配置阿里yum源（三台主机操作）:

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

[kubernetes]

name=Kubernetes

baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=0

repo\_gpgcheck=0

EOF

编辑hosts文件（三台主机操作）：

vim /etc/hosts

添加：

192.168.1.20 k8smaster

192.168.1.21 k8snode1

192.168.1.22 k8snode2

发送给k8snode节点：

scp /etc/yum.repos.d/Ali-k8s.repo [root@k8snode1:/etc/yum.repos.d/](mailto:root@k8snode1:/etc/yum.repos.d/)

scp /etc/yum.repos.d/Ali-k8s.repo [root@k8snode2:/etc/yum.repos.d/](mailto:root@k8snode2:/etc/yum.repos.d/)

scp /etc/hosts root@k8snode1:/etc/hosts

scp /etc/hosts root@k8snode2:/etc/hosts

修改主机名（三台主机操作）：

hostname k8smaster/k8snode1/k8snode2

bash

设置docker服务开机自启（三台主机操作）：

systemctl enable docker.service

升级内核（三台主机操作）：

wget -O/etc/yum.repos.d/aliyun-yilai.repo http://mirrors.aliyun.com/repo/Centos-7.repo

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

systemctl enable kubelet.service

rpm --import <https://www.elrepo.org/RPM-GPG-KEY-elrepo.org> //ELRepo 仓库

yum -y install <https://www.elrepo.org/elrepo-release-7.0-4.el7.elrepo.noarch.rpm> //ELRepo 仓库

yum install --enablerepo=elrepo-kernel kernel-ml -y //安装最新内核

grub2-set-default 0 //设置默认内核为最新的

grub2-mkconfig -o /boot/grub2/grub.cfg //生成 grub 配置文件

重启（三台主机操作）：

reboot

验证（三台主机操作）：

uname -r



关机，调整，虚拟机至少2核4G内存

确保主机一致性（三台主机操作）：

# 查看mac地址确认

cat /sys/class/net/ens33/address

# 查看确认UUID

cat /sys/class/dmi/id/product\_uuid

关闭防火墙（三台主机操作）：

systemctl disable firewalld.service && systemctl stop firewalld.service

关闭selinux（三台主机操作）：

[root@localhost ~]# vi /etc/selinux/config

SELINUX=disabled

关闭swap（三台主机操作）：

# K8s 1.8版本之后要求必须关闭

[root@localhost ~]# swapoff -a

[root@localhost ~]# cp /etc/fstab /etc/fstab\_bak //备份

[root@localhost ~]# vi /etc/fstab //删除swap，swap挂载注释掉

[root@localhost ~]# vi /etc/sysctl.conf

vm.swappiness = 0

[root@localhost ~]# sysctl -p

# 查看swap被关闭，全为0（三台主机操作）：

[root@localhost ~]# free -m

# 配置各节点系统内核参数使流过网桥的流量也进入iptables/netfilter框架中（三台主机操作）：

[root@localhost ~]# vi /etc/sysctl.conf

[root@localhost ~]# sysctl -p

net.ipv4.ip\_forward = 1

vm.swappiness = 0

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

重启（三台主机操作）：

reboot

拉取核心组件镜像（翻墙不稳定可能需要多拉几次）：

[root@localhost ~]# kubeadm config images pull

kubeadm version //查看版本

kubeadm config images list --kubernetes-version=v1.15.1

下载镜像

systemctl enable kubelet.service

systemctl stop kubelet.service

systemctl restart kubelet.service

kubeadm version //查看版本

查看版本需要的镜像：

kubeadm config images list --kubernetes-version=v1.15.1

然后从docker的google的克隆镜像拉取：

docker pull mirrorgooglecontainers/kube-apiserver:v1.15.1

docker pull mirrorgooglecontainers/kube-controller-manager:v1.15.1

docker pull mirrorgooglecontainers/kube-scheduler:v1.15.1

docker pull mirrorgooglecontainers/kube-proxy:v1.15.1

docker pull mirrorgooglecontainers/pause:3.1

docker pull mirrorgooglecontainers/etcd:3.3.10

docker pull mirrorgooglecontainers/coredns:1.3.1

#这个下不动，改用阿里云的

docker pull registry.cn-hangzhou.aliyuncs.com/google\_containers/coredns:1.3.1

#因为国内网络原因失败个好几次很正常

#因为这些镜像3台主机都要下载，但下载的效率极低，最好一台下载完成后打包传过去

批量修改镜像标签：

docker images | grep mirrorgooglecontainers | awk '{print "docker tag",$1":"$2,$1":"$2}' | sed -e 's#mirrorgooglecontainers#k8s.gcr.io#2' | sh -x

