# **# [centos7使用kubeadm安装部署kubernetes 1.15](https://www.cnblogs.com/ding2016/p/10784620.html).0**

**应用背景**：

截止目前为止，kubernetes版本已经发布至1.15.0，在此记录一下安装部署步骤和过程中的问题排查。

部署k8s一般两种方式：kubeadm（官方称目前已经GA，可以在生产环境使用）；二进制安装（比较繁琐）。

这里暂且采用kubeadm方式部署测试。

**测试环境**：

|  |  |  |
| --- | --- | --- |
| **System** | **Hostname** | **IP** |
| CentOS 7.6 | k8s-master | **172.18.23.106** |
| CentOS 7.6 | k8s-worker01 | **172.18.23.95** |
| CentOS 7.6 | k8s-worker02 | **172.18.23.xxx** |

**172.18.23.106 k8s-master**

**172.18.23.95 k8s-worker01**

**具体步骤**：

1. **环境预设（在所有主机上操作）**

关闭firewalld：

ystemctl stop firewalld && systemctl disable firewalld

关闭SElinux：

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

关闭Swap：  
swapoff -a && sed -i "s/\/dev\/mapper\/centos-swap/\#\/dev\/mapper\/centos-swap/g" /etc/fstab

使用阿里云yum源：  
wget -O /etc/yum.repos.d/CentOS7-Aliyun.repo http://mirrors.aliyun.com/repo/Centos-7.repo

（注意：更新/etc/hosts 件：在每一台主机的该文件中添加k8s所有节点的IP和对应主机名，否则初始化的时候回出现告警甚至错误。）

**2. 安装docker引擎（在所有主机上操作）**

安装阿里云docker源：   
wget -O /etc/yum.repos.d/docker-ce http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

安装docker：  
yum install -y docker-ce

启动docker：  
systemctl enable docker && systemctl start docker

调整docker部分参数：  
mkdir -p /etc/docker  
  
tee /etc/docker/daemon.json << 'EOF'

{

"registry-mirrors": ["https://5twf62k1.mirror.aliyuncs.com"],　　　// 改为阿里镜像

"exec-opts": ["native.cgroupdriver=systemd"]　　// 默认cgroupfs，k8s官方推荐systemd，否则初始化出现Warning

}

EOF  
  
systemctl daemon-reload  
systemctl restart docker

检查确认docker的Cgroup Driver信息：  
[root@localhost ~]# docker info |grep Cgroup

Cgroup Driver: systemd 原来是：cgroupfs

**3. 安装kubernetes初始化工具（在所有主机上操作）**

使用阿里云的kubernetes源：  
cat <<EOF > /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

EOF

安装工具：yum install -y kubelet kubeadm kubectl // 此时最新版本1.15.0

启动kubelet：systemctl enable kubelet && systemctl start kubelet  //启动不成功正常，后面初始化的时候会变成功

**4. 预下载相关镜像（在master节点上操作）**

查看集群初始化所需镜像及对应依赖版本号：

[root@k8s-master~]# kubeadm config images list

W0709 03:25:05.715766 12809 version.go:98] could not fetch a Kubernetes version from the internet: unable to get URL "https://dl.k8s.io/release/stable-1.txt": Get https://dl.k8s.io/release/stable-1.txt: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)

W0709 03:25:05.715907 12809 version.go:99] falling back to the local client version: v1.15.0

