Ubuntu系统部署k8s

1.安装K8S

1.让apt支持SSL传输

```
sudo apt-get update sudo apt-get -y install apt-transport-https ca-certificates
```

2.下载 gpg 密钥

```
curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | sudo apt-key add -
```

3.添加 k8s 镜像源

创建/etc/apt/sources.list.d/kubernetes.list文件，并添加阿里的K8S源。

```
sudo touch /etc/apt/sources.list.d/kubernetes.list sudo vi
/etc/apt/sources.list.d/kubernetes.list
```

添加的内容如下：

```
deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main
```

4.安装k8s

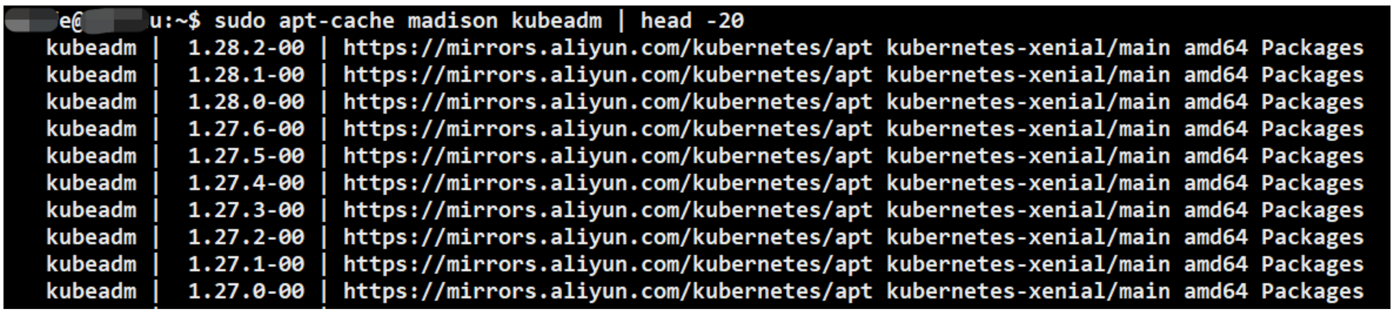
更新软件源

```
sudo apt-get update
```

查看版本

```
sudo apt-cache madison kubeadm sudo apt-cache madison kubelet sudo apt-cache madison
kubectl
```

三个软件的版本都是下边这样（我为了只看前几行，加了个| head -20，否则会输出很多）



```
[redacted]e@[redacted]u:~$ sudo apt-cache madison kubeadm | head -20
   kubeadm |  1.28.2-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
   kubeadm |  1.28.1-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
   kubeadm |  1.28.0-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
   kubeadm |  1.27.6-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
   kubeadm |  1.27.5-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
   kubeadm |  1.27.4-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
   kubeadm |  1.27.3-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
   kubeadm |  1.27.2-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
   kubeadm |  1.27.1-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
   kubeadm |  1.27.0-00 | https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
```

安装指定版本（本处我安装最新版本（1.28.2-00））

```
sudo apt-get install -y kubelet=1.28.2-00 kubeadm=1.28.2-00 kubectl=1.28.2-00
```

