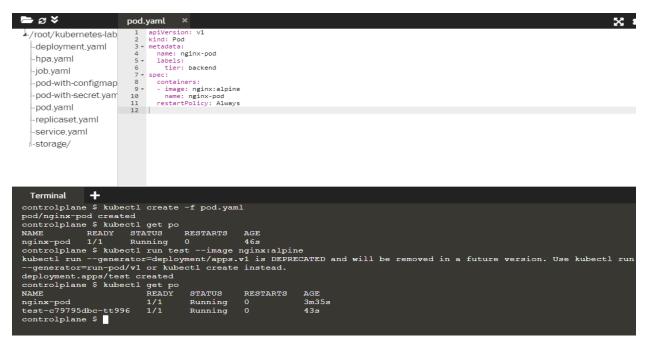
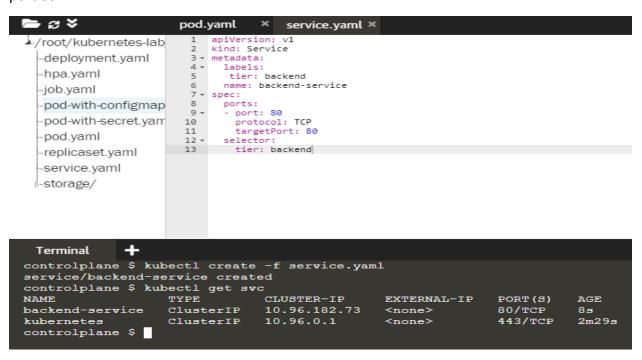
Kubernetes Lab 2

- 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.



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



```
Terminal
controlplane $ kubectl create -f service.yaml
service/backend-service created
controlplane $ kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP backend-service ClusterIP 10.96.182.73 <none> kubernetes ClusterIP 10.96.0.1 <none>
                                                                  PORT (S)
                                                                             AGE
                                                                             2m29s
                                                                  443/TCP
controlplane $ kubectl describe svc backend-service
                  backend-service
Name:
Namespace:
                    default
                     tier=backend
Annotations:
                     <none>
                    tier=backend
Selector:
                     ClusterIP
Type:
                     10.96.182.73
Port:
                     <unset> 80/TCP
TargetPort:
                    80/TCP
                     10.32.0.193:80
Endpoints:
Session Affinity: None
Events:
                     <none>
controlplane $ kubectl get po -o wide
                        READY STATUS
                                                                                   NODE
                                                                                             NOMINATED NODE READINESS
                                             RESTARTS AGE
NAME
                                                         6m53s 10.32.0.193 node01
6m36s 10.32.0.194 node01
                                                                                   node01
test-c79795dbc-9k8qh 1/1 Running 0 6 controlplane $ kubectl get endpoints backend-service
                                  Running
                                                                                            <none>
                                                                                                                <none>
                   ENDPOINTS
                                      AGE
                   10.32.0.193:80
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
           % Received % Xferd Average Speed Time Time
 % Total
                                                              Time Current
                               Dload Upload Total
                                                      Spent
                                                              Left Speed
100 615 100<!DOCTYPE html> 0
                                        0 --:--:- 0:00:02 --:--:-
<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>
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
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.
Thank you for using nginx.
</body>
</html>
  615
                   245
                            0 0:00:02 0:00:02 --:--:-
              0
controlplane $ |
```

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



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
                      1 apiVersion: v1
                          kind: Service
  deployment.yaml
                       3 → metadata:
                           labels:
  hpa.yaml
                           name: my-web-app-service
  job.yaml
                       7 ▼ spec:
                          ports:
- port: 80
  pod-with-configmap
                       9 +
  pod-with-secret.yam
                          protocol: TCP
targetPort: 80
                      11
  pod.yaml
                             nodePort: 30083
                      13 → selector:
  replicaset.yaml
                      14 app: my-web-a
15 type: NodePort
                            app: mv-web-app
  service.yaml
  -storage/
  test-pod.yaml
  -web-svc.yaml
 Terminal
controlplane $ kubectl create -f web-svc.yaml
service/my-web-app-service created
controlplane $ kubectl get svc my-web-app-service
                      TYPE
                                   CLUSTER-IP
                                                      EXTERNAL-IP
                                                                      PORT (S)
                                                                                      AGE
                                                                     80:30083/TCP
my-web-app-service
                                   10.108.126.109
                      NodePort
                                                                                      18s
                                                      <none>
controlplane $ kubectl get ep web-app-service
                  ENDPOINTS AGE <none> 8m41s
NAME
web-app-service <none>
controlplane $ kubectl get ep my-web-app-service
                     ENDPOINTS
my-web-app-service
                       10.32.0.197:80,10.32.0.198:80
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