k8s.gcr.io/kube-apiserver:v1.15.0

k8s.gcr.io/kube-controller-manager:v1.15.0

k8s.gcr.io/kube-scheduler:v1.15.0

k8s.gcr.io/kube-proxy:v1.15.0

k8s.gcr.io/pause:3.1

k8s.gcr.io/etcd:3.3.10

k8s.gcr.io/coredns:1.3.1

因为这些重要镜像都被墙了，所以要预先单独下载好，然后才能初始化集群。

下载脚本：

#!/bin/bash

set -e

KUBE\_VERSION=v1.15.0

KUBE\_PAUSE\_VERSION=3.1

ETCD\_VERSION=3.3.10

CORE\_DNS\_VERSION=1.3.1

GCR\_URL=k8s.gcr.io

ALIYUN\_URL=registry.cn-hangzhou.aliyuncs.com/google\_containers

images=(kube-proxy:${KUBE\_VERSION}

kube-scheduler:${KUBE\_VERSION}

kube-controller-manager:${KUBE\_VERSION}

kube-apiserver:${KUBE\_VERSION}

pause:${KUBE\_PAUSE\_VERSION}

etcd:${ETCD\_VERSION}

coredns:${CORE\_DNS\_VERSION})

for imageName in ${images[@]} ; do

docker pull $ALIYUN\_URL/$imageName

docker tag $ALIYUN\_URL/$imageName $GCR\_URL/$imageName

docker rmi $ALIYUN\_URL/$imageName

done

*v1.15.0: Pulling from google\_containers/kube-proxy*

*346aee5ea5bc: Pull complete*

*1e695dec1fee: Pull complete*

*100690d61cf6: Pull complete*

*Digest: sha256:020d25ff45a33ec7958d7128308cb499d5b24cdaa228a2344514bcab9b7296c0*

*Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-proxy:v1.15.0*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-proxy:v1.15.0*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-proxy@sha256:020d25ff45a33ec7958d7128308cb499d5b24cdaa228a2344514bcab9b7296c0*

*v1.15.0: Pulling from google\_containers/kube-scheduler*

*346aee5ea5bc: Already exists*

*b88909b8f99f: Pull complete*

*Digest: sha256:650dd7e101652a2b98db5fe18e9ff0c09b5530b211529ccf5f62241d21db1ee2*

*Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-scheduler:v1.15.0*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-scheduler:v1.15.0*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-scheduler@sha256:650dd7e101652a2b98db5fe18e9ff0c09b5530b211529ccf5f62241d21db1ee2*

*v1.15.0: Pulling from google\_containers/kube-controller-manager*

*346aee5ea5bc: Already exists*

*f4db69ee8ade: Pull complete*

*Digest: sha256:f6ddbc332516d73afc7c81fabe47ed6b1e0a43461a0b861aae9608a4692602c0*

*Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-controller-manager:v1.15.0*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-controller-manager:v1.15.0*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-controller-manager@sha256:f6ddbc332516d73afc7c81fabe47ed6b1e0a43461a0b861aae9608a4692602c0*

*v1.15.0: Pulling from google\_containers/kube-apiserver*

*346aee5ea5bc: Already exists*

*7f0e834d5a94: Pull complete*

*Digest: sha256:0c8710b83841950515d3bdea9ad052df00dc730ac22ac07b27c02adaaad30a36*

*Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-apiserver:v1.15.0*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-apiserver:v1.15.0*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-apiserver@sha256:0c8710b83841950515d3bdea9ad052df00dc730ac22ac07b27c02adaaad30a36*

*3.1: Pulling from google\_containers/pause*

*cf9202429979: Pull complete*

*Digest: sha256:759c3f0f6493093a9043cc813092290af69029699ade0e3dbe024e968fcb7cca*

*Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/google\_containers/pause:3.1*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/pause:3.1*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/pause@sha256:759c3f0f6493093a9043cc813092290af69029699ade0e3dbe024e968fcb7cca*

*3.3.10: Pulling from google\_containers/etcd*

*90e01955edcd: Pull complete*

*6369547c492e: Pull complete*

*bd2b173236d3: Pull complete*

*Digest: sha256:240bd81c2f54873804363665c5d1a9b8e06ec5c63cfc181e026ddec1d81585bb*

*Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/google\_containers/etcd:3.3.10*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/etcd:3.3.10*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/etcd@sha256:240bd81c2f54873804363665c5d1a9b8e06ec5c63cfc181e026ddec1d81585bb*

