Mohanad Mahmoud Sayed

Lab3

1- How many DaemonSets are created in the cluster in all namespaces?

→There are 2 DaemonSets

```
controlplane:~$ kubectl get daemonsets --all-namespaces
NAMESPACE     NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   canal        2         2         2       2            2           kubernetes.io/os=linux   32d
kube-system   kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   32d
controlplane:~$
```

2- what DaemonSets exist on the kube-system namespace?

```
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
canal        2         2         2       2            2           kubernetes.io/os=linux   32d
kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   32d
controlplane:~$
```

3- What is the image used by the POD deployed by the kube-proxy DaemonSet

```
controlplane:~$ kubectl describe daemonset kube-proxy -n kube-system | grep Image:
    Image:      registry.k8s.io/kube-proxy:v1.32.1
controlplane:~$
```

4- Deploy a DaemonSet for FluentD Logging. Use the given specifications. Name: elasticsearch Namespace: kube-system Image: k8s.gcr.io/fluentd-elasticsearch:1.20

Firstly, let’s create fluentd-daemonset.yml:

```
Editor   Tab 1   +
 GNU nano 7.2                                  fluentd-daemonset.yml *
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: elasticsearch
  namespace: kube-system
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      containers:
      - name: fluentd
        image: k8s.gcr.io/fluentd-elasticsearch:1.20
```

Finally, apply and check:

```
controlplane:~$ kubectl apply -f fluentd-daemonset.yml
daemonset.apps/elasticsearch created
controlplane:~$ kubectl get daemonsets --all-namespaces
NAMESPACE     NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   canal           2         2         2       2            2           kubernetes.io/os=linux   32d
kube-system   elasticsearch   2         2         2       2            2           <none>                   56s
kube-system   kube-proxy      2         2         2       2            2           kubernetes.io/os=linux   32d
controlplane:~$
```

5- Deploy a pod named nginx-pod using the nginx:alpine image with the labels set to tier=backend.

Firstly, create nginx-pod.yml file:

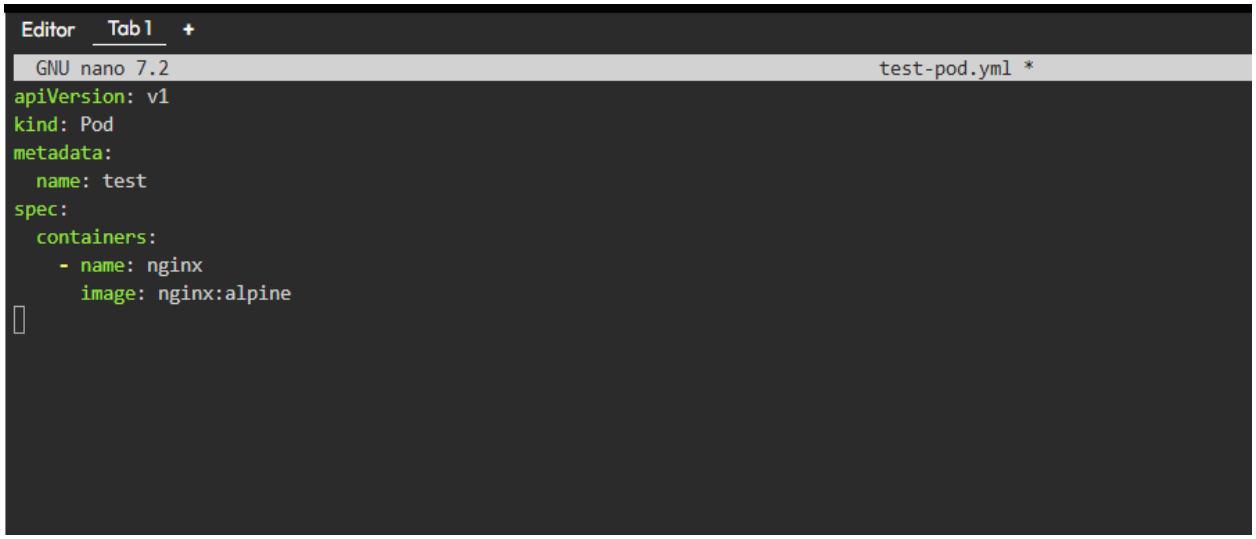
Finally, apply and check:

```
controlplane:~$ kubectl apply -f nginx-pod.yml
pod/nginx-pod created
```

```
controlplane:~$ kubectl get pods -l tier=backend
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          2m57s
controlplane:~$
```

