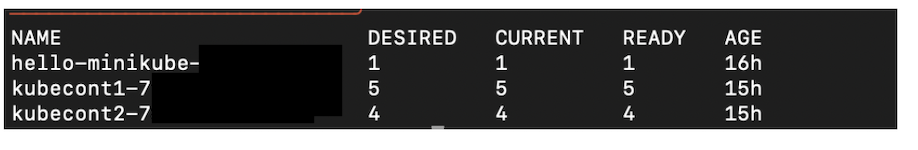
#expose port 80 & publishing type(NodePort)  
kubectl expose deployment kubecont1 --type=NodePort --port=80



#command that displays the description of the 2 deployments  
kubectl get services kubecont1 kubecont2



#command to view the replica sets  
kubectl get rs



**Step 2.**

**Create a service to expose your deployment from the internet**

* In order to understand the changes, you have to see what you’re changing. So we have to DEFINE the NGINX deployment by creating a deployment.yml file.

vi deployment.yml

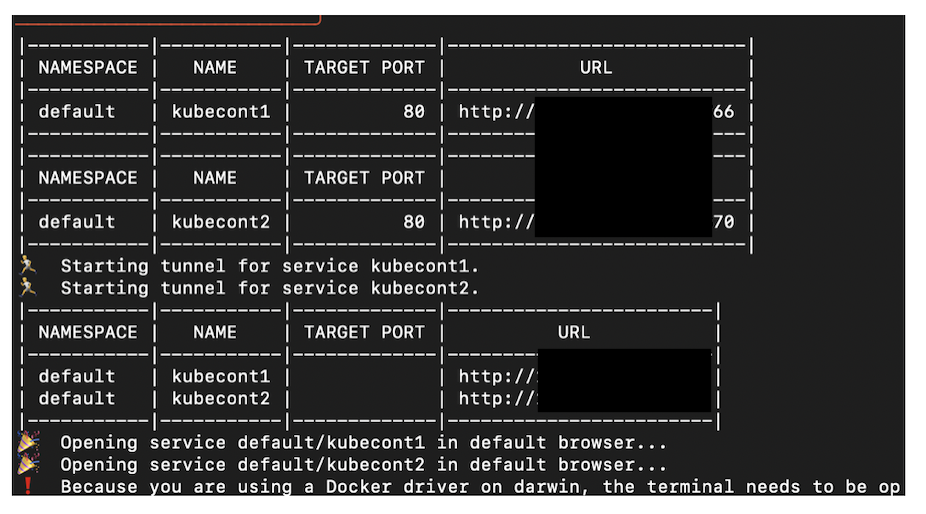
#enter the following to create the .yml file  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: nginx  
 labels:  
 name: nginx  
spec:  
 selector:  
 matchLabels:  
 name: nginx  
 template:  
 metadata:  
 labels:  
 name: nginx  
 spec:  
 containers:  
 - name: nginx  
 image: nginx:1.21.6-alpine  
 ports:  
 - containerPort: 80

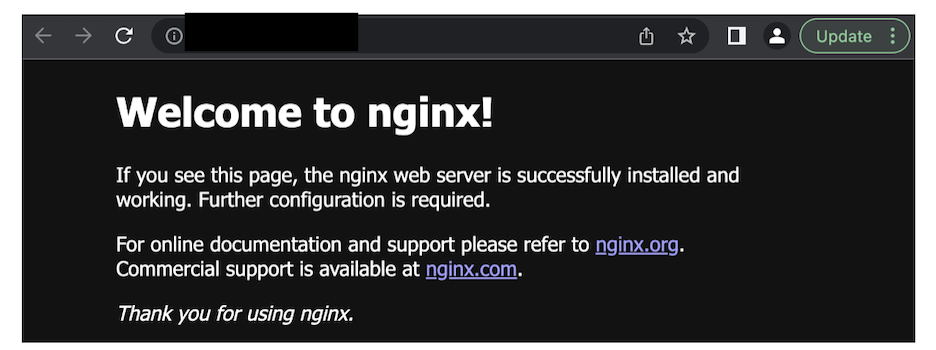
* Hit “esc” then :wq! to save the .yml file.
* Apply the changes to put it all together.

kubectl apply -f deployment.yml

* Test the service to see if IP address will launch a web browser & to make sure port 80 is exposed using the following command. You could also use the “curl” command to view file:

minikube service kubecont1 kubecont2



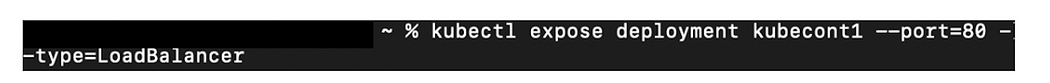


Using the above command will automatically display the NGINX HTML page

**Step 3.**

Create a ConfigMap that points to a custom index.html page that contains the line “This web page is housed on a Pod running NGINX”.