kubeadm用于初始化环境，kubectl用于操作kubelet。

5.启动k8s

启动k8s

```
sudo systemctl start kubelet
```

设置开机启动

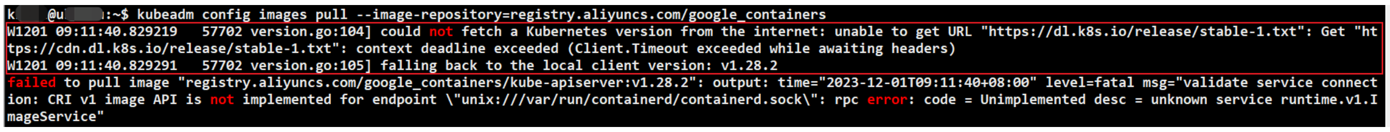
```
sudo systemctl enable kubelet
```

6.命令自动补全

 这几个命令没有自动补全，用起来不方便。启用自动补全的方法见：这里

2.配置K8S环境

执行命令时有时候会报下边错误，不用管它。报错原因：去k8s官网查看发布版列表，但是国内访问不了这个网站，连不上；此时会自动使用本地客户端的版本，本处是：1.28.2。



```
k[redacted]@u[redacted]:~$ kubeadm config images pull --image-repository=registry.aliyuncs.com/google_containers
W1201 09:11:40.829219   57702 version.go:104] could not fetch a Kubernetes version from the internet: unable to get URL "https://dl.k8s.io/release/stable-1.txt": Get "ht
tps://cdn.dl.k8s.io/release/stable-1.txt": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
W1201 09:11:40.829291   57702 version.go:105] falling back to the local client version: v1.28.2
failed to pull image "registry.aliyuncs.com/google_containers/kube-apiserver:v1.28.2": output: time="2023-12-01T09:11:40+08:00" level=fatal msg="validate service connect
ion: CRI v1 image API is not implemented for endpoint \"unix:///var/run/containerd/containerd.sock\": rpc error: code = Unimplemented desc = unknown service runtime.v1.I
mageService"
```

1.禁用防火墙和swap

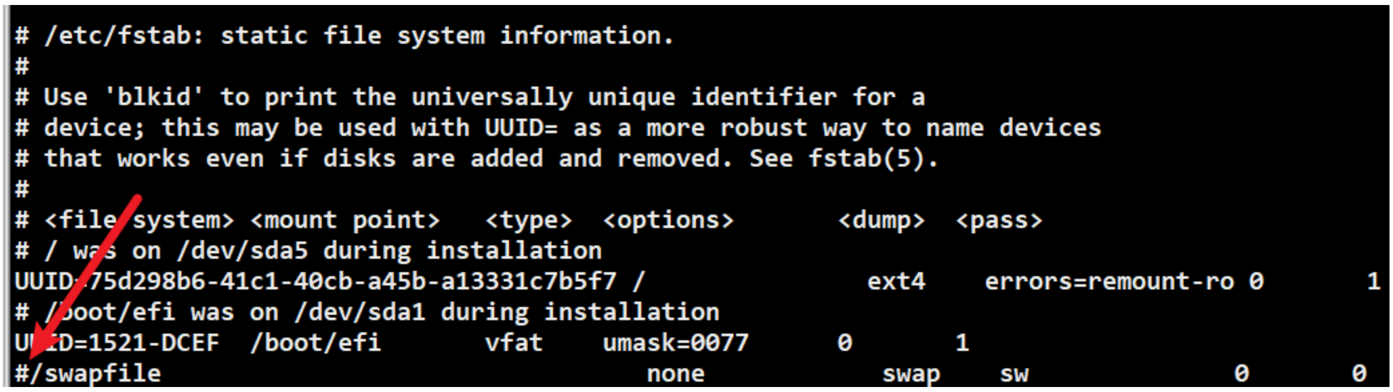
这两步必须操作！不然k8s无法正常运行。

1.禁用防火墙

```
sudo systemctl stop ufw sudo systemctl disable ufw
```

2.禁用swap

修改/etc/fstab文件，将swap所在的行注释掉



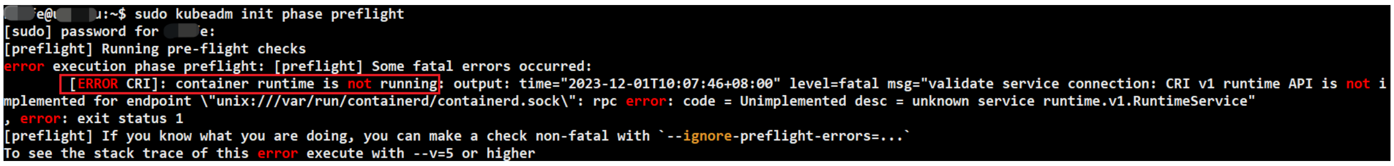
```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda5 during installation
UUID=75d298b6-41c1-40cb-a45b-a13331c7b5f7 /               ext4    errors=remount-ro 0       1
# /boot/efi was on /dev/sda1 during installation
UUID=1521-DCEF  /boot/efi       vfat    umask=0077      0       1
#/swapfile                                 none            swap    sw              0       0
```

2.预检

执行安装之前，会做一系列的系统预检查，以确保主机环境符合安装要求，如果检查失败，就直接终止，不进行init操作。因此可以通过命令执行预检查操作，确保系统就绪后再进行init操作。会检查内存大小等。

```
sudo kubeadm init phase preflight
```

