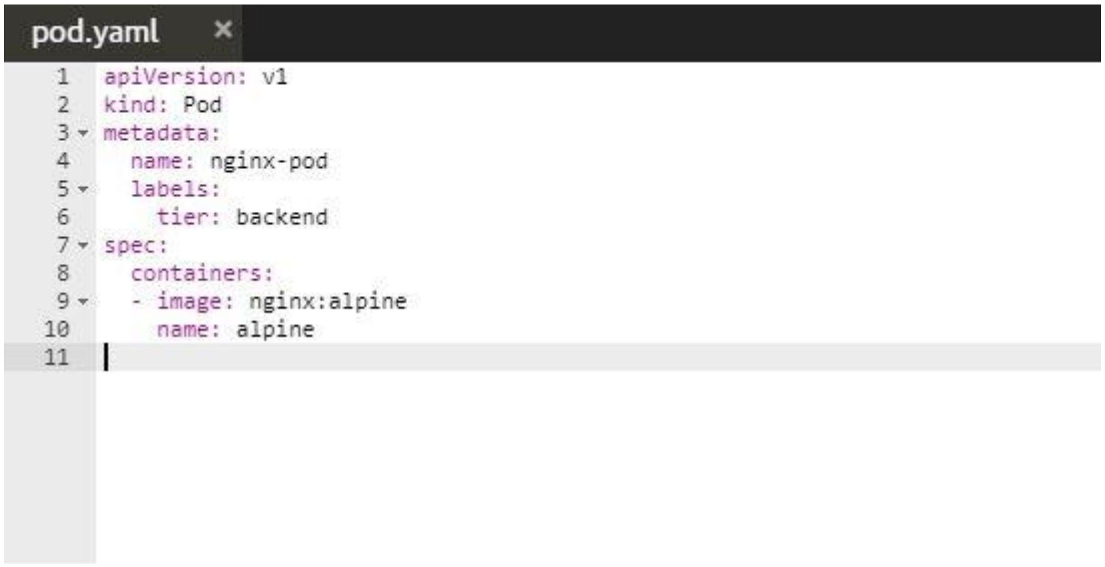
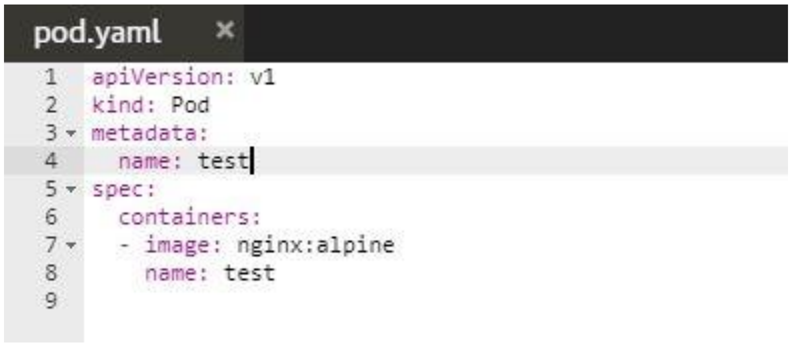
1 - Deploy a pod named `nginx-pod` using the `nginx:alpine` image with the labels set to `tier=backend`.



