K8's - 4

-----------

Kubectl Commands Syntax:

-----------------------------------------

kubectl <Operation\_command> <Type\_command> <Name\_command>

- Operations:- Get, Create, Delete, Describe, Logs, etc

- Type:- Pods, Deployments, Jobs, Namespace, etc

- Name:- Seach-pod, Cart-Deployment, Ui-Service, mail-jobs, etc

eg:- kubectl get pod test-pod

Steps:

---------

1. To Check whether my kops has been installed we can use (kops validate cluster)

2. To Check the namespaces use (kubectl get namespace)

3. To Check the pods in default namespace use (kubectl get pods -n default) [Note: -n is defining namespace]

4. To Check the pods in kube-system use (kubectl get pods -n kube-system). If need to know the pods are running which machine

we can go with (kubectl get pods -n kube-system -o wide)

=========================================================================

There are two ways in which we can create the resources in K8's:

---------------------------------------------------------------------------------------

Imperative way ( CLI )

- kubectl commands

Declarative way

- yaml file

[Note: For system to system communication the language will be Yaml & JSON]

-----------------------------------------------------------------------------------------

Kubernetes Docs Link:

---------------------------------

Kubernetes API reference docs --> https://kubernetes.io/docs/reference/

Kubernetes Cheat Sheet --> https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Kubernetes Troubleshooting Guide --> https://learnk8s.io/troubleshooting-deployments

==========================================================================

For Creating a Namespace using yaml:

-----------------------------------------------------

apiVersion: v1

kind: Namespace

metadata:

name: <insert-namespace-name-here>

To execute this yaml file use (kubectl apply -f <yaml file name>)

==========================================================================

PODS:

--------

- Single Container on a Namespace

- Multiple Container with Error

- Request & Limit

--------------------------------------------------

Single Container on a Namespace

For Creating a Pod:

-------------------------

apiVersion: v1

kind: Pod

metadata:

name: mypod

spec:

containers:

- name: cont1

image: httpd

ports:

- containerPort: 80

After kubectl apply.

For Checking the pods use (kukbectl get pods -n <namespace name>)

-------------------------------------------------------------------------------------------------

For Creating a Pod with Namespace:

--------------------------------------------------

apiVersion: v1

kind: Pod

metadata:

name: mypod

namespace: green

spec:

containers:

- name: cont1

image: httpd

ports:

- containerPort: 80

After apply my pod will be created and running in the namespace we have given.

1.To check the pods in all namespaces use (kubectl get pods --all-namespaces)

2. To specifically describe the pod use (kubectl describe pod <pod name> -n <namespace name>)

3. To check the logs of pod use (kubectl logs <pod name> -c <cont name> -n <namespace name>)

4. To check the extra information of a Pod use ( kuebcet get pods -n default -o wide )

[NOTE: HOW WILL U TROUBLESHOOT A POD?]

-------------------------------------------------------------------------------------

Multiple Container with Error

apiVersion: v1

kind: Pod

metadata:

name: mypod

spec:

containers:

- name: cont1

image: httpd

ports:

- containerPort: 80

- name: cont2

image: ngnix

ports:

- containerPort: 80

[NOTE: We need to check the pods describtion and the logs]

--------------------------------------------------------------------------------------------

Request & Limit is based upon the CPU & Memory we set to run the Pod

apiVersion: v1

kind: Pod

metadata:

name: mypod

spec:

containers:

- name: cont1

image: httpd

ports:

- containerport: 80

resources:

requests:

memory: "64Mi"

cpu: "200m"

limits:

memory: "128Mi"

cpu: "400m"

[NOTE: TO DELETE POD USE (kubectl delete pod <pod name> -n <namespace name>)]

=============================================================================

ReplicaSet:

----------------

- ReplicaSet --> ReplicaSet + Pods

- no of replicas --> no of pods

To delete replicaset use (kubectl delete replicaset <replicasetname>)

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: replicaset-example

spec:

replicas: 3

selector:

matchLabels:

app: httpd

template:

metadata:

labels:

app: httpd

spec:

containers:

- name: httpd

image: httpd

ports:

- containerPort: 80

DaemonSet:

-----------------

- DaemonSet --> DaemonSet + Pods

- no of w.nodes --> no of Pods

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: daemonset-example

spec:

selector:

matchLabels:

app: httpd

template:

metadata:

labels:

app: httpd

spec:

containers:

- name: httpd

image: httpd

ports:

- containerPort: 80

Deployment:

------------------

- Deployment + ReplicaSet + Pods

For deployment we can use replicaset yaml file.

apiVersion: apps/v1

kind: Deployment

metadata:

name: deployment-example

spec:

replicas: 3

selector:

matchLabels:

app: httpd

template:

metadata:

labels:

app: httpd

spec:

containers:

- name: httpd

image: httpd

ports:

- containerPort: 80

Rolling update:

----------------------

3 nginix

3 nginix + 1 httpd

2 nginix + 2 httpd

1 nginix + 3 httpd

3 httpd

For rolling update just change the image and again run the yaml file.

apiVersion: apps/v1

kind: Deployment

metadata:

name: deployment-example

spec:

replicas: 3

selector:

matchLabels:

app: ngnix

template:

metadata:

labels:

app: ngnix

spec:

containers:

- name: ngnix

image: httpd

ports:

- containerPort: 80

[NOTE: WE ROLL BACK THE DEPLOYMENT IF WE NEED USING K8's CHEAT SHEET]

(kubectl rollout undo deployment/<deployment name>)