



Exercise 12.1: Assign Pods Using Labels

Overview

While allowing the system to distribute Pods on your behalf is typically the best route, you may want to determine which nodes a Pod will use. For example you may have particular hardware requirements to meet for the workload. You may want to assign VIP Pods to new, faster hardware and everyone else to older hardware.

In this exercise we will use `labels` to schedule Pods to a particular node. Then we will explore `taints` to have more flexible deployment in a large environment.

1. Begin by getting a list of the nodes. They should be in the ready state and without added labels or taints.

```
student@cp:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
cp	Ready	control-plane	44h	v1.27.1
worker	Ready	<none>	43h	v1.27.1

2. View the current labels and taints for the nodes.

```
student@cp:~$ kubectl describe nodes |grep -A5 -i label
```

```
Labels:      beta.kubernetes.io/arch=amd64
             beta.kubernetes.io/os=linux
             kubernetes.io/arch=amd64
             kubernetes.io/hostname=scp
             kubernetes.io/os=linux
             node-role.kubernetes.io/control-plane=
--
Labels:      beta.kubernetes.io/arch=amd64
             beta.kubernetes.io/os=linux
             kubernetes.io/arch=amd64
             kubernetes.io/hostname=worker
             kubernetes.io/os=linux
             system=secondOne
```

```
student@cp:~$ kubectl describe nodes |grep -i taint
```

```
Taints:      <none>
Taints:      <none>
```

3. Get a count of how many containers are running on both the cp and worker nodes. There are about 24 containers running on the cp in the following example, and eight running on the worker. There are status lines which increase the **wc** count. You may have more or less, depending on previous labs and cleaning up of resources. Take note of the number of containers, and then notice the numbers change due to scheduling. The change between nodes is the important information, not the particular number. If you are using **cri-o** you can view containers using **crictl ps**.

```
student@cp:~$ kubectl get deployments --all-namespaces
```

NAMESPACE	NAME	READY	UP-TO-DATE	AVAILABLE	AGE
accounting	nginx-one	1/1	1	1	19h
default	anotherweb-apache	1/1	1	1	8h
default	web-one	1/1	1	1	45m
default	web-two	1/1	1	1	45m
kube-system	calico-kube-controllers	1/1	1	1	35h

```
<output_omitted>
```

```
student@cp:~$ sudo crictl ps | wc -l
```

```
24
```

```
student@worker:~$ sudo crictl ps | wc -l
```

```
21
```

4. For the purpose of the exercise we will assign the cp node to be VIP hardware and the secondary node to be for others.

```
student@cp:~$ kubectl label nodes cp status=vip
```

```
node/cp labeled
```

```
student@cp:~$ kubectl label nodes worker status=other
```

```
node/worker labeled
```

5. Verify your settings. You will also find there are some built in labels such as hostname, os and architecture type. The output below appears on multiple lines for readability.

```
student@cp:~$ kubectl get nodes --show-labels
```

```
NAME      STATUS    ROLES    AGE   VERSION   LABELS
cp        Ready     control-plane   35h   v1.27.1   beta.kubernetes.io/arch=amd64,
beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=cp,
kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=,
node.kubernetes.io/exclude-from-external-load-balancers=,status=vip
worker    Ready     <none>        35h   v1.27.1   beta.kubernetes.io/arch=amd64,
beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=worker,
kubernetes.io/os=linux,status=other,system=secondOne
```

6. Create `vip.yaml` to spawn four busybox containers which sleep the whole time. Include the `nodeSelector` entry.

```
student@cp:~$ vim vip.yaml
```

YAML

vip.yaml

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: vip
5 spec:
6   containers:
7   - name: vip1
8     image: busybox
9     args:
10    - sleep
11    - "1000000"
12   - name: vip2
13     image: busybox
14     args:
15    - sleep
16    - "1000000"
17   - name: vip3
18     image: busybox
```



```

19     args:
20     - sleep
21     - "1000000"
22   - name: vip4
23     image: busybox
24     args:
25     - sleep
26     - "1000000"
27   nodeSelector:
28     status: vip

```

7. Deploy the new pod. Verify the containers have been created on the cp node. It may take a few seconds for all the containers to spawn. Check both the cp and the secondary nodes. From this point forward use **crictl** where the step lists **docker** if you have deployed your cluster with cri-o.

```
student@cp:~$ kubectl create -f vip.yaml
```

```
pod/vip created
```

```

student@cp:~$ sudo cat <<EOF | sudo tee /etc/crictl.yaml
runtime-endpoint: unix:///run/containerd/containerd.sock
image-endpoint: unix:///var/run/dockerhim.sock
timeout: 10
debug: false
EOF

```

```
student@cp:~$ sudo crictl ps |wc -l
```

```
28
```

```
student@worker:~$ sudo crictl ps |wc -l
```

```
21
```

8. Delete the pod then edit the file, commenting out the `nodeSelector` lines. It may take a while for the containers to fully terminate.

```
student@cp:~$ kubectl delete pod vip
```

```
pod "vip" deleted
```

```
student@cp:~$ vim vip.yaml
```

```

....
# nodeSelector:
#   status: vip

```

9. Create the pod again. Containers should now be spawning on either node. You may see pods for the daemonsets as well.

```
student@cp:~$ kubectl get pods
```

```
<output_omitted>
```

```
student@cp:~$ kubectl create -f vip.yaml
```

```
pod/vip created
```

10. Determine where the new containers have been deployed. They should be more evenly spread this time. Again, the numbers may be different, the change in numbers is what we are looking for. Due to lack of `nodeSelector` they could go to either node.

```
student@cp:~$ sudo crictl ps |wc -l
```

```
24
```

```
student@worker:~$ sudo crictl ps |wc -l
```

```
25
```

11. Create another file for other users. Change the names from vip to others, and uncomment the `nodeSelector` lines.

```
student@cp:~$ cp vip.yaml other.yaml
```

```
student@cp:~$ sed -i s/vip/other/g other.yaml
```

```
student@cp:~$ vim other.yaml
```

YAML

other.yaml

```
1  ....
2  nodeSelector:
3  status: other
```

12. Create the other containers. Determine where they deploy.

```
student@cp:~$ kubectl create -f other.yaml
```

```
pod/other created
```

```
student@cp:~$ sudo crictl ps |wc -l
```

```
24
```

```
student@worker:~$ sudo crictl ps |wc -l
```

```
25
```

13. Shut down both pods and verify they terminated. Only our previous pods should be found.

```
student@cp:~$ kubectl delete pods vip other
```

```
pod "vip" deleted
pod "other" deleted
```

```
student@cp:~$ kubectl get pods
```

```
<output_omitted>
```