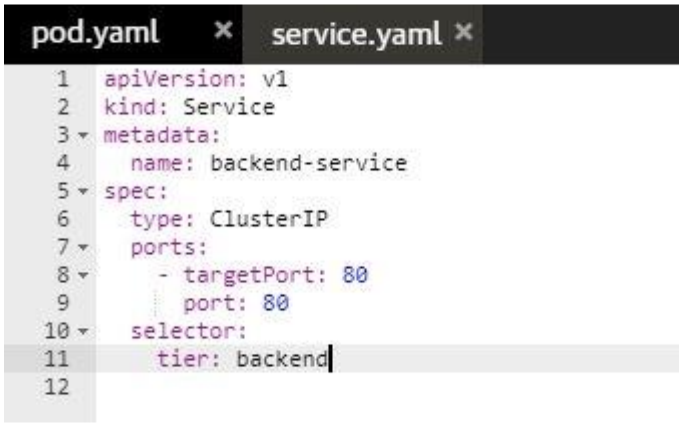
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    tier: backend
spec:
  containers:
  - image: nginx:alpine
    name: alpine
```

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



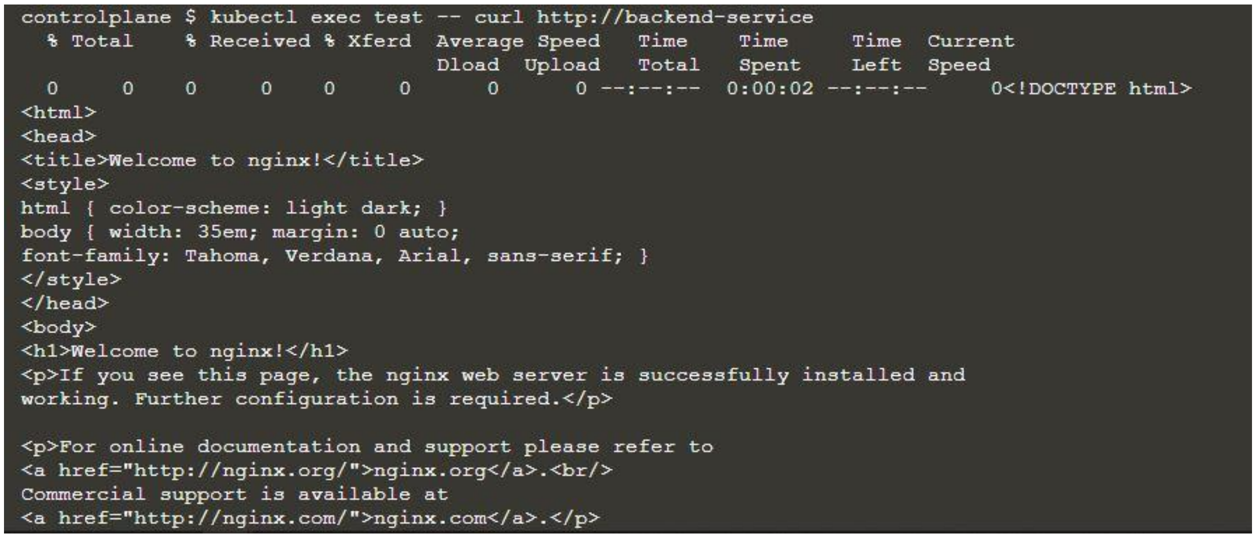
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test
spec:
  containers:
  - image: nginx:alpine
    name: test
```

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



```yaml
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  type: ClusterIP
  ports:
    - targetPort: 80
      port: 80
  selector:
    tier: backend
```

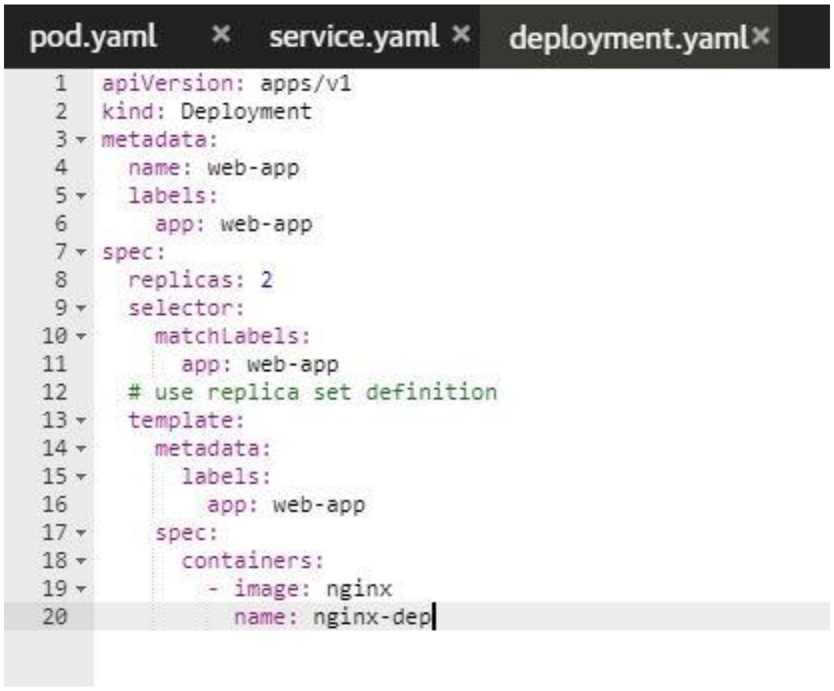
4- try to curl the backend-service from the test pod. What is the response?



```
controlplane $ kubectl exec test -- curl http://backend-service
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:02 --:--:--     0<!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>
```