到这一步一般会报错：



```
[redacted]e@[redacted]u:~$ sudo kubeadm init phase preflight
[sudo] password for [redacted]e:
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR CRI]: container runtime is not running: output: time="2023-12-01T10:07:46+08:00" level=fatal msg="validate service connection: CRI v1 runtime API is not i
mplemented for endpoint \"unix:///var/run/containerd/containerd.sock\": rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService"
, error: exit status 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

原因是：默认关闭了cri（容器运行时），需要启用它。 本处先不处理，下边第4步安装容器运行时之后，这个就解决了。

3.网络设置

转发 IPv4 并让 iptables 看到桥接流量。 官网：[这里]

执行如下指令：

1.向k8s.conf写入两行配置并启用它

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf overlay br_netfilter EOF sudo modprobe overlay sudo modprobe br_netfilter
```

2.设置所需的 sysctl 参数，参数在重新启动后保持不变

下边命令会向k8s.conf写入三行内容

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1 net.ipv4.ip_forward                 = 1 EOF
```

3.应用 sysctl 参数而不重新启动

```
sudo sysctl --system
```

4.安装容器运行时

官网：[这里]

需要在集群内每个节点上安装一个容器运行器以使 Pod 可以运行在上面。容器运行器实现了CRI（容器运行时接口），常见的容器运行器有：

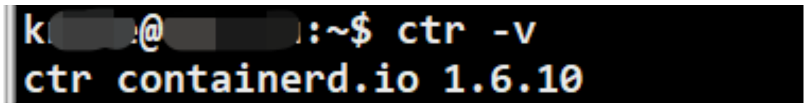
1. containerd
2. CRI-O
3. Docker（使用 cri-dockerd 适配器来将 Docker 与 Kubernetes 集成）

4. Mirantis Container Runtime

我是用containerd，这是现在最常用的。做完上边第一步（1.安装K8S）后，默认会安装contrainerd，可以用此命令查看：

```
ctr -v
```

结果



```
k[redacted]@[redacted]:~$ ctr -v
ctr containerd.io 1.6.10
```

1.生成配置并修改

先切换到root用户，然后执行如下命令：

```
containerd config default > /etc/containerd/config.toml
```

会生成配置文件：/etc/containerd/config.toml，修改如下配置项：

修改1：修改cgroup为systemd

原内容

```
SystemdCgroup = false
```

修改为

```
SystemdCgroup = true
```

kubelet 的 cgroup 驱动 默认是systemd，这里需要保持一致。详见：官网

修改2：沙箱使用阿里云的源

原内容

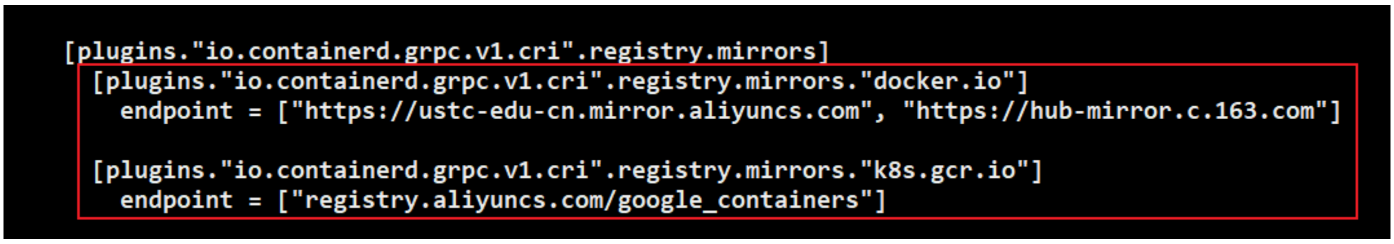
```
sandbox_image = "registry.k8s.io/pause:3.6"
```

修改为

```
sandbox_image = "registry.aliyuncs.com/google_containers/pause:3.6"
```

修改3：使用国内的镜像源

在plugins."io.containerd.grpc.v1.cri".registry.mirrors下边添加如下内容：



```
[plugins."io.containerd.grpc.v1.cri".registry.mirrors]
  [plugins."io.containerd.grpc.v1.cri".registry.mirrors."docker.io"]
    endpoint = ["https://ustc-edu-cn.mirror.aliyuncs.com", "https://hub-mirror.c.163.com"]

  [plugins."io.containerd.grpc.v1.cri".registry.mirrors."k8s.gcr.io"]
    endpoint = ["registry.aliyuncs.com/google_containers"]
```

内容为：