6- Deploy a test pod using the nginx:alpine image.

Firstly, create test-pod.yml file:



Finally, apply and check:

```
controlplane:~$ kubectl apply -f test-pod.yml
pod/test created
controlplane:~$ k get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          2m12s
test        1/1     Running   0          10s
controlplane:~$
```

7- Create a service backend-service to expose the backend application within the cluster on port 80.

Firstly, create backend-service.yml file:

```
Editor   Tab 1   +
 GNU nano 7.2                                          backend-service.yml *
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    tier: backend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP
```

Finally, apply and check:

```
controlplane:~$ kubectl apply -f backend-service.yml
service/backend-service created
controlplane:~$ kubectl get svc backend-service
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
backend-service   ClusterIP   10.105.222.81   <none>        80/TCP    6s
controlplane:~$
```

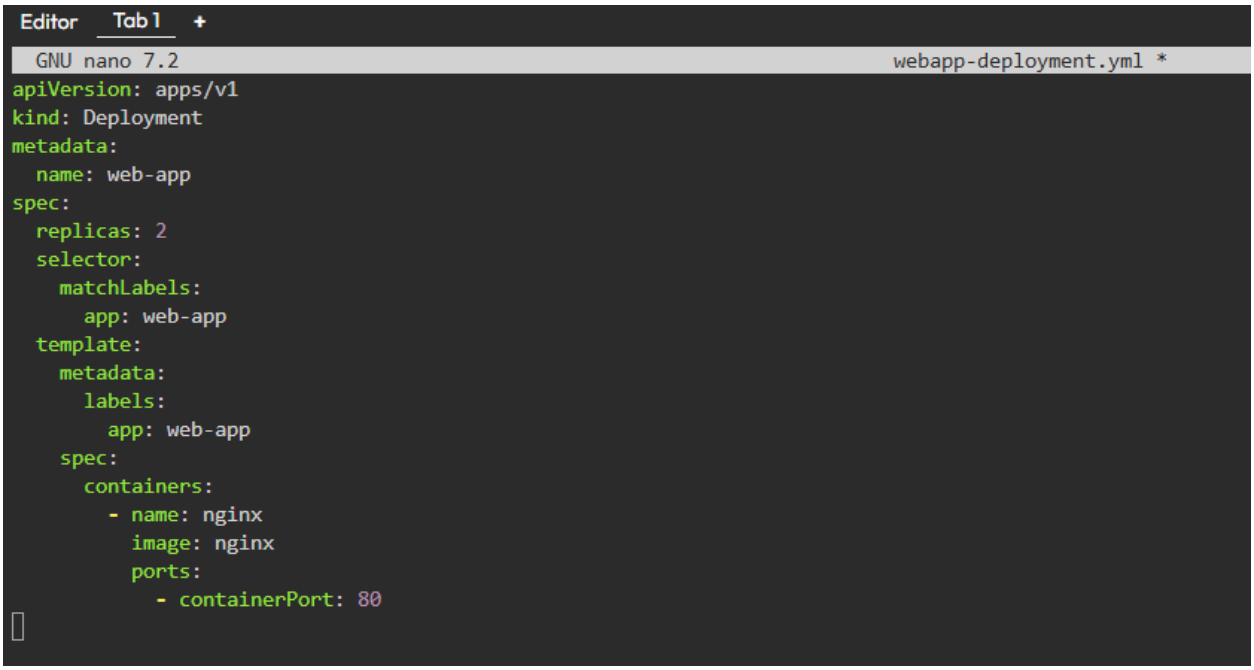
8- try to curl the backend-service from the test pod. What is the response?

```
controlplane:~$ kubectl exec -it test -- /bin/sh
/ # curl backend-service
<!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>
/ #
```

→The response is: the HTML of the default Nginx welcome page, indicating successful access to the backend service via the ClusterIP.

