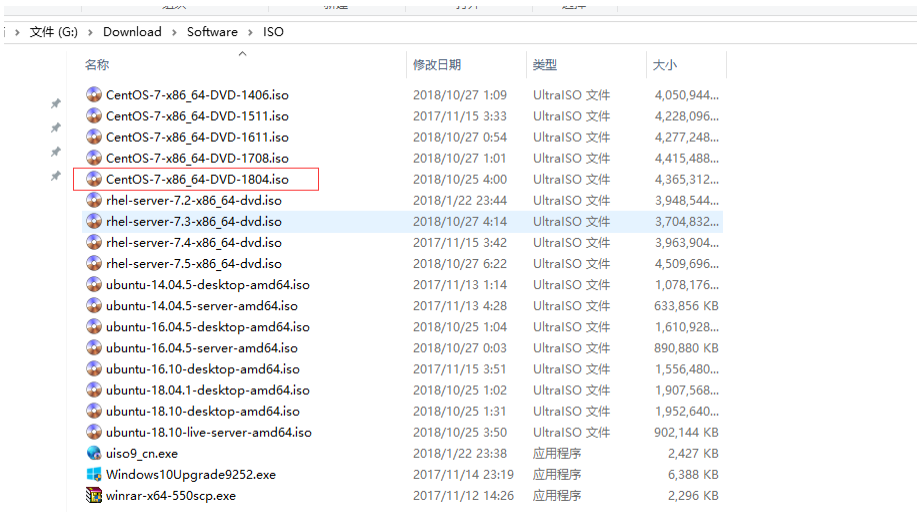
K8S部署

本教程旨在Windows 10企业版上，通过Hyper-V构建3台虚拟机，然后搭建Kubernetes-1.14.0版本集群，节点系统为CentOS-7.5

宿主机信息：

需要提前准备CentOS-7.5的ISO镜像



0 准备前规划

角色	IP	操作系统	KUBERNETES-VERSION	虚拟机规格
k8s-master	192.168.1.106	CentOS-7.5	1.14.0	2核1G

角色	IP	操作系统	KUBERNETES-VERSION	虚拟机规格
k8s-node1	192.168.1.107	CentOS-7.5	1.14.0	1核1G
k8s-node2	192.168.1.108	CentOS-7.5	1.14.0	1核1G

- k8s-master配置成2核是因为后面安装k8s时要求master节点的核数大于1
- **后续安装的时候，我们先安装一台机器，一般是k8s-master，然后再复制出另外两台Node节点，这样可以节省时间**。
- Hyper-V创建虚拟机的步骤就暂时省略了，后面直接开始操作。

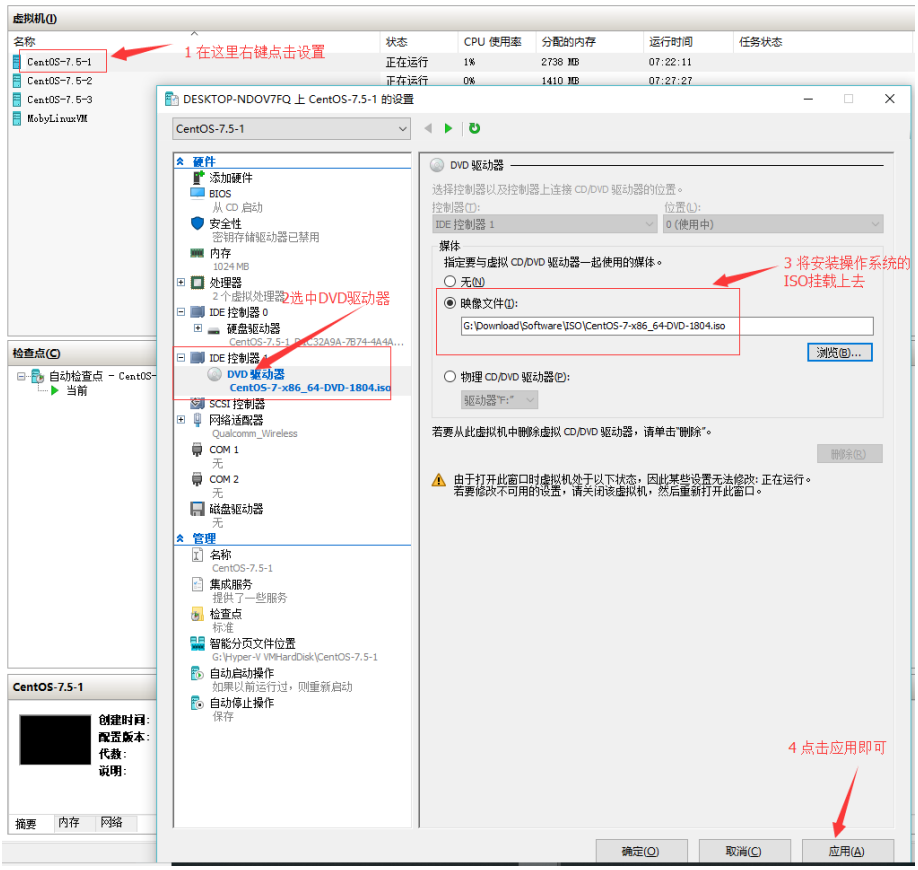
1 安装操作系统

准备CentOS-7.5 ISO，虚拟机的默认硬盘大小是127G

硬盘大小	600G分区安排	127G
/boot	1G	1G
/swap	4G	2G
/	200G	50G
/var/lib/docker	200G	30G
/var/lib/registry	200G	44G

2 设置光盘为本地yum源

2.1挂载本地操CentOS-7.5 ISO到系统




```
mkdir -p /mnt/cdrom
mount /dev/sdr0 /mnt/cdrom
```

2.2 配置yum源

```
[local]
name=local
baseurl=file:///mnt/cdrom
enabled=1
gpgcheck=0
gpgkey=file:///mnt/cdrom/PRM-GPG-KEY-CentOS-7
```

2.3更新根本地源，安装部分必要软件vim、net-tools与curl

```
yum makecache
yum -y install vim net-tools curl wget
```

2.4配置系统IP

2.4.1 ifconfig查看系统网口信息

```
[root@localhost yum.repos.d]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 00:15:5d:41:01:5b  txqueuelen 1000  (Ethernet)
        RX packets 2408  bytes 298833 (291.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@localhost yum.repos.d]# _
```

2.4.2查看网口的Link状态

```
[root@localhost yum.repos.d]# ethtool eth0
Settings for eth0:
        Supported ports: [ ]
        Supported link modes:   Not reported
        Supported pause frame use: No
        Supports auto-negotiation: No
        Supported FEC modes: Not reported
        Advertised link modes:  Not reported
        Advertised pause frame use: No
        Advertised auto-negotiation: No
        Advertised FEC modes: Not reported
        Speed: 150Mb/s
        Duplex: Full
        Port: Other
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: off
        Link detected: yes
[root@localhost yum.repos.d]#
```

2.4.3配置网口IP并验证

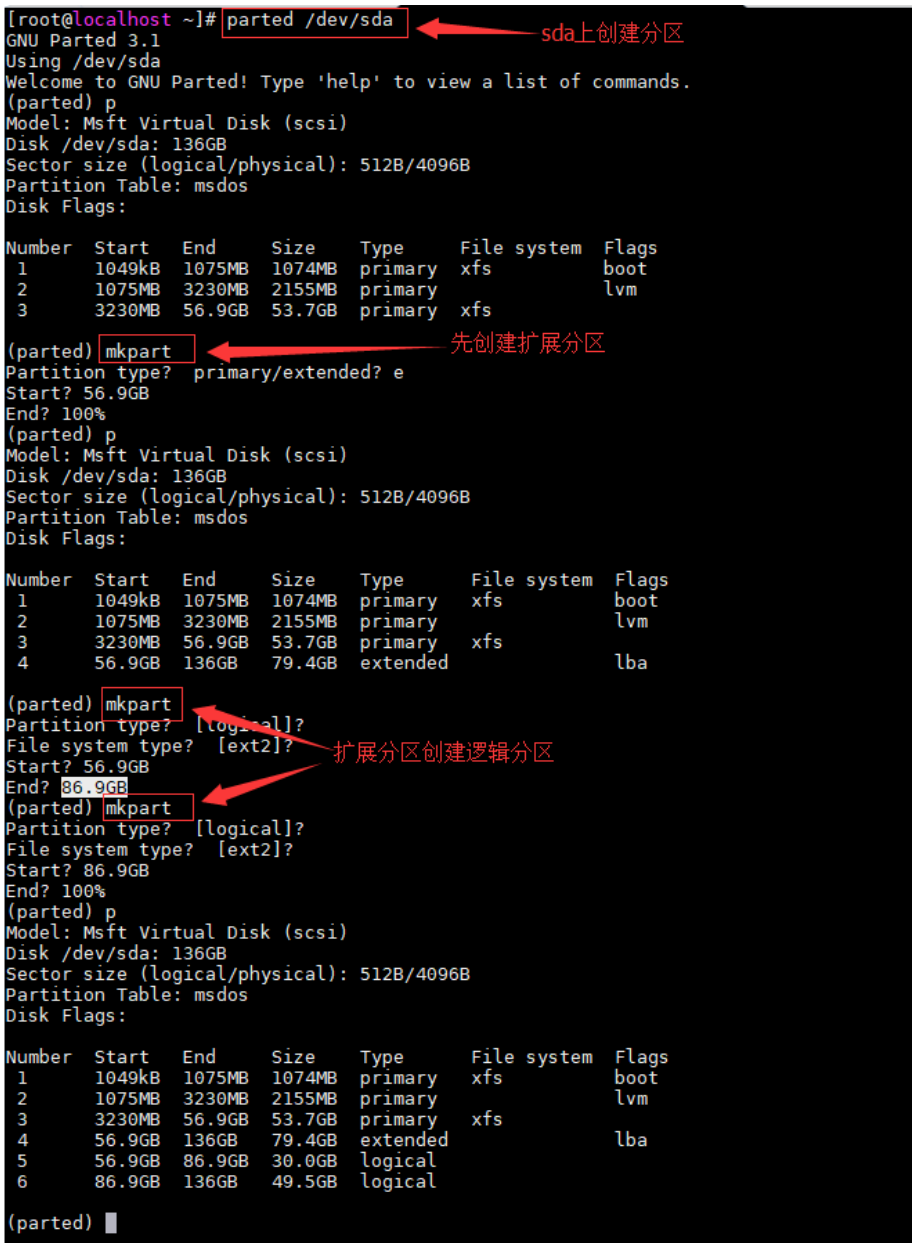
```
[root@localhost yum.repos.d]# ifconfig eth0 192.168.1.106
[root@localhost yum.repos.d]# ping 192.168.1.101
PING 192.168.1.101 (192.168.1.101) 56(84) bytes of data.
64 bytes from 192.168.1.101: icmp_seq=1 ttl=64 time=0.673 ms
^C
--- 192.168.1.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.673/0.673/0.673/0.000 ms
[root@localhost yum.repos.d]#
```

