Kubernets K8S

- CentOS 7.7 CentOS 7 64
- Docker

Kubernetes

- K8S
- https://kubernetes.io/
-
  - ``
  - ``
  - ``
  - ``
  - ``
  - ``
- Master Node

- Kubernetes 1.13

- 2C2G 2C4G
- 1.13 1.13
- kubeadm
-
  - issues https://github.com/kubernetes/kubeadm
  - https://github.com/kubernetes/kubernetes/tree/master/cmd/kubeadm
  - https://kubernetes.io/docs/setup/independent/install-kubeadm/
  - https://kubernetes.io/docs/setup/independent/install-kubeadm/#before-you-begin
  - https://kubernetes.io/docs/setup/independent/install-kubeadm/#verify-the-mac-address-and-product-uuid-are-unique-for-every-node
  - https://kubernetes.io/docs/setup/independent/install-kubeadm/#check-required-ports
  - **Docker** **18.06** https://kubernetes.io/docs/setup/release/notes/#sig-cluster-lifecycle
-
  - `kubeadm`: the command to bootstrap the cluster.
  - `kubelet`: the component that runs on all of the machines in your cluster and does things like starting pods and containers.
    - Pod Docker
  - `kubectl`: the command line util to talk to your cluster.
    - k8s

- https://kubernetes.io/docs/setup/independent/troubleshooting-kubeadm/
-
    - https://github.com/coreos/tectonic-installer
    - https://github.com/kubernetes-incubator/kubespray
    - https://github.com/apprenda/kismatic

- Kubernetes 1.13.3

-
    - master-1 `192.168.0.127`
    - node-1 `192.168.0.128`
    - node-2 `192.168.0.129`
- https://github.com/kubernetes/kubernetes/releases
- 1.13 changelog https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG-1.13.md
- **Docker 18.06**
    - o(∩_∩)o
    - `yum list docker-ce --showduplicates`
- kubernetes repo Kubeadm Kubelet Kubectl
- Kubeadm `--image-repository` 1.13

- `systemctl start chronyd.service && systemctl enable chronyd.service`
- selinux swap

```
systemctl stop firewalld.service
systemctl disable firewalld.service
systemctl disable iptables.service

iptables -P FORWARD ACCEPT


setenforce 0 && sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config


echo "vm.swappiness = 0" >> /etc/sysctl.conf
swapoff -a && sysctl -w vm.swappiness=0
```

- hostname hosts

```
hostnamectl --static set-hostname  k8s-master-1
hostnamectl --static set-hostname  k8s-node-1
hostnamectl --static set-hostname  k8s-node-2
```

```
vim /etc/hosts
192.168.0.127 k8s-master-1
192.168.0.128 k8s-node-1
192.168.0.129 k8s-node-2
```

- master

```
ssh-keygen -t rsa

cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys



ssh localhost


ssh-copy-id -i ~/.ssh/id_rsa.pub -p 22 root@k8s-node-1       k8s-node-1
ssh-copy-id -i ~/.ssh/id_rsa.pub -p 22 root@k8s-node-2       k8s-node-2


ssh k8s-master-1
ssh k8s-node-1
ssh k8s-node-2
```

- yum

```
vim /etc/yum.repos.d/kubernetes.repo

[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg



scp -r /etc/yum.repos.d/kubernetes.repo root@k8s-node-1:/etc/yum.repos.d/
scp -r /etc/yum.repos.d/kubernetes.repo root@k8s-node-2:/etc/yum.repos.d/
```

- master flannel

```
mkdir -p /etc/cni/net.d && vim /etc/cni/net.d/10-flannel.conflist
```

```
{
    "name": "cbr0",
    "plugins": [
        {
            "type": "flannel",
            "delegate": {
                "hairpinMode": true,
                "isDefaultGateway": true
            }
        },
        {
            "type": "portmap",
            "capabilities": {
                "portMappings": true
            }
        }
    ]
}
```

```
vim /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward=1
vm.swappiness=0



scp -r /etc/sysctl.d/k8s.conf root@k8s-node-1:/etc/sysctl.d/
scp -r /etc/sysctl.d/k8s.conf root@k8s-node-2:/etc/sysctl.d/

modprobe br_netfilter && sysctl -p /etc/sysctl.d/k8s.conf
```

```
yum install -y kubelet-1.13.3 kubeadm-1.13.3 kubectl-1.13.3 --disableexcludes=kubernetes
```

```
vim  /etc/systemd/system/kubelet.service.d/10-kubeadm.conf


      Environment="KUBELET_CGROUP_ARGS=--cgroup-driver=cgroupfs"
```

```
systemctl enable kubelet && systemctl start kubelet
```

```
kubeadm version
kubectl version
```

