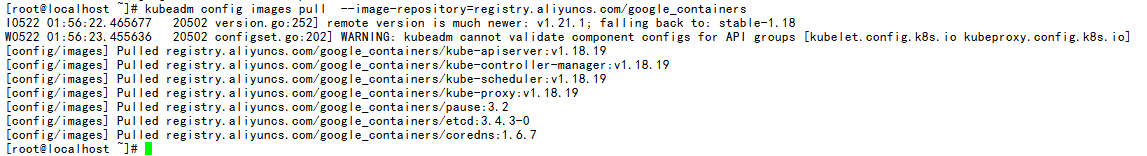
阿里云镜像仓库：registry.aliyuncs.com

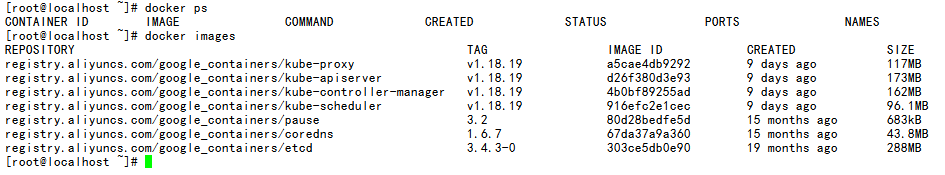
找一个可以联网的服务器或者虚拟机，安装kubeadm、docker、kubelet

使用命令下载k8s的docker镜像

# kubeadm config images pull --image-repository=registry.aliyuncs.com/google\_containers



# docker images



使用docker命令导出所有的镜像文件

# docker save registry.aliyuncs.com/google\_containers/kube-proxy:v1.18.19 \

registry.aliyuncs.com/google\_containers/kube-apiserver:v1.18.19 \

registry.aliyuncs.com/google\_containers/kube-controller-manager:v1.18.19 \

registry.aliyuncs.com/google\_containers/kube-scheduler:v1.18.19 \

registry.aliyuncs.com/google\_containers/pause:3.2 \

registry.aliyuncs.com/google\_containers/coredns:1.6.7 \

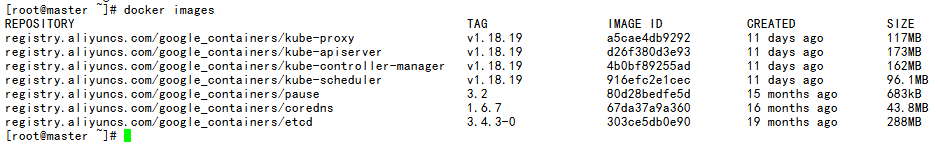
registry.aliyuncs.com/google\_containers/etcd:3.4.3-0 -o k8s.tar

注意：这里需要注意的是，如果只使用id导出镜像文件，则再次导入docker镜像后，将丢失tag标签，所以这里使用tag标签来导出docker镜像文件

# 上传到k8s服务器，导入docker镜像文件

# docker load -i k8s.tar

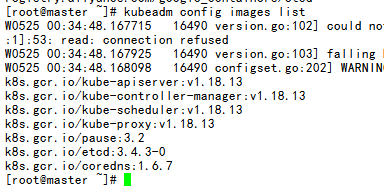
# docker images



此时，docker镜像已经存在，但是不能直接使用，因为kubeadm命令会默认使用k8s.gcr.io源，需要修改images的tag标签为k8s.gcr.io

查看k8s使用的镜像名称：

# kubeadm config images list



# docker tag registry.aliyuncs.com/google\_containers/kube-proxy:v1.23.3 k8s.gcr.io/kube-proxy:v1.23.3

# docker tag registry.aliyuncs.com/google\_containers/kube-apiserver:v1.23.3 k8s.gcr.io/kube-apiserver:v1.23.3

# docker tag registry.aliyuncs.com/google\_containers/kube-controller-manager:v1.23.3 k8s.gcr.io/kube-controller-manager:v1.23.3