2.5 ssh远程连接后配置docker的相关目录

2.5.1先创建与docker相关的目录

```
mkdir -p /var/lib/docker     #容器数据卷volume所在目录
mkdir -p /var/lib/registey   #私有镜像仓库存储镜像目录
```

2.5.2创建分区



2.5.6格式化逻辑分区

```
[root@localhost ~]# mkfs.xfs /dev/sda5 -f
meta-data=/dev/sda5              isize=512    agcount=4, agsize=1829952 blks
         =                       sectsz=4096  attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=7319808, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=3574, version=2
         =                       sectsz=4096  sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[root@localhost ~]# mkfs.xfs /dev/sda6 -f
meta-data=/dev/sda6              isize=512    agcount=4, agsize=3019072 blks
         =                       sectsz=4096  attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=12076288, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=5896, version=2
         =                       sectsz=4096  sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[root@localhost ~]# partprobe
[root@localhost ~]#
```

2.5.7 生成uuid，加入到/etc/fstab中

```
[root@localhost ~]# blkid /dev/sda5
/dev/sda5: UUID="33feff19-8e8c-4b60-ae00-7b0fac13ff82" TYPE="xfs"
[root@localhost ~]# blkid /dev/sda6
/dev/sda6: UUID="94363be7-2e17-4388-8c08-f73dd04da0f3" TYPE="xfs"
```

```
[root@localhost yum.repos.d]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Sun Mar 31 09:19:20 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=bf083b89-26f6-49e7-b743-41c884c58f73 /                       xfs     defaults        0 0
UUID=08b800aa-24a2-4aba-86c6-43fe58d0b76d /boot                   xfs     defaults        0 0
/dev/mapper/centos-swap swap                    swap    defaults        0 0
UUID=33feff19-8e8c-4b60-ae00-7b0fac13ff82 /var/lib/docker         xfs     defaults        0 0
UUID=94363be7-2e17-4388-8c08-f73dd04da0f3 /var/lib/registry       xfs     defaults        0 0
[root@localhost yum.repos.d]# mount -a
```

执行mount命令

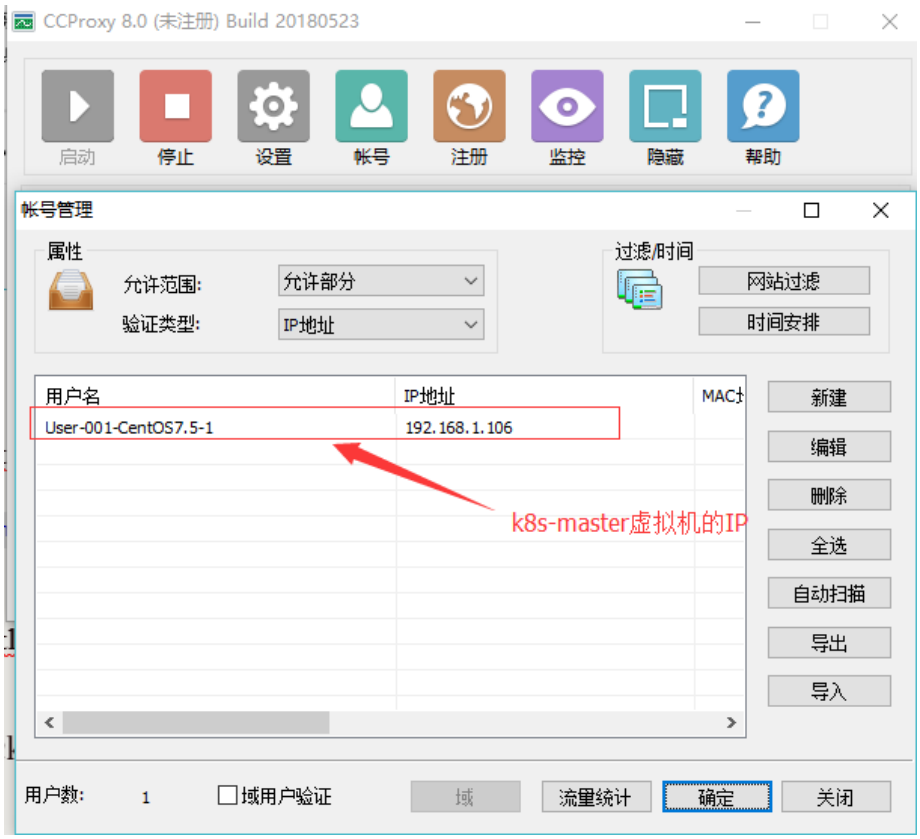
2.6 配置cntlm代理

2.6.1 准备CCProxy

由于是使用Hyper-V上的虚拟机上安装，为了使虚拟机内能够方便的访问外网，需要使用CCProxy进行代理，相关的安装包可以在网上下载到，本教程中使用的是

ccproxy2010_118231.rar版本

使用的是如下配置，宿主机IP是192.168.1.101

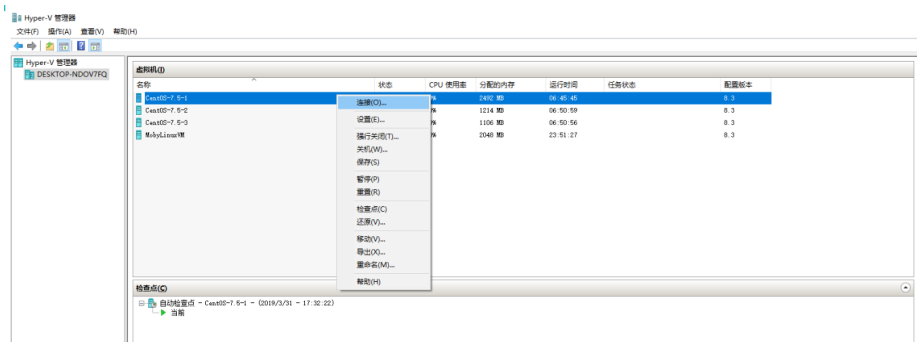


2.6.2 下载cntlm的安装包cntlm-0.92.3-1.x86_64.rpm，然后rpm安装

```
rpm -vih cntlm-0.92.3-1.x86_64.rpm
```

