Chapter 11

Scheduling

11.1 Labs

Exercise 11.1: 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.

Assign Pods Using Labels

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

```
student@lfs458-node-1a0a:~$ kubectl get nodes
NAME              STATUS    ROLES   AGE       VERSION
lfs458-node-2b2b   Ready     <none>  2d        v1.9.1
lfs458-node-1a0a   Ready     master  2d        v1.9.1
```

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

```
student@lfs458-node-1a0a:~$ kubectl describe nodes |grep -i label
Labels:                        beta.kubernetes.io/arch=amd64
Labels:                        beta.kubernetes.io/arch=amd64

student@lfs458-node-1a0a:~$ kubectl describe nodes |grep -i taint
Taints:                        <none>
Taints:                        <none>
```

3. Verify there are no deployments running. If there are, delete them. Get a count of how many containers are running on both the master and secondary nodes. There are about 17 containers running on the master in the following example, and five running on the secondary. There are status lines which increase the **wc** count. You may have more or less, depending on previous labs and cleaning up of resources.

```
student@lfs458-node-1a0a:~$ kubectl get deployments --all-namespaces
NAMESPACE     NAME       DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kube-system   kube-dns   1         1         1            1           1d

student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
19

student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
7
```

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

```
student@lfs458-node-1a0a:~$ kubectl label nodes lfs458-node-1a0a status=vip
node "lfs458-node-1a0a" labeled

student@lfs458-node-1a0a:~$ kubectl label nodes lfs458-node-2b2b status=other
node "lfs458-node-2b2b" 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@lfs458-node-1a0a:~$ kubectl get nodes --show-labels
NAME              STATUS    AGE       VERSION   LABELS
lfs458-node-2b2b    Ready     2d        v1.9.1
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=
linux,kubernetes.io/hostname=lfs458-node-2b2b,status=other
lfs458-node-1a0a   Ready     2d        v1.9.1
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=
linux,kubernetes.io/hostname=lfs458-node-1a0a,
node-role.kubernetes.io/master=,status=vip
```

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

```
student@lfs458-node-1a0a:~$ vim vip.yaml


apiVersion: v1
kind: Pod
metadata:
  name: vip
spec:
  containers:
  - name: vip1
    image: busybox
    args:
    - sleep
    - "1000000"
  - name: vip2
    image: busybox
    args:
    - sleep
    - "1000000"
  - name: vip3
    image: busybox
    args:
```



```
    - sleep
    - "1000000"
  - name: vip4
    image: busybox
    args:
    - sleep
    - "1000000"
  nodeSelector:
    status: vip
```

7. Deploy the new pod. Verify the containers have been created on the master node. It may take a few seconds for all the containers to spawn. Check both the master and the secondary nodes.

```
student@lfs458-node-1a0a:~$ kubectl create -f vip.yaml
pod "vip" created


student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
24

student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
7
```

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@lfs458-node-1a0a:~$ kubectl delete pod vip
pod "vip" deleted

student@lfs458-node-1a0a:~$ vim vip.yaml
....
#  nodeSelector:
 #    status: vip
```

9. Create the pod again. Containers should now be spawning on both nodes.

```
student@lfs458-node-1a0a:~$ kubectl get pods
NAME      READY     STATUS        RESTARTS   AGE
vip       0/4       Terminating   0          5m

student@lfs458-node-1a0a:~$ kubectl get pods
No resources found.

student@lfs458-node-1a0a:~$ kubectl create -f vip.yaml
pod "vip" created
```

10. Determine where the new containers have been deployed.

```
student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
19

student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
12
```

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

```
student@lfs458-node-1a0a:~$ cp vip.yaml other.yaml

student@lfs458-node-1a0a:~$ sed -i s/vip/other/g other.yaml

student@lfs458-node-1a0a:~$ vim other.yaml
.
  nodeSelector:
    status: other
```


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

```
student@lfs458-node-1a0a:~$ kubectl create -f other.yaml
pod "other" created

student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
19

student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
17
```

13. Shut down both pods and verify they are terminating.