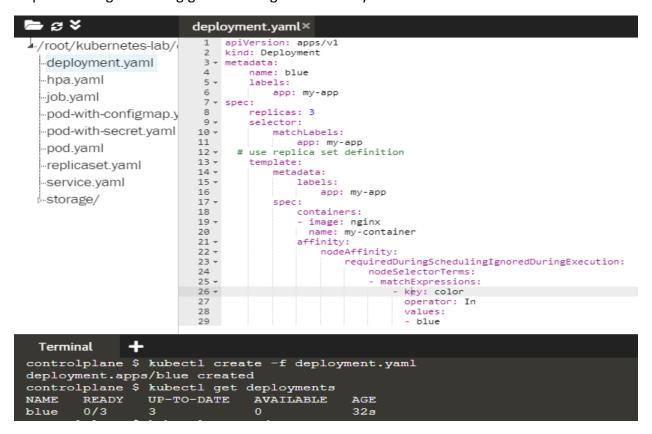
Taints: node-role.kubernetes.io/master:NoSchedule

controlplane $
```

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

```
Terminal
controlplane $ kubectl get nodes --show-labels
NAME STATUS ROLES AGE VERSION controlplane Ready master 4h27m v1.14.0
                                                    LABELS
                                                    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
               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



- 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-keepalived-vip 1
                                                                                                  4h47m
kube-system
                                                                                   <none>
kube-system kube-proxy
                                                                                                  4h47m
                                                                                   <none>
kube-system weave-net
                                                                                   <none>
                                                                                                  4h47m
controlplane $ kubectl get daemonset -n kube-system
                                                           AVAILABLE NODE SELECTOR AGE
                  DESIRED CURRENT READY UP-TO-DATE
kube-keepalived-vip 1
kube-proxy 2
                                                                      <none>
                                                                                     4h50m
                                                                      <none>
                                                                                     4h50m
weave-net
                                                                      <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
kube-proxy-gdlvx
                                             1/1
                                                     Running
                                              1/1
                                                     Running
                                                                                      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×
                           1 apiVersion: apps/v1
//root/kubernetes-lab/
                           2 kind: DaemonSet
  ....my-daemonset.yaml
                           3 → metadata:
                           4
                               name: elasticsearch
   -daemonset.yaml
                           5
                                namespace: kube-system
                           6 → labels:
   deployment.yaml
                           7
                                 app-type: test-app-type
   -hpa.yaml
                           8 ≠ spec:
                           9 +
                               template:
   --job.yaml
                                  metadata:
                          10 -
                          11 -
                                   labels:
   -pod-with-configmap.y
                         12
                                    name: test-daemonset-container
                          13 -
   -pod-with-secret.yaml
                          14
                                     containers:
   --pod.yaml
                          15 -
                                     - image: k8s.gcr.io/fluentd-elasticsearch:1.20
                          16
                                       name: my-container
   replicaset.yaml
                          17 -
                               selector:
                          18 -
                                 matchLabels:
   -service.yaml
                          19
                                  name: test-daemonset-container
                          20
  -storage/
  -undefined/
  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

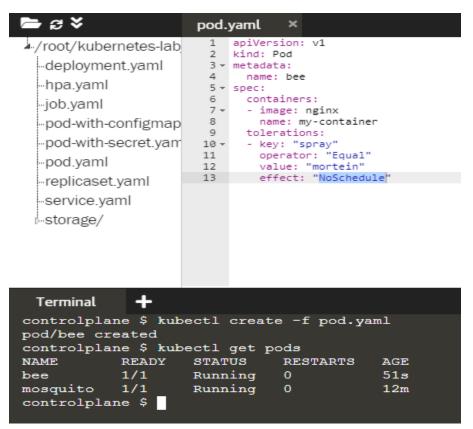
```
Terminal Terminal 2 +

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?

```
- ≈ ¥
                       pod.yaml
                           apiVersion:
                      1 apiVersion
2 kind: Pod
  /root/kubernetes-lab
   deployment.yaml
                       3 → metadata:
                            name: mosquito
   hpa.yaml
                       5 → spec:
                           containers:
  -job.yaml
                            - image: nginx
name: my-container
   pod-with-configmap
                       9
   pod-with-secret.yam
   pod.yaml
   replicaset.yaml
   service.yaml
 -storage/
  Terminal
controlplane $ kubectl create -f pod.yaml
pod/mosquito created
controlplane $ kubectl get pods
                                RESTARTS
NAME
            READY
                      STATUS
                                               AGE
mosquito 1/1
                      Running
                                               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



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
                Running
                                     29m
controlplane $ kubectl describe pod mosquito
Name:
                 mosquito
Namespace:
                  default
Priority:
PriorityClassName: <none>
10.32.0.193
IP:
Containers:
  my-container:
    Container ID: docker://2b4668e9be999cce160efbc5ba270148e72c6c6bb938e6d6b1ff1c3e24e7987e
   Image: nginx
Image ID: docker-pullable://nginx@sha256:853b221d3341add7aaadf5f81dd088ea943ab9c918766e295
   FoIt: <none>
Host Port: <none>
State:
                 Running
     Started:
                  Mon, 20 Sep 2021 03:14:46 +0000
    Ready:
                   True
    Restart Count: 0
    Environment:
                  <none>
   Mounts:
     /war/run/secrets/kubernetes.io/serviceaccount from default-token-szwdf (ro)
Conditions:
  Type
                   Status
  Initialized
                  True
```