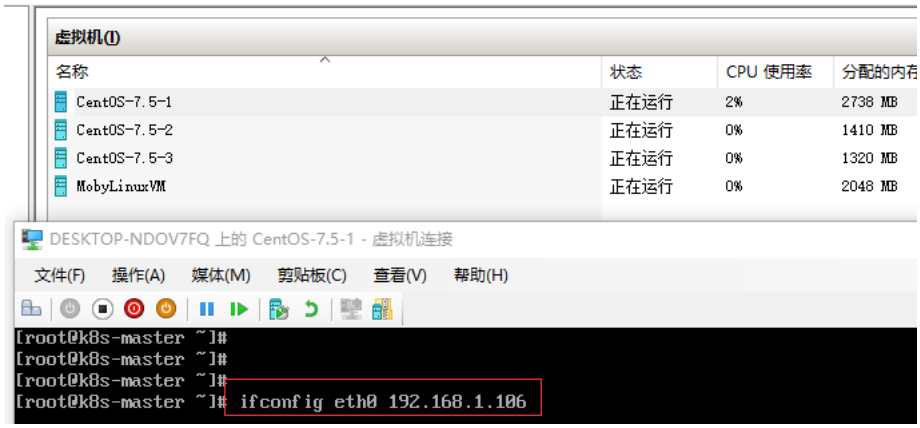
2.6.3 修改cntlm的配置文件

通过Hyper-V连接到k8s-master的虚拟机

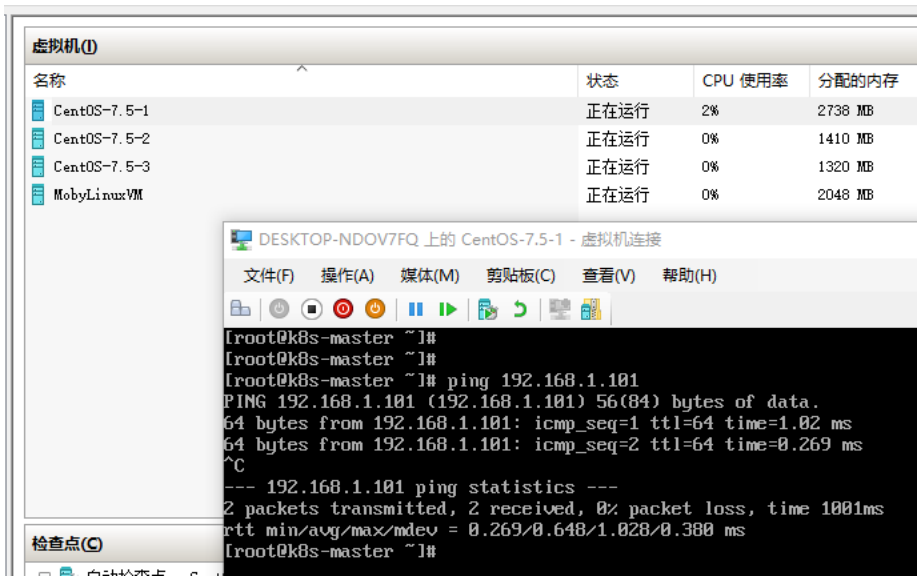


临时配置本机IP地址

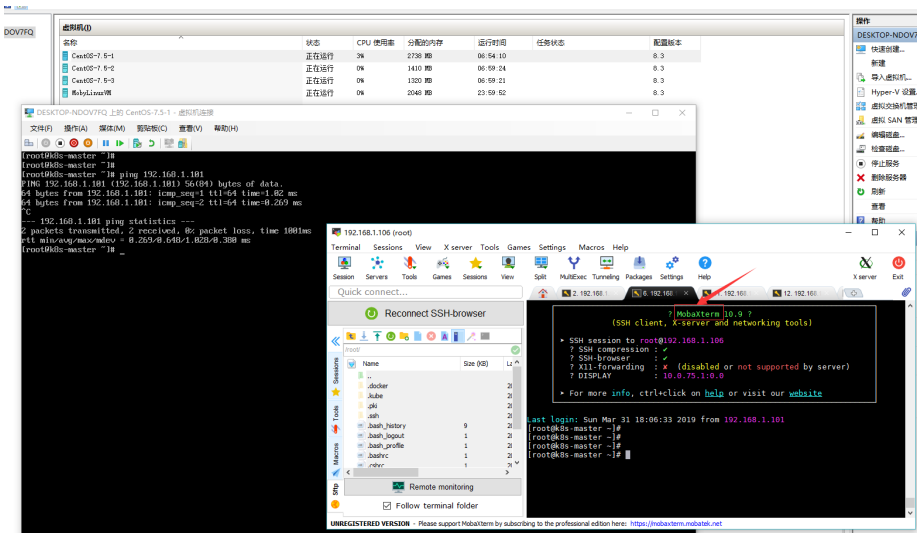
```
ifconfig eth0 192.168.1.106
```



然后就可以试着ping一下宿主机的IP，我这里是192.168.1.101



然后就可以通过ssh软件，比如MobaxTerm或者Xshell远程连接到机器方便操作

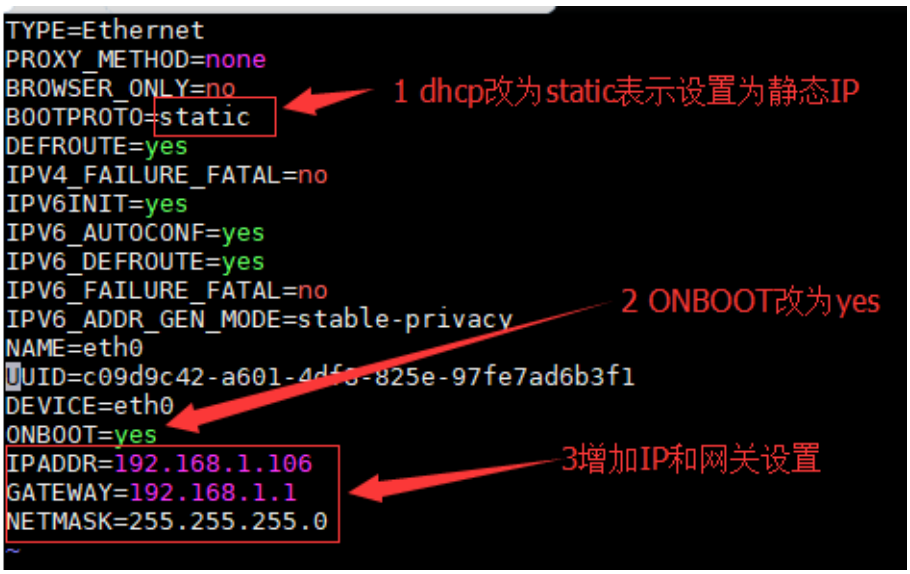


后面的步骤就完全切换到MobaXterm中来操作。

首先，让我们来将机器的IP设置为固定IP，k8s-master=192.168.1.101

```
vim /etc/sysconfig/network-scripts/ifcfg-eth0
```

按照如下修改即可：



ifcfg-eth0模板文件如下：

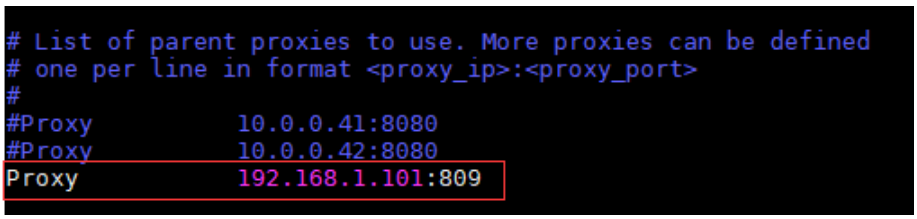
```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
```

```
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eth0
UUID=c09d9c42-a601-4df8-825e-97fe7ad6b3f1
DEVICE=eth0
ONBOOT=yes
IPADDR=192.168.1.106
GATEWAY=192.168.1.1
NETMASK=255.255.255.0
```

其次，让我们修改cntlm的代理设置

```
vim /etc/cntlm.conf
```

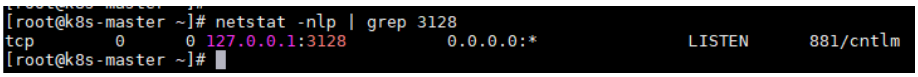
将其中的代理配置先注释掉，然后增加宿主机CCProxy设置的IP和端口



然后既可以启动cntlm了

```
cntlm -c /etc/cntlm.conf
```

然后使用如下命令就可以看到cntlm启动了，且在监听3128端口



这样做只会让cntlm本次生效，系统重启之后需要重新执行cntlm -c 命令，为了达到cntlm在开机时自动配置好，可以如下设置

```
vim /etc/rc.local
#在其尾部加入：
cntlm -c /etc/cntlm.conf

#然后给/etc/rc.d/rc.local增加可执行权限
chmod +x /etc/rc.d/rc.local
```

这样cntlm在每次系统启动之后就生效了，稍微解释一下：

系统启动的时候会执行/etc/rc.local中的命令，而/etc/rc.local是/etc/rc.d/rc.local的软链接，如果想/etc/rc.local中的命令生效，就需要给/etc/rc.d/rc.local设置可执行权限。

2.6.4 设置系统环境变量，增加http_proxy等代理设置

```
vim /etc/profile
#在文件尾部添加:
export http_proxy=http://localhost:3128
export https_proxy=${http_proxy}
export ftp_proxy=${http_proxy}
export no_proxy="localhost,127.0.0.1,192.168.*"
```

2.6.5 执行source /etc/profile使代理生效

在/etc/profile中设置代理，是为了使这些和代理有关的环境变量在系统启动的时候就设置好，因为系统启动的时候回执行/etc/profile里面的指令

2.6.6 使用curl测试外网联通

直接curl一下www.baidu.com，看到如下输出表示成功访问外网

3 安装docker

3.1 准备好访问外网的yum文件

3.1.1 aliyun-centos7.repo

```
# CentOS-Base.repo
#
```

```
# The mirror system uses the connecting IP address of the
client and the
# update status of each mirror to pick mirrors that are
updated to and
# geographically close to the client.  You should use this
for CentOS updates
# unless you are manually picking other mirrors.
#
# If the mirrorlist= does not work for you, as a fall back
you can try the
# remarked out baseurl= line instead.
#
#

[base]
name=CentOS-$releasever - Base - mirrors.aliyun.com
failovermethod=priority
baseurl=http://mirrors.aliyun.com/centos/$releasever/os/$b
asearch/

http://mirrors.aliyuncs.com/centos/$releasever/os/$basearc
h/

http://mirrors.cloud.aliyuncs.com/centos/$releasever/os/$b
asearch/
gpgcheck=1
gpgkey=http://mirrors.aliyun.com/centos/RPM-GPG-KEY-
CentOS-7

#released updates
[updates]
name=CentOS-$releasever - Updates - mirrors.aliyun.com
failovermethod=priority
baseurl=http://mirrors.aliyun.com/centos/$releasever/updat
es/$basearch/

http://mirrors.aliyuncs.com/centos/$releasever/updates/$ba
search/

http://mirrors.cloud.aliyuncs.com/centos/$releasever/updat
es/$basearch/
gpgcheck=1
gpgkey=http://mirrors.aliyun.com/centos/RPM-GPG-KEY-
CentOS-7

#additional packages that may be useful
[extras]
name=CentOS-$releasever - Extras - mirrors.aliyun.com
```

```
failovermethod=priority
baseurl=http://mirrors.aliyun.com/centos/$releasever/extra
s/$basearch/

http://mirrors.aliyuncs.com/centos/$releasever/extras/$bas
earch/

http://mirrors.cloud.aliyuncs.com/centos/$releasever/extra
s/$basearch/
gpgcheck=1
gpgkey=http://mirrors.aliyun.com/centos/RPM-GPG-KEY-
CentOS-7

#additional packages that extend functionality of existing
packages
[centosplus]
name=CentOS-$releasever - Plus - mirrors.aliyun.com
failovermethod=priority
baseurl=http://mirrors.aliyun.com/centos/$releasever/cento
splus/$basearch/

http://mirrors.aliyuncs.com/centos/$releasever/centosplus/
$basearch/

http://mirrors.cloud.aliyuncs.com/centos/$releasever/cento
splus/$basearch/
gpgcheck=1
enabled=0
gpgkey=http://mirrors.aliyun.com/centos/RPM-GPG-KEY-
CentOS-7

#contrib - packages by Centos Users
[contrib]
name=CentOS-$releasever - Contrib - mirrors.aliyun.com
failovermethod=priority
baseurl=http://mirrors.aliyun.com/centos/$releasever/contr
ib/$basearch/

http://mirrors.aliyuncs.com/centos/$releasever/contrib/$ba
search/

http://mirrors.cloud.aliyuncs.com/centos/$releasever/contr
ib/$basearch/
gpgcheck=1
enabled=0
gpgkey=http://mirrors.aliyun.com/centos/RPM-GPG-KEY-
CentOS-7
```