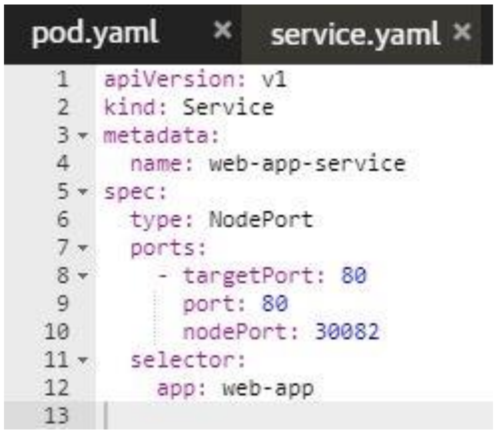
5- Create a deployment named `web-app` using the image `nginx` with `2` replicas



pod.yaml × service.yaml × deployment.yaml×

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
  labels:
    app: web-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-app
  # use replica set definition
  template:
    metadata:
      labels:
        app: web-app
    spec:
      containers:
        - image: nginx
          name: nginx-dep
```

6- Expose the `web-app` as service `web-app-service` application on port `30082` on the nodes on the cluster

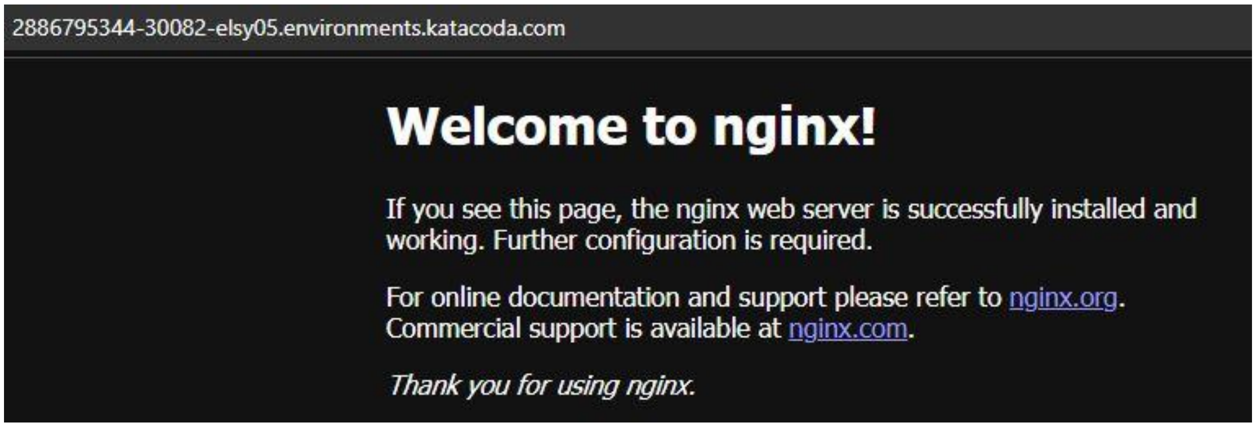


pod.yaml × service.yaml ×

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web-app-service
spec:
  type: NodePort
  ports:
    - targetPort: 80
      port: 80
      nodePort: 30082
  selector:
    app: web-app

```

7- access the web app from the node



2886795344-30082-elsy05.environments.katacoda.com

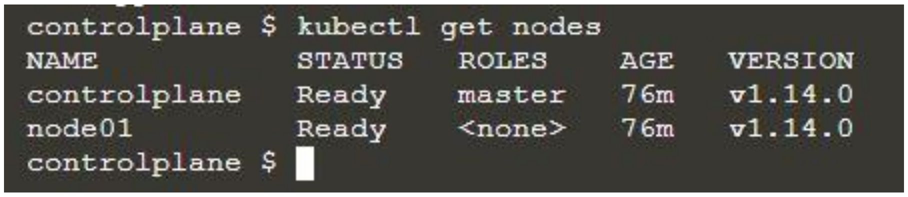
Welcome to nginx!

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.

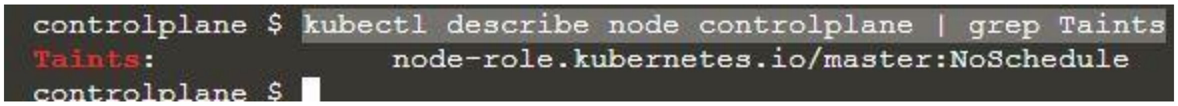
Thank you for using nginx.