- master

```
echo 1 > /proc/sys/net/ipv4/ip_forward



kubeadm init \
--image-repository registry.cn-hangzhou.aliyuncs.com/google_containers \
--pod-network-cidr 10.244.0.0/16 \
--kubernetes-version 1.13.3 \
--ignore-preflight-errors=Swap

   10.244.0.0/16   flannel       ip

         docker

[init] Using Kubernetes version: v1.13.3
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet
connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-
flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [k8s-master-1 localhost] and IPs
[192.168.0.127 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [k8s-master-1 localhost] and IPs
[192.168.0.127 127.0.0.1 ::1]
```

```
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [k8s-master-1 kubernetes
kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs
[10.96.0.1 192.168.0.127]
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from
directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 19.001686 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-
system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.13" in namespace kube-system with the
configuration for the kubelets in the cluster
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API
object "k8s-master-1" as an annotation
[mark-control-plane] Marking the node k8s-master-1 as control-plane by adding the label "node-
role.kubernetes.io/master=''"
[mark-control-plane] Marking the node k8s-master-1 as control-plane by adding the taints
[node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: 8tpo9l.jlw135r8559kaad4
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order
for nodes to get long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically
approve CSRs from a Node Bootstrap Token
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client
certificates in the cluster
```

```
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes master has initialized successfully!

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

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/

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

  kubeadm join 192.168.0.127:6443 --token 8tpo9l.jlw135r8559kaad4 --discovery-token-ca-cert-
hash sha256:d6594ccc1310a45cbebc45f1c93f5ac113873786365ed63efcf667c952d7d197
```

- master

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

- master

```
kubeadm token list


kubectl cluster-info
```

- master Flannel

```
cd /opt && wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

kubectl apply -f /opt/kube-flannel.yml
```

- node

```
echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables

kubeadm join 192.168.0.127:6443 --token 8tpo9l.jlw135r8559kaad4 --discovery-token-ca-cert-hash sha256:d6594ccc1310a45cbebc45f1c93f5ac113873786365ed63efcf667c952d7d197




[preflight] Running pre-flight checks
[discovery] Trying to connect to API Server "192.168.0.127:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://192.168.0.127:6443"
[discovery] Requesting info from "https://192.168.0.127:6443" again to validate TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "192.168.0.127:6443"
[discovery] Successfully established connection with API Server "192.168.0.127:6443"
[join] Reading configuration from the cluster...
[join] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.13" ConfigMap in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "k8s-node-1" as an annotation

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.
```

```
Run 'kubectl get nodes' on the master to see this node join the cluster.
```

- nod`kubeadm reset` join
- master`kubectl get cs`

```
NAME                 STATUS    MESSAGE              ERROR
controller-manager   Healthy   ok
scheduler            Healthy   ok
etcd-0               Healthy   {"health": "true"}
    Healthy                  `kubeadm reset`
```

- master`kubectl get nodes`

```
    NotReady       kubectl get pods --all-namespaces
  Pending/ContainerCreating/ImagePullBackOff    Pod              Pod
kubectl describe pod <Pod Name> --namespace=kube-system
  kubectl logs <Pod Name> -n kube-system
tail -f /var/log/messages
```

- Master
  - https://kubernetes.io/docs/concepts/overview/components/
  - kube-apiserver API
  - kube-scheduler
  - Kube-Controller-Manager
  - Etcd
  - Kube-proxy Service cluster
  - Kube-DNS DNS
- node
- `Pods`

```



   PID   IPC  UTS



```

- `Volumes`

```
Pod


          – emptyDir, hostpath, gcePersistentDisk, awsElasticBlockStore, nfs, iscsi, glusterfs,
```

- `Labels`

```
      Pod  key/value
```

- `Replication Controllers`

```
           Pod
```

- `Services`

```
    Pod
  IP   DNS
      DNS

     – ClusterIP, NodePort, LoadBalancer
```

- `etcd`

```
   Key/Value
  apiserver
  etcd
```

- `apiserver`

```
Kubernetes     REST
```

- `kube-scheduler`

```



  /  /



```

- `kube-controller-manager`

```
Replication controller
Endpoint controller
Namespace controller
Serviceaccount controller
```

- `kubelet`

```

       Pod
```

- `kube-proxy`

```
Pod
TCP/UDP
    Round Robin
```

-

```

DNS — kube2sky  etcd skydns
```

-

```



  Pod       IP
```

-

```
kubelet    master
      kubelet
Etcd

```

```
  apiserver
Master    kube-scheduler kube-controller-manager
```

- “ ” Kubernetes
-

Revision #2
Created 20 March 2020 15:24:49 by
Updated 14 August 2020 06:18:41 by