3.1.2 docker-ce.repo

```
[docker-ce-stable]
name=Docker CE Stable - $basearch
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/$basearch/stable
enabled=1
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-stable-debuginfo]
name=Docker CE Stable - Debuginfo $basearch
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/debug-$basearch/stable
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-stable-source]
name=Docker CE Stable - Sources
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/source/stable
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-edge]
name=Docker CE Edge - $basearch
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/$basearch/edge
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-edge-debuginfo]
name=Docker CE Edge - Debuginfo $basearch
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/debug-$basearch/edge
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-edge-source]
```

```
name=Docker CE Edge - Sources
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/source/edge
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-test]
name=Docker CE Test - $basearch
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/$basearch/test
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-test-debuginfo]
name=Docker CE Test - Debuginfo $basearch
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/debug-$basearch/test
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-test-source]
name=Docker CE Test - Sources
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/source/test
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-nightly]
name=Docker CE Nightly - $basearch
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/$basearch/nightly
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-nightly-debuginfo]
name=Docker CE Nightly - Debuginfo $basearch
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/debug-$basearch/nightly
```

```
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg

[docker-ce-nightly-source]
name=Docker CE Nightly - Sources
baseurl=https://mirrors.aliyun.com/docker-
ce/linux/centos/7/source/nightly
enabled=0
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/docker-
ce/linux/centos/gpg
```

3.1.3 epel.repo

```
[epel]
name=Extra Packages for Enterprise Linux 7 - $basearch
baseurl=http://mirrors.aliyun.com/epel/7/$basearch
failovermethod=priority
enabled=1
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

[epel-debuginfo]
name=Extra Packages for Enterprise Linux 7 - $basearch -
Debug
baseurl=http://mirrors.aliyun.com/epel/7/$basearch/debug
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
gpgcheck=0

[epel-source]
name=Extra Packages for Enterprise Linux 7 - $basearch -
Source
baseurl=http://mirrors.aliyun.com/epel/7/SRPMS
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
gpgcheck=0
```

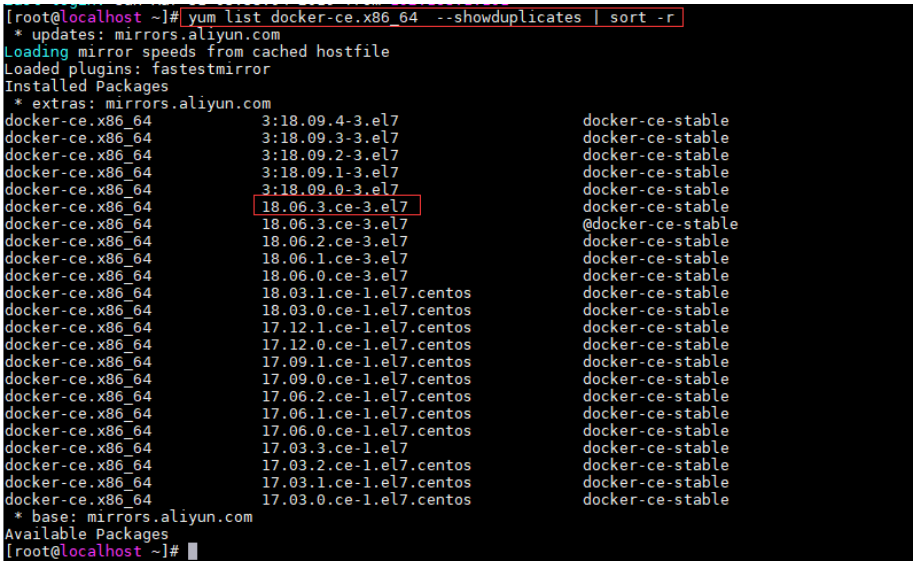
3.2 移走local.repo后再更新yum缓存

```
yum makecache
```

3.3 安装docker

3.3.1 先查看yum里面的docker 版本信息

```
yum list docker-ce.x86_64  --showduplicates | sort -r
```



3.3.2 安装最新的docker

```
yum install docker-ce-18.06.3.ce-3.el7.x86_64 docker-ce-selinux-18.06.3.ce-3.el7.x86_64 -y
```

3.4 配置docker CDN加速

```
mkdir /etc/docker
vim daemon.json
#写入:
{
    "registry-mirrors":
["https://69959mok.mirror.aliyuncs.com"]
}
```

3.5 配置docker代理

```
vim /etc/systemd/system/docker.service.d/http-proxy.conf
#写入:
[Service]
Environment="HTTP_PROXY=http://localhost:3128"
Environment="HTTPS_PROXY=http://localhost:3128"
Environment="NO_PROXY=localhost,127.0.0.0,192.168.*"
```

3.6 启动docker

```
systemctl start docker
systemctl enable docker
```

3.7 检查docker 是否启动

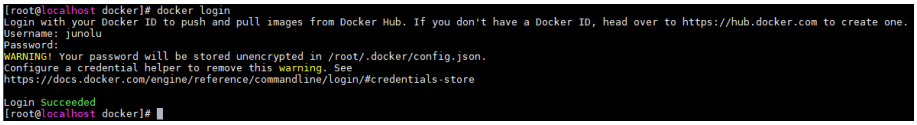
```
docker info
```

```
[root@localhost docker]# docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 18.06.3-ce
Storage Driver: overlay2
 Backing Filesystem: xfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 468a545b9edcd5932818eb9de8e72413e616e86e
runc version: a592beb5bc4c4092b1b1bac971afed27687340c5
init version: fec3683
Security Options:
 seccomp
  Profile: default
Kernel Version: 3.10.0-862.el7.x86_64
Operating System: CentOS Linux 7 (Core)
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 910.2MiB
Name: localhost.localdomain
ID: MPHL:RNCX:44WG:AAHP:XWPH:6WAZ:GKU6:2ETA:JVJ5:VUWH:KFU6:PLLF
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
HTTP Proxy: http://localhost:3128
HTTPS Proxy: http://localhost:3128
No Proxy: localhost,127.0.0.0,192.168.*
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
 192.168.1.101:5000
 127.0.0.0/8
Registry Mirrors:
 https://69959mok.mirror.aliyuncs.com/
Live Restore Enabled: false

[root@localhost docker]#
```

3.8 检查docker环境变量

```
[root@localhost docker]# systemctl show docker --property Environment
Environment=HTTP_PROXY=http://localhost:3128 HTTPS_PROXY=http://localhost:3128 NO_PROXY=localhost,127.0.0.0,192.168.*
[root@localhost docker]#
```