8- How many Nodes exist on the system?



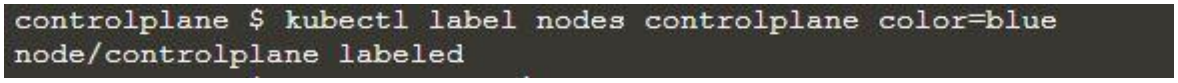
```
controlplane $ kubectl get nodes
NAME           STATUS   ROLES    AGE   VERSION
controlplane   Ready    master   76m   v1.14.0
node01         Ready    <none>   76m   v1.14.0
controlplane $ 
```

9- Do you see any taints on `master?`

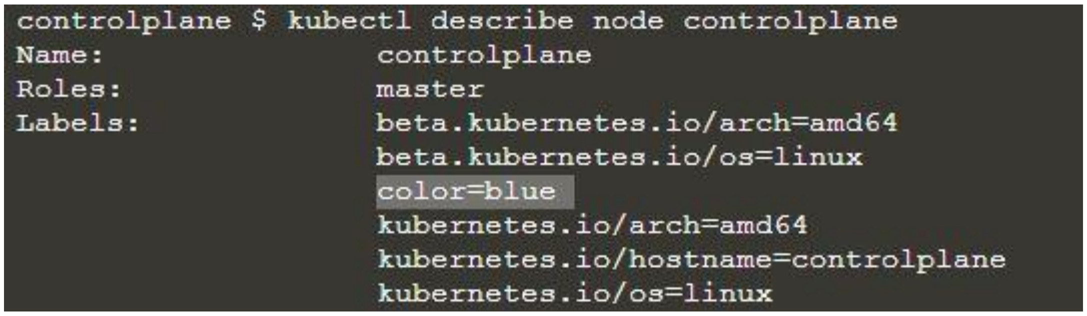


```
controlplane $ kubectl describe node controlplane | grep Taints
Taints:             node-role.kubernetes.io/master:NoSchedule
controlplane $ 
```

10- Apply a label `color=blue` to the master node

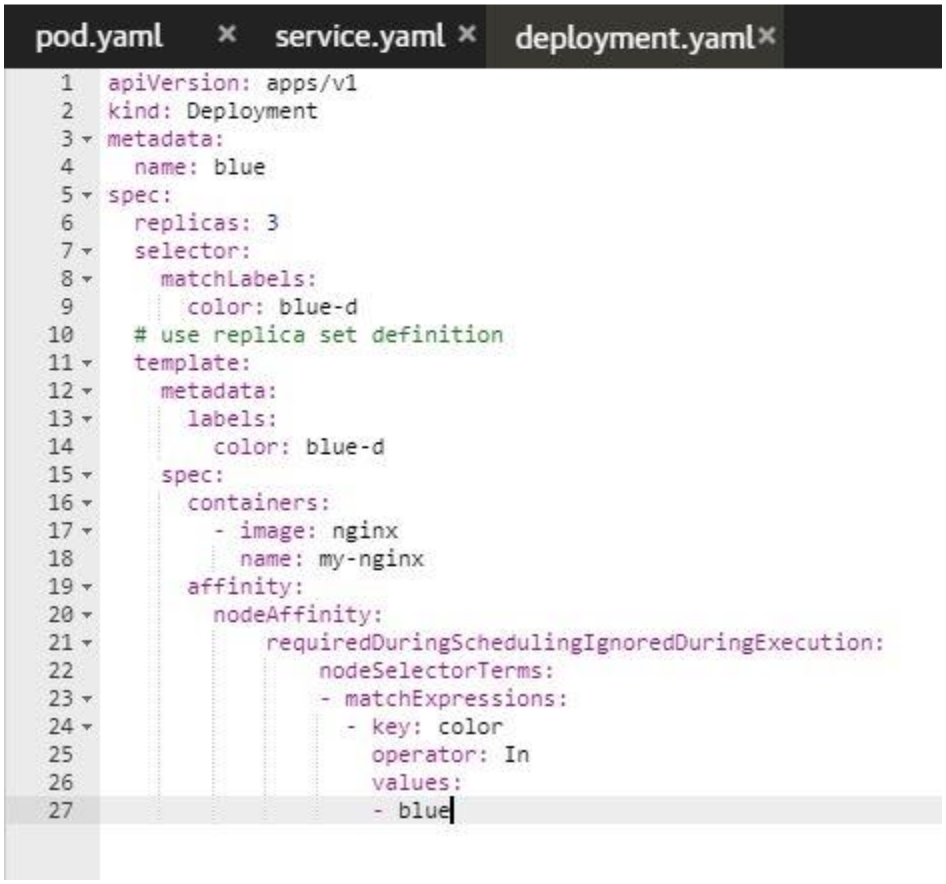


```
controlplane $ kubectl label nodes controlplane color=blue
node/controlplane labeled
```



