Introductions of the Kubernetes

1. Introductions

Kubernetes is a powerful open-source system, initially developed by Google, for managing containerized applications in a clustered environment. It aims to provide better ways of managing related, distributed components and services across varied infrastructure.

2. What is Kubernetes?

Kubernetes is basic leave, a system for running and coordinating containerized applications across the cluster of system. It's a platform designed to completely managed the containerized applications and services using methods that provide predictability scalability ans high available As Kubernetes used, can define how the applications should run and the way that should be able to interact with other applications. You can scale up or down, perform graceful rolling update and switch traffic between version of the applications to test features or rollback deployment. Kubernetes provides interfaces and composable platform primitives that allow to define and manage the applications with high degrees of flexibility, power, and reliability.

3. Kubernetes Architecture.

Kubernetes is visualized as a system build in layer, as a base Kubernetes bring together individual system(physical or virtual machines) into the cluster using share the network to communication between each server. This cluster is the physical platform where all kubernetes components, capabilities and workloads are configured.

i. Master server: its act as the primary control plane for Kubernetes cluster & serves as the main contact point for administrators and users, and also provides many cluster-wide systems for the relatively unsophisticated worker nodes. Overall, the components on the master server work together to accept user requests, determine the best ways to schedule workload containers, authenticate clients and nodes, adjust cluster-wide networking, and manage scaling and health checking responsibilities.
ii. Node server: in Kubernetes, servers which perform work to running containers are know as nodes. node servers have a few requirements that are necessary for communicating with master components, configuring the container networking, and running the actual workloads assigned to them.

4. Installation of the Kubernetes.

To install the kubernetesneed min 2 system, one system for master cluster & other one for slave node service

Need to install the below package in all system

a) "Kubeadm" the command to bootstrap the cluster.
b) "kubelet" the component the run on all of the machines in cluster and done think like starting pods and containers.
c) "kubectl" the command line util to talk to cluster

Note: kubeadm will not install or manage kubelet or kubectl

5. Installations step for Kubernetes
   a) Disable the selinux & swap

```
$ setenforce 0
 $ sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

```
$ swapoff -a
```

b) Configure Kubernetes repository

For Centos Linux

```
$ cat <<EOF > /etc/yum.repos.d/kubernetes.repo
> [kubernetes]
> name=Kubernetes
> baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
> enable=1
> gpgcheak=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
```

For Ubuntu Linux

```
$ apt-get update && apt-get install -y
& apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
& cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
> deb https://apt.kubernetes.io/ kubernetes-xenial main
> EOF
$ apt-get update
```

c) Before install the kubernetes. We need to install the docker-ce(version 18.6)

d) After install the docker services, now install the kubernetes

For centos Linux

```
$ yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
```

For Ubuntu Linux

```
$ apt-get purge kubeadm kubectl kubelet kubernetes-cni kube*
```

e) After install the kubernetes now enable the service

```
$ systemctl enable --now kubelet
```

f) To install the master or cluster

Note: below step done only in master server

```
$ kubeadm init
```

g) To make kubectl work for your non-root user, run these commands, which are also part of the kubeadm init output

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

h) Apply the pods network in the kubernetes master server only

```
$ kubectl apply -f
https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/rbac-kdd.yaml
$ kubectl apply -f
https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/kubernetes-datastore/calico-networking/1.7/calico.yaml

                    (or)
$ export kubever=$(kubectl version | base64 | tr -d '\n')
$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"
```

i) Ones the pods network install in master server, confirm that it's working by checking that the coreDNS pod is running in output

```
$ kubectl get pods --all-namespaces
```

Ones the coreDNS is up and running, then continue by joining the node

j) To join the nodes to master cluster, in node server execute step 'a' to 'e'
Note: when run the command "kubeadm init" in master server below command in the out put

```
You can now join any number of machines by running the following on each nodes
as root:

  kubeadm join 192.168.4.150:6443 --token 6dzi0s.czg16b1atjocijgo --discovery-token-ca-cert-hash
sha256:8edf00acf5d77f3590d23b482fcb9b48fd103bd0d46d8e769399fef5728503a5
```

The above command "kubeadm join <>" run on the node machines,

k) For new token created

```
$ kubeadm token create    ---to create new token
$ kubeadm token list      ---to check the list of token
```

l) To check the cluster information

```
$ kubectl cluster-info
```

m) To check the node status

```
$ kubectl get nodes
```

n) To check the all service, deployment, pods & nodes

```
$ kubectl get all
```

Output

```
[root@centos bharath]# kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/tomcat-deployment-68ddf7989f-q8l2x   1/1     Running   0          10m

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP          44m
service/tomcat       NodePort    10.108.201.108   <none>        8080:30863/TCP   9m31s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/tomcat-deployment   1/1     1            1           10m

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/tomcat-deployment-68ddf7989f   1         1         1       10m
```

o) To access the tomcat service in the web browser

```
http://192.168.5.96:30863/
```

p) To reset the cluster or clear the cluster configure

```
$ kubeadm reset
```

q) To delete the node from the master

```
$ kubectl delete nodes
```

r) To deploy the application in the Kubernetes Container, need to build the docker images, for that pull the docker images from docker hub or repository through docker command or build the images using the Dockerfile

```
$ docker pull <images name>
$ docker build -t <name:tag> <dockerfile parth> ---build the images through dockerfile.
```

s) Created the deployment & service yaml file,

Sample for deployment & service yaml file below

Deployment.yml file sample

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: tomcat
spec:
  selector:
    matchLabels:
      app: tomcat
  replicas: 1 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: 192.168.5.96:5000/tomcat:8.5.38
        ports:
        - containerPort: 8080
        imagePullSecrets:
        - name: my-registry
```

Service,yml file sample

```
apiVersion: v1
kind: Service
metadata:
  name: tomcat
spec:
  selector:
    app: tomcat
  ports:
    - protocol: "TCP"
      port: 8080
      targetPort: 8080
  type: NodePort
```

Note: we creates the deployment & service yaml in single also.

t) To created the deployment in kubernetes

```
$ kubectl create -f < of the yaml file/deployment.yml>
```

u) To created the service in kubernetes

```
$ kubectl create -f < of the yaml file/service.yml>
```

v) To execute two or more yaml file

```
$ kubectl create -f < of the yaml file/service.yml> < of the yaml file/service.yml>
```

w) To export the services or port to access the container

```
$ kubectl export service <container name> --port=<app port no>
```

x) To run the image without yaml

```
$ kubectl run --image=<image name> <container name> --port=<port no> --env=<cluster or
nodes>(“DOMAIN=node1”)
```

y) To deleted the kubernetes container

```
$ kubectl deleted deployment <container name>
```

z) To check the logs of the container

```
$ kubectl logs -f <container name>
```