**KUBERNATES**

https://www.youtube.com/watch?v=6uvHVVNq34w&list=PLMPZQTftRCS8Pp4wiiUruly5ODScvAwcQ&index=18

Kubectl syntax

**kubectl [command] [resource type] [resource name] [flag]**

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

**Flags**

-f

-o : output

-l : label

-c : container name

-record : record the command in history

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

kubectl create -f pod.yml

Kubectl create -f < directoty name>

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

Kubectl get [resource type] [Name] [flag]

kubectl get pods

this will display high level infomation about the pods

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

Kubectl describe [type] [name] -- display complete information 1 or more resources.

kubectl describe pods

kubectl describe pods <pod-name>

kubectl describe node <node-name>

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

kubectl delete pods <pod name>

kubectl delete pods,servcie -l name=<labelname>

-l flag is for Label

kubectl delete pods -all

it will delete all the pods

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

Kubectl exec --> execute a command against a container inside a pod

kubectl exec <pod-name> date ( when there is one container inside a pod)

kubectl exec <pod-name> -c <conatainer name > date ( when there is more than one conatainer in pod)

kubectl exec <pod-name> -it <pod-name> /bin/bash ( to get inside container)

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

kubectl log pod-name

kubectl log -f pod-name

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

**PODS :-**

Continer inside pod share same IP address ,same volumes

all pods can communicate with each other

**Pod life cycle:-**

we write the mainfeast file and submit it to API server.

it will create container using Mainfest file.

it will be in pending state until all conatiners is created.

once all conatiners are up and running it will go to running state.

once its work its completed it go to successded state.

if any failure happens it will go to failed state.

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

Pod Config

#nginix-pod.yaml

apiVersion: v1

kind: Pod

metadta:

name: nginix

label:

app: nginix

tier: dev

spec:

container:

- name: nginix-container

image: nginix

Command to excuete (youtube vdeio 21:00)

1) kubectl create -f niginx-pod.ymal

2) kubectl get pods

3) kubectl get pod -o wide ( show wide output or more option like Ipaddress , nOde)

4) kubectl get pod nginix-pod -o yaml ( it will give detail in yaml format)

kubectl get pod nginix-pod -o yaml | more

5) kubectl decribe pod nginix-pod ( detailed output of nginx pod

6) kubectl exec -it pod nginix-pod -- /bin/bash ( get inside continer)

run some commnds

hostname

exit

7) kubectl expose pod nginix-pod --type=NodePort --port:80 ( Port forwarding)

8)kubectl delete pod nginix-pod

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

Replication controller

It ensure specefic number ( mentioned in yaml file) of pods are always running. If there excess pods it gets killed and vice versa.

Replication COntroller and Pods are assosciated with Labels

Advantages :

High availability

Load balaancing

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

Deployement

Features

Multiple replicas

upgrade

rollback

scale up or down

pause and resume

Deployement types

1) Recreate :- its is a dummy deployemnt process. in this we will first shutdown version A and once version A is shutdown,

then we will deploy version B. while switching from versio A to version B there will be a down time of server

2) Rolling update:- in this , it will slowly update the version of the app by replacing instances one after the other until

all instances are succesfully rolled out

default stragey , easy to use but take some time.

3)Canary deployemnt:- it is gradually shifting form vesion A to Verion B

4) Blue/green deployemnt:- here blue means version A and Green means Verison B. we will create exact same no of version B instance

and shify the traffic to version B form version A.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Deployment mainfest file.

