k8S安装部署详细教程

firesuperman 于 2022-03-17 17:41:39 发布

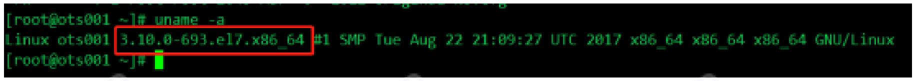
k8S安装部署教程

要求： 三台虚拟机

以下步骤在三台虚拟机上均要执行

检查虚拟机内核版本

```
uname -r
```

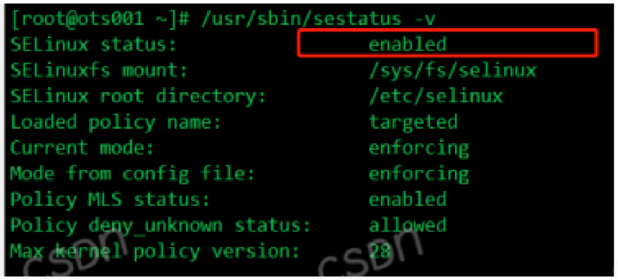


```
[root@ots001 ~]# uname -a
Linux ots001 3.10.0-693.el7.x86_64 #1 SMP Tue Aug 22 21:09:27 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
[root@ots001 ~]# 
```

关闭SELinux

检查SELinux状态,enabled为开启状态

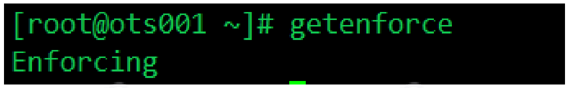
```
/usr/sbin/sestatus -v
```



```
[root@ots001 ~]# /usr/sbin/sestatus -v
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

或者使用getenforce命令查看：

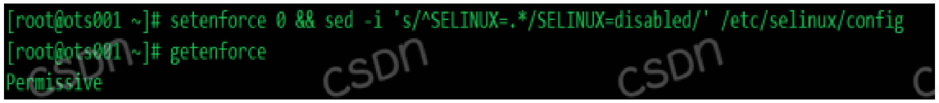
```
getenforce
```



```
[root@ots001 ~]# getenforce
Enforcing
```

关闭SELinux，修改配置文件，将SELINUX设置为disabled，并且修改SELinux模式为Premissive

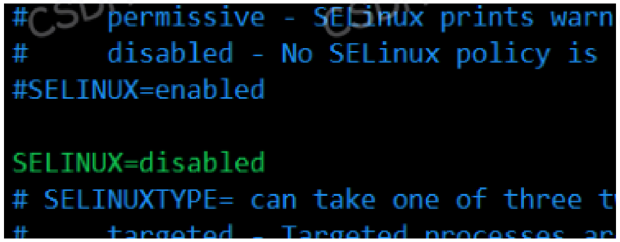
```
setenforce 0 && sed -i 's/^SELINUX=.*/SELINUX=disabled/' /etc/selinux/config
```



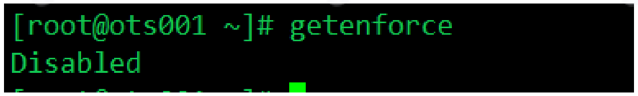
```
[root@ots001 ~]# setenforce 0 && sed -i 's/^SELINUX=.*/SELINUX=disabled/' /etc/selinux/config
[root@ots001 ~]# getenforce
Permissive
```

或者修改配置文件，将SELINUX设置为disabled,然后重启虚拟机生效

```
vim /etc/selinux/config
```



```
#      permissive - SELinux prints warn
#      disabled - No SELinux policy is
#SELINUX=enabled

SELINUX=disabled
# SELINUXTYPE= can take one of three t
#      targeted - Targeted processes ar
```



```
[root@ots001 ~]# getenforce
Disabled
```

关闭防火墙

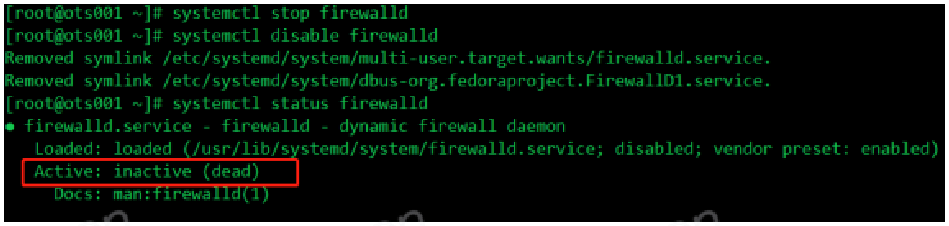
```
systemctl stop firewalld
```

关闭防火墙开机自启动

```
systemctl disable firewalld
```

检查防火墙状态，inactive为关闭状态，active为运行状态

```
systemctl status firewalld
```



```
[root@ots001 ~]# systemctl stop firewalld
[root@ots001 ~]# systemctl disable firewalld
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@ots001 ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

