# Kubernetes Lab 2

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

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

```
/root/kubernetes-lab
deployment.yaml
hpa.yaml
job.yaml
pod-with-configmap
pod-with-secret.yam
pod.yaml
replicaset.yaml
service.yaml
storage/
```

pod.yaml

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

Terminal

```
controlplane $ kubectl create -f pod.yaml
pod/nginx-pod created
controlplane $ kubectl get po
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          46s
controlplane $ kubectl run test --image nginx:alpine
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run
--generator=run-pod/v1 or kubectl create instead.
deployment.apps/test created
controlplane $ kubectl get po
NAME                    READY   STATUS    RESTARTS   AGE
nginx-pod               1/1     Running   0          3m35s
test-c79795dbc-tt996    1/1     Running   0          43s
controlplane $
```

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

```
/root/kubernetes-lab
deployment.yaml
hpa.yaml
job.yaml
pod-with-configmap
pod-with-secret.yam
pod.yaml
replicaset.yaml
service.yaml
storage/
```

pod.yaml | service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
   tier: backend
  name: backend-service
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    tier: backend
```

Terminal

```
controlplane $ kubectl create -f service.yaml
service/backend-service created
controlplane $ kubectl get svc
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
backend-service   ClusterIP   10.96.182.73   <none>        80/TCP    8s
kubernetes        ClusterIP   10.96.0.1      <none>        443/TCP   2m29s
controlplane $
```

```
Terminal
controlplane $ kubectl create -f service.yaml
service/backend-service created
controlplane $ kubectl get svc
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
backend-service   ClusterIP   10.96.182.73   <none>        80/TCP    8s
kubernetes        ClusterIP   10.96.0.1      <none>        443/TCP   2m29s
controlplane $ kubectl describe svc backend-service
Name:              backend-service
Namespace:         default
Labels:            tier=backend
Annotations:       <none>
Selector:          tier=backend
Type:              ClusterIP
IP:                10.96.182.73
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         10.32.0.193:80
Session Affinity:  None
Events:            <none>
controlplane $ kubectl get po -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP            NODE     NOMINATED NODE   READINESS G
S
nginx-pod               1/1     Running   0          6m53s   10.32.0.193   node01   <none>           <none>
test-c79795dbc-9k8qh    1/1     Running   0          6m36s   10.32.0.194   node01   <none>           <none>
controlplane $ kubectl get endpoints backend-service
NAME              ENDPOINTS        AGE
backend-service   10.32.0.193:80   7m15s
controlplane $
```

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

```
Terminal
controlplane $ kubectl exec test -- curl http://backend-service
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   615  100<!DOCTYPE html> 0      0      0 --:--:--  0:00:02 --:--:--     0
<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>
   615    0     0    245      0  0:00:02  0:00:02 --:--:--   245
controlplane $
```

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

pod.yaml | service.yaml | deployment.yaml

./root/kubernetes-lab
- deployment.yaml
- hpa.yaml
- job.yaml
- pod-with-configmap
- pod-with-secret.yam
- pod.yaml
- replicaset.yaml
- service.yaml
- storage/

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

Terminal

```
controlplane $ kubectl create -f deployment.yaml
deployment.apps/web-app created
controlplane $ kubectl get deployment
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
test       1/1     1            1           18m
web-app    2/2     2            2           19s
controlplane $
```

6- Expose the web-app as service web-app-service application on port 30082 on the nodes on the cluster 7- access the web app from the node

pod.yaml × deployment.yaml × service.yaml × web-svc.yaml × test-pod.yaml ×

/root/kubernetes-lab
- deployment.yaml
- hpa.yaml
- job.yaml
- pod-with-configmap
- pod-with-secret.yam
- pod.yaml
- replicaset.yaml
- service.yaml
- storage/
- test-pod.yaml
- web-svc.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: my-web-app
  name: my-web-app-service
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
    nodePort: 30083
  selector:
    app: my-web-app
  type: NodePort
```

Terminal

```
controlplane $ kubectl create -f web-svc.yaml
service/my-web-app-service created
controlplane $ kubectl get svc my-web-app-service
NAME                  TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
my-web-app-service    NodePort   10.108.126.109   <none>        80:30083/TCP   18s
controlplane $ kubectl get ep web-app-service
NAME                ENDPOINTS   AGE
web-app-service     <none>      8m41s
controlplane $ kubectl get ep my-web-app-service
NAME                  ENDPOINTS                                   AGE
my-web-app-service    10.32.0.197:80,10.32.0.198:80               2m20s
controlplane $ kubectl get nodes
NAME           STATUS   ROLES    AGE    VERSION
controlplane   Ready    master   147m   v1.14.0
node01         Ready    <none>   147m   v1.14.0
controlplane $
```

8- How many Nodes exist on the system? 2 nodes [ Controlplane(master) and node01 ]

9- Do you see any taints on master ?