批量删除旧：

docker images |grep mirrorgooglecontainers |awk '{print "docker rmi ", $1":"$2}' |sh -x

coredns仍然需要但需修改：

docker tag registry.cn-hangzhou.aliyuncs.com/google\_containers/coredns:1.3.1 k8s.gcr.io/coredns:1.3.1

docker rmi registry.cn-hangzhou.aliyuncs.com/google\_containers/coredns:1.3.1

#镜像打包

docker save -o scheduler k8s.gcr.io/kube-scheduler

docker save -o c k8s.gcr.io/kube-controller-manager

docker save -o p k8s.gcr.io/kube-proxy

docker save -o a k8s.gcr.io/kube-apiserver

docker save -o dns k8s.gcr.io/coredns

docker save -o etcd k8s.gcr.io/etcd

docker save -o pause k8s.gcr.io/pause

#xftp传过去，导入镜像

docker load < a

docker load < c

docker load < dns

docker load < etcd

docker load < p

docker load < pause

docker load < scheduler

[root@k8smaster ~]# mkdir -p /etc/cni/net.d/

vim /etc/cni/net.d/10-flannel.conf

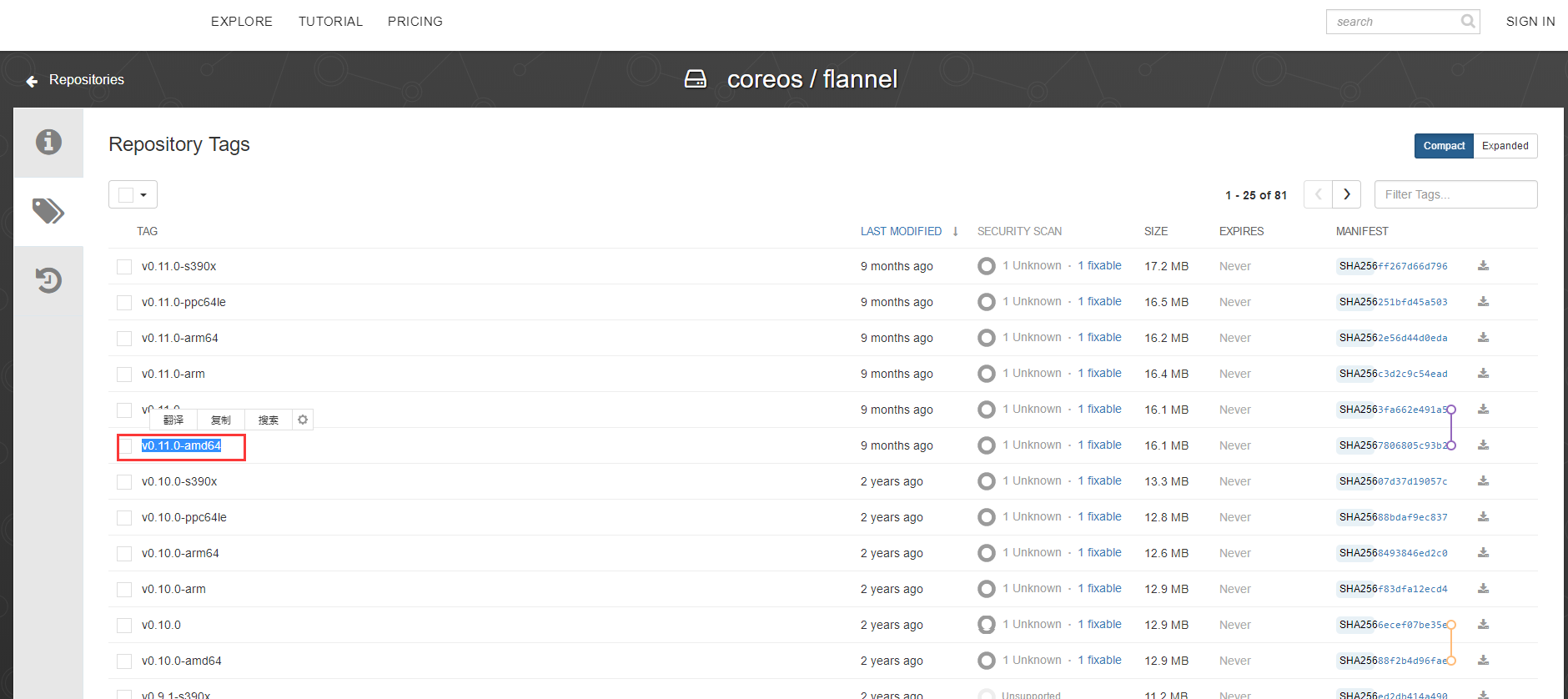
写入：

{"name":"cbr0","type":"flannel","delegate": {"isDefaultGateway": true}}

scp /etc/cni/net.d/10-flannel.conf [root@k8snode1:/etc/cni/net.d/](mailto:root@k8snode1:/etc/cni/net.d/) //cp给节点1