安装依赖包

```
yum install -y conntrack ntpdate ntp ipvsadm ipset jq iptables curl     sysstat libseccomp wget vim net-tools git
```

设置防火墙为iptables并设置空规则

```
yum -y install iptables-services && systemctl start iptables && systemctl enable iptables && iptables -F && service iptables save
```

永久关闭虚拟内存

```
swapoff -a && sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

调整内核参数，写入配置文件中

在/etc/sysctl.d/目录下创建kubernetes.conf文件，写入配置数据：

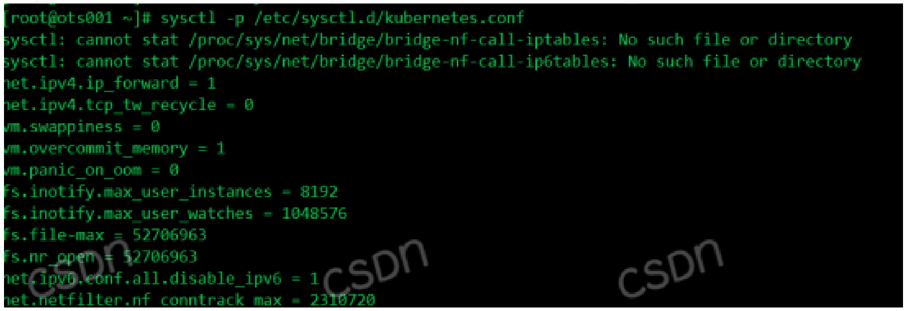
```
 vim /etc/sysctl.d/kubernetes.conf
```

```
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1
net.ipv4.ip_forward=1
net.ipv4.tcp_tw_recycle=0
vm.swappiness=0
vm.overcommit_memory=1
vm.panic_on_oom=0
```

```
fs.inotify.max_user_instances=8192
fs.inotify.max_user_watches=1048576
fs.file-max=52706963
fs.nr_open=52706963
net.ipv6.conf.all.disable_ipv6=1
net.netfilter.nf_conntrack_max=2310720
```

保存退出后刷新配置立马生效

```
 sysctl -p /etc/sysctl.d/kubernetes.conf
```



```
[root@ots001 ~]# sysctl -p /etc/sysctl.d/kubernetes.conf
sysctl: cannot stat /proc/sys/net/bridge/bridge-nf-call-iptables: No such file or directory
sysctl: cannot stat /proc/sys/net/bridge/bridge-nf-call-ip6tables: No such file or directory
net.ipv4.ip_forward = 1
net.ipv4.tcp_tw_recycle = 0
vm.swappiness = 0
vm.overcommit_memory = 1
vm.panic_on_oom = 0
fs.inotify.max_user_instances = 8192
fs.inotify.max_user_watches = 1048576
fs.file-max = 52706963
fs.nr_open = 52706963
net.ipv6.conf.all.disable_ipv6 = 1
net.netfilter.nf_conntrack_max = 2310720
```

关闭系统不需要的服务

```
systemctl stop postfix && systemctl disable postfix
```

日志配置

systemd-journald 用于检索 systemd 的日志，是 systemd 自带的日志系统。设置rsyslogd 和systemd journald
创建持久化日志目录

```
 mkdir /var/log/journal
```

创建配置文件目录

```
mkdir /etc/systemd/journald.conf.d
```

创建99-prophet.conf配置文件，写入配置

```
vim /etc/systemd/journald.conf.d/99-prophet.conf
```

```
[Journal]
#持久化保存到磁盘
Storage=persistent

#压缩历史日志
Compress=yes

SyncIntervalSec=5m
RateLimitInterval=30s
RateLimitBurst=1000

#最大占用空间10G
SystemMaxUse=10G

#单日志文件最大200M
SystemMaxFileSize=200M

#日志保存时间2周
MaxRetentionSec=2week

#不将日志转发到syslog
ForwardToSyslog=no
```

重启日志

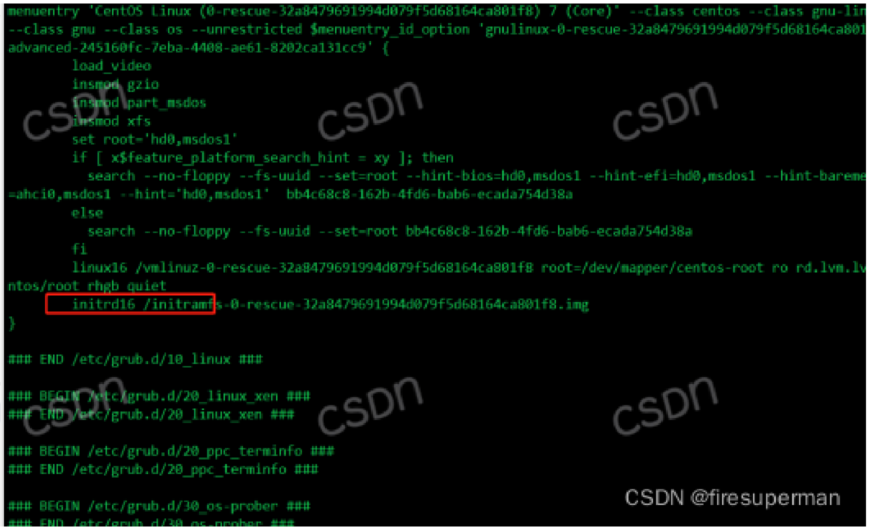
```
systemctl restart systemd-journald
```

升级内核到4.44或以上版本