```
    [plugins."io.containerd.grpc.v1.cri".registry.mirrors."docker.io"]        endpoint =
["https://ustc-edu-cn.mirror.aliyuncs.com", "https://hub-mirror.c.163.com"]
[plugins."io.containerd.grpc.v1.cri".registry.mirrors."k8s.gcr.io"]        endpoint =
["registry.aliyuncs.com/google_containers"]
```

2.重启containerd

修改完毕后，重启containerd

```
sudo systemctl daemon-reload sudo systemctl restart containerd
```

让containerd开机自启

```
sudo systemctl enable containerd
```

5.搭建K8S集群

方案概述：使用kubeadm init命令创建master节点，然后再用kubeadm join命令创建node节点并加入master。

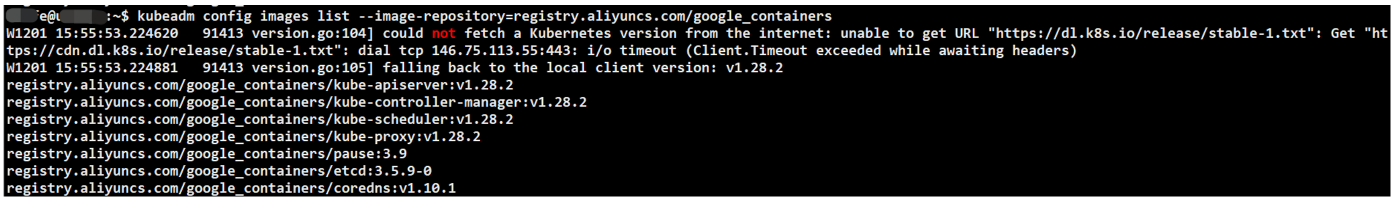
1.下载镜像

为了加快kubeadm创建集群的过程，可以预先下载所有镜像。

查看镜像

```
kubeadm config images list --image-repository=registry.aliyuncs.com/google_containers
```

结果

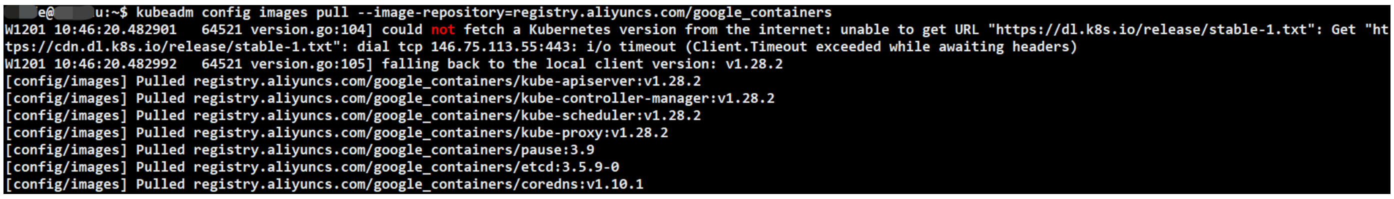


```
[redacted]e@[redacted]:~$ kubeadm config images list --image-repository=registry.aliyuncs.com/google_containers
W1201 15:55:53.224620   91413 version.go:104] could not fetch a Kubernetes version from the internet: unable to get URL "https://dl.k8s.io/release/stable-1.txt": Get "ht
tps://cdn.dl.k8s.io/release/stable-1.txt": dial tcp 146.75.113.55:443: i/o timeout (Client.Timeout exceeded while awaiting headers)
W1201 15:55:53.224881   91413 version.go:105] falling back to the local client version: v1.28.2
registry.aliyuncs.com/google_containers/kube-apiserver:v1.28.2
registry.aliyuncs.com/google_containers/kube-controller-manager:v1.28.2
registry.aliyuncs.com/google_containers/kube-scheduler:v1.28.2
registry.aliyuncs.com/google_containers/kube-proxy:v1.28.2
registry.aliyuncs.com/google_containers/pause:3.9
registry.aliyuncs.com/google_containers/etcd:3.5.9-0
registry.aliyuncs.com/google_containers/coredns:v1.10.1
```

下载镜像（阿里云）

```
kubeadm config images pull --image-repository=registry.aliyuncs.com/google_containers
```

结果