scp /etc/cni/net.d/10-flannel.conf [root@k8snode2:/etc/cni/net.d/](mailto:root@k8snode2:/etc/cni/net.d/) //cp给节点2

flanneld网络网址：quay.io



拉取flannel镜像：

docker pull quay-mirror.qiniu.com/coreos/flannel:v0.10.0-amd64

docker tag quay-mirror.qiniu.com/coreos/flannel:v0.10.0-amd64 quay.io/coreos/flannel:v0.10.0-amd64

mkdir /usr/share/oci-umount/oci-umount.d -p

mkdir /run/flannel/

vim /run/flannel/subnet.env

写入:

FLANNEL\_NETWORK=172.100.0.0/16

FLANNEL\_SUBNET=172.100.1.0/24

FLANNEL\_MTU=1450

FLANNEL\_IPMASQ=true

所有节点及master上都要添加flannel：

scp /run/flannel/subnet.env root@k8snode1:/run/flannel/ //cp给节点1

scp /run/flannel/subnet.env root@k8snode2:/run/flannel/ //cp给节点2

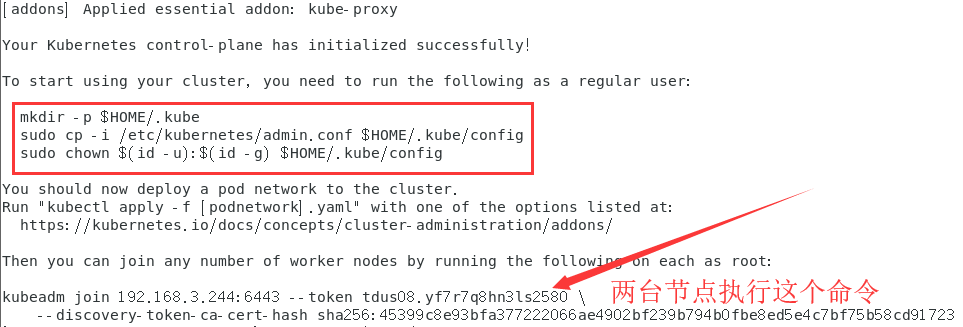
systemctl daemon-reload

systemctl restart kubelet

systemctl restart docker

创建Master节点：

kubeadm init --apiserver-advertise-address 192.168.1.20 --pod-network-cidr=10.244.0.0/16



根据它的提示，为了让 kubelet 为非root用户使用，需要作如下的一些配置：

mkdir -p $HOME/.kube

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

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

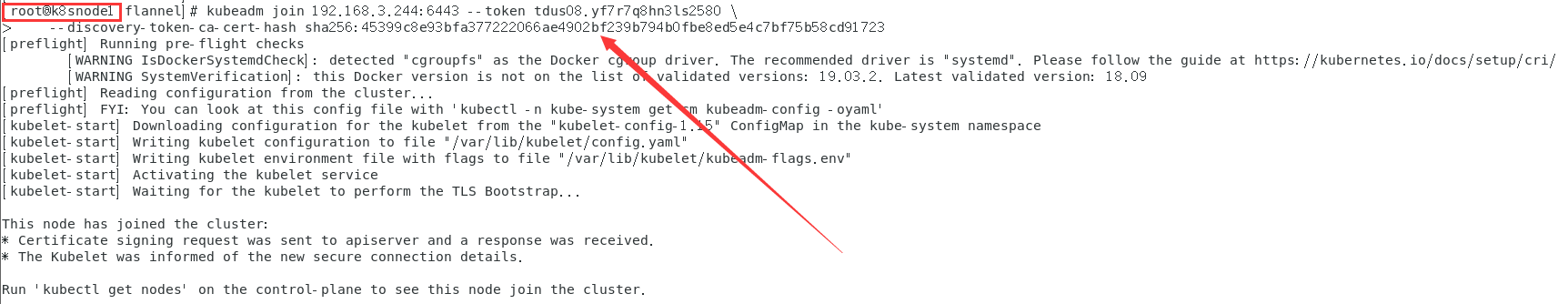
#ps 如果你中间出错了，使用了kubeadm reset（从主）命令的话，上面的$HOME/.kube需要删除