*1.3.1: Pulling from google\_containers/coredns*

*e0daa8927b68: Pull complete*

*3928e47de029: Pull complete*

*Digest: sha256:638adb0319813f2479ba3642bbe37136db8cf363b48fb3eb7dc8db634d8d5a5b*

*Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/google\_containers/coredns:1.3.1*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/coredns:1.3.1*

*Untagged: registry.cn-hangzhou.aliyuncs.com/google\_containers/coredns@sha256:638adb0319813f2479ba3642bbe37136db8cf363b48fb3eb7dc8db634d8d5a5b*

**5. 初始化集群（在master节点上操作）  
kubeadm init --kubernetes-version=v1.15.0 --pod-network-cidr=192.168.0.0/16**

（注意：初始化之后会安装网络插件，这里选择了calico，所以修改**--pod-network-cidr=192.168.0.0/16 ）**初始化输出记录样例：

[root@k8s-master ~]# kubeadm init --kubernetes-version=v1.15.0 --pod-network-cidr=192.168.0.0/16

[init] Using Kubernetes version: v1.15.0

[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 "ca" certificate and key

[certs] Generating "apiserver" certificate and key

[certs] apiserver serving cert is signed for DNS names [k8s-master kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.18.23.106]

[certs] Generating "apiserver-kubelet-client" certificate and key

[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/peer" certificate and key

[certs] etcd/peer serving cert is signed for DNS names [k8s-master localhost] and IPs [172.18.23.106 127.0.0.1 ::1]

[certs] Generating "apiserver-etcd-client" certificate and key

[certs] Generating "etcd/healthcheck-client" certificate and key

[certs] Generating "etcd/server" certificate and key

[certs] etcd/server serving cert is signed for DNS names [k8s-master localhost] and IPs [172.18.23.106 127.0.0.1 ::1]

[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 15.502689 seconds

[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace

[kubelet] Creating a ConfigMap "kubelet-config-1.15" in namespace kube-system with the configuration for the kubelets in the cluster

[upload-certs] Skipping phase. Please see --upload-certs

[mark-control-plane] Marking the node k8s-master as control-plane by adding the label "node-role.kubernetes.io/master=''"

[mark-control-plane] Marking the node k8s-master as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]

[bootstrap-token] Using token: 94ttw4.i9c73eulwru8wera

[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles

[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials

[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token

[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster

[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace

[addons] Applied essential addon: CoreDNS

[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane 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/

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

kubeadm join 172.18.23.106:6443 --token 94ttw4.i9c73eulwru8wera \

--discovery-token-ca-cert-hash sha256:c67aab094026e3116e30fa3231d2811ae96951d28243bd6c629e04eb9d28ffcb

[root@k8s-master ~]#

#以上输出显示初始化成功，并给出了接下来的必要步骤和节点加入集群的命令，照着做即可。

[root@k8s-master ~]# mkdir -p $HOME/.kube

[root@k8s-master ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[root@k8s-master ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config

#查看已经运行的pod

[root@k8s-master ~]# kubectl get pod -n kube-system -owide

NAME READY STATUS ESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

coredns-5c98db65d4-gs8fk 0/1 Pending 0 6m13s <none> <none> <none> <none>

coredns-5c98db65d4-zksmd 0/1 Pending 0 6m13s <none> <none> <none> <none>

etcd-k8s-master 1/1 Running 0 5m14s 172.18.23.106 k8s-master <none> <none>

kube-apiserver-k8s-master 1/1 Running 0 5m26s 172.18.23.106 k8s-master <none> <none>

