Labs with Minicube - for practice and test

08:53 PM

Hands-On With Docker and Kubernetes

Minikube is a tool that makes it easy to run Kubernetes locally.

[minikube start | minikube (k8s.io)](https://minikube.sigs.k8s.io/docs/start/)
https://minikube.sigs.k8s.io/docs/start/

Allow all traffic to in nsg while creating vm.

Ubuntu

sudo su # change to root use sudo -i

Now install docker

sudo apt update && apt -y install docker.io

Sudo apt upgrade -y

install Kubectl

curl -LO https://storage.googleapis.com/kubern... -s https://storage.googleapis.com/kubern... && chmod +x ./kubectl && sudo mv ./kubectl /usr/local/bin/kubectl

Which kubectl

```
sudo apt-get update && sudo apt-get install -y apt-transport-https
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubectl
```

install Minikube

curl -Lo minikube https://storage.googleapis.com/miniku... && chmod +x minikube && sudo mv minikube /usr/local/bin/

Apt-get install conntrack # to work minikube properly , supported file

Minikube start --vm-drive=none // to start

Minikube status

Kubectl version

Kubectl get nodes

Your machine ip and details your will get

Kubectl Describe node node-ip // to see details of node

In kubertnete we write manifest - yml

Manifest format .yml/yaml
--- //optional
#

First pod

1 pod

****

```
kind: Pod
apiVersion: v1
metadata: #name of pos                    ""
  name: testpod
spec:
  containers:    # conatinerr in po; d
    - name: c00   # name of conato;ner
      image: ubuntu     #which command need to run once started
      command: ["/bin/bash", "-c", "while true; do echo Hello-World; sleep 5 ; done"]
  restartPolicy: Never        # Defaults to Always
```

Save and quite file

Kubectl create -f pod1.yml

kubectl apply -f pod1.yml // create pod and run container

Kubectl get pods // get details of pods

Kubectl get pods -o wide //get excact details where or which woker node it got create

Kctl describe pod namepod #testpod or

Kctl describe pod pod/namepod # pod/testpod

Kctl logs -f testpod # nameofof - to see logs inside pod

Kubeclt logs -f testpod -c c00 // to check inside container

Kubectl delete pod testpod # name of pod - to delete pod

Kubectl delete pod1.yml # delete by podfile

kubectl exec -i -t my-pod --container main-app -- /bin/bash // to go inside container in pod

Kubecelt exec -it testpod -c c02 -- /bin/bash ==#

Kubectl delete pod --all // going detel all pods
Kubectl delete pods --all

****

Annotation - add extra info

```
Metadata:
 name: testpod
Annotations:
  description: this is test message for other plp
```

MULTI CONTAINER POD ENVIRONMENT
Pod2.yml

#2

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod3
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo HTBS ; sleep 5 ; done"]
    - name: c01
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-world; sleep 5 ; done"]
```

2/2 // to containers in pod

```
Kubectl logs -f testpod3 -c c00

Kubectl exec testpod3 -it -c c00 -- /bin/bash # to go inside container
Kubectl exec testpod3 -it -c c00 -- hostname -i # to get ip address

Ps

Ps -ef // to check docker commands inside containers

Exit
```

-f pod2.yml delete by file name

****

POD ENVIRONMENT VARIABLES

Variable can get used inside conatiner

```
kind: Pod
apiVersion: v1
metadata:
  name: environments
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-wolrd; sleep 5 ; done"]
      env:                        # List of environment variables to be used inside the conatiner
       - name: MYNAME
         value: THBS
```

-p 8081:8

-itd --name

:wq

Create pod by command
Get pods

Get inside conatiner

Env # to list info about

Echo $MYNAME

Exit
Delete by filename or pod name

****

POD WITH PORTS

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod4
spec:
  containers:
    - name: c00
      image: httpd
      ports:
       - containerPort: 80
```

********************END**

Create pods

Kubectl get pods -o wide

Curl ip:80

Delete pod

Get pods

Object is the task or work you want to do.