```
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm
```

安装完成后检查 /boot/grub2/grub.cfg 中对应内核menuentry中是否包含initrd16配置，如果没有，再安装一次

```
cat /boot/grub2/grub.cfg
```



```
menuentry 'CentOS Linux (0-rescue-32a8479691994d079f5d68164ca801f8) 7 (Core)' --class centos --class gnu-lin
--class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-0-rescue-32a8479691994d079f5d68164ca801
advanced-245160fc-7eba-4408-ae61-8202ca131cc9' {
        load_video
        insmod gzio
        insmod part_msdos
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-bareme
=ahci0,msdos1 --hint='hd0,msdos1'  bb4c68c8-162b-4fd6-bab6-ecada754d38a
        else
          search --no-floppy --fs-uuid --set=root bb4c68c8-162b-4fd6-bab6-ecada754d38a
        fi
        linux16 /vmlinuz-0-rescue-32a8479691994d079f5d68164ca801f8 root=/dev/mapper/centos-root ro rd.lvm.lv
ntos/root rhgb quiet
        initrd16 /initramfs-0-rescue-32a8479691994d079f5d68164ca801f8.img
}

### END /etc/grub.d/10_linux ###

### BEGIN /etc/grub.d/20_linux_xen ###
### END /etc/grub.d/20_linux_xen ###

### BEGIN /etc/grub.d/20_ppc_terminfo ###
### END /etc/grub.d/20_ppc_terminfo ###

### BEGIN /etc/grub.d/30_os-prober ###
### END /etc/grub.d/30_os-prober ###
```

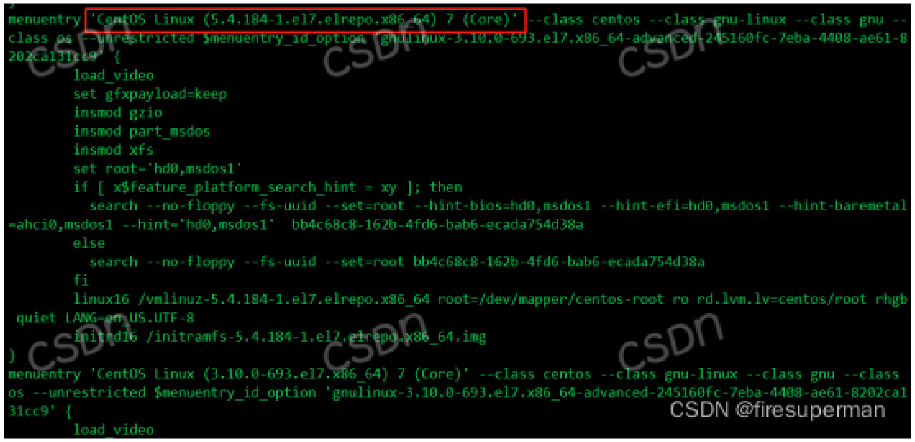
CSDN @firesuperman

安装内核

```
yum --enablerepo=elrepo-kernel install -y kernel-lt
```

查看内核

```
cat /boot/grub2/grub.cfg
```



```
menuentry 'CentOS Linux (5.4.184-1.el7.elrepo.x86_64) 7 (Core)' --class centos --class gnu-linux --class gnu --
class os --unrestricted $menuentry_id_option 'gnulinux-3.10.0-693.el7.x86_64-advanced-245160fc-7eba-4408-ae61-8
202ca131cc9' {
        load_video
        set gfxpayload=keep
        insmod gzio
        insmod part_msdos
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal
=ahci0,msdos1 --hint='hd0,msdos1'  bb4c68c8-162b-4fd6-bab6-ecada754d38a
        else
          search --no-floppy --fs-uuid --set=root bb4c68c8-162b-4fd6-bab6-ecada754d38a
        fi
        linux16 /vmlinuz-5.4.184-1.el7.elrepo.x86_64 root=/dev/mapper/centos-root ro rd.lvm.lv=centos/root rhgb
 quiet LANG=en_US.UTF-8
        initrd16 /initramfs-5.4.184-1.el7.elrepo.x86_64.img
}
menuentry 'CentOS Linux (3.10.0-693.el7.x86_64) 7 (Core)' --class centos --class gnu-linux --class gnu --class
os --unrestricted $menuentry_id_option 'gnulinux-3.10.0-693.el7.x86_64-advanced-245160fc-7eba-4408-ae61-8202ca1
31cc9' {
        load_video
```

CSDN @firesuperman

设置开机从新内核启动，下载的内核版本可能不一样，将命令中冒号里的内核版本换成对应的版本再执行

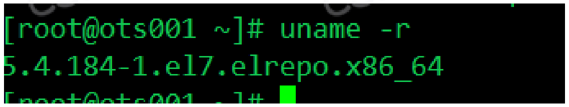
```
grub2-set-default 'CentOS Linux (5.4.184-1.el7.elrepo.x86_64) 7 (Core)'
```

然后重启虚拟机

