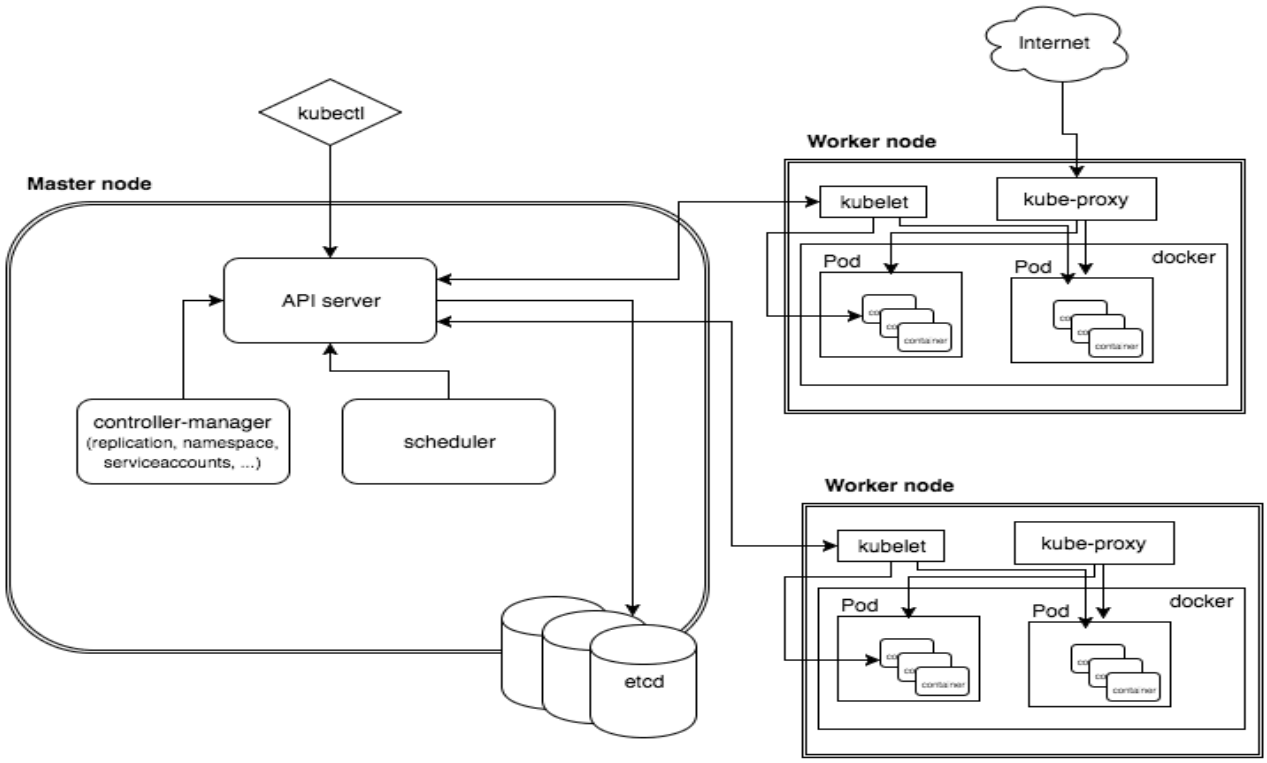
Kubernetes System Deploy



######### Kubernetes-Cluster-Master-and-Node on Centos - 8 ########

Describe Kubernetes Cluster Master and Worker Node,

First off all we create a Three Virtual Machine One for Master Node and two [2] Worker Node.######

****My Site****

Master ip = 192.168.2.121

Master Hostname = k8master.pc

and

Worker Node01 ip = 192.168.2.122

Worker Node02 ip = 192.168.2.123

Worker Node01 Hostname = k8snode01.pc

Worker Node02 Hostname = k8snode02.pc

Kubernetes System Deploy

Step [1] First Off your swap mamery

$ vi /etc/fstab

disable swap Copy and paste the line if already swap memory inclue your machine you Just Change comment [#]

#/dev/mapper/cl-swap swap swap defaults 0 0

$ swapoff -a

Check It's Disable or not

$ free -m

Step [2] Stop firewalld & disable firewalld service

$ systemctl stop firewalld && systemctl disable firewalld

Check Status

$ systemctl status firewalld

Step [3] Disable selinux service

$ vi /etc/selinux/config

Line Number 7 Change the comment [disabled]

SELINUX=disabled

**This module is required to enable transparent masquerading and to facilitate

Virtual Extensible LAN (VxLAN) traffic for communication between Kubernetes pods across the cluster**

$ modprobe br_netfilter

Step [4] Change your Hostname

$ hostname

$ hostnamectl set-hostname k8master.pc

Config your Hostname [192.168.2.212 k8master.pc]

