Docker, Kubernetes, VMware Tanzu and RedHat OCP
14Oct2022
Day08
Kubernetes Workloads III

Workloads
https://kubernetes.io/docs/concepts/workloads/
A workload is an application running on Kubernetes.
1-Pod
2-ReplicaSet
3-Deployment
5-DaemonSet
6-Init Containers
7-Staticp Pod

DaemonSet
https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/
A DaemonSet ensures that all Nodes run a copy of a Pod.
As nodes are added to the cluster, Pods are added to them.
As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.
ex
```
# kubectl get nodes
# kubectl api-resources | grep -i "daemon"
daemonsets                    ds        apps/v1                     true       DaemonSet
# vim  rsyslogds.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: rsyslogds
spec:
  selector:
    matchLabels:
      name: rsyslog
  template:
    metadata:
      labels:
        name: rsyslog
    spec:
      containers:
        - name: rsyslogcnt
          image: rsyslog/syslog_appliance_alpine                    ->selected image rom specific registry
:wq!
# kubectl apply -f rsyslogds.yaml --dry-run=client
# kubectl apply -f rsyslogds.yaml
# watch kubectl get pods -o wide
# kubectl get ds
# kubectl describe ds rsyslogds
```

Init Containers
https://kubernetes.io/docs/concepts/workloads/pods/init-containers/
-initContainers: specialized containers that run before app containers in a Pod.
-initContainers can contain utilities or setup scripts not present in an app image.
-initContainers, which are run before the app containers are started.
-each initContainer must complete success before the next one will start.
ex
```
# vim initCnt.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  initContainers:
    - name: busyboxcnt
      image: busybox
  containers:
    - name: nginxcnt
      image: nginx
:wq!
# kubectl apply -f initCnt.yaml
# watch kubectl get pods
```

static Pods

https://kubernetes.io/docs/tasks/configure-pod-container/static-pod/

-Static Pods are managed directly by the kubelet daemon on a specific node, without the API server observing them.
-Unlike Pods that are managed by the control plane; instead, the kubelet watches each static Pod.
-default location for static pod is **/etc/kubernetes/manifests/**

How change StaticPod default location

in MasterNode/ControlPlane edit:

```
# vim /var/lib/kubelet/config.yaml
41 staticPodPath: /etc/kubernetes/manifests                    ->this path should be change
:wq!
```

Now, StaticPods will store in new location

ex

create StaticPod on Node3

```
root@master1:~# hostname
master1.example.com
root@master1:~#  kubectl get node -o wide
root@master1:~#  ssh root@node3.example.com
root@node3:~# hostname
node3.example.com
root@node3:~# echo "autocmd FileType yaml setlocal ai ts=2 sw=2 et cursorcolumn" >~/.vimrc
root@node3:~# cd /etc/kubernetes/manifests/
root@node3:/etc/kubernetes/manifests# pwd
/etc/kubernetes/manifests
root@node3:/etc/kubernetes/manifests# vim redis.yaml
kind: Pod
apiVersion: v1
metadata:
  name: redispod
spec:
  containers:
    - name: rediscnt
      image: redis
:wq!
# ls
redis.yaml
root@node3:~# exit
root@master1:~# kubectl get pods -o wide
NAME                         READY   STATUS      RESTARTS   AGE   IP          NODE
redispod-node3.example.com   1/1     Running     0          87s   10.85.0.5   node3.example.com
```

How assign Pod to specific Node

1-nodeSelector field
2-nodeName field
3-Affinity and anti-affinity

1-nodeSelector field

```
# kubectl get nodes node2.example.com --show-labels
# kubectl label nodes node2.example.com lbl=node
# kubectl run redispod --image redis -o yaml --dry-run=client >redispod.yaml
# vim redispod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: redispod
spec:
  containers:
  - image: redis
    name: rediscnt
  nodeSelector:
    lbl: node
:wq!
# kubectl create -f redispod.yaml
# kubectl get pods -o wide
redispod          1/1     Running        0       79s    10.85.0.5   node2.example.com
```

2-nodeName field

```
# kubectl run nginx-pod --image nginx -o yaml --dry-run=client >nginxpod.yaml
# vim nginxpod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginxpod
spec:
  containers:
  - image: nginx
    name: nginxcnt
  nodeName: node4.example.com
:wq!
# kubectl apply -f nginxpod.yaml --dry-run=client
# kubectl apply -f nginxpod.yaml
# kubectl get pods -o wide
nginxpod          1/1     Running        0       51s    10.85.0.3   node4.example.com
```

Mohammad Ali Naghval
ali@prodevans.com

Attach specific namespace to specific Node/Nodes

step1- enable **PodNodeSelector** on kube-api component
step2- create namespace
step3- label Node and Namespace

step1- Create Namespace

```
# kubectl create namespace test1 -o yaml --dry-run=client >test1.yaml
# ls
test1.yaml
# cat test1.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: test1
:wq!
# kubectl apply -f test1.yaml
# kubectl get ns
```

step2- enable **PodNodeSelector** on kube-api component

```
# vim /etc/kubernetes/manifests/kube-apiserver.yaml
20     - --enable-admission-plugins=PodNodeSelector
:wq!
# watch kubectl get nodes
```

step3- label Node and Namespace

add

```
# kubectl label nodes node5.example.com ns=test1
# kubectl get nodes node5.example.com --show-labels
```

remove

```
# kubectl label nodes node5.example.com ns-
# kubectl get nodes node5.example.com --show-labels
```

update label on Namespace

```
# vim test1.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: test1
  annotations:
    scheduler.alpha.kubernetes.io/node-selector: ns=test1
:wq!
# kubectl apply -f test1.yaml
```

verify

```
# kubectl run nginx --image nginx --namespace test1
# watch kubectl get pods --namespace test1 -o wide
nginx   1/1     Running   0        82s   10.85.0.3    node5.example.com
```

Mohammad Ali Naghval
ali@prodevans.com