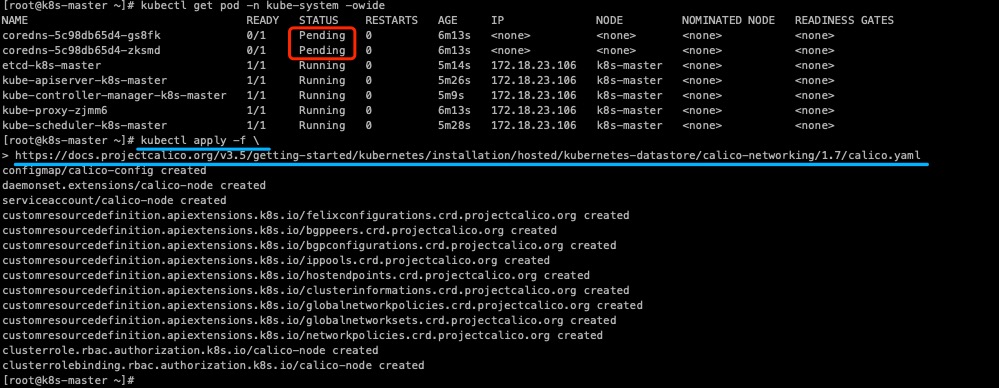
kube-controller-manager-k8s-master 1/1 Running 0 5m9s 172.18.23.106 k8s-master <none> <none>

kube-proxy-zjmm6 1/1 Running 0 6m13s 172.18.23.106 k8s-master <none> <none>

kube-scheduler-k8s-master 1/1 Running 0 5m28s 172.18.23.106 k8s-master <none> <none>

[root@k8s-master ~]#

# coredns状态:Pending 这是正常的，因为还没有网络插件，接下来安装calico后就变为正常running了。



**6. 安装calico（在master节点上操作）**

Calico官网：<https://docs.projectcalico.org/v3.6/getting-started/kubernetes/>

[root@k8s-master ~]# kubectl apply -f \

https://docs.projectcalico.org/v3.5/getting-started/kubernetes/installation/hosted/kubernetes-datastore/calico-networking/1.7/calico.yaml

configmap/calico-config created

daemonset.extensions/calico-node created

serviceaccount/calico-node created

customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created

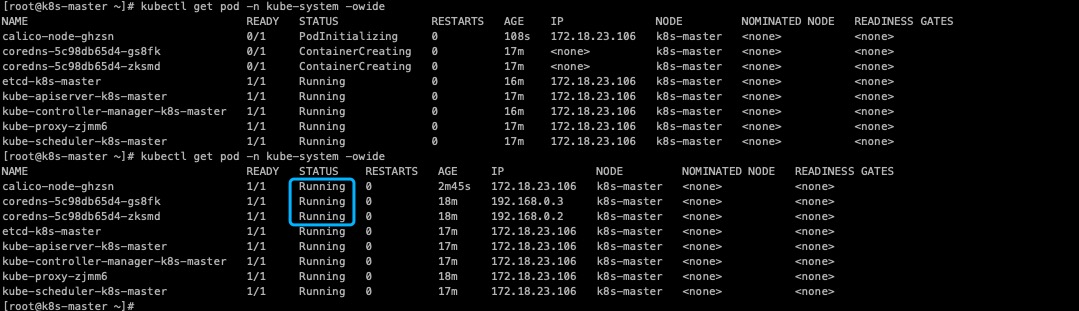
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created

clusterrole.rbac.authorization.k8s.io/calico-node created

clusterrolebinding.rbac.authorization.k8s.io/calico-node created

[root@k8s-master ~]#   


# 应用官方的yaml文件之后，过一会查看所有pod已经正常running状态了，也分配出了对应IP：

[root@k8s-master ~]# kubectl get pod -n kube-system -owide

NAME READY STATUS ESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

calico-node-ghzsn 1/1 Running 0 2m45s 172.18.23.106 k8s-master <none> <none>

coredns-5c98db65d4-gs8fk 1/1 Running 0 18m 192.168.0.3 k8s-master <none> <none>

coredns-5c98db65d4-zksmd 1/1 Running 0 18m 192.168.0.2 k8s-master <none> <none>

etcd-k8s-master 1/1 Running 0 5m14s 172.18.23.106 k8s-master <none> <none>

kube-apiserver-k8s-master 1/1 Running 0 5m26s 172.18.23.106 k8s-master <none> <none>

kube-controller-manager-k8s-master 1/1 Running 0 5m9s 172.18.23.106 k8s-master <none> <none>

kube-proxy-zjmm6 1/1 Running 0 6m13s 172.18.23.106 k8s-master <none> <none>

kube-scheduler-k8s-master 1/1 Running 0 5m28s 172.18.23.106 k8s-master <none> <none>