# docker tag registry.aliyuncs.com/google\_containers/kube-scheduler:v1.23.3 k8s.gcr.io/kube-scheduler:v1.23.3

# docker tag registry.aliyuncs.com/google\_containers/pause:3.6 k8s.gcr.io/pause:3.6

# docker tag registry.aliyuncs.com/google\_containers/coredns:v1.8.6 k8s.gcr.io/coredns/coredns:v1.8.6

# docker tag registry.aliyuncs.com/google\_containers/etcd:3.5.1-0 k8s.gcr.io/etcd:3.5.1-0

初始化前需要关闭防火墙，关闭色Linux，关闭swap分区

设置防火墙：

iptables -P FORWARD ACCEPT

在1.13版本之后，k8s集群默认设置为DROP（原因未找到）

# vi /etc/sysctl.conf

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

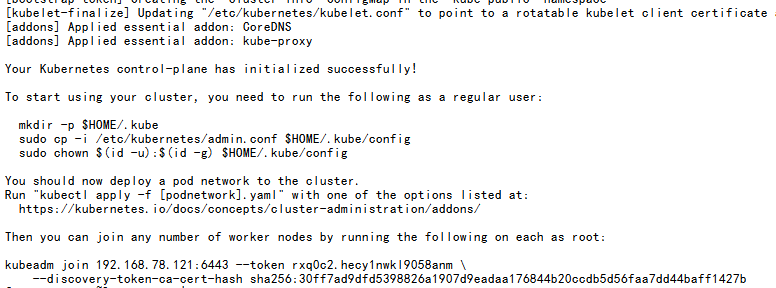
# sysctl -p

因为下载的images镜像版本是v1.18.19，而kubeadm命令查询到的默认使用版本是v1.18.13，则初始化的命令如下：

# kubeadm init --apiserver-advertise-address 192.168.78.121 --kubernetes-version v1.18.19 --pod-network-cidr 10.1.1.0/16

手动指定版本即可

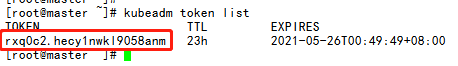
注意：pod-network-cidr必须指定，否则初始化的集群会有问题



1. 初始化完成后，就可以在其余的node节点上，执行提示的命令，将node节点加入到k8s集群中。

如果不是在第一时间将节点加入集群，后续加入，需要查看master的token信息

# kubeadm token list



在node节点上执行命令，加入k8s集群

# kubeadm join 192.168.78.121:6443 --token rxq0c2.hecy1nwkl9058anm --discovery-token-unsafe-skip-ca-verification

说明：

--discovery-token-unsafe-skip-ca-verification 不验证CA证书

或者：忘记初始master节点时的node节点加入集群命令

kubeadm token create $token --print-join-command --ttl=1h

说明：--ttl=1h 代表一小时之后这个token将自动删除，也可以删除1m，0代表永久有效

master复制kube配置文件到普通用户，最新的版本，默认不允许使用root用户执行kubectl命令

# mkdir /home/ap/safm/.kube

# cp -i /etc/kubernetes/admin.conf /home/ap/safm/.kube/config

# chown -R safm.safm /home/ap/safm/.kube

切换到safm用户

$ kubectl get nodes

clipboard.png

