Kubernetes

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

Menions: This is an individual node used in Kubernetes.

Combination of these minions is called as Kubernetes cluster

Master is the main machine which triggers the container orchestration.

It distributes the work load to the Slaves.

Slaves are the nodes that accept the work load from the master

and handle activities load balancing, auto scaling, high availability etc

Kubernetes uses various of types of Object

1 Pod: This is a layer of abstraction on top of a container. This is the smallest

object that kubernetes can work on. In the Pod we have a container.

The advantage of using a Pod is that kubectl commands will work on the Pod and the Pod communicates these instructions to the container. In this way we can use the same kubectl irrespective of which technology containers are in the Pod.

2 Service: This is used for port mapping and network load balancing

3 NameSpace: This is used for creating partitions in the cluster. Pods running

in a namespace cannot communicate with other pods running in other namespace

4 Secrets: This is used for passing encrypted data to the Pods

5 ReplicationController: This is used for managing multiple replicas of PODs

and also performing scaling

6 ReplicaSet: This is similar to replicationcontroller but it is more advanced

where features like selector can be implemented

7 Deployment: This used for performing all activities that a Replicaset can do

it can also handle rolling update

8 Volume: Used to preserve the data even when the pods are deleted

9 Statefulsets: These are used to handle stateful application like data bases

where consistency in read write operations has to be maintained.

Setup of Kubernetes

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

Free

===========

1 http://katakoda.com

(or)

2 http://playwithk8s.com

Paid

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

1 Signup for a Google cloud account

2 Click on Menu icon on top right corner--->Click on Kubernetes Engine-->Clusters

3 Click on Create cluster--->Click on Create

|  |  |
| --- | --- |
| MASTER | SLAVE |
| Container run time | Container run time |
| Kubelet | Kubelet |
| Api server | Kubeproxy |
| Schedular |  |
| Controller |  |
| etcd |  |

Day 23

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

Manual setup of Kubernetes on centos/RHEL

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

Install, start and enable docker service

yum install -y -q yum-utils device-mapper-persistent-data lvm2 > /dev/null 2>&1

yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo > /dev/null 2>&1

yum install -y -q docker-ce >/dev/null 2>&1

systemctl start docker

systemctl enable docker

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

Disable SELINUX🡪 it’s a security feature which prevents you from installing some softwere. We need to disable this.

setenforce 0

sed -i --follow-symlinks 's/^SELINUX=enforcing/SELINUX=disabled/' /etc/sysconfig/selinux

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

Disable SWAP

sed -i '/swap/d' /etc/fstab

swapoff -a

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

Update sysctl settings for Kubernetes networking : This installs necessary firewalls for kubernetes

cat >>/etc/sysctl.d/kubernetes.conf<<EOF

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sysctl --system

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

Add Kubernetes to yum repository

cat >>/etc/yum.repos.d/kubernetes.repo<<EOF

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg

https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

EOF

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

Install Kubernetes

yum install -y kubeadm kubelet kubectl

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

Enable and start Kubernetes service

systemctl start kubelet

systemctl enable kubelet

vim /etc/hostname

remove content and give names master and worker1 for our convenience

init 6

to restart

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

Repeat the above steps on Master and slaves

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

On Master=============

===========

Initilise the Kubernetes cluster

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

kubeadm init --apiserver-advertise-address=private\_ip\_of\_master --pod-network-cidr=192.168.0.0/16

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

To be able to use kubectl command to connect and interact with the cluster,

the user needs kube config file.

mkdir /home/centos/.kube

sudo cp /etc/kubernetes/admin.conf /home/centos/.kube/config

sudo chown -R centos:centos /home/centos/.kube

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

Deploy calico network

Exit from root

kubectl create -f https://docs.projectcalico.org/v3.9/manifests/calico.yaml

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

For slaves to join the cluster

kubeadm token create --print-join-command

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

Check the pods of kube-system are running

kubectl get pods -n kube-system

\*note: when we stop and start ec2 machines we need to fire following commands:

Master:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

Slave:

kubeadm reset

systemctl restart kubelet

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

Day 24

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

Kubernetes Architecture

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

Master Componentes

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

Container runtime: This can be docker or any other container technology

apiServer: Users interact with the apiServer using some client like ui, command line tool like kubelet. It is the apiServer which is the gateway to the cluster.

It works as a gatekeeper for authentication and it validates if a specific

user is having permissions to execute a specific command. Example if we want to deploy a pod or a deployment first apiServers validates if the user is authorised to perform that action and if so it passes to the next process i.e the "Scheduler".

Scheduler: This process accepts the instructions from apiServer after validation

and starts an application on a specific node or set of nodes. It estimates

how much amount of h/w is required for an application and then checks which

slave have the necessary h/w resources and instructs the kubelet to deploy

the application.

kubelet: This is the actual process that takes the orders from scheduler and

deploy an application on a slave. This kubelet is present on both master and slave.

controller manager: This check if the desired state of the cluster is always

maintained. If a pod dies it recreates that pod to maintain the desired state.

etcd: Here the cluster state is maintained in key value pairs.

It maintains info about the slaves and the h/w resources available on

the slaves and also the pods running on the slaves.

The scheduler and the control manager read the info from this etcd

and schedule the pods and maintain the desired state

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

Worker components

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

containerrun time: Docker or some other container technology

kubelet: This process interacts with container run time and the node

and it start a pod with a container in it

kubeproxy: This will take the request from services to pod

It has the intelligence to forward a request to

a near by pod. Eg If an application pod wants to communicate with a db pod

then kubeproxy will take that request to the nearby pod

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

UseCase

===========

Create nginx as a pod and name it webserver

kubectl run --image nginx webserver

To see the list of pods running

kubectl get pods

To see more info about the pods like their ip and slave where they are running

kubectl get pods -o wide

To delete the pod

kubectl delete pods webserver

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

UseCase

=========

Create mysql pod and name it mydb and go into its interactive terminal and create few tables

kubectl run --image mysql:5 mydb --env MYSQL\_ROOT\_PASSWORD=intelliqit

To check the pods

kubectl get pods

To go into the interactive terminal

kubectl exec -it mydb -- bash

To login into the db

mysql -u root -p

Password: intellqiit

Create tables here