[root@k8s-master ~]#

#查看节点状态

[root@k8s-master ~]# kubectl get node -owide

NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME

k8s-master Ready master 26m v1.15.0 172.18.23.106 <none> CentOS Linux 7 (Core) 3.10.0-693.21.1.el7.x86\_64 docker://18.9.7

[root@k8s-master ~]#

至此，集群初始化和主节点都准备就绪，接下来就是加入其他工作节点至集群中。

**7. 加入集群（在非master节点上操作）**

先在需要加入集群的节点上下载必要镜像，下载脚本如下：

#!/bin/bash

set -e

KUBE\_VERSION=v1.15.0

KUBE\_PAUSE\_VERSION=3.1

GCR\_URL=k8s.gcr.io

ALIYUN\_URL=registry.cn-hangzhou.aliyuncs.com/google\_containers

images=(kube-proxy-amd64:${KUBE\_VERSION}

pause:${KUBE\_PAUSE\_VERSION})

for imageName in ${images[@]} ; do

docker pull $ALIYUN\_URL/$imageName

docker tag $ALIYUN\_URL/$imageName $GCR\_URL/$imageName

docker rmi $ALIYUN\_URL/$imageName

done

# 然后在主节点初始化输出中获取加入集群的命令，复制到工作节点执行即可：  
  
[root@k8s-worker01 ~]# kubeadm join 172.18.23.106:6443 --token 94ttw4.i9c73eulwru8wera \

--discovery-token-ca-cert-hash sha256:c67aab094026e3116e30fa3231d2811ae96951d28243bd6c629e04eb9d28ffcb

[preflight] Running pre-flight checks

[preflight] Reading configuration from the cluster...

[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'

[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.15" 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

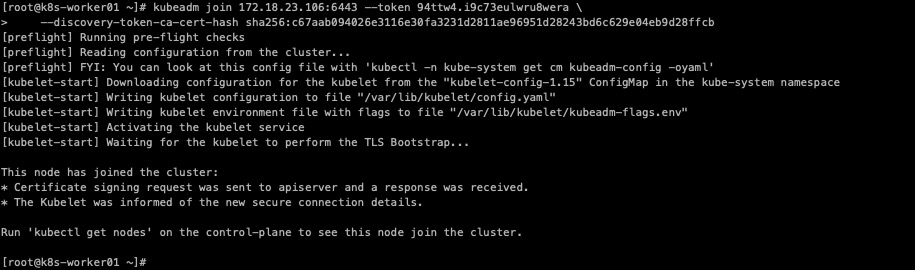
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

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 control-plane to see this node join the cluster.[root@k8s-master ~]#



1. **在master节点上查看各节点工作状态**

[root@k8s-master ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-master Ready master 12h v1.15.0