```
 reboot
```

查看系统内核版本是否不低于4.44版本

```
uname -r
```



```
[root@ots001 ~]# uname -r
5.4.184-1.el7.elrepo.x86_64
[root@ots001 ~]# 
```

kube-proxy开启ipvs前置条件

加载netfilter模块

```
modprobe br_netfilter
```

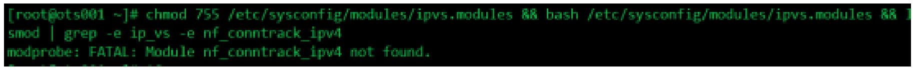
写入依赖配置文件

```
cat > /etc/sysconfig/modules/ipvs.modules <<EOF
#!/bin/bash
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack_ipv4
EOF
```

给配置文件赋权，引导配置文件

```
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && lsmod | grep -e ip_vs -e nf_conntrack_ipv4
```

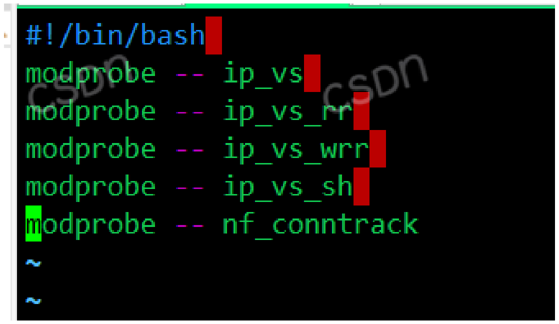
如果报FATAL: Module nf_conntrack_ipv4 not found的错误，是因为高版本的centos内核nf_conntrack_ipv4被nf_conntrack替换了，所以装不了，执行以下操作可解决，配置正常则无需执行



```
[root@ots001 ~]# chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && l
smod | grep -e ip_vs -e nf_conntrack_ipv4
modprobe: FATAL: Module nf_conntrack_ipv4 not found.
```

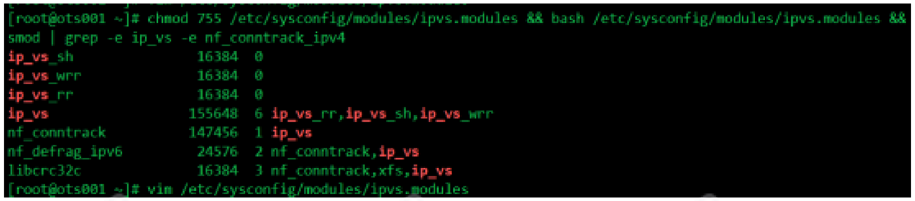
修改ipvs.modules文件，nf_conntrack_ipv4修改为nf_conntrack,可解决

```
vim /etc/sysconfig/modules/ipvs.modules
```



```
#!/bin/bash
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack
~
~
```

再次执行之前的命令



```
[root@ots001 ~]# chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && l
smod | grep -e ip_vs -e nf_conntrack_ipv4
ip_vs_sh               16384  0
ip_vs_wrr              16384  0
ip_vs_rr               16384  0
ip_vs                 155648  6 ip_vs_rr,ip_vs_sh,ip_vs_wrr
nf_conntrack          147456  1 ip_vs
nf_defrag_ipv6         24576  2 nf_conntrack,ip_vs
libcrc32c              16384  3 nf_conntrack,xfs,ip_vs
[root@ots001 ~]# vim /etc/sysconfig/modules/ipvs.modules
```

安装Docker软件

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

导入阿里云Docker镜像仓库

```
sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

安装docker-ce

```
yum update -y && yum install -y docker-ce
```

安装完docker-ce相关组件重启服务器

```
reboot
```

重启后，Linux内核又变回原来的，所以要重新设置成4.4以上版本并再次重启

```
grub2-set-default 'CentOS Linux (5.4.184-1.el7.elrepo.x86_64) 7 (Core)' && reboot
```