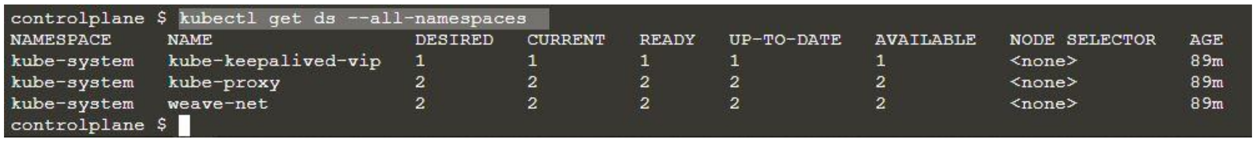
```
controlplane $ kubectl describe node controlplane
Name:               controlplane
Roles:              master
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    color=blue
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=controlplane
                    kubernetes.io/os=linux
```

11- Create a new deployment named `blue` with the `nginx` image and 3 replicas Set Node Affinity to the deployment to place the pods on `master` only
- NodeAffinity: requiredDuringSchedulingIgnoredDuringExecution
- Key: color
- values: blue



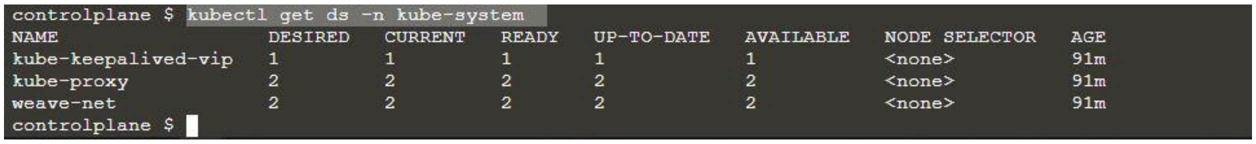
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: blue
spec:
  replicas: 3
  selector:
    matchLabels:
      color: blue-d
  # use replica set definition
  template:
    metadata:
      labels:
        color: blue-d
    spec:
      containers:
        - image: nginx
          name: my-nginx
      affinity:
        nodeAffinity:
            requiredDuringSchedulingIgnoredDuringExecution:
                nodeSelectorTerms:
                - matchExpressions:
                  - key: color
                    operator: In
                    values:
                    - blue
```

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



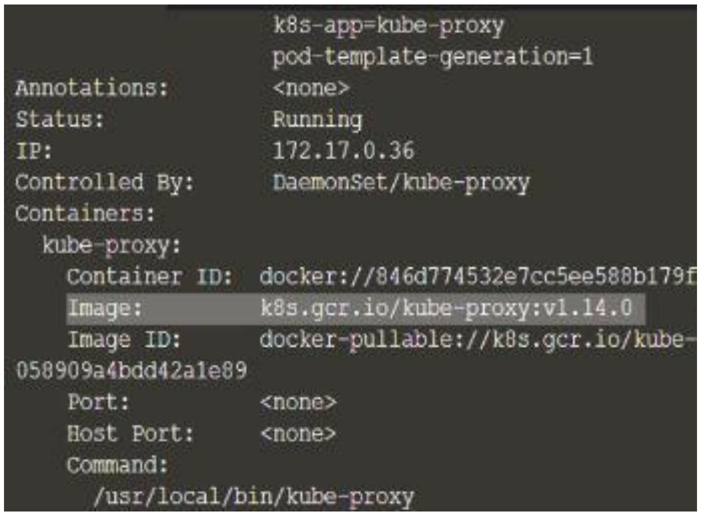
```
controlplane $ kubectl get ds --all-namespaces
NAMESPACE     NAME                  DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
kube-system   kube-keepalived-vip   1         1         1       1            1           <none>          89m
kube-system   kube-proxy            2         2         2       2            2           <none>          89m
kube-system   weave-net             2         2         2       2            2           <none>          89m
controlplane $ 
```

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



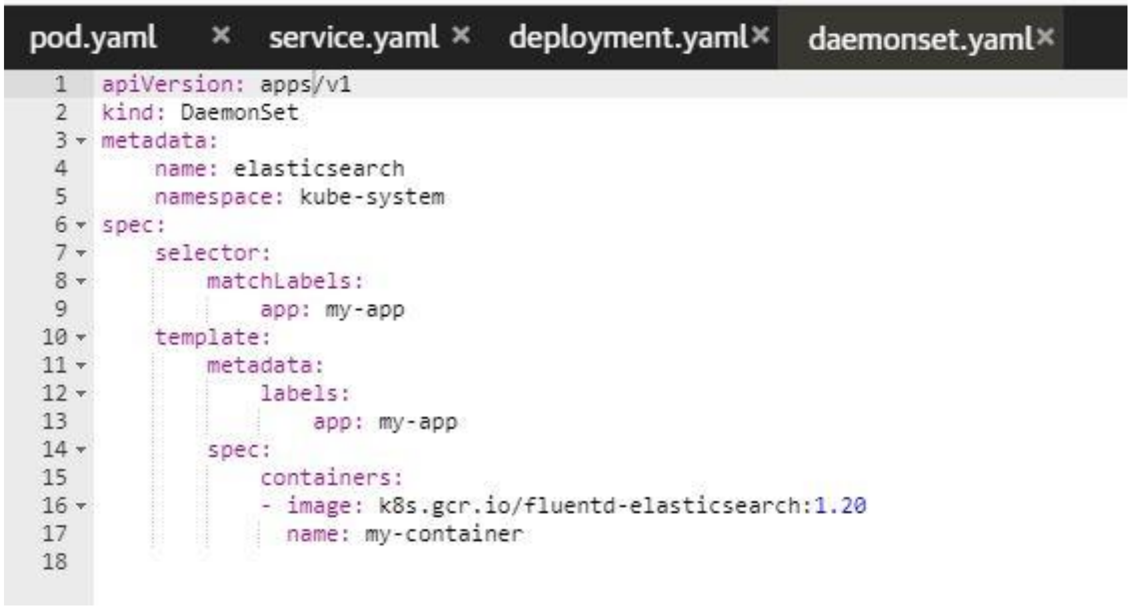
```
controlplane $ kubectl get ds -n kube-system
NAME                  DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