k8s-worker01 Ready <none> 11h v1.15.0

[root@k8s-master ~]# kubectl get pod -n kube-system -owide

NAME READY STATUS ESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

calico-node-ghzsn 1/1 Running 0 2m45s 172.18.23.106 k8s-master <none> <none>

coredns-5c98db65d4-gs8fk 1/1 Running 0 18m 192.168.0.3 k8s-master <none> <none>

coredns-5c98db65d4-zksmd 1/1 Running 0 18m 192.168.0.2 k8s-master <none> <none>

etcd-k8s-master 1/1 Running 0 5m14s 172.18.23.106 k8s-master <none> <none>

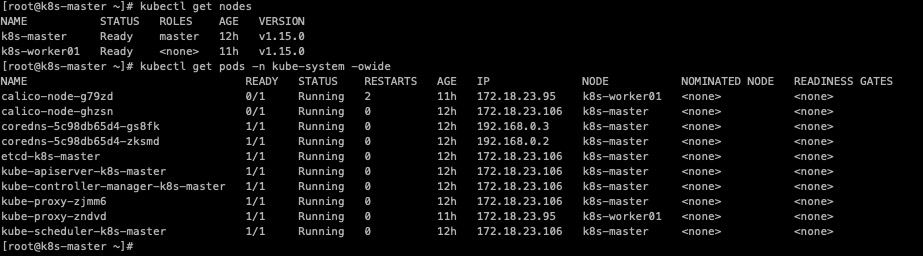
kube-apiserver-k8s-master 1/1 Running 0 5m26s 172.18.23.106 k8s-master <none> <none>

kube-controller-manager-k8s-master 1/1 Running 0 5m9s 172.18.23.106 k8s-master <none> <none>

kube-proxy-zjmm6 1/1 Running 0 6m13s 172.18.23.106 k8s-master <none> <none>

kube-scheduler-k8s-master 1/1 Running 0 5m28s 172.18.23.106 k8s-master <none> <none>

[root@k8s-master ~]#



至此，最简单的集群已经部署完成。

**# Q&A:**  
  
1、**初始化集群（在master节点上操作）报错：**

[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at <https://kubernetes.io/docs/setup/cri/>  
解决：  
[root@k8s-master ~]# docker info

Containers: 1

Running: 0

Paused: 0

Stopped: 1

Images: 17

Server Version: 18.09.7

Storage Driver: overlay2

Backing Filesystem: xfs

Supports d\_type: true

Native Overlay Diff: true

Logging Driver: json-file

Cgroup Driver: systemd # 看这里，没有修改之前应该是cgroupfs

#修改或者创建:

vim /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"]

}

#修改 docker文件驱动, 修改保存后 重新启动 docker, docker重新启动方法(systemctl restart docker)

1. **初始化集群（在master节点上操作）**报错：

[root@k8s-master ~]# kubeadm init --kubernetes-version=v1.15.0 --pod-network-cidr=192.168.0.0/16

[init] Using Kubernetes version: v1.15.0

[preflight] Running pre-flight checks

error execution phase preflight: [preflight] Some fatal errors occurred:

[ERROR KubeletVersion]: Kubelet version "1.9.6" is lower than kubeadm can support. Please upgrade kubelet

[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`

# 解决：  
[root@k8s-master ~]# which kubectl

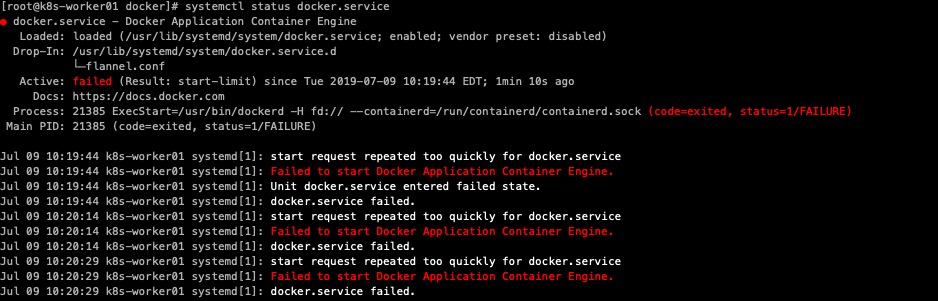
/usr/local/bin/kubectl

[root@k8s-master ~]# /usr/local/bin/kubelet version

……  
I0709 09:34:14.808144 1472 server.go:182] Version: v1.9.6

……  
[root@k8s-master ~]# rm -rf /usr/local/bin/kub\*

1. docker启动报错：



# 解决：  
rm /etc/docker/key.json

rm -rf /var/lib/docker

4、将worker加入k8s集群 **（在worker01节点上操作）**报错：  
[root@k8s-worker01 docker]# kubeadm join 172.18.23.106:6443 --token 94ttw4.i9c73eulwru8wera \