启动docker

```
systemctl start docker
```

设置docker开机自启

```
systemctl enable docker
```

创建deamon.json的配置文件

```
vim /etc/docker/deamon.json
```

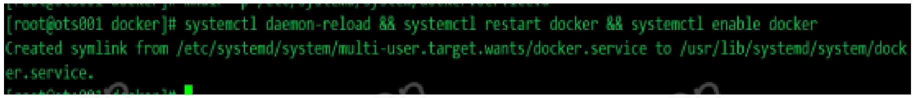
```
{
  "exec-opts":["native.cgroupdriver=systemd"],
  "log-driver":"json-file",
  "log-opts":{ "max-size":"100m" }
}
```

创建docker.service.d文件夹，用于存放docker相关服务

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

重启docker服务

```
systemctl daemon-reload && systemctl restart docker && systemctl enable docker
```



```
[root@ots001 docker]# mkdir -p /etc/systemd/system/docker.service.d
[root@ots001 docker]# systemctl daemon-reload && systemctl restart docker && systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/dock
er.service.
[root@ots001 docker]#
```

安装Kubeadm

先配置阿里镜像源

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

写入列内容后保存退出

```
[kubernetes]
name=Kubernetes
baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
```

安装kubeadm、kubectl和kubelet

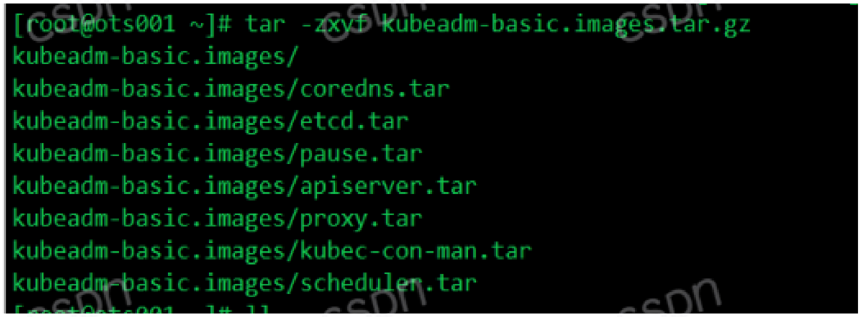
```
yum -y install kubeadm-1.15.1 kubectl-1.15.1 kubelet-1.15.1
```

设置启动开机自启kubelet服务

```
systemctl enable kubelet.service
```

Kubeadm在初始化k8s集群的时候，会从谷歌云服务器取pull所需的镜像，由于国内网络的原因，无法直接从谷歌pull镜像，所以在初始化之前要先将镜像导入到k8s集群中。
Kubeadm-basic.images镜像国内下载地址,如果下载地址失效请搜索其他下载地址
云盘下载地址
https://blog.csdn.net/weixin_45632212/article/details/119995901
下载之后将kubeadm-basic.images.tar.gz移动到虚拟机中/root目录下
解压镜像压缩文件

```
tar -zxvf kubeadm-basic.images.tar.gz
```



```
[root@ots001 ~]# tar -zxvf kubeadm-basic.images.tar.gz
kubeadm-basic.images/
kubeadm-basic.images/coredns.tar
kubeadm-basic.images/etcd.tar
kubeadm-basic.images/pause.tar
kubeadm-basic.images/apiserver.tar
kubeadm-basic.images/proxy.tar
kubeadm-basic.images/kubec-con-man.tar
kubeadm-basic.images/scheduler.tar
[root@ots001 ~]# ll
```

创建load-images.sh，用于批量导入kubeadm-basic镜像，内容如下：

```
vim load-images.sh
```

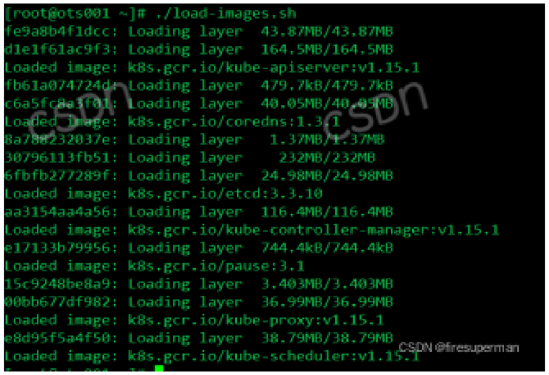
```
#!/bin/bash
ls /root/kubeadm-basic.images > /tmp/image-list.txt
cd /root/kubeadm-basic.images
for i in $(cat /tmp/image-list.txt)
do
   docker load -i $i
done
rm -rf /tmp/image-list.txt
```