kube-keepalived-vip   1         1         1       1            1           <none>          91m
kube-proxy            2         2         2       2            2           <none>          91m
weave-net             2         2         2       2            2           <none>          91m
controlplane $ 
```

14- What is the image used by the POD deployed by the `kube-proxy` DaemonSet.



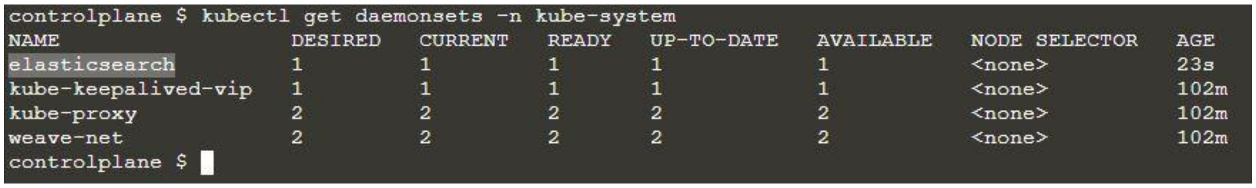
```
                    k8s-app=kube-proxy
                    pod-template-generation=1
Annotations:        <none>
Status:             Running
IP:                 172.17.0.36
Controlled By:      DaemonSet/kube-proxy
Containers:
  kube-proxy:
    Container ID:  docker://846d774532e7cc5ee588b179f
    Image:         k8s.gcr.io/kube-proxy:v1.14.0
    Image ID:      docker-pullable://k8s.gcr.io/kube-
058909a4bdd42a1e89
    Port:          <none>
    Host Port:     <none>
    Command:
      /usr/local/bin/kube-proxy
```

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



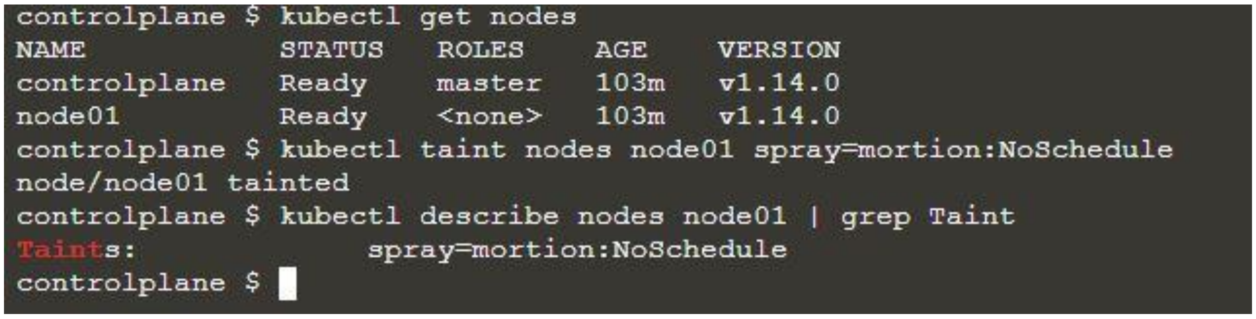
```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
    name: elasticsearch
    namespace: kube-system
spec:
    selector:
        matchLabels:
            app: my-app
    template:
        metadata:
            labels:
                app: my-app
        spec:
            containers:
            - image: k8s.gcr.io/fluentd-elasticsearch:1.20
              name: my-container

```



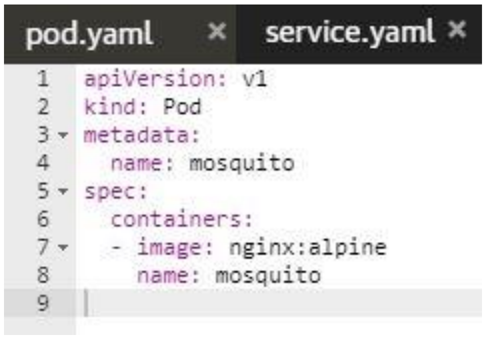
```
controlplane $ kubectl get daemonsets -n kube-system
NAME                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
elasticsearch        1         1         1       1            1           <none>          23s
kube-keepalived-vip  1         1         1       1            1           <none>          102m
kube-proxy           2         2         2       2            2           <none>          102m
weave-net            2         2         2       2            2           <none>          102m
controlplane $
```

16- Create a taint on node01 with key of `spray`, value of `mortein` and effect of `NoSchedule`