```
student@lfs458-node-1a0a:~$ kubectl delete pods vip other
pod "vip" deleted
pod "other" deleted

student@lfs458-node-1a0a:~$ kubectl get pods
NAME      READY     STATUS        RESTARTS   AGE
other     4/4       Terminating   0          5m
vip       4/4       Terminating   0          10m
```

Exercise 11.2: Using Taints to Control Pod Deployment

Use taints to manage where Pods are deployed or allowed to run. In addition to assigning a Pod to a group of nodes, you may also want to limit usage on a node or fully evacuate Pods. Using taints is one way to achieve this. You may remember that the master node begins with a `NoSchedule` taint. We will work with three taints to limit or remove running pods.

1. Verify that the master and secondary node have the minimal number of containers running.
2. Create a deployment which will deploy eight **nginx** containers. Begin by creating a `YAML` file.

```
student@lfs458-node-1a0a:~$ vim taint.yaml


apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: taint-deployment
spec:
  replicas: 8
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name:  nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

3. Apply the file to create the deployment.

```
student@lfs458-node-1a0a:~$ kubectl apply -f taint.yaml
deployment "taint-deployment" created
```

4. Determine where the containers are running. In the following example three have been deployed on the master node and five on the secondary node. Remember there will be other housekeeping containers created as well.


```
student@lfs458-node-1a0a:~$ sudo docker ps |grep nginx
00c1be5df1e7        nginx@sha256:e3456c851a152494c3e..
<output_omitted>

student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
25
student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
17
```

5. Delete the deployment. Verify the containers are gone.

```
student@lfs458-node-1a0a:~$ kubectl delete deployment taint-deployment
deployment "taint-deployment" deleted

student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
19
```

6. Now we will use a taint to affect the deployment of new containers. There are three taints, `NoSchedule`, `PreferNoSchedule` and `NoExecute`. The taints having to do with schedules will be used to determine newly deployed containers, but will not affect running containers. The use of `NoExectute` will cause running containers to move.

   Taint the secondary node, verify it has the taint then create the deployment again. We will use the key of `bubba` to illustrate the key name is just some string an admin can use to track Pods.

```
student@lfs458-node-1a0a:~$ kubectl taint nodes lfs458-node-2b2b bubba=value:PreferNoSchedule
node "lfs458-node-2b2b" tainted

student@lfs458-node-1a0a:~$ kubectl describe node |grep Taint
Taints:                         bubba=value:PreferNoSchedule
Taints:                         <none>
student@lfs458-node-1a0a:~$ kubectl apply -f taint.yaml
deployment "taint-deployment" created
```

7. Locate where the containers are running. We can see that more containers are on the master, but there still were some created on the secondary. Delete the deployment when you have gathered the numbers.

```
student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
29

student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
13

student@lfs458-node-1a0a:~$ kubectl delete deployment taint-deployment
deployment "taint-deployment" deleted
```

8. Remove the taint, verify it has been removed. Note that the key is used with a minus sign appended to the end.

```
student@lfs458-node-1a0a:~$ kubectl taint nodes lfs458-node-2b2b bubba-
node "lfs458-node-2b2b" untainted

student@lfs458-node-1a0a:~$ kubectl describe node |grep Taint
Taints:                         <none>
Taints:                         <none>
```

9. This time use the `NoSchedule` taint, then create the deployment again. The secondary node should not have any new containers.

```
student@lfs458-node-1a0a:~$ kubectl taint nodes lfs458-node-2b2b bubba=value:NoSchedule
node "lfs458-node-2b2b" tainted

student@lfs458-node-1a0a:~$ kubectl apply -f taint.yaml
```



```
deployment "taint-deployment" created

student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
35

student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
7
```