到这里，node节点以及加入到k8s集群中，状态为“NotReady”，是因为我们没有安装pod网络插件，接下来安装flannel网络插件

在有网络的服务器上，下载flannel的yml文件

# wget <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

# cat kube-flannel.yml | grep image

clipboard.png

由此可知，flannel使用的镜像是 quay.io/coreos/flannel:v0.14.0

在服务器上设置代理，下载flannle的镜像文件

# export HTTP\_PROXY="http://192.168.0.173:1081"

# export HTTPS\_PROXY="http://192.168.0.173:1081"

# export ALL\_PROXY="http://192.168.0.173:1081"

# docker pull quay.io/coreos/flannel:v0.14.0

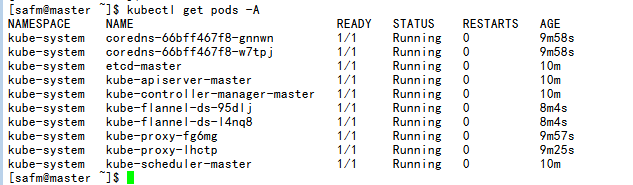
导出flannel镜像，上传到k8s服务器中

使用flannel.yml文件创建flannel资源

$ kubectl create -f kube-flannel.yaml

$ kubectl get pods -A

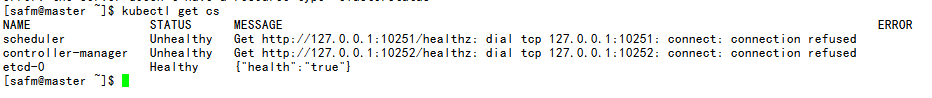
可以看到所有的pods已经正常运行



检查k8s集群状态

$ kubectl get cs

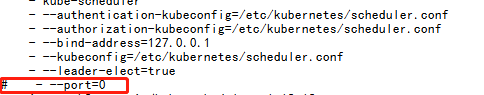
cs是componentstatus的缩写，component是组件的意思



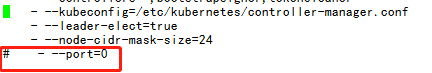
可以看到scheduler和controller-manager的状态不正常

出现这种情况，是/etc/kubernetes/manifests/下的kube-controller-manager.yaml和kube-scheduler.yaml设置的默认端口是0导致的，解决方式是注释掉对应的port即可，并重启kubelet服务即可

# vi /etc/kubernetes/manifests/kube-scheduler.yaml

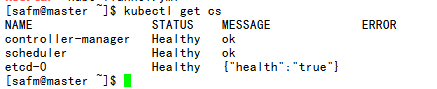


# vi /etc/kubernetes/manifests/kube-controller-manager.yaml



# systemctl restart kubelet

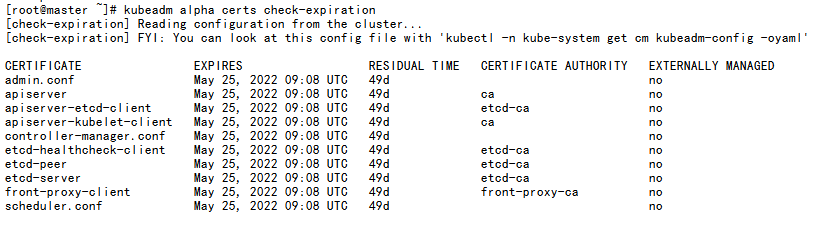
$ kubectl get cs



至此，k8s集群搭建完成，不过目前集群证书的有效期只有一年

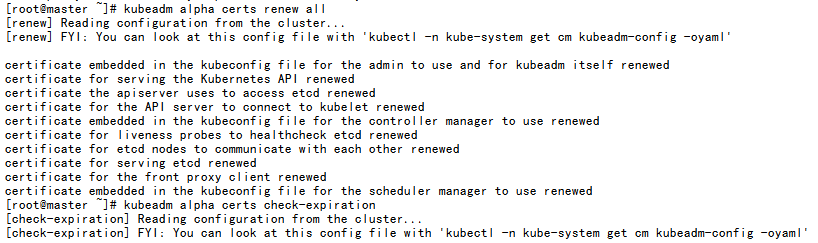
查看当前k8s集群证书的有效时间

# kubeadm alpha certs check-expiration



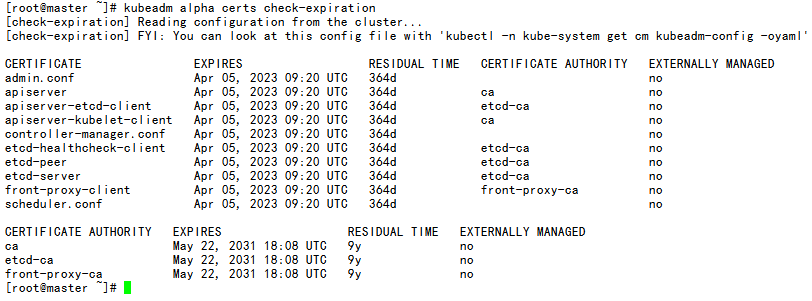
更新集群证书有效时间：

# kubeadm alpha certs renew all



查看更新后的集群证书有效时间：

# kubeadm alpha certs check-expiration



可以看到证书的有效期已经更新，可以设置一月一次的定时任务，定时更新证书

注意：alpha命令为测试版本，有可能有各种BUG，生产环境中慎用