```
controlplane $ kubectl get nodes
NAME           STATUS   ROLES    AGE    VERSION
controlplane   Ready    master   103m   v1.14.0
node01         Ready    <none>   103m   v1.14.0
controlplane $ kubectl taint nodes node01 spray=mortion:NoSchedule
node/node01 tainted
controlplane $ kubectl describe nodes node01 | grep Taint
Taints:             spray=mortion:NoSchedule
controlplane $
```

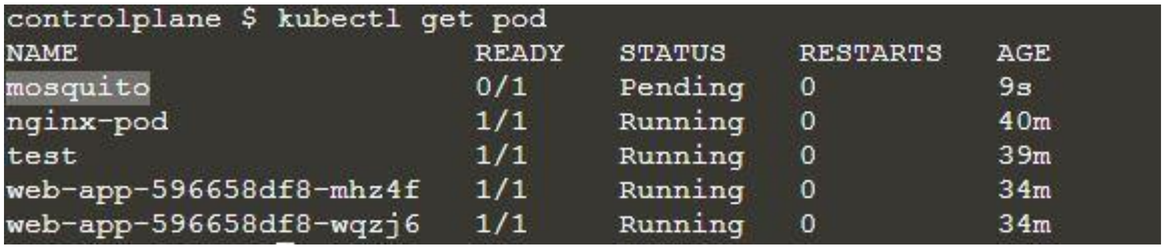
17- Create a new pod with the NGINX image, and Pod name as mosquito



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mosquito
spec:
  containers:
  - image: nginx:alpine
    name: mosquito