（rm -rf $HOME/.kube），后重新添加不然报错509x

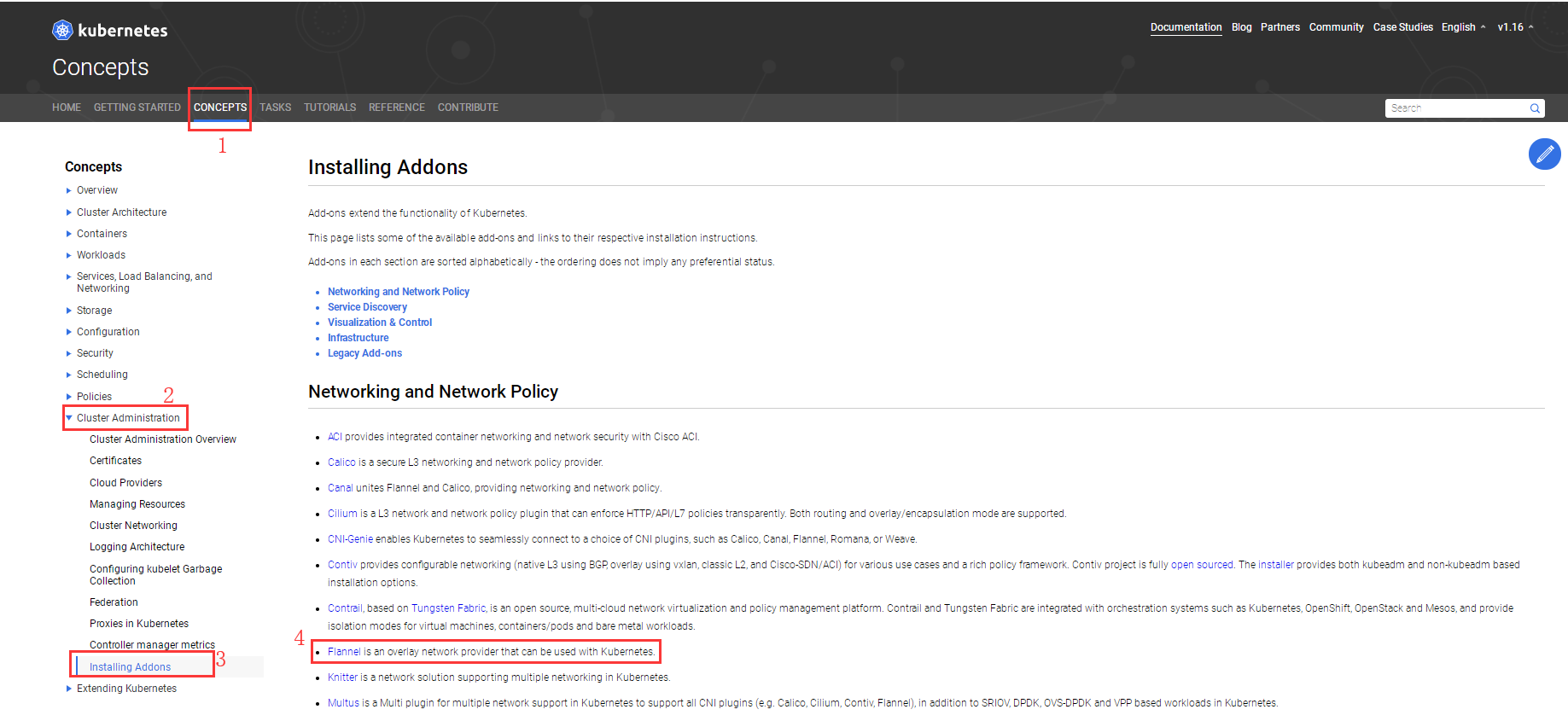
#从节点加入群集

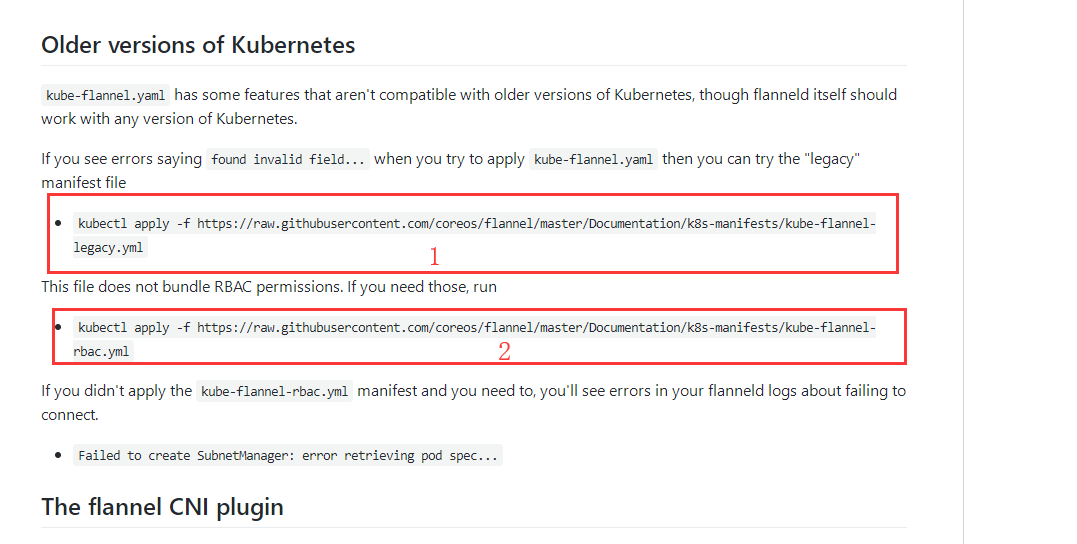
kubeadm join 192.168.1.20:6443 --token j4c6zt.z5awfr9319e01cry \

--discovery-token-ca-cert-hash sha256:711fd5833c9f6896230edc02f4c3c3d5d7d73e49ee2192518016eebcb5f2fedd



打开提示的网页：kubernetes.io

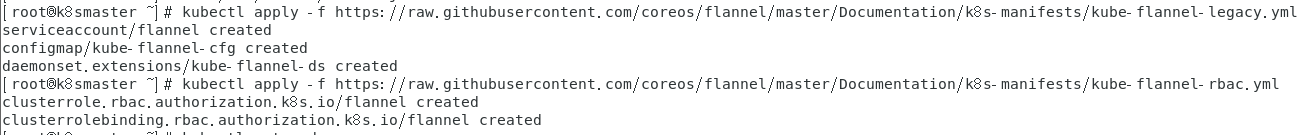




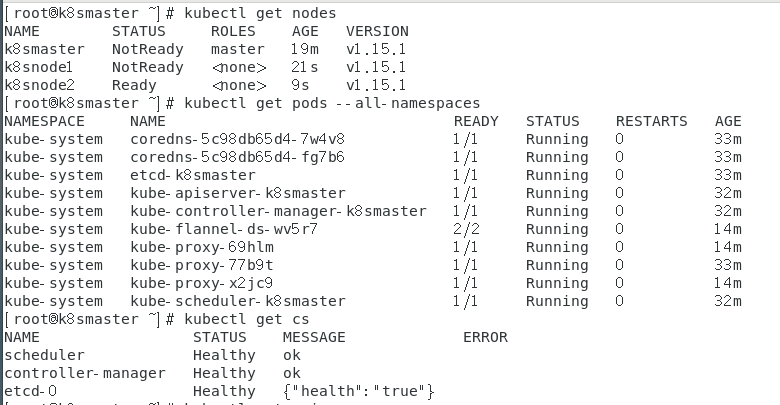
master主机输入这两条指令：

kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-legacy.yml>

kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-rbac.yml>



查看集群状态：



#整理以下这些东西都是些什么

kube-system coredns-5c98db65d4-pcwqb 1/1 Running 0 7m52s

kube-system coredns-5c98db65d4-wvndx 1/1 Running 0 7m52s

#上面这俩应该是从节点之间的域名解析

kube-system kube-flannel-ds-4zkdt 2/2 Running 0 11m

kube-system kube-flannel-ds-jxqct 2/2 Running 0 11m

#这是主节点于两台从节点通信使用的网络，如果不是2/2，那么应该是镜像不对，可能是版本或者其他因素

kube-system etcd-k8smaster 1/1 Running 0 6m50s

kube-system kube-apiserver-k8smaster 1/1 Running 0 7m10s

kube-system kube-controller-manager-k8smaster 1/1 Running 0 7m11s

#主服务器上存放的etcd数据库，节点接口，以及期望节点数量控制

kube-system kube-proxy-74n6s 1/1 Running 0 7m52s

kube-system kube-proxy-jgr2s 1/1 Running 0 7m27s

kube-system kube-proxy-sqdt5 1/1 Running 0 7m23s

#3个proxy 负责节点的负载均衡

kube-system kube-scheduler-k8smaster 1/1 Running 0 7m2s

#负责调度任务

开启TAB键：

[root@k8smaster ~]# source /usr/share/bash-completion/bash\_completion

[root@k8smaster ~]# source <(kubectl completion bash)