给创建的脚本赋权

```
chmod a+x load-images.sh
```

执行脚本批量导入镜像

```
./load-images.sh
```



```
[root@ots001 ~]# ./load-images.sh
fe9a8b4f1dcc: Loading layer  43.87MB/43.87MB
d1e1f61ac9f3: Loading layer  164.5MB/164.5MB
Loaded image: k8s.gcr.io/kube-apiserver:v1.15.1
fb61a074724d: Loading layer  479.7kB/479.7kB
c6a5fc8a3f01: Loading layer  40.05MB/40.05MB
Loaded image: k8s.gcr.io/coredns:1.3.1
8a788232037e: Loading layer   1.37MB/1.37MB
30796113fb51: Loading layer    232MB/232MB
6fbfb277289f: Loading layer  24.98MB/24.98MB
Loaded image: k8s.gcr.io/etcd:3.3.10
aa3154aa4a56: Loading layer  116.4MB/116.4MB
Loaded image: k8s.gcr.io/kube-controller-manager:v1.15.1
e17133b79956: Loading layer  744.4kB/744.4kB
Loaded image: k8s.gcr.io/pause:3.1
15c9248be8a9: Loading layer  3.403MB/3.403MB
00bb677df982: Loading layer  36.99MB/36.99MB
Loaded image: k8s.gcr.io/kube-proxy:v1.15.1
e8d95f5a4f50: Loading layer  38.79MB/38.79MB
Loaded image: k8s.gcr.io/kube-scheduler:v1.15.1
```

CSDN @firesuperman

将镜像文件kubeadm-basic.images与批量导入脚本load-images.sh复制到其他节点执行
注意：以上所有操作需在所有节点上操作

**

初始化主节点(以下步骤只在主节点执行，其他虚拟机不执行)

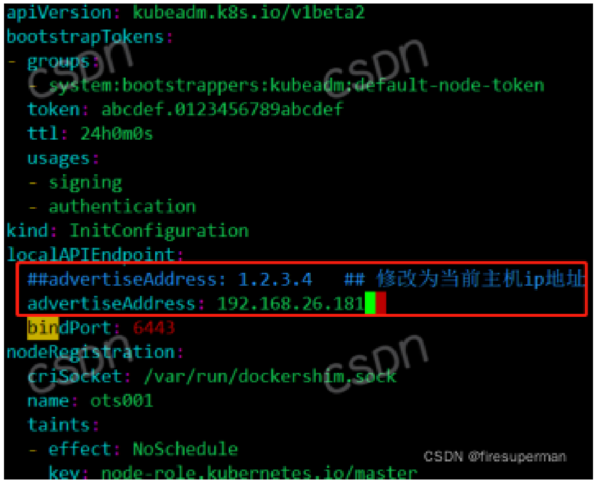
**
将kubeadm配置输出到kubeadm-config.yaml

```
kubeadm config print init-defaults > kubeadm-config.yaml
```

打开kubeadm-config.yaml，并修改相关配置

```
vim kubeadm-config.yaml
```

将advertiseAddress修改当前主机ip地址

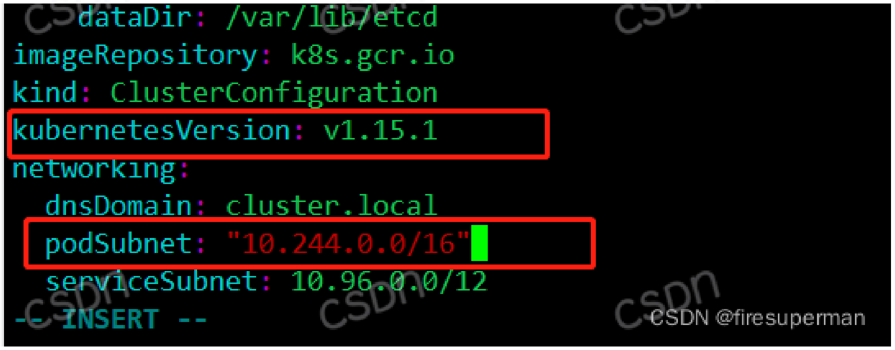


```
apiVersion: kubeadm.k8s.io/v1beta2
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  ##advertiseAddress: 1.2.3.4   ## 修改为当前主机ip地址
  advertiseAddress: 192.168.26.181
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: ots001
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
```

CSDN @firesuperman

修改kubernetes版本，kubernetesVersion: v1.15.1

在dnsDomain行下添加podSubnet: “10.244.0.0/16”,注意缩进

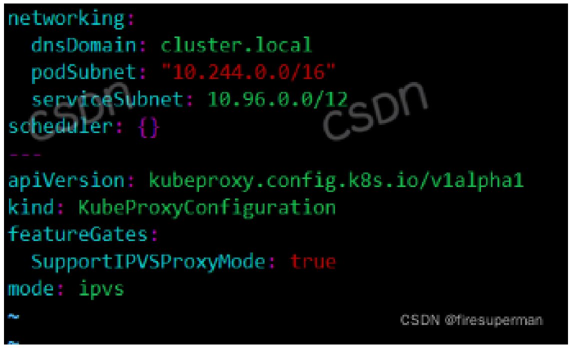


```
  dataDir: /var/lib/etcd
imageRepository: k8s.gcr.io
kind: ClusterConfiguration
kubernetesVersion: v1.15.1
networking:
  dnsDomain: cluster.local
  podSubnet: "10.244.0.0/16"
  serviceSubnet: 10.96.0.0/12
-- INSERT --
```