```

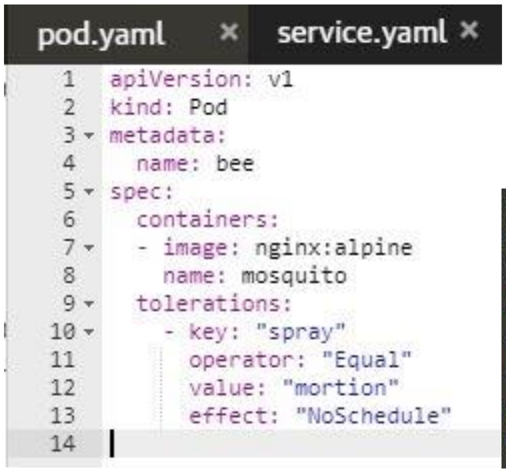
18- What is the state of the `mosquito` POD?



```
controlplane $ kubectl get pod
NAME                      READY   STATUS    RESTARTS   AGE
mosquito                  0/1     Pending   0          9s
nginx-pod                 1/1     Running   0          40m
test                      1/1     Running   0          39m
web-app-596658df8-mhz4f   1/1     Running   0          34m
web-app-596658df8-wqzj6   1/1     Running   0          34m
```

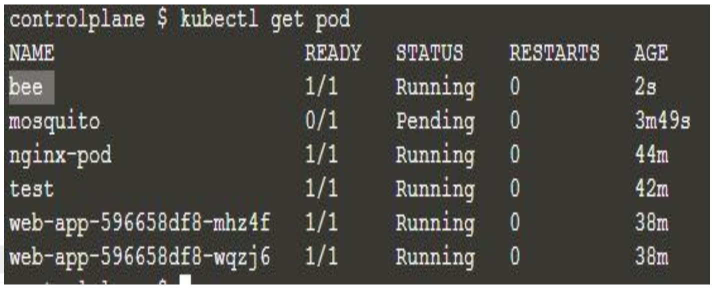
19- Create another pod named `bee` with the NGINX image, which has a toleration set to the

- taint `Mortein –` Image name: nginx
  - Key: spray - Value: mortion - Effect: NoSchedule
  - Status: Running



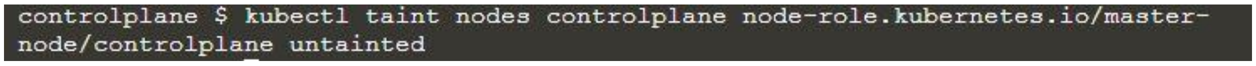
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: bee
spec:
  containers:
  - image: nginx:alpine
    name: mosquito
  tolerations:
    - key: "spray"
      operator: "Equal"
      value: "mortion"
      effect: "NoSchedule"

```



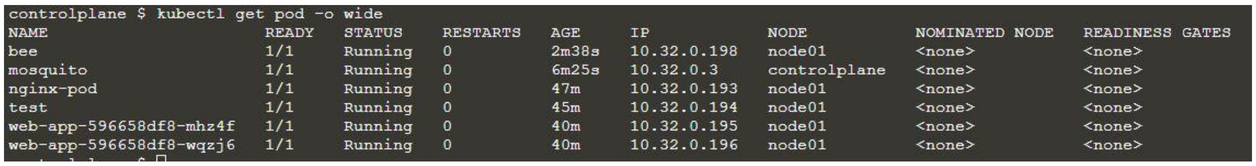
```
controlplane $ kubectl get pod
NAME                      READY   STATUS    RESTARTS   AGE
bee                       1/1     Running   0          2s
mosquito                  0/1     Pending   0          3m49s
nginx-pod                 1/1     Running   0          44m
test                      1/1     Running   0          42m
web-app-596658df8-mhz4f   1/1     Running   0          38m
web-app-596658df8-wqzj6   1/1     Running   0          38m
```

20- Remove the taint on master/controlplane, which currently has the taint effect of NoSchedule



```
controlplane $ kubectl taint nodes controlplane node-role.kubernetes.io/master-
node/controlplane untainted
```

21- What is the state of the pod `mosquito` now and Which node is the POD `mosquito` on?



```
controlplane $ kubectl get pod -o wide
NAME                       READY   STATUS    RESTARTS   AGE     IP            NODE           NOMINATED NODE   READINESS GATES
bee                        1/1     Running   0          2m38s   10.32.0.198   node01         <none>           <none>
mosquito                   1/1     Running   0          6m25s   10.32.0.3     controlplane   <none>           <none>
nginx-pod                  1/1     Running   0          47m     10.32.0.193   node01         <none>           <none>
test                       1/1     Running   0          45m     10.32.0.194   node01         <none>           <none>
web-app-596658df8-mhz4f    1/1     Running   0          40m     10.32.0.195   node01         <none>           <none>
web-app-596658df8-wqzj6    1/1     Running   0          40m     10.32.0.196   node01         <none>           <none>
```