$ vi /etc/hosts

Now reboot your Host PC

Kubernetes System Deploy

Step [5] Install Docker Engine on Your Host Machine

Step1: IF install old Docker engine first Uninstall old versions

$ sudo yum remove docker \

docker-client \

docker-client-latest \

docker-common \

docker-latest \

docker-latest-logrotate \

docker-logrotate \

docker-engine

Step2: Installation methods

1. Install using the repository

$ sudo yum install -y yum-utils

$ sudo yum-config-manager \

--add-repo \

https://download.docker.com/linux/centos/docker-ce.repo

Install Docker Engine

Install the latest version of Docker Engine and Container

$ sudo yum install -y docker-ce docker-ce-cli containerd.io

Start Docker Service

$ systemctl enable docker && systemctl start docker

Check Docker Service

$ service docker status

Kubernetes System Deploy

Check Docker Version

$ docker version

Check Docker Container running

$ docker ps

Check Docker Container all

$ docker ps -a

Create a Directory

$ mkdir /etc/docker

Copy and Paste the Commande / file

```
$ cat > /etc/docker/daemon.json <<EOF
  {
    "exec-opts": ["native.cgroupdriver=systemd"],
    "log-driver": "json-file",
    "log-opts": {
    "max-size": "100m"
  },
    "storage-driver": "overlay2",
    "storage-opts": [
    "overlay2.override_kernel_check=true"
     ]
     }
EOF
```

Now Again create a Directory

$ mkdir -p /etc/systemd/system/docker.service.d

Reload daemon

$ systemctl daemon-reload

Kubernetes System Deploy

Step [6] Install kubernetes on linux machine

****Copy and Paste the Command / file****

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


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

****Install Command on Kubernetes****

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


cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
```

****Chweck sysctl System****

```
$ sysctl --system
```

****Enable Kubelet****

```
$ systemctl enable kubelet
```

***** Note: This Step Follow All Three Machine, Master and Worker Node *****

Kubernetes System Deploy

########## Task to do in only Master node ########## If you can choice Option anyone

$ kubeadm init --apiserver-advertise-address=192.168.2.121 --pod-network-cidr=10.244.0.0/16

or

$ kubeadm init --control-plane-endpoint "192.168.2.212:6443" --pod-network-cidr=10.244.0.0/16

or

$ kubeadm init --apiserver-advertise-address=192.168.2.212 --pod-network-cidr=192.168.0.0/16 --service-cidr=192.168.1.0/24

***Note: Take a few minute

Copy All file Your Txt file (Example below)

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube

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

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

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.2.121:6443 --token p8r1a2.msnjnqjh36ft443w \

--discovery-token-ca-cert-hash sha256:33e34bfcbe5d0e1a9a58757e1534eb90ef1076625e9d15c652425d921e4a8e91

And Copy token (Example below) For Worker Node uses

kubeadm join 192.168.2.121:6443 --token p8r1a2.msnjnqjh36ft443w \

--discovery-token-ca-cert-hash sha256:33e34bfcbe5d0e1a9a58757e1534eb90ef1076625e9d15c652425d921e4a8e91

Now Run the Command and process

$ mkdir -p $HOME/.kube

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

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

Kubernetes System Deploy

Configure Pod Network with Flannel

$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/2140ac876ef134e0ed5af15c65e414cf26827915/Documentation/kube-flannel.yml

Untainted master

$ kubectl taint nodes --all node-role.kubernetes.io/master-

Check Nodes are Master Machine

$ kubectl get nodes -A

Check all pods in service

$ kubectl get pods --all-namespaces -o wide

Check all nodes in service

$ kubectl get nodes --all-namespaces -o wide

pode delete command

$ kubectl delete deployment <kubernetes-dashboard> --namespace=<kubernetes-dashboard>

$ kubectl delete deployment <dashboard-metrics-scraper> --namespace=<kubernetes-dashboard>

Task to do in Worker Node

1. worker node connect with cluster using token.(Copy and Paste)

Example: kubeadm join 192.168.2.121:6443 --token ab9lu9.l9pmvfid0lbx0ska \

--discovery-token-ca-cert-hash sha256:1f91ec32fcb126f55c3b9a7c613fbd8570e08bd4c14e0a14d3043a6c6a15963e

Kubernetes System Deploy

###For worker node when problem####

swapoff -a

kubeadm reset --force

reboot

swapoff -a

kubeadm reset Command

$ kubeadm reset

Install Kubernetes Dash-Board

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.2.0/aio/deploy/recommended.yaml

Create Admin User

$ kubectl create serviceaccount -n kubernetes-dashboard admin-user

Create yml file and Define role

$ vi rbac.yml

Copy and paste the yml file

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

Kubernetes System Deploy

Apply / run yml file

$ kubectl apply -f rbac.yml

confirm security token of the account

$ kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')

if access from other client hosts, set port-forwarding

$ kubectl -n kubernetes-dashboard edit service kubernetes-dashboard

***** Just Change [ClusterIP to NodePort]**

Show Services Port

$ kubectl get services --all-namespaces

Show all namespaces service

kubernetes-dashboard kubernetes-dashboard NodePort 10.103.154.32 <none> 443:30690/TCP 6m57s

Now Hit your Web Browser

https://192.168.2.121:30690

Or

You run kube-proxy

$ kubectl proxy

Now Hit your Web Browser

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/