#command to add a loadbalancer & expose the deployment to the internet   
kubectl expose deployment kubecont1 --port=80 --type=LoadBalancer



#command to create configmap & html  
vi index-html-configmap.yml

#Once you're in the .yml file. Configure the file with the index.html info.  
  
apiVersion: v1  
kind: ConfigMap  
metadata:  
name: index-html-configmap  
namespace: default  
data:  
index.html:|  
<html>  
<h1>Yayy! Looks like you found my Kube webpage!</h1>  
</br>  
</html  
<h1>This web page is housed on a Pod running Nginx</h1>  
</html

**Step 4.**

**Ensure you can reach the service from your web browser.**

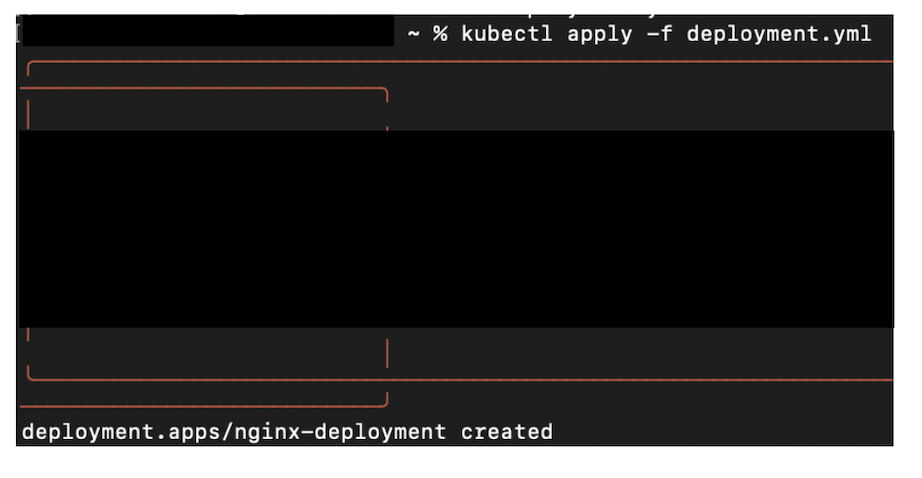
* We need to use the ConfigMap.yml file to overwrite the default of the Nginx html page above with our custom HTML file BUT you have to now create a deployment file using volume mounts to execute it.
* Once you’re in the .yml file. Configure the file with the info.

#command to create the .yml file  
vi <name of the deployment>.yml  
vi deployment.yml  
  
#edit the .yml file and enter the volumemounts parameters.   
Setting this up with replace the default NGINX HTML   
  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
name: nginx-deployment  
namespace: default  
spec:  
selector:  
matchLabels:  
app: nginx  
replicas: 5  
template:  
metadata:  
labels:  
app: nginx  
spec:  
containers:  
name: nginx  
image: nginx:latest  
ports:  
- containerport: 80  
volumeMounts:  
name: nginx-index-file  
mountPath:/usr/share/nginx/html/  
volumes:  
name: nginx-index-file

* Hit “esc” then :wq! to save the .yml file.
* Apply the changes & put it all together to ensure that the custom HTML will replace the NGINX HTML.

#the deployment.yml file with the volume mounts  
kubectl apply -f deployment.yml

#the configmap with our index-html.yml file  
kubectl apply -f index-html-configmap.yml

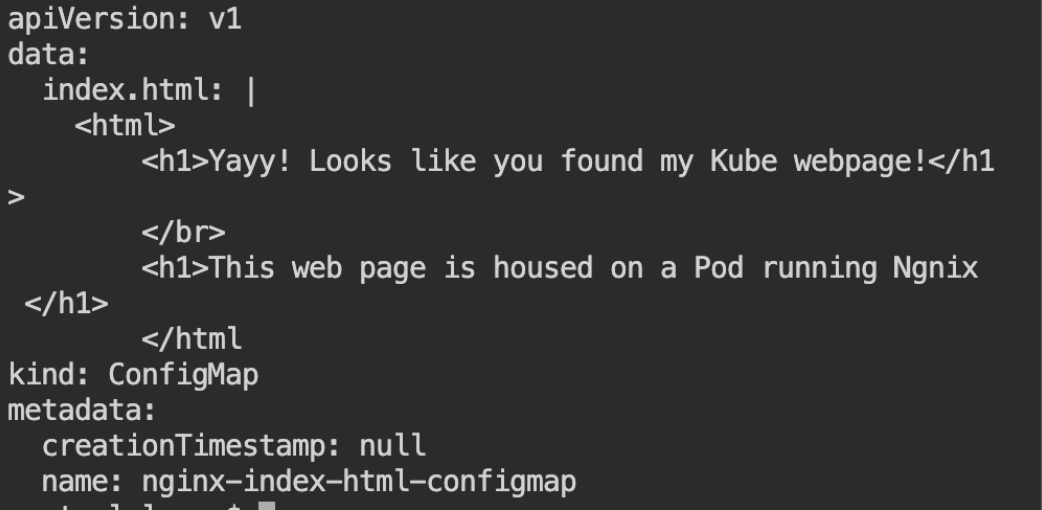


* Earlier we used the “get service” command to obtain the IP address. Run that same command again to get the service IP address that the configMap was mounted to the NordPort.

kubectl get services <deployment name>

You could either use the “curl” command or the minikube service <deployment name>

#command to test the IP address  
curl localhost 10.222.334.21



We have successfully replaced default NGINX HTML with the custom HTML.