10. Remove the taint and delete the deployment. When you have determined that all the containers are running create the deployment again. Without any taint the containers should be spread across both nodes.

```
student@lfs458-node-1a0a:~$ kubectl delete deployment taint-deployment
deployment "taint-deployment" deleted

student@lfs458-node-1a0a:~$ kubectl taint nodes lfs458-node-2b2b bubba-
node "lfs458-node-2b2b" untainted

student@lfs458-node-1a0a:~$ kubectl apply -f taint.yaml
deployment "taint-deployment" created

student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
25

student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
17
```

11. Now use the `NoExecute` to taint the secondary node. Wait a minute then determine if the containers have moved. The DNS containers can take a while to shutdown.

```
student@lfs458-node-1a0a:~$ kubectl taint nodes lfs458-node-2b2b bubba=value:NoExecute
node "lfs458-node-2b2b" tainted
student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
35

student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
1
```

12. Remove the taint. Wait a minute. Note that all of the containers did not return to their previous placement.

```
student@lfs458-node-1a0a:~$ kubectl taint nodes lfs458-node-2b2b bubba-
node "lfs458-node-2b2b" untainted

student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
35

student@lfs458-node-2b2b:~$ sudo docker ps |wc -l
7
```

13. In addition to the ability to taint a node you can also set the status to drain. First view the status, then destroy the existing deployment. Note that the status reports Ready, even though it will not allow containers to be executed. Also note that the output mentioned that `DaemonSet`-managed pods are not affected by default.

    In v1.9.1 there appears to be an issue where the existing containers are not moved, but no new containers are created. You may receive an error `error: unable to drain node "<your node>", aborting command...`

```
student@lfs458-node-1a0a:~$ kubectl get nodes
NAME              STATUS    AGE       VERSION
lfs458-node-1a0a   Ready     master    2d        v1.9.1
lfs458-node-2b2b   Ready     <none>    2d        v1.9.1

student@lfs458-node-1a0a:~$ kubectl drain lfs458-node-2b2b
node "lfs458-node-2b2b" cordoned
error: DaemonSet-managed pods (use --ignore-daemonsets to ignore): kube-flannel-ds-fx3tx, kube-proxy-q2q4k
```


14. Verify the state change of the node. It should indicate no new Pods will be scheduled.

```
student@lfs458-node-1a0a:~$ kubectl get nodes
NAME              STATUS                     ROLES     AGE    VERSION
lfs458-node-1a0a   Ready                      master    2d     v1.9.1
lfs458-node-2b2b   Ready,SchedulingDisabled   <none>    2d     v1.9.1
```

15. Delete the deployment to destroy the current Pods.

```
student@lfs458-node-1a0a:~$ kubectl delete deployment taint-deployment
deployment "taint-deployment" deleted
```

16. Create the deployment again and determine where the containers have been deployed.

```
student@lfs458-node-1a0a:~$ kubectl apply -f taint.yaml
deployment "taint-deployment" created

student@lfs458-node-1a0a:~$ sudo docker ps |wc -l
35
```

17. Return the status to Ready, then destroy and create the deployment again. The containers should be spread across the nodes. Begin by removing the cordon on the node.

```
student@lfs458-node-1a0a:~$ kubectl uncordon lfs458-node-2b2b
node "lfs458-node-2b2b" uncordoned

student@lfs458-node-1a0a:~$ kubectl get nodes
NAME              STATUS    ROLES     AGE       VERSION
lfs458-node-1a0a   Ready     master    2d        v1.9.1
lfs458-node-2b2b   Ready     <none>    2d        v1.9.1
```

18. Delete and re-create the deployment.

```
student@lfs458-node-1a0a:~$ kubectl delete deployment taint-deployment
deployment "taint-deployment" deleted

student@lfs458-node-1a0a:~$ kubectl apply -f taint.yaml
deployment "taint-deployment" created
```

19. View the **docker ps** output again. Both nodes should have almost the same number of containers deployed. The master will have a few more, due to its role.

20. Remove the deployment a final time to free up resources.

```
student@lfs458-node-1a0a:~$ kubectl delete deployment taint-deployment
```