CSDN @firesuperman

在最后添加如下内容，将默认的调度方式改为ipvs调度模式，注意缩进（2个空格），之后保存退出

```
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
featureGates:
  SupportIPVSProxyMode: true
mode: ipvs
```



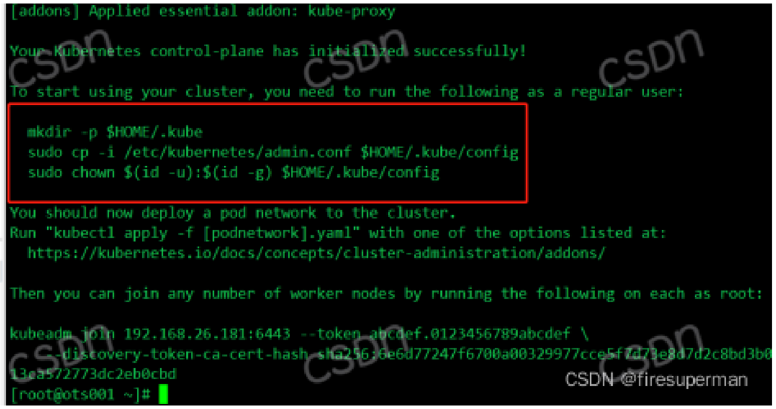
```
networking:
  dnsDomain: cluster.local
  podSubnet: "10.244.0.0/16"
  serviceSubnet: 10.96.0.0/12
scheduler: {}
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
featureGates:
  SupportIPVSProxyMode: true
mode: ipvs
~
```

CSDN @firesuperman

指定从yml文件初始化安装，以及自动颁发证书

```
kubeadm init --config=kubeadm-config.yaml --experimental-upload-certs | tee kubeadm-init.log
```

查看日志是否提示安装成功，同时要求需要执行以下命令



```
[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 192.168.26.181:6443 --token abcdef.0123456789abcdef \
    --discovery-token-ca-cert-hash sha256:6e6d77247f6700a00329977cce5f7d73e8d7d2c8bd3b0
13ea572773dc2eb0cbd
[root@ots001 ~]#
```

CSDN @firesuperman

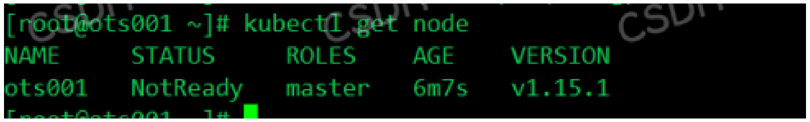
```
mkdir -p $HOME/.kube
```

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

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

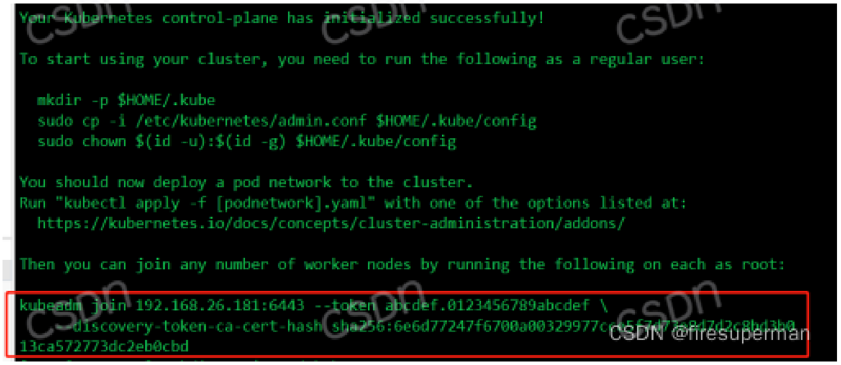
执行kubectl get node命令查看节点状态

```
kubectl get node
```



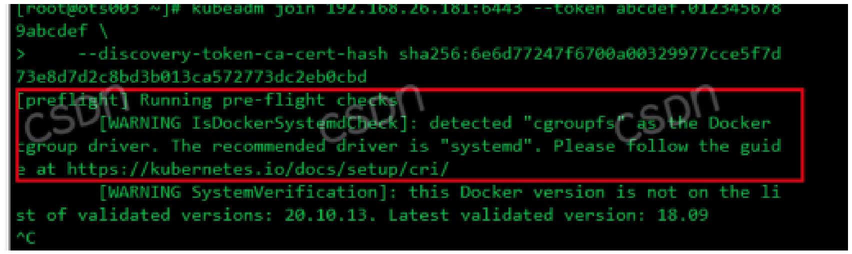
```
[root@ots001 ~]# kubectl get node
NAME     STATUS     ROLES    AGE    VERSION
ots001   NotReady   master   6m7s   v1.15.1
[root@ots001 ~]#
```