3.9 登录docker hub账号



4 安装Kubernetes

4.1 准备工作

4.1.1 准备kubernetets yum源

kubernetes.repo

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

4.1.2 更新kubernetes的yum源

```
yum makecache
```

4.1.3 查看kubernetes版本，可以看到1.14.0版本

```
yum list kubelet.x86_64  --showduplicates | sort -r
```

```
[root@localhost yum.repos.d]cayum list kubelet.x86_64  --showduplicates | sort -r
 * updates: mirrors.aliyun.com
Loading mirror speeds from cached hostfile
Loaded plugins: fastestmirror
kubelet.x86_64                          1.9.9-0                         kubernetes
kubelet.x86_64                          1.9.8-0                         kubernetes
kubelet.x86_64                          1.9.7-0                         kubernetes
kubelet.x86_64                          1.9.6-0                         kubernetes
kubelet.x86_64                          1.9.5-0                         kubernetes
kubelet.x86_64                          1.9.4-0                         kubernetes
kubelet.x86_64                          1.9.3-0                         kubernetes
kubelet.x86_64                          1.9.2-0                         kubernetes
kubelet.x86_64                          1.9.11-0                        kubernetes
kubelet.x86_64                          1.9.1-0                         kubernetes
kubelet.x86_64                          1.9.10-0                        kubernetes
kubelet.x86_64                          1.9.0-0                         kubernetes
kubelet.x86_64                          1.8.9-0                         kubernetes
kubelet.x86_64                          1.8.8-0                         kubernetes
kubelet.x86_64                          1.8.7-0                         kubernetes
kubelet.x86_64                          1.8.6-0                         kubernetes
kubelet.x86_64                          1.8.5-1                         kubernetes
kubelet.x86_64                          1.8.5-0                         kubernetes
kubelet.x86_64                          1.8.4-1                         kubernetes
kubelet.x86_64                          1.8.4-0                         kubernetes
kubelet.x86_64                          1.8.3-1                         kubernetes
kubelet.x86_64                          1.8.3-0                         kubernetes
kubelet.x86_64                          1.8.2-1                         kubernetes
kubelet.x86_64                          1.8.2-0                         kubernetes
kubelet.x86_64                          1.8.15-0                        kubernetes
kubelet.x86_64                          1.8.14-0                        kubernetes
kubelet.x86_64                          1.8.13-0                        kubernetes
kubelet.x86_64                          1.8.12-0                        kubernetes
kubelet.x86_64                          1.8.1-1                         kubernetes
kubelet.x86_64                          1.8.11-0                        kubernetes
kubelet.x86_64                          1.8.1-0                         kubernetes
kubelet.x86_64                          1.8.10-0                        kubernetes
kubelet.x86_64                          1.8.0-1                         kubernetes
kubelet.x86_64                          1.8.0-0                         kubernetes
kubelet.x86_64                          1.7.9-1                         kubernetes
kubelet.x86_64                          1.7.9-0                         kubernetes
kubelet.x86_64                          1.7.8-2                         kubernetes
kubelet.x86_64                          1.7.8-1                         kubernetes
kubelet.x86_64                          1.7.7-2                         kubernetes
kubelet.x86_64                          1.7.7-1                         kubernetes
kubelet.x86_64                          1.7.6-2                         kubernetes
kubelet.x86_64                          1.7.6-1                         kubernetes
kubelet.x86_64                          1.7.5-1                         kubernetes
kubelet.x86_64                          1.7.5-0                         kubernetes
kubelet.x86_64                          1.7.4-1                         kubernetes
kubelet.x86_64                          1.7.4-0                         kubernetes
```

4.1.4 下载github上的kubernetes二进制安装包

相关链接(以kubernetes v1.14.0为例):

```
https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG-1.14.md#kubernetes-v114-release-notes
```

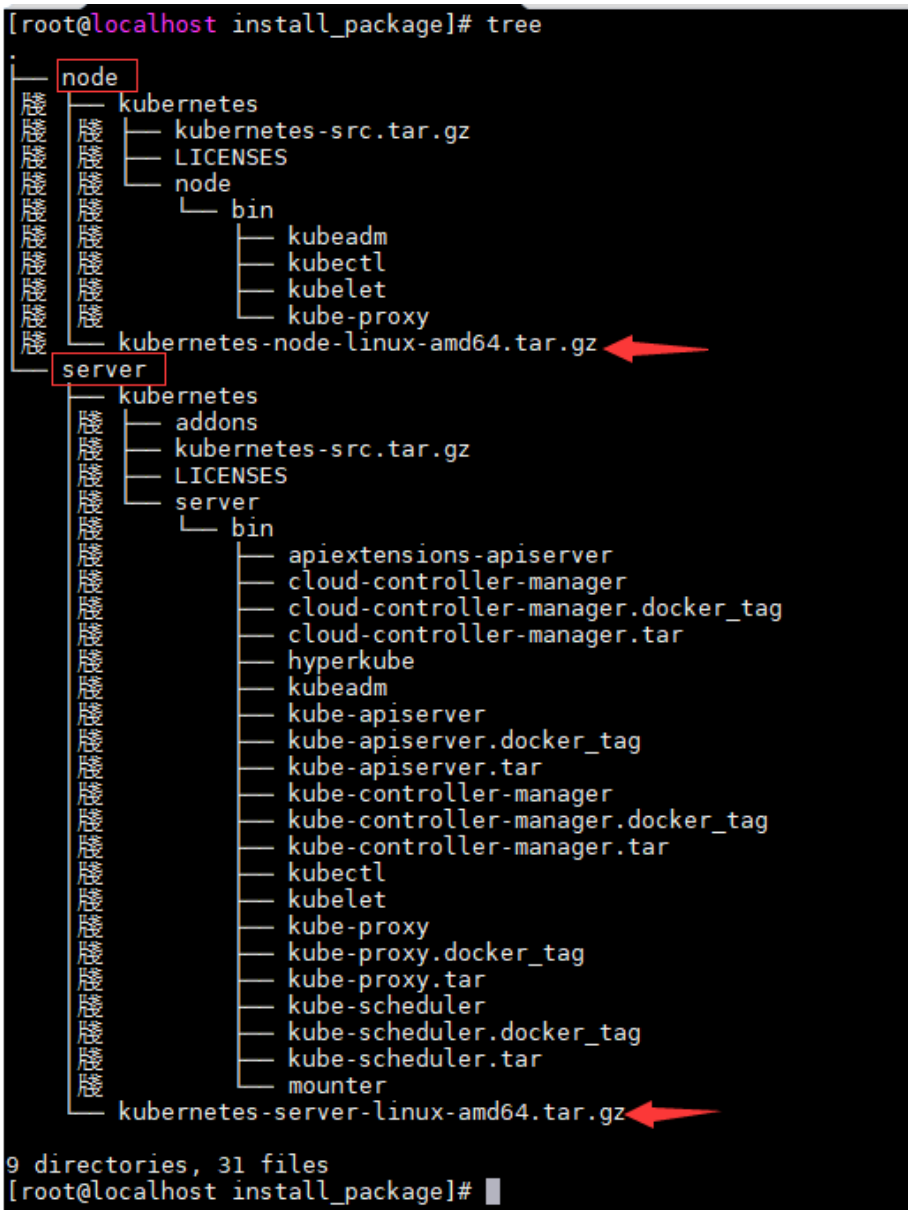
Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	25739802a641517a8bbb933b69000a943e8dd38e616b8778149dd0138737abacf377683da2ff35fdd0bbb305b88b
kubernetes-server-linux-arm.tar.gz	c1dbba77a4ff5661eb36c55182a753b88ccc9b89ca31e162b06672126743cfea115b2f8ea8658b12344c36df1795
kubernetes-server-linux-arm64.tar.gz	ad346bbe2a053c1106b51e5125698737dc7b76fa3bf439e14d4b4ba1c262678fede9c507c1098aac6e14d2c742c5
kubernetes-server-linux-ppc64le.tar.gz	49f9bd1c751620ecf4b5c152f287d72b36abca21fd1dfe99443d984473c6efa051a910de585c42f5447ef7c18d7d
kubernetes-server-linux-s390x.tar.gz	d6be847f2a0358755a69dea26181e5fc1a80ac4939b8b04a3875e1f6693553cad562452bfad21b2e380ddda1839a

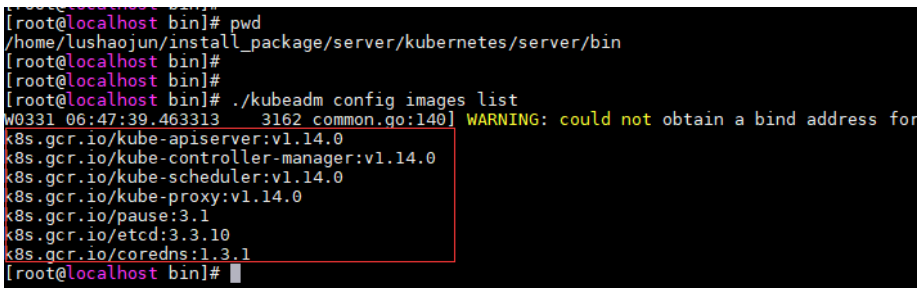
Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	75dc99919d1084d7d471a53ab60c743dc399145c99e83f37c6ba3c241b2c0b2ecc2c0d1b94690ff912e2a15b7c55
kubernetes-node-linux-arm.tar.gz	49013a4f01be8086fff332099d94903082688b9b295d2f34468462656da4709360025e9d84b069410c608977ef80
kubernetes-node-linux-arm64.tar.gz	f8c0cb0c089cd1d1977c049002620b8cf748d193c1b76dd1d3aac01ff9273549c06a1e3dfe983dc40a95ee8b0719

拷贝到k8s-master机器上并使用tar -xf解压



4.1.5 检查安装Kubernetes-1.14.0需要的docker镜像



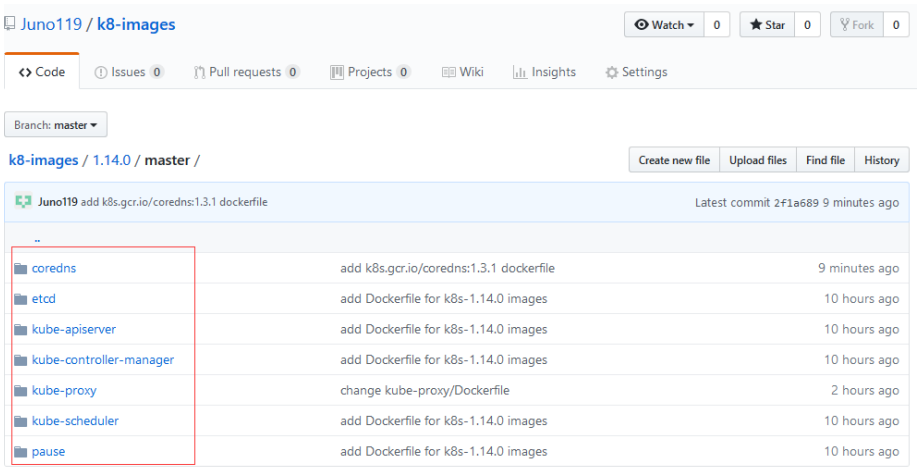
可以看到Kubernetes-1.14.0需要这些镜像

```
k8s.gcr.io/kube-apiserver:v1.14.0
k8s.gcr.io/kube-controller-manager:v1.14.0
k8s.gcr.io/kube-scheduler:v1.14.0
k8s.gcr.io/kube-proxy:v1.14.0
k8s.gcr.io/pause:3.1
k8s.gcr.io/etcd:3.3.10
k8s.gcr.io/coredns:1.3.1

#其实后面还会用到2个镜像，后面可以一起准备
quay.io/coreos/flannel:v0.11.0-amd64
k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1
```