#niginix-deploy.yaml

apiVersion: apps/v1

kind: Deployment

metadta:

name: nginix-deploy

label:

app: nginix-app

spec:

replica: 3

selector:

matchLabels: nginix-app

template:

metadata:

label:

app: niginix-app

spec:

container:

- name: nginix-container

image: nginix

port:

- containerPort: 80

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Commands:

1 kubectl create -f niginix-deploy.yaml

2) kubectl get deploy -l app=niginix-app

3) kubectl get rs -l app=niginix-app ( dispaly replica set)

4) kubectl get pods -l app=niginix-app

5) kubectl describe deploy nginix-deployemnt

------update Deployemnt----------------

there are two ways to update it

1) SET command

kubectl set image deploy nginx deployement ngnix-conatiner=nginix:1.9.1

2) edit command

kubectl edit deploy nginx-deployement

in this configuration will open in Vi editor , make the changes and close it, configuraton will save automatically

check the status using

kuebctl rollout status deployemnt/nginx-deployment

kubectl get deploy

------ RollBack Deployemnt-------------------------

suppose if we gave the wrong version number as 1.9.1 as 1.91 then

1) kubectl set image deploy nginx deployement ngnix-conatiner=nginix:1.91 --record

record will record this command in history

once the command is run output shows that image is updated.

2)kubectl rollout status deployemnt/nginx-deployment

but when you check the statuus it will show that it is stuck or waiting for somethibg

3) kubectl rollout history deployemnt/nginx-deployemnt

to check which commandwe have executed earlier

we will notice that incorrect version number. to undo this deployment will execute below command

4) kubectl rollout undo deployment/nginix-deployment

now again you can check the status

5)kubectl rollout status deployemnt/nginx-deployment

Scale Up and Scale Down

1) kubectl scale deployment nginix-deployment --replicas=5

it will scale deployment form 3 to 5

2) kubectl get deploy

3) kubectl het pods

4) kubectl scale deployment nginix-deployment --replicas=2

it will scale deployment form 5 to 2

Delete

Kubectl delete -g nginx-deploy.yaml

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

5) kubectl get deploy

6) kubectl describe deploy

7) kubectl get rs

8) kubectl apply -f deploy.yml --record

9) kubectl rollout status deployment sample-deploy

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

Horizontal scaling means that you scale by adding more machines

into your pool of resources whereas Vertical scaling means that you scale by adding more power (CPU, RAM) to an existing machine.

deployment object

scling up is esy

exposing to external world is esy

rolling updtae

service object

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

Kubernates architecture

Kubernates master

1)managing each operatiion and node.

2) checking each node is healthy , pod is healthy ,scheduling deployemnet of pod , controlling replicas.

-Master components:

-API server : API is used to make communication between two pod or two service in kubernates. to manage all this API's we

have API server . it will manage communication between different component of KUBERNATES.

it isimp component in Kubernates cluster. it is front end component of Kubernates master. we define our cluster in yaml file.

API server will validate that file and deploy continer to cluster.

-etcd : It is a place where you store your password. it is store data in form of key value pair.

-Kube controller mnager : handle all the controller that is present in kubernates.

node controller : it will check wheather a node is created , in progress or ready state.

replica controller : it will ensure that actual state is equal to desired state.

it is having many controller in cluster which will check wheather desired state of cluster is

equal to actual state of cluster.if desired state is not equal to actual state . controller manager will run watch loop at

backend and bring desired stte equl to actual state.

-kube scheduler : it will distribute the work load to nodes in kubernates cluster. example which job needs to run on which node.

-clould controller manager : making kubenates cluster avaiable to hardware of the cloud. it act as bridge between the

aws hardware ( sevices how aws works ) and kubernates API ( how thwy will work).

-Kubernates Nodes

- Kubelet : it is major compont .it is kubernates agent on the node.it will interct with api server and deploy the contaner.

it will take work load from kubernates master

-Kubeproxy : it will see the newtwork aspect betwen user and container.it will check netwrk

connection betwee user and contaienr how they are expose to external world.

-cAdvisor : container run time environment. which container we are using example DOcker , Rocket.

- pod: very imp. continer will run inside pod. it will create runtime env for container. provide layer of abstarction.

as standard we will run 1 container per pod. we can run many.

-kubectl : its a way of acesing the cluster. it is a command line tool.