添加其他节点
在初始化日志中也提示了对加如节点的操作，**复制到其他节点机器上执行**即可



```
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 192.168.26.181:6443 --token abcdef.0123456789abcdef \
    --discovery-token-ca-cert-hash sha256:6e6d77247f6700a00329977cce5f7d73e8d7d2c8bd3b0
13ca572773dc2eb0cbd
```

可能出现的问题

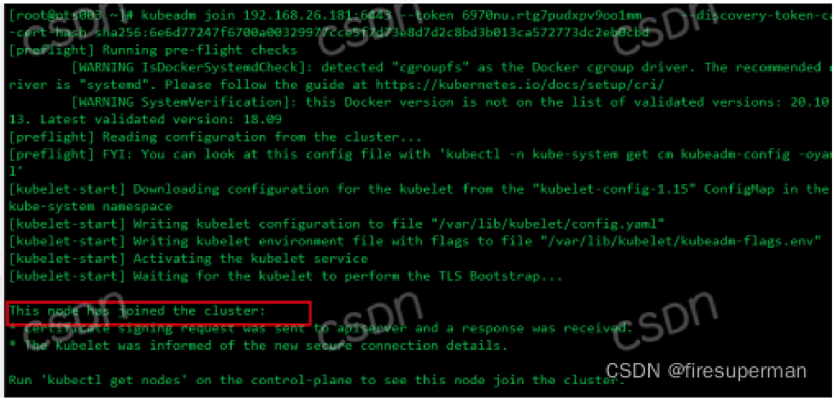


```
[root@ots003 ~]# kubeadm join 192.168.26.181:6443 --token abcdef.012345678
9abcdef \
>     --discovery-token-ca-cert-hash sha256:6e6d77247f6700a00329977cce5f7d
73e8d7d2c8bd3b013ca572773dc2eb0cbd
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker
cgroup driver. The recommended driver is "systemd". Please follow the guid
e at https://kubernetes.io/docs/setup/cri/
        [WARNING SystemVerification]: this Docker version is not on the li
st of validated versions: 20.10.13. Latest validated version: 18.09
^C
```

因为kubeadm在使用过程中token的有效期只有24h，需要重新生成，才能解决上述问题，在安装过程中因为其他节点是1天以后才安装的，所以导致了该问题
解决办法：
在主节点上执行命令生产新的加入节点命令

```
kubeadm token create --print-join-command
```

使用输出的新的命令到节点上执行即可
加入节点成功



```
[root@ots003 ~]# kubeadm join 192.168.26.181:6443 --token 6970nu.rtg7pudxpv9oo1mm     --discovery-token-c
-cert-hash sha256:6e6d77247f6700a00329977cc5f7d73e8d7d2c8bd3b013ca572773dc2eb0cbd
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended d
river is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 20.10
13. Latest validated version: 18.09
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oya
l'
[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.
```

CSDN @firesuperman

部署网络Flannel

Flannel是 CoreOS 团队针对 Kubernetes 设计的一个覆盖网络（Overlay Network）工具，其目的在于帮助每一个使用 Kuberentes 的 CoreOS 主机拥有一个完整的子网。
在主节点上创建文件夹

```
mkdir -p k8s_install/core
```

```
mkdir -p k8s_install/plugin/flannel
```

将 kubeadm-init.log kubeadm-config.yaml放到core目录，其他文件则可以删除

```
mv kubeadm-init.log kubeadm-config.yaml  k8s_install/core
```

进入到flannel目录下

```
cd k8s_install/plugin/flannel
```

然后执行命令：

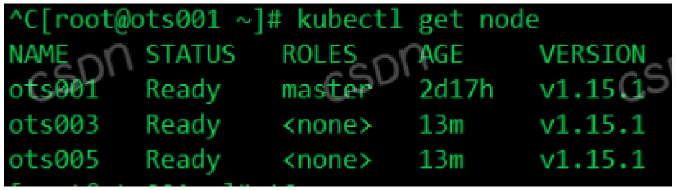
```
wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

之后执行:

```
kubectl create -f kube-flannel.yml
```

添加子节点后可通过命令查看个节点状态

```
kubectl get node
```



```
^C[root@ots001 ~]# kubectl get node
NAME     STATUS   ROLES    AGE     VERSION
ots001   Ready    master   2d17h   v1.15.1
ots003   Ready    <none>   13m     v1.15.1
ots005   Ready    <none>   13m     v1.15.1
```

三个节点状态都是 Ready则说明部署成功
如果其他节点状态是NoReady,可能是节点在初始化过程中，等几分再执行命令查看节点状态