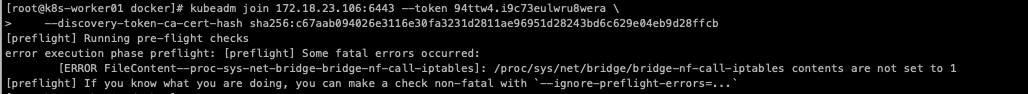
> --discovery-token-ca-cert-hash sha256:c67aab094026e3116e30fa3231d2811ae96951d28243bd6c629e04eb9d28ffcb

[preflight] Running pre-flight checks

error execution phase preflight: [preflight] Some fatal errors occurred:

[ERROR FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables contents are not set to 1

[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`



# 解决：  
echo "1" >/proc/sys/net/bridge/bridge-nf-call-iptables

5、集群创建后，报错：  
[root@k8s-master ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-master Ready master 11h v1.15.0

k8s-worker01 NotReady <none> 10h v1.15.0

[root@k8s-master ~]# kubectl get pods -n kube-system -owide

NAME EADY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

calico-node-g79zd 0/1 Init:0/1 0 10h 172.18.23.95 k8s-worker01 <none> <none>

calico-node-ghzsn 1/1 Running 0 11h 172.18.23.106 k8s-master <none> <none>

coredns-5c98db65d4-gs8fk 1/1 Running 0 11h 192.168.0.3 k8s-master <none> <none>

coredns-5c98db65d4-zksmd 1/1 Running 0 11h 192.168.0.2 k8s-master <none> <none>

etcd-k8s-master 1/1 Running 0 11h 172.18.23.106 k8s-master <none> <none>

kube-apiserver-k8s-master 1/1 Running 0 11h 172.18.23.106 k8s-master <none> <none>

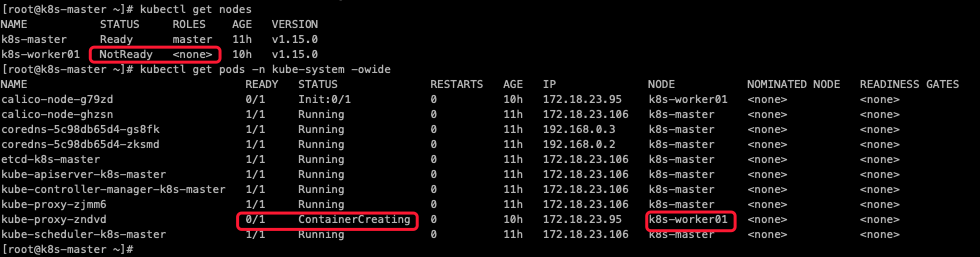
kube-controller-manager-k8s-master 1/1 Running 0 11h 172.18.23.106 k8s-master <none> <none>

kube-proxy-zjmm6 1/1 Running 0 11h 172.18.23.106 k8s-master <none> <none>

kube-proxy-zndvd 0/1 ContainerCreating 0 10h 172.18.23.95 k8s-worker01 <none> <none>

kube-scheduler-k8s-master 1/1 Running 0 11h 172.18.23.106 k8s-master <none> <none>