Relationships between objects
- Pod manages container
- replicasets manages pods
- Services expose pod process to outside world
- Configsmaps and secrete help you to config pod

You create these and run with kubectl

State of the object
- Replicas
- Name
- Port
- Volume

Pod manifest file:

```
$ vi hello-pod.yml
apiVersion: v1
kind: Pod
metadata:
 name: hello-pod
 labels:
 zones: prod
 version: v1
spec:
 containers:
 - name: hello-ctr
 image: nigelpoulton/pluralsight-docker-ci:latest
 ports:
 - containerPort: 8080
```

```
$ kubectl create -f hello-pod.yml
```

pod/hello-pod created

To access our hello-pod/ Replication Controller /Deployment we need to expose the pods though a service:

kubectl expose

When a pod is created, without a service, we cannot access to the app running within container in the pod. The most obvious way is to create a service for the pod either via Load Balancer or NodePort.

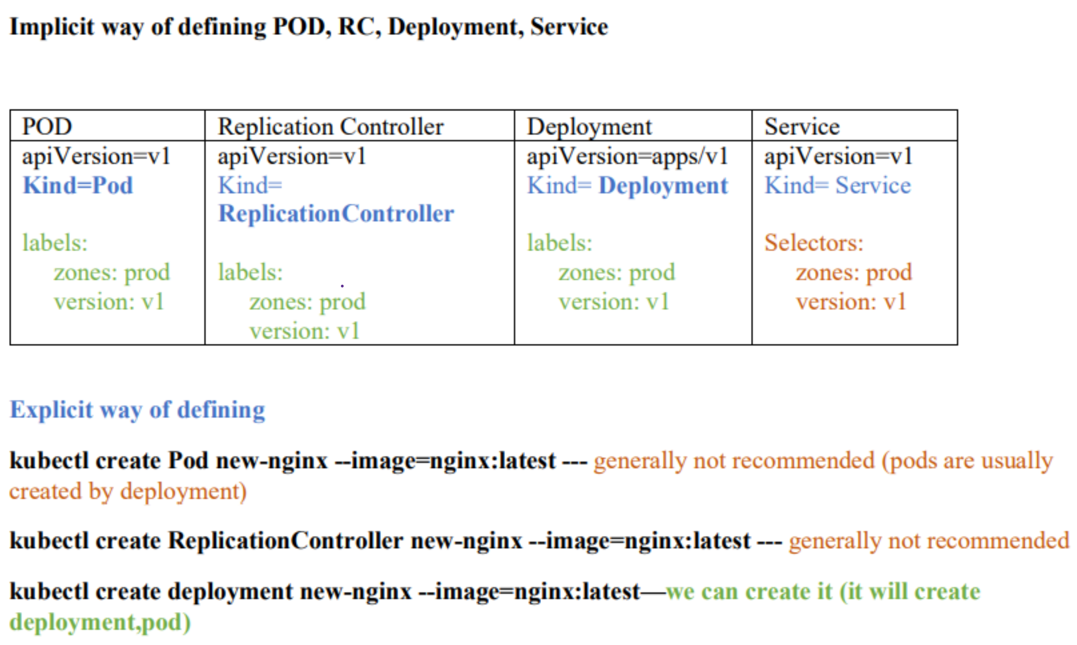
```
$ kubectl expose pod hello-pod --type=NodePort --target-port=80 -o yaml
$ kubectl describe pod hello-pod | grep -i ip
IP: 10.244.0.83
$ curl http://localhost:30779
```

http://10.244.0.83:30779/

```
$ kubectl get svc hello-pod -o wide

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR
hello-pode NodePort 10.107.83.213 <none> 8080:30779/TCP 14m version=v1,zones=prod
```

Kubetnets object management
- Direct command line
- Or write manifest file and run



Implicit way of defining POD, RC, Deployment, Service

POD	Replication Controller	Deployment	Service
apiVersion=v1 **Kind=Pod** labels: zones: prod version: v1	apiVersion=v1 Kind= **ReplicationController** labels: zones: prod version: v1	apiVersion=apps/v1 Kind= **Deployment** labels: zones: prod version: v1	apiVersion=v1 Kind= Service Selectors: zones: prod version: v1

Explicit way of defining

kubectl create Pod new-nginx --image=nginx:latest --- generally not recommended (pods are usually created by deployment)

kubectl create ReplicationController new-nginx --image=nginx:latest --- generally not recommended

kubectl create deployment new-nginx --image=nginx:latest—we can create it (it will create deployment,pod)

Kubectl run nginx --image=nginx --port=80 --restart=Never // create pod with cli