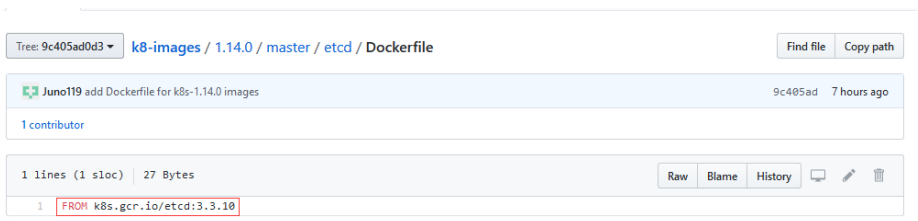
4.1.6 准备镜像

由于国内没法访问k8s.gcr.io仓库，除非使用VPN，所以通过github和docker hub来制作镜像。

a 创建github镜像仓库

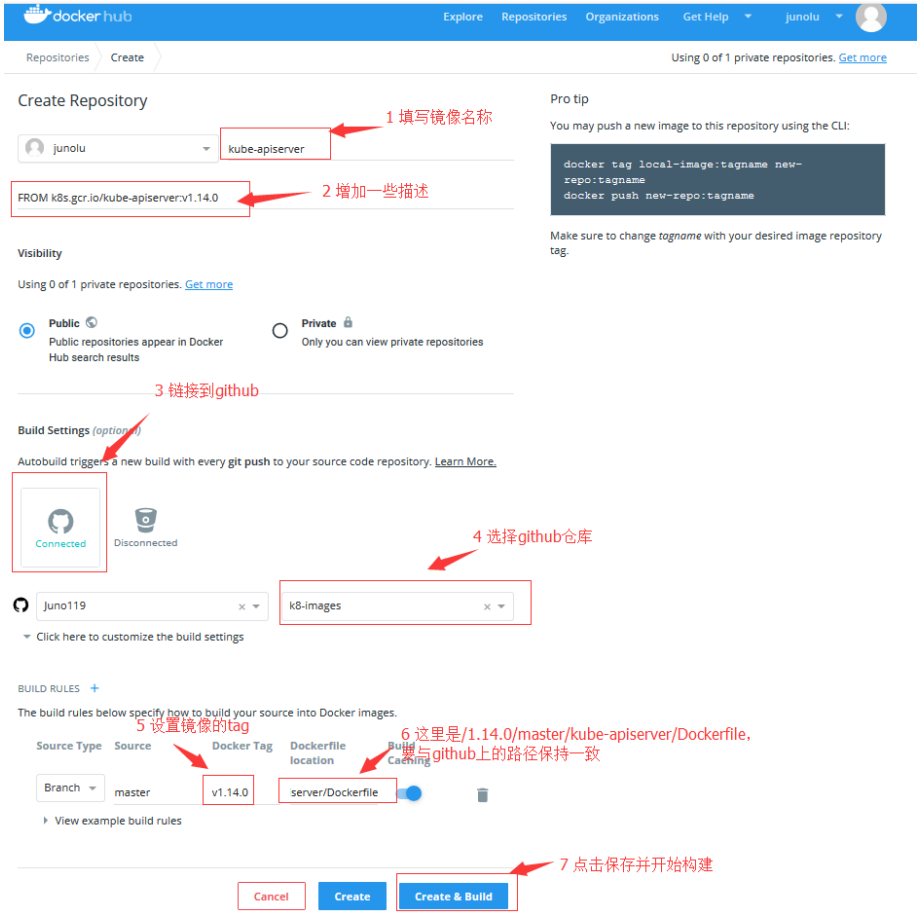


每个目录下放置分别放置构建这个镜像的Dockerfile，以etcd为例：

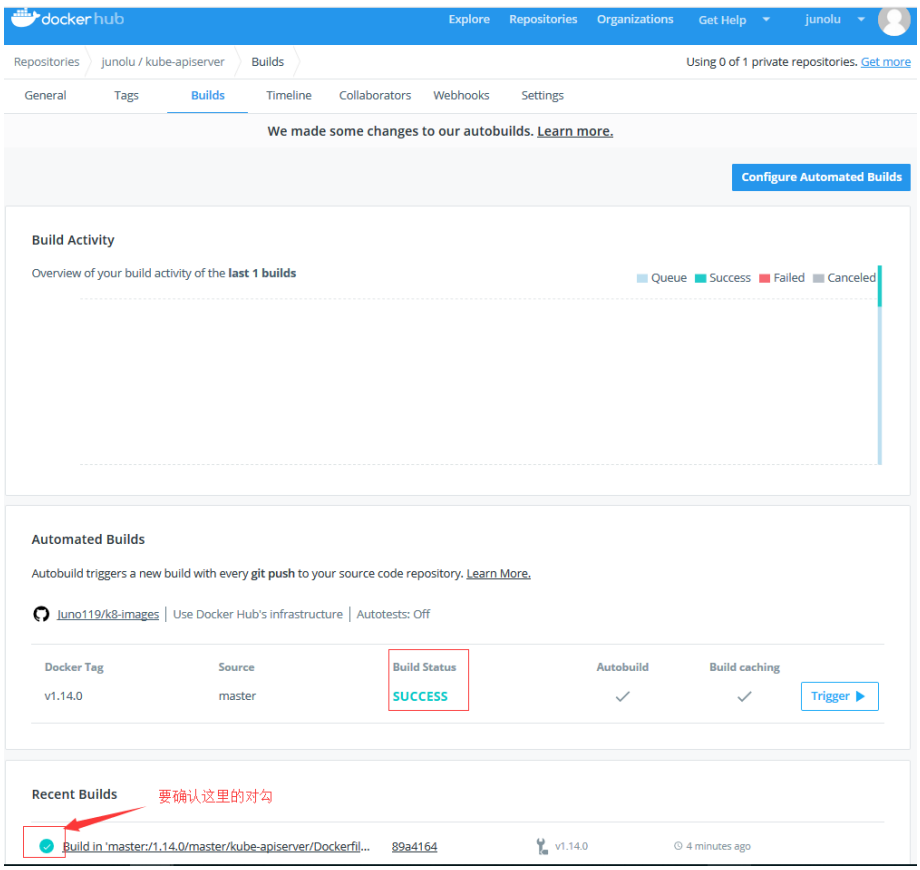


其实就是为了利用docker hub的在线构建机制。后续可以通过将docker hub链接到github，让docker hub在后台帮我们把镜像pull下来，我们再从dokcer hub上pull。

b 创建docker hub镜像仓库



然后耐心等待一会就可以看到kube-apiserver构建成功，此期间什么都不要点击



然后采用同样的方法就可以完成其他镜像的构建了

c docker pull所有构建的镜像，然后重新打tag

因为我们pull下来的image的tag都带有自己docker hub仓库的名字，而安装Kubernetes只认k8s.gcr.io仓库下的，所以我们pull下来之后需要重新打标签

以etcd为例：

```
#pull镜像
docker pull junolu/etcd:3.3.10

#重新打tag
docker tag junolu/etcd:3.3.10 k8s.gcr.io/etcd:3.3.10

#删除旧的tag
docker rmi junolu/etcd:3.3.10
```

其它的镜像依次同样操作，最终的结果：

```
[root@k8s-master install_package]# docker images
REPOSITORY                                TAG                 IMAGE ID            CREATED             SIZE
k8s.gcr.io/kube-proxy                     v1.14.0             5cd54e388aba        5 days ago          82.1MB
k8s.gcr.io/kube-controller-manager        v1.14.0             b95b1efa0436        5 days ago          158MB
k8s.gcr.io/kube-scheduler                 v1.14.0             00638a24688b        5 days ago          81.6MB
k8s.gcr.io/kube-apiserver                 v1.14.0             ecf910f40d6e        5 days ago          210MB
quay.io/coreos/flannel                    v0.11.0-amd64       ff281650a721        2 months ago        52.6MB
k8s.gcr.io/coredns                        1.3.1               eb516548c180        2 months ago        40.3MB
k8s.gcr.io/kubernetes-dashboard-amd64     v1.10.1             f9aed6605b81        3 months ago        122MB
k8s.gcr.io/etcd                           3.3.10              2c4adeb21b4f        4 months ago        258MB
quay.io/coreos/flannel                    v0.10.0-amd64       f0fad859c909        14 months ago       44.6MB
k8s.gcr.io/pause                          3.1                 da86e6ba6ca1        15 months ago       742kB
[root@k8s-master install_package]#
```

4.2 安装kubernetes-1.14.0相关的包

k8s-master和k8s-node均要执行

```
yum install kubelet-1.14.0-0.x86_64 kubeadm-1.14.0-
0.x86_64 kubectl-1.14.0-0.x86_64 -y

#然后执行
systemctl enable kubelet
#注意：这一步不能直接执行 systemctl start kubelet，否侧会报错，
kubelet也起动不成功
```