Terminal

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

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

Terminal

```
controlplane $ kubectl get nodes --show-labels
NAME           STATUS   ROLES    AGE    VERSION   LABELS
controlplane   Ready    master   4h27m  v1.14.0   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.
io/arch=amd64,kubernetes.io/hostname=controlplane,kubernetes.io/os=linux,node-role.kubernetes.io/master=
node01         Ready    <none>   4h27m  v1.14.0   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.
io/arch=amd64,kubernetes.io/hostname=node01,kubernetes.io/os=linux
controlplane $ kubectl label nodes controlplane color=blue
node/controlplane labeled
controlplane $ kubectl get nodes --show-labels
NAME           STATUS   ROLES    AGE    VERSION   LABELS
controlplane   Ready    master   4h28m  v1.14.0   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,color=blue,
kubernetes.io/arch=amd64,kubernetes.io/hostname=controlplane,kubernetes.io/os=linux,node-role.kubernetes.io/master=
node01         Ready    <none>   4h27m  v1.14.0   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.
io/arch=amd64,kubernetes.io/hostname=node01,kubernetes.io/os=linux
controlplane $
```

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

```
/root/kubernetes-lab/
  deployment.yaml
  hpa.yaml
  job.yaml
  pod-with-configmap.y
  pod-with-secret.yaml
  pod.yaml
  replicaset.yaml
  service.yaml
  storage/
```

deployment.yaml

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

Terminal

```
controlplane $ kubectl create -f deployment.yaml
deployment.apps/blue created
controlplane $ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
blue    0/3     3            0           32s
```

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

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

Terminal

```
controlplane $ kubectl get daemonsets --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>          4h47m
kube-system   kube-proxy           2         2         2       2            2           <none>          4h47m
kube-system   weave-net            2         2         2       2            1           <none>          4h47m
controlplane $ kubectl get daemonset -n kube-system
NAME                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
kube-keepalived-vip  1         1         1       1            1           <none>          4h50m
kube-proxy           2         2         2       2            2           <none>          4h50m
weave-net            2         2         2       2            1           <none>          4h50m
controlplane $
```

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

```
Terminal
controlplane $ kubectl get pods -n kube-system | grep proxy
kube-proxy-cbdsm                           1/1     Running            0          4h54m
kube-proxy-gdlvx                           1/1     Running            0          4h54m
controlplane $ kubectl describe daemonset kube-proxy -n kube-system
Name:           kube-proxy
Selector:       k8s-app=kube-proxy
Node-Selector:  <none>
Labels:         k8s-app=kube-proxy
Annotations:    deprecated.daemonset.template.generation: 1
Desired Number of Nodes Scheduled: 2
Current Number of Nodes Scheduled: 2
Number of Nodes Scheduled with Up-to-date Pods: 2
Number of Nodes Scheduled with Available Pods: 2
Number of Nodes Misscheduled: 0
Pods Status:  2 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:           k8s-app=kube-proxy
  Service Account:  kube-proxy
  Containers:
   kube-proxy:
    Image:      k8s.gcr.io/kube-proxy:v1.14.0
    Port:       <none>
    Host Port:  <none>
    Command:
      /usr/local/bin/kube-proxy
      --config=/var/lib/kube-proxy/config.conf
      --hostname-override=$(NODE_NAME)
    Environment:
      NODE_NAME:   (v1:spec.nodeName)
    Mounts:
      /lib/modules from lib-modules (ro)
      /run/xtables.lock from xtables-lock (rw)
```

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

deployment.yaml × daemonset.yaml ×

/root/kubernetes-lab/
- .my-daemonset.yaml
- daemonset.yaml
- deployment.yaml
- hpa.yaml
- job.yaml
- pod-with-configmap.y
- pod-with-secret.yaml
- pod.yaml
- replicaset.yaml
- service.yaml
- storage/
- undefined/

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

Terminal | Terminal 2 | +

```
controlplane $ kubectl create -f daemonset.yaml
daemonset.apps/elasticsearch created
```

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

Terminal | Terminal 2 | +

```
controlplane $ kubectl taint nodes node01 spray=mortein:NoSchedule
node/node01 tainted
controlplane $
```

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

18- What is the state of the mosquito POD?

```
/root/kubernetes-lab
deployment.yaml
hpa.yaml
job.yaml
pod-with-configmap
pod-with-secret.yam
pod.yaml
replicaset.yaml
service.yaml
storage/
```

pod.yaml

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

Terminal

```
controlplane $ kubectl create -f pod.yaml
pod/mosquito created
controlplane $ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
mosquito    1/1     Running   0          12s
controlplane $
```

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: mortein Effect: NoSchedule Status: Running

```
/root/kubernetes-lab
deployment.yaml
hpa.yaml
job.yaml
pod-with-configmap
pod-with-secret.yam
pod.yaml
replicaset.yaml
service.yaml
storage/
```

pod.yaml

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

Terminal

```
controlplane $ kubectl create -f pod.yaml
pod/bee created
controlplane $ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
bee         1/1     Running   0          51s
mosquito    1/1     Running   0          12m
controlplane $
```

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

```
Terminal
controlplane $ kubectl describe node controlplane | grep Taint
Taints:             node-role.kubernetes.io/master:NoSchedule
controlplane $ kubectl taint nodes controlplane node-role.kubernetes.io/master:NoSchedule
error: at least one taint update is required
controlplane $ kubectl taint nodes controlplane node-role.kubernetes.io/master:NoSchedule-
node/controlplane untainted
controlplane $
```

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

```
Terminal
controlplane $ kubectl get pod mosquito
NAME       READY   STATUS    RESTARTS   AGE
mosquito   1/1     Running   0          29m
controlplane $ kubectl describe pod mosquito
Name:               mosquito
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               node01/172.17.0.39
Start Time:         Mon, 20 Sep 2021 03:14:39 +0000
Labels:             <none>
Annotations:        <none>
Status:             Running
IP:                 10.32.0.193
Containers:
  my-container:
    Container ID:   docker://2b4668e9be999cce160efbc5ba270148e72c6c6bb938e6d6b1ff1c3e24e7987e
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:853b221d3341add7aaadf5f81dd088ea943ab9c918766e295
e
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 20 Sep 2021 03:14:46 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-szvdf (ro)
Conditions:
  Type              Status
  Initialized       True
```