[root@k8s-master ~]#



[root@k8s-worker01 ~]# journalctl -f

-- Logs begin at Sat 2019-04-20 01:47:26 EDT. --

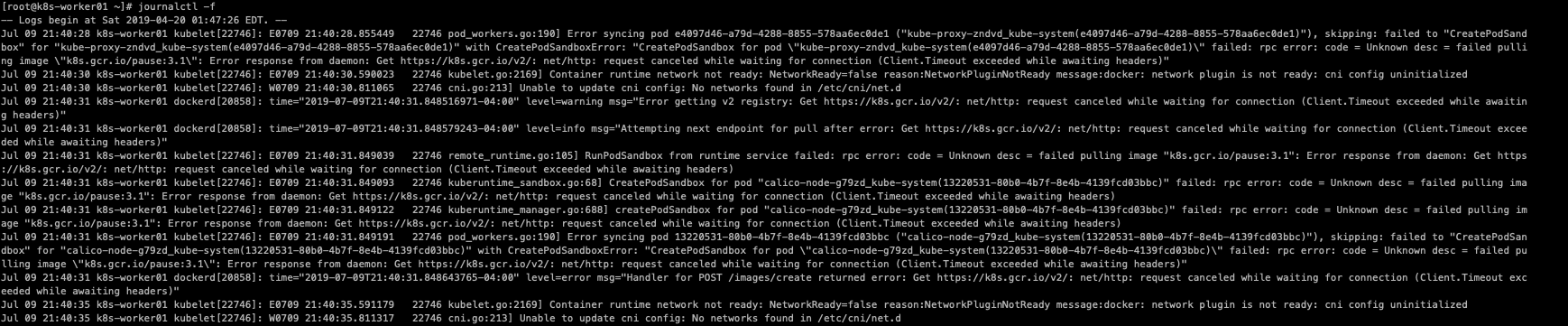
Jul 09 21:40:28 k8s-worker01 kubelet[22746]: E0709 21:40:28.855449 22746 pod\_workers.go:190] Error syncing pod e4097d46-a79d-4288-8855-578aa6ec0de1 ("kube-proxy-zndvd\_kube-system(e4097d46-a79d-4288-8855-578aa6ec0de1)"), skipping: failed to "CreatePodSandbox" for "kube-proxy-zndvd\_kube-system(e4097d46-a79d-4288-8855-578aa6ec0de1)" with CreatePodSandboxError: "CreatePodSandbox for pod \"kube-proxy-zndvd\_kube-system(e4097d46-a79d-4288-8855-578aa6ec0de1)\" failed: rpc error: code = Unknown desc = failed pulling image \"k8s.gcr.io/pause:3.1\": Error response from daemon: Get https://k8s.gcr.io/v2/: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)"

Jul 09 21:40:30 k8s-worker01 kubelet[22746]: E0709 21:40:30.590023 22746 kubelet.go:2169] Container runtime network not ready: NetworkReady=false reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config uninitialized

Jul 09 21:40:30 k8s-worker01 kubelet[22746]: W0709 21:40:30.811065 22746 cni.go:213] Unable to update cni config: No networks found in /etc/cni/net.d

Jul 09 21:40:31 k8s-worker01 dockerd[20858]: time="2019-07-09T21:40:31.848516971-04:00" level=warning msg="Error getting v2 registry: Get https://k8s.gcr.io/v2/: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)"

Jul 09 21:40:31 k8s-worker01 dockerd[20858]: time="2019-07-09T21:40:31.848579243-04:00" level=info msg="Attempting next endpoint for pull after error: Get https://k8s.gcr.io/v2/: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)"

……

# 解决：在worker01上再次执行以下脚本，然后 journalctl -f 查看详情；

#!/bin/bash

set -e

KUBE\_VERSION=v1.15.0

KUBE\_PAUSE\_VERSION=3.1

GCR\_URL=k8s.gcr.io

ALIYUN\_URL=registry.cn-hangzhou.aliyuncs.com/google\_containers

images=(kube-proxy:${KUBE\_VERSION}

pause:${KUBE\_PAUSE\_VERSION})

for imageName in ${images[@]} ; do

docker pull $ALIYUN\_URL/$imageName

docker tag $ALIYUN\_URL/$imageName $GCR\_URL/$imageName

docker rmi $ALIYUN\_URL/$imageName

done