设置时区和节点名称

```
timedatectl set-timezone Asia/Shanghai  #如果安装系统时选的时
区时shanghai就不用执行，否则各个节点都要执行
hostnamectl set-hostname k8s-master   #k8s-master执行，这里
我们安装master，就只执行这一句
hostnamectl set-hostname k8s-node1    #k8s-node1执行
hostnamectl set-hostname k8s-node2    #k8s-node2执行
```

配置/etc/hosts

```
vim /etc/hosts
在文件尾部追加：
192.168.1.106 k8s-master
192.168.1.107 k8s-node1
192.168.1.108 k8s-node2
```

关闭所有节点的seliux以及firewalld

```
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/selinux/config
setenforce 0
systemctl disable firewalld
systemctl stop firewalld
```

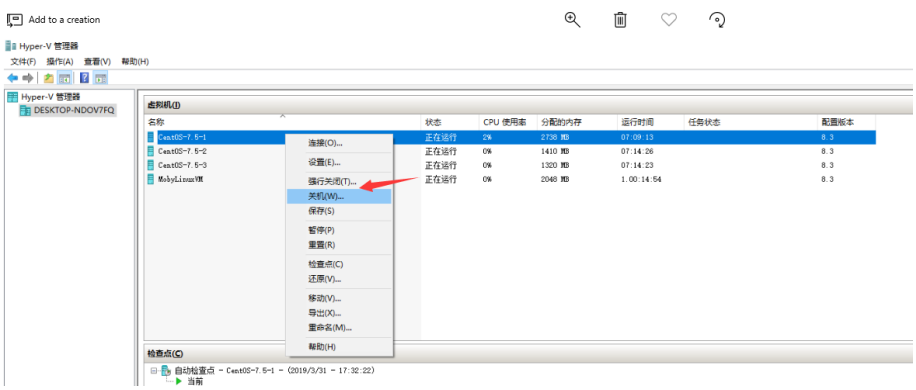
关闭swap，及修改iptables，不然后面kubeadm会报错

```
swapoff -a

vi /etc/fstab    #然后将swap一行注释

cat <<EOF >  /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sysctl --system
```

4.3 复制2个Node节点

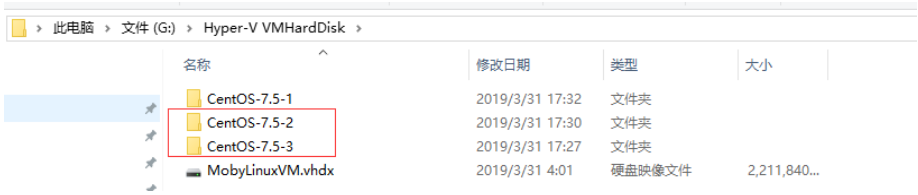
以上的步骤将master和node的公共操作就做完了，现在我们来复制两个node节点，这样就不需要从头开始安装操作系统了，而且docker镜像也完全准备好了。

4.3.1 关闭k8s-master虚拟机

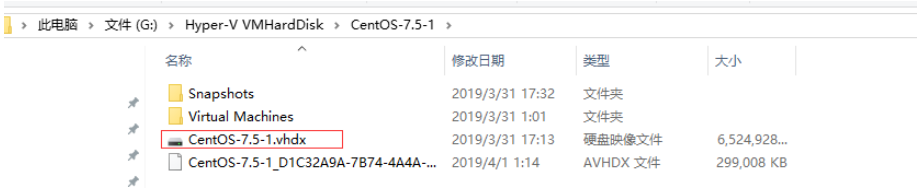


关闭虚拟机是为了让Hyper-V进行合并操作，将虚拟机的修改持久化到硬盘，免得中间有些步骤丢失了。

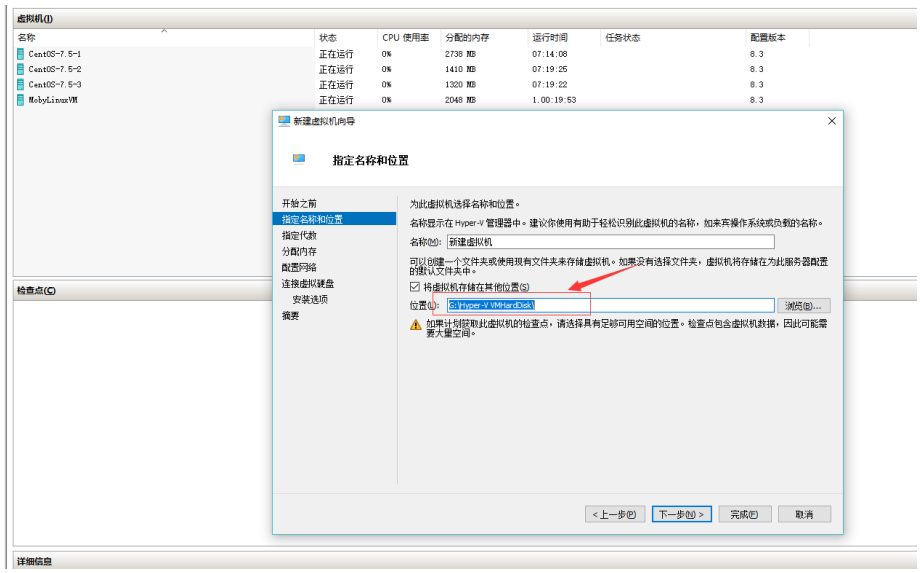
4.3.2 复制k8s-master的虚拟磁盘



这里CentOS-7.5-1是k8s-master的虚拟磁盘，我们看一下里面的内容

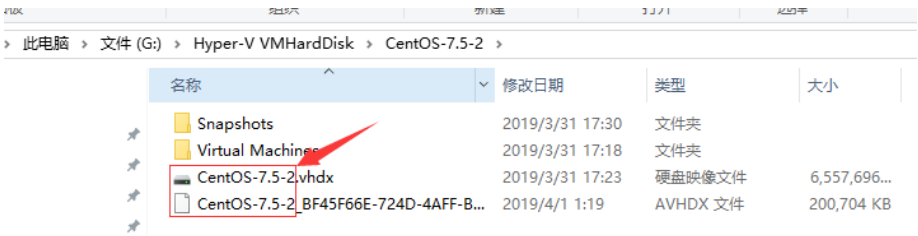


这里的路径就是创建虚拟机的时候设置保存的路径



然后复制出两个node节点，也就是上面截图中的CentOS-7.5-2(k8s-node1)和CentOS-7.5-3（k8s-node2）。

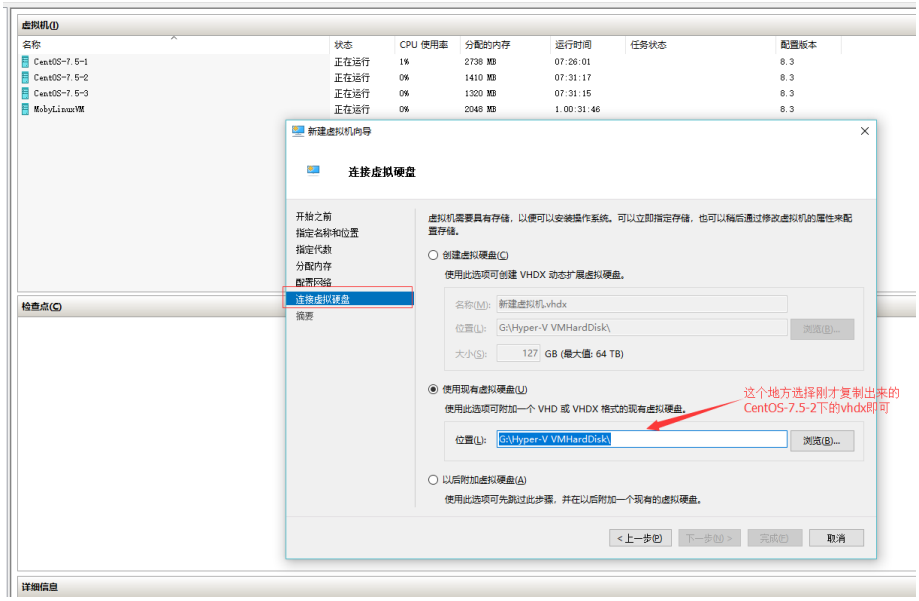
以CentOS-7.5-2为例，将目录下带有CentOS-7.5-1的都改为CentOS-7.5-2