```
[redacted]e@[redacted]u:~$ kubeadm config images pull --image-repository=registry.aliyuncs.com/google_containers
W1201 10:46:20.482901   64521 version.go:104] could not fetch a Kubernetes version from the internet: unable to get URL "https://dl.k8s.io/release/stable-1.txt": Get "ht
tps://cdn.dl.k8s.io/release/stable-1.txt": dial tcp 146.75.113.55:443: i/o timeout (Client.Timeout exceeded while awaiting headers)
W1201 10:46:20.482992   64521 version.go:105] falling back to the local client version: v1.28.2
[config/images] Pulled registry.aliyuncs.com/google_containers/kube-apiserver:v1.28.2
[config/images] Pulled registry.aliyuncs.com/google_containers/kube-controller-manager:v1.28.2
[config/images] Pulled registry.aliyuncs.com/google_containers/kube-scheduler:v1.28.2
[config/images] Pulled registry.aliyuncs.com/google_containers/kube-proxy:v1.28.2
[config/images] Pulled registry.aliyuncs.com/google_containers/pause:3.9
[config/images] Pulled registry.aliyuncs.com/google_containers/etcd:3.5.9-0
[config/images] Pulled registry.aliyuncs.com/google_containers/coredns:v1.10.1
```

2.部署 master

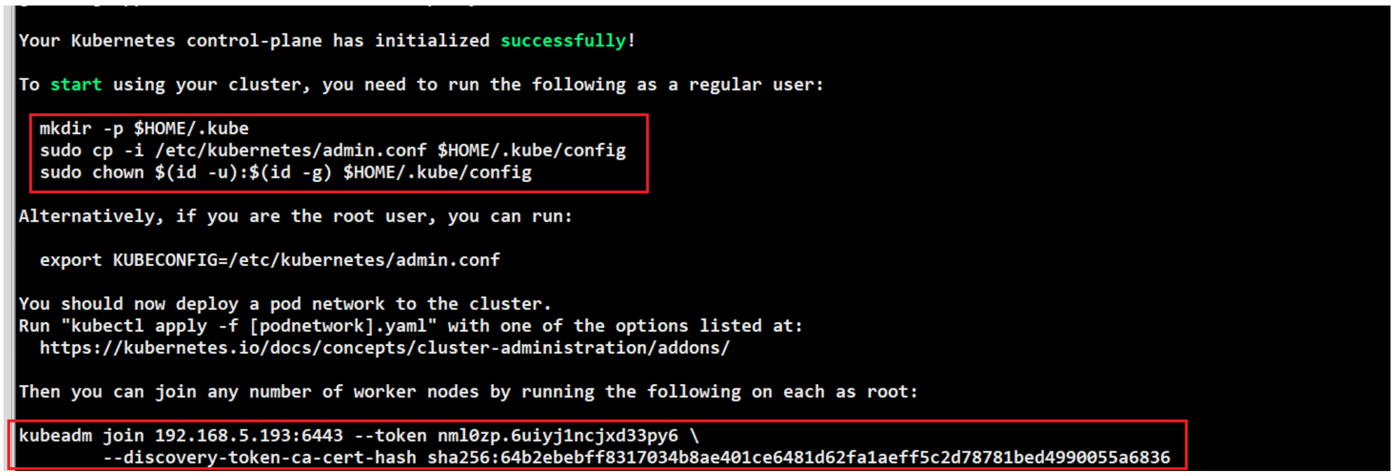
1.初始化master

`apiserver-advertise-address`后写自己电脑当前网络接口的ip地址，注意要设置为静态ip。

```
sudo kubeadm init \  --apiserver-advertise-address 192.168.5.193 \  --image-repository registry.aliyuncs.com/google_containers \  --kubernetes-version v1.28.2 \  --service-cidr 10.100.0.0/16 \  --pod-network-cidr 10.96.0.0/16
```

参数	说明
apiserver-advertise-address	API 服务器所公布的其正在监听的 IP 地址。如果未设置，则使用默认网络接口
image-repository	镜像拉取的仓库，填写国内镜像源
kubernetes-version	K8s 版本，本文值为 v1.28.2
service-cidr	为服务的虚拟 IP 地址另外指定 IP 地址段。默认值：“10.96.0.0/12”
pod-network-cidr	指明 Pod 网络可以使用的 IP 地址段。如果使用 Flannel 网络，必须配置这个字段。

结果：



```
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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 192.168.5.193:6443 --token nml0zp.6uiyj1ncjxd33py6 \
        --discovery-token-ca-cert-hash sha256:64b2ebebff8317034b8ae401ce6481d62fa1aeff5c2d78781bed4990055a6836
```

备注

如果你想重新kubeadm init，可以先kubeadm reset，再kubeadm init。

注意要删除创建的.kube目录：

```
rm -rf .kube/
```

2.配置环境

执行上边的命令：

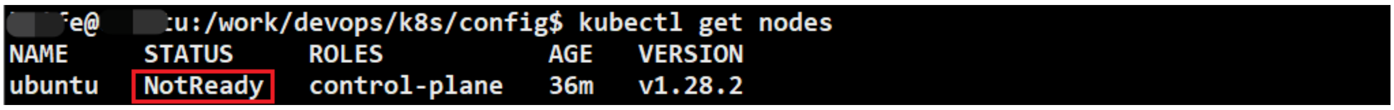
```
mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown
$(id -u):$(id -g) $HOME/.kube/config
```

3.查看状态

查看节点状态

```
kubectl get nodes
```

结果



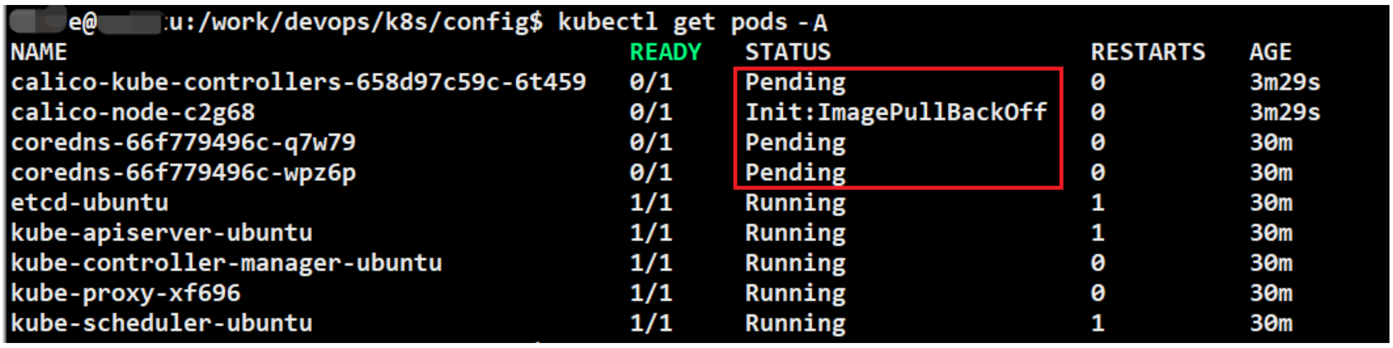
```
[redacted]e@[redacted]u:/work/devops/k8s/config$ kubectl get nodes
NAME     STATUS     ROLES           AGE   VERSION
ubuntu   NotReady   control-plane   36m   v1.28.2
```

可以发现，是“NotReady” 状态，因为还没有安装网络插件（不用管它，后边会安装网络插件）。

查看pods状态

```
kubectl get pods -A
```

结果



```
[redacted]e@[redacted]u:/work/devops/k8s/config$ kubectl get pods -A
NAME                                        READY   STATUS                  RESTARTS   AGE
calico-kube-controllers-658d97c59c-6t459    0/1     Pending                 0          3m29s
calico-node-c2g68                           0/1     Init:ImagePullBackOff   0          3m29s
coredns-66f779496c-q7w79                    0/1     Pending                 0          30m
coredns-66f779496c-wpz6p                    0/1     Pending                 0          30m
etcd-ubuntu                                 1/1     Running                 1          30m
kube-apiserver-ubuntu                       1/1     Running                 1          30m
kube-controller-manager-ubuntu              1/1     Running                 0          30m
kube-proxy-xf696                            1/1     Running                 0          30m
kube-scheduler-ubuntu                       1/1     Running                 1          30m
```

4.安装网络插件

K8S中网络架构是很重要的。CNI(Container Network Interface)意为容器网络接口，它是一种标准设计，目的是进行数据转发、让用户在容器创建或销毁时能够更容易地配置容器网络。

CNI网络插件只在master安装即可。

有如下常见的CNI网络插件产品：

1. Flannel（比较常用。简单易用，但缺乏高级功能，例如配置网络策略和防火墙）
2. Calico（最常用。高性能、高灵活性、功能强大）
3. Weave
4. Canal

本文使用Calico。

1.下载calico配置文件

先创建个文件夹，用于存放配置文件：

```
mkdir -p /work/devops/k8s/config cd /work/devops/k8s/config
```

下载calico配置文件

```
wget https://docs.tigera.io/archive/v3.25/manifests/calico.yaml
```

2.修改calico配置文件

vi calico.yaml

原来的配置