9- Create a deployment named web-app using the image nginx with 2 replicas

Firstly, create webapp-deployment.yml file:

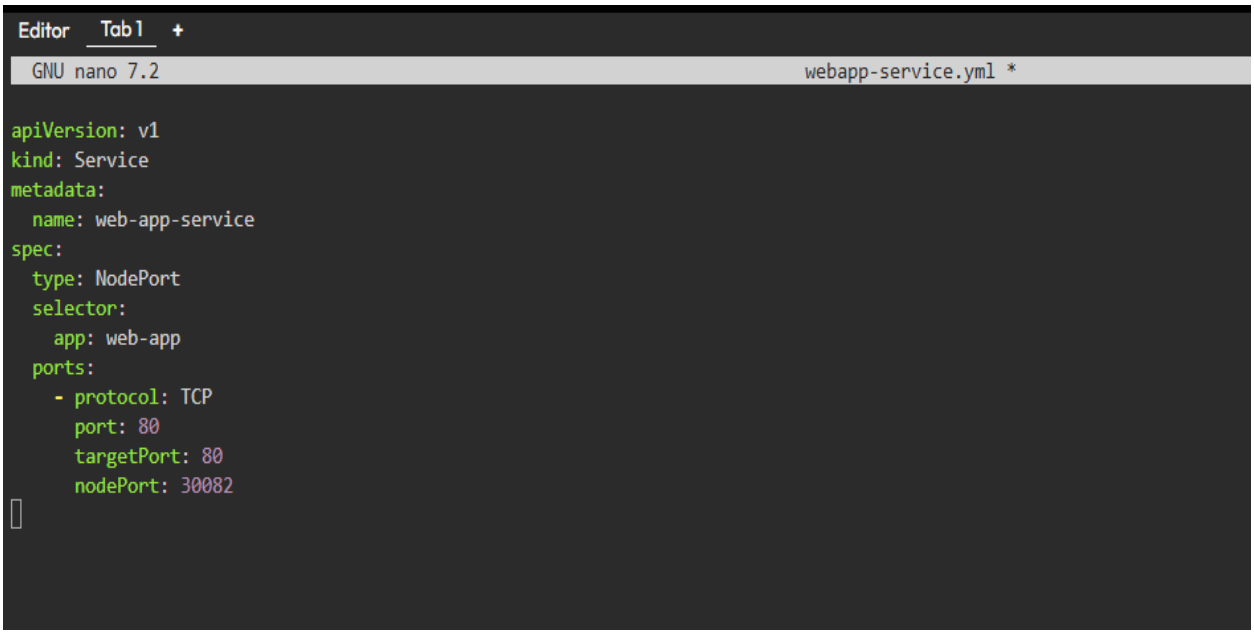


Finally, apply and check:

```
controlplane:~$ kubectl apply -f webapp-deployment.yml
deployment.apps/web-app created
controlplane:~$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
web-app   2/2     2            2           9s
controlplane:~$ kubectl get pods -l app=web-app
NAME                       READY   STATUS    RESTARTS   AGE
web-app-64cd7668-dlgv5   1/1     Running   0          26s
web-app-64cd7668-h4b41   1/1     Running   0          26s
controlplane:~$
```

10- Expose the web-app as service web-app-service application on port 80 and nodeport 30082 on the nodes on the cluster

Firstly, create webapp-service.yml file:



Finally, apply and check:

```
controlplane:~$ kubectl apply -f webapp-service.yml
service/web-app-service created
controlplane:~$ kubectl get svc web-app-service
NAME              TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
web-app-service   NodePort   10.103.66.75   <none>        80:30082/TCP   7s
controlplane:~$
```

11- access the web app from the node

Firstly, let's get the ip address of the node:

```
controlplane:~$ kubectl get nodes -o wide
NAME         STATUS   ROLES          AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RUNTIME
controlplane   Ready    control-plane   32d   v1.32.1   172.30.1.2    <none>        Ubuntu 24.04.1 LTS   6.8.0-51-generic   containerd://1.7.24
node01         Ready    <none>         32d   v1.32.1   172.30.2.2    <none>        Ubuntu 24.04.1 LTS   6.8.0-51-generic   containerd://1.7.24
controlplane:~$
```

Then, let’s access the web app using the ip address of the node and port number 30082

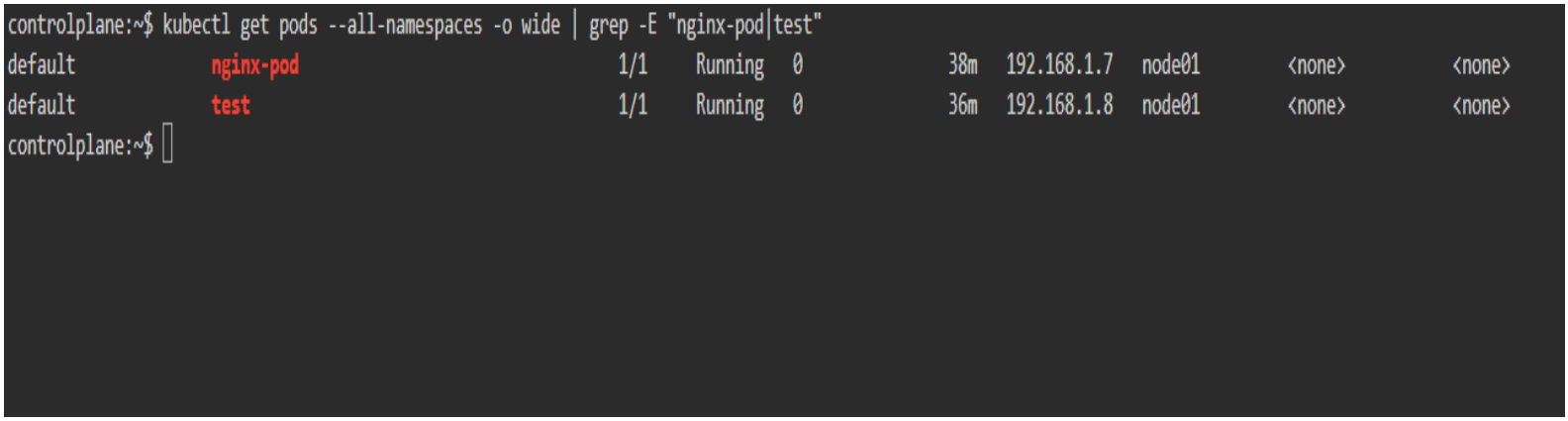
```
controlplane:~$ curl http://172.30.1.2:30082
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you 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>
controlplane:~$
```

The response is: The default Nginx welcome page, confirming that the NodePort service is correctly exposing the web-app on port 30082 on the node IP (172.30.1.2).

12- How many static pods exist in this cluster in all namespaces?

```
Editor  Tab 1  +
controlplane:~$ ls /etc/kubernetes/manifests | wc -l
4
controlplane:~$ kubectl get pods --all-namespaces -o json | jq '.items[] | select(.metadata.ownerReferences == null) | .metadata.name'
"nginx-pod"
"test"
controlplane:~$
```

13-On which nodes are the static pods created currently?



→Static pods created currently on worker node its name is node01