CentOS-7.5-3做同样修改即可。

4.3.3 新建两个虚拟机，然后选定刚才复制出来的vhdx

以CentOS-7.5-2为例：



这样两个node节点的虚拟机就创建好了。

4.3.4 启动两个node节点虚拟机

node节点启动之后应该做的修改一些地方，下面以k8s-node1为例进行修改。

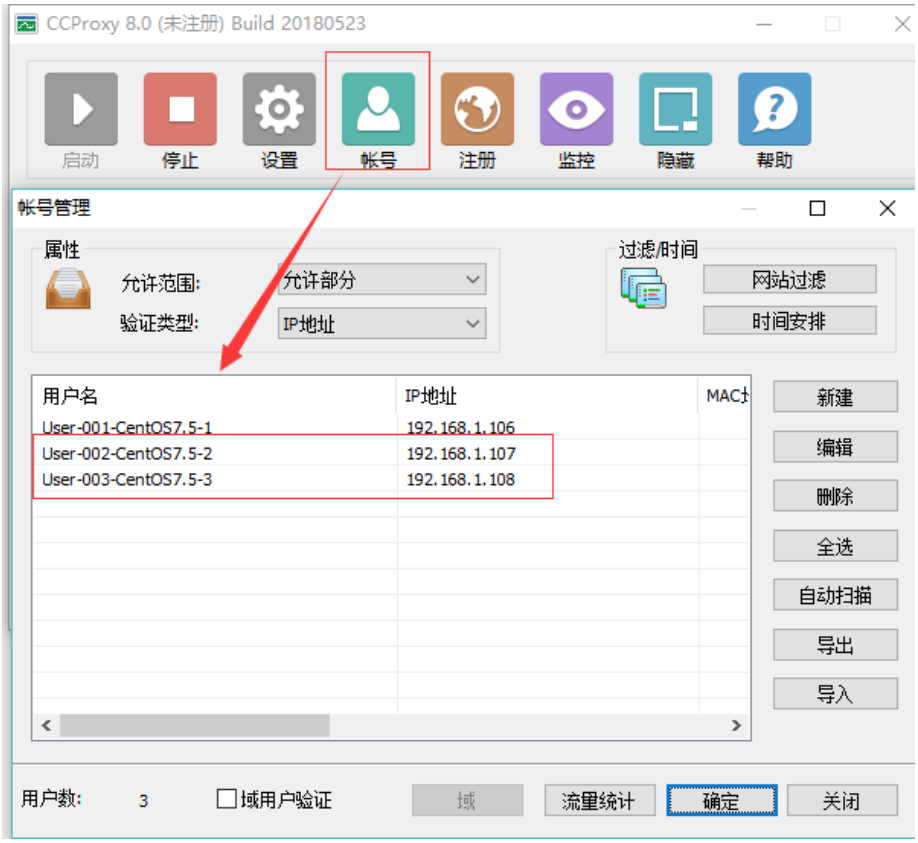
a 设置节点名称，需要将节点的名称设置为k8s-node1

```
hostnamectl set-hostname k8s-node1
```

b 修改本机的IP地址

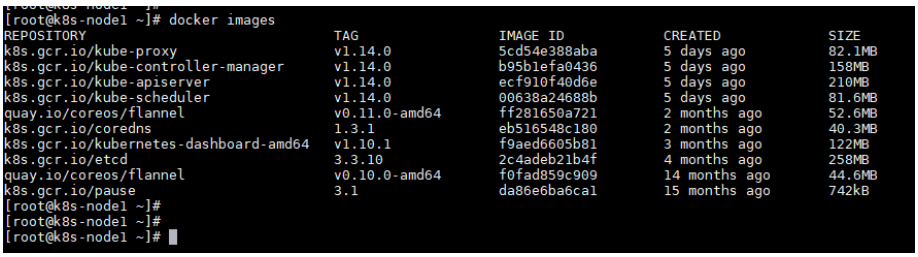
k8s-node1规划的IP是192.168.1.107

c 在CCProxy上增加账号



d 检查代理通不通

e 查看docker images是否和master相同



4.4 部署k8s-master节点

重新启动master节点，然后在master节点执行

```
#如果配置了master节点的核数为2则直接执行
kubeadm init --kubernetes-version=v1.14.0 --pod-network-
cidr=10.244.0.0/16

#如果忘了设置为2，而是只有1个核的话，应该执行下面这条，否则安装会报错
kubeadm init --kubernetes-version=v1.14.0 --pod-network-
cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU
```

等待一会执行完成之后就会打印下面这些信息

```
Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following
as a regular user:

#这三步是接下来要执行的
  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/

You can now join any number of machines by running the
following on each node
as root:

#这句话得记录下来，接下来Node节点加入就要执行这句话了
  kubeadm join 192.168.1.106:6443 --token
wct45y.tq23fogetd7rp3ck --discovery-token-ca-cert-hash
sha256:c267e2423dba21fdf6fc9c07e3b3fa17884c4f24f0c03f2283a
230c70b07772f
```

按要求执行后面的指令

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

这样k8s-master就安装好了，可以使用kubectl命令查看节点，节点状态应该显示NotReady，而NotReady是因为还未部署网络插件。类似这样：

```
[root@master1 kubernetes1.10]# kubectl get node
NAME       STATUS     ROLES     AGE       VERSION
master1    NotReady   master    3m        v1.10.1
```

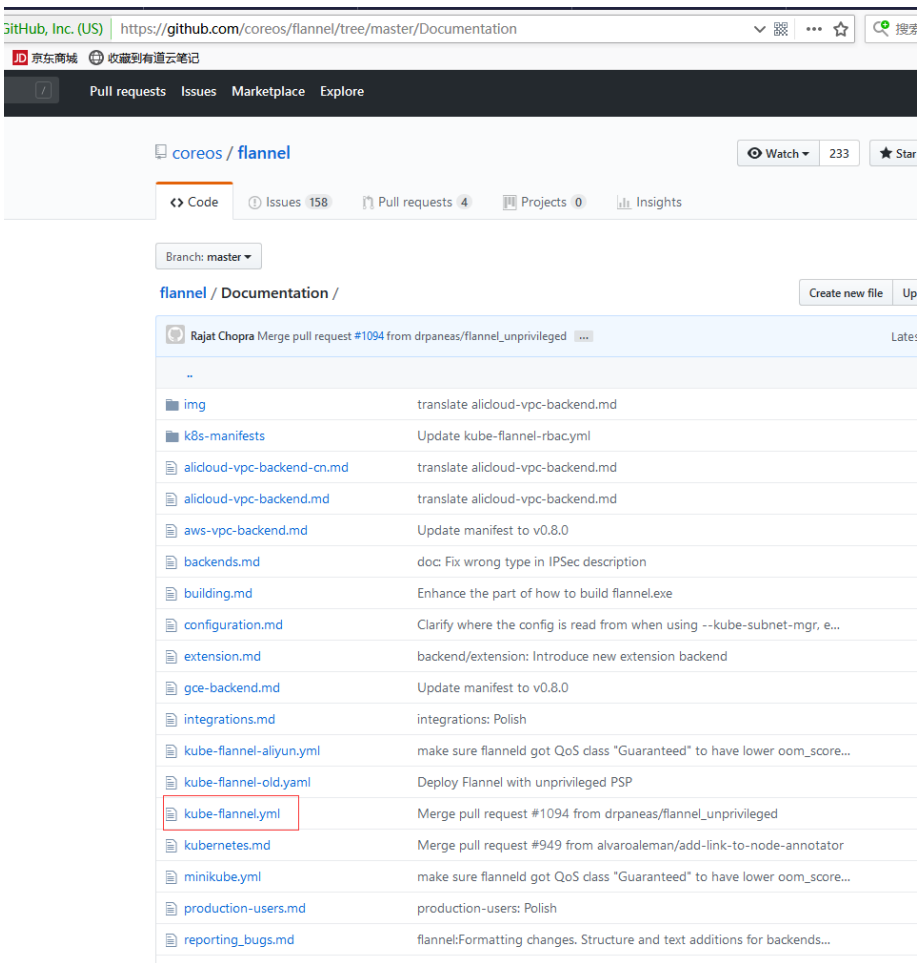
查看所有的pod，kubectl get pod --all-namespaces。kubedns也依赖于容器网络，此时pending是正常的

类似如下（这些图都是网上找的，安装的过程中没来的及记录）

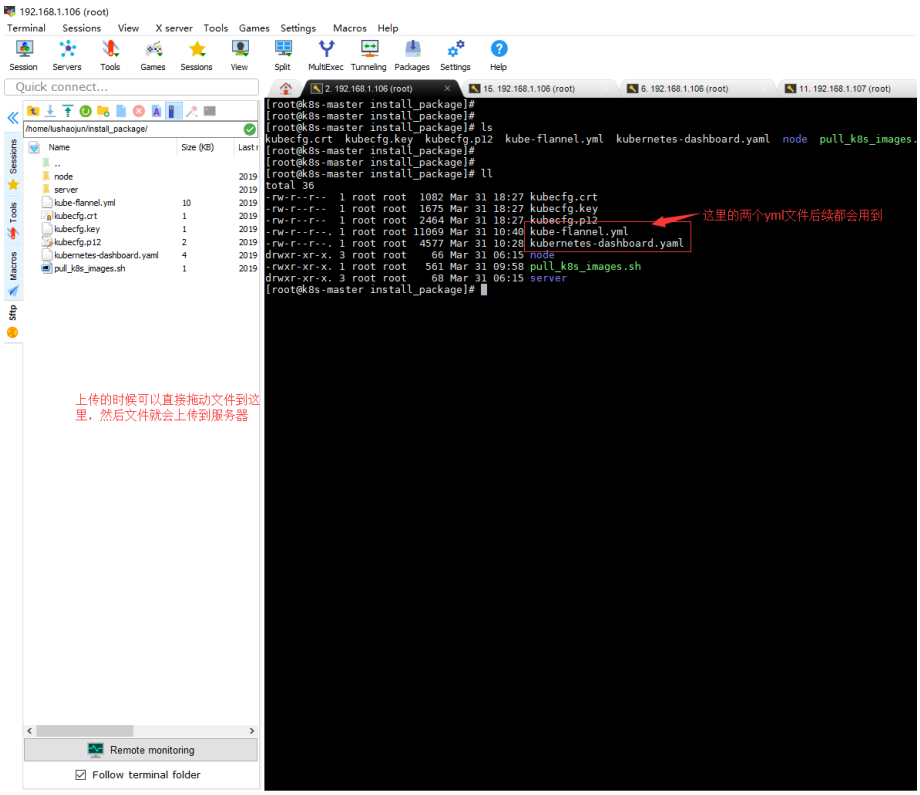
```
[root@master1 kubernetes1.10]# kubectl get pod —all-namespaces
NAMESPACE     NAME                              READY     STATUS    RESTARTS   AGE
kube-system   etcd-master1                