```
# - name: CALICO_IPV4POOL_CIDR #   value: "192.168.0.0/16"
```

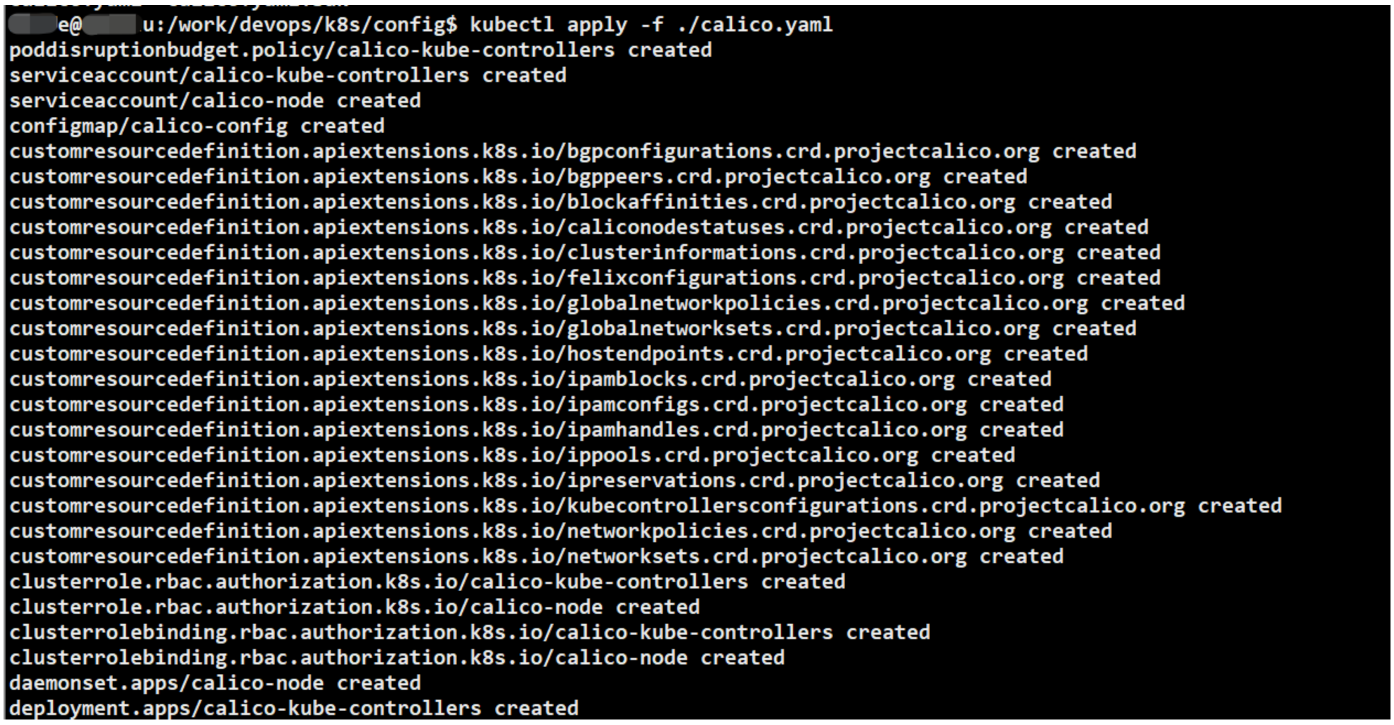
修改后的配置（注意：value与上边kubeadm init的–pod-network-cidr参数保持一致）

```
- name: CALICO_IPV4POOL_CIDR  value: "10.96.0.0/12"
```

3.启用calico

```
kubectl apply -f ./calico.yaml
```

结果：



```
[redacted]e@[redacted]u:/work/devops/k8s/config$ kubectl apply -f ./calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.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/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
```

4*.calico镜像拉取失败

删除 `calico.yaml` 文件中定义的所有 Kubernetes 资源。

```
kubectl delete -f ./calico.yaml
```

cd 进入calico.yaml所在文件夹下

```
cat calico.yaml | grep image
```

查看配置文件中的镜像源，可能是`docker.io/calico`

在网上搜取自己所需对应版本的镜像（与处理器型号有关），保存到阿里云免费镜像仓库

使用 `sed -i 's/Hello/Hi/g' example.yaml` 命令批量替换文本

```
sed -i 's/docker.io\/calico/registry.cn.beijing.aliyuncs.com\/calico-system/g' calico.yaml
```

确认是否提替换成功:

```
cat calico.yaml | grep image
```

启用calico:

```
kubectl apply -f ./calico.yaml
```

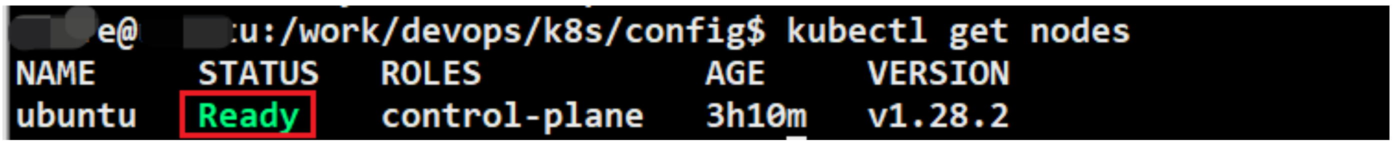
5.再次查看状态

需要过一会儿才会成为正常状态

查看节点状态

```
kubectl get nodes
```

结果



```
[redacted]e@[redacted]:u:/work/devops/k8s/config$ kubectl get nodes
NAME      STATUS    ROLES           AGE      VERSION
ubuntu    Ready     control-plane   3h10m    v1.28.2
```

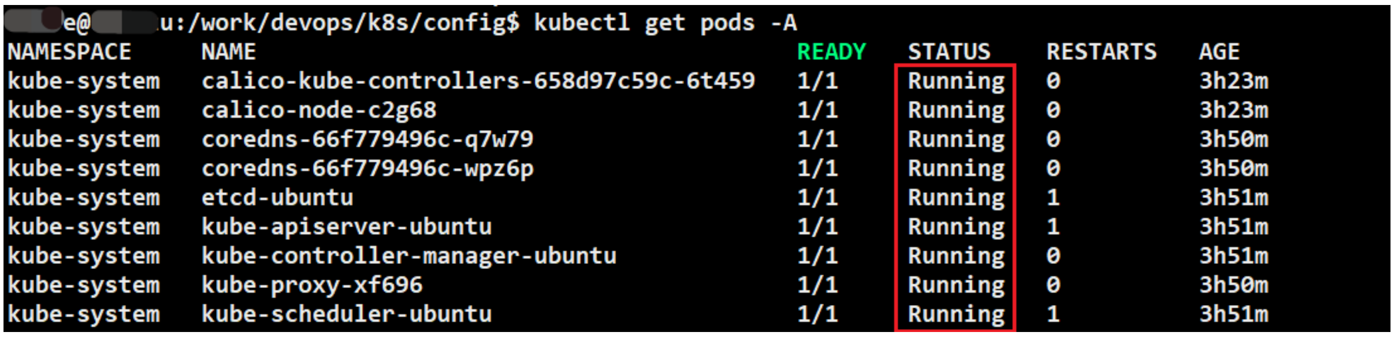
查看pod状态

```
kubectl get pods -A
```

或者:

```
kubectl get pods -n kube-system
```

结果



```
[redacted]e@[redacted]u:/work/devops/k8s/config$ kubectl get pods -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   calico-kube-controllers-658d97c59c-6t459   1/1     Running   0          3h23m
kube-system   calico-node-c2g68                          1/1     Running   0          3h23m
kube-system   coredns-66f779496c-q7w79                   1/1     Running   0          3h50m
kube-system   coredns-66f779496c-wpz6p                   1/1     Running   0          3h50m
kube-system   etcd-ubuntu                                1/1     Running   1          3h51m
kube-system   kube-apiserver-ubuntu                      1/1     Running   1          3h51m
kube-system   kube-controller-manager-ubuntu             1/1     Running   0          3h51m
kube-system   kube-proxy-xf696                           1/1     Running   0          3h50m
kube-system   kube-scheduler-ubuntu                      1/1     Running   1          3h51m
```

3.部署node节点（非必须）

将上边的部署命令复制下来，到node机器上去执行即可。

```
kubeadm join 192.168.5.193:6443 --token nml0zp.6uiyj1ncjxd33py6 \        --discovery-
token-ca-cert-hash sha256:64b2ebebff8317034b8ae401ce6481d62fa1aeff5c2d78781bed4990055a6836
```

注意：这段kubeamd join命令的token只有24h，24h就过期，在master上执行kubeadm token create –print-join-command 重新生成。

4.部署dashboard

见：Ubuntu安装k8s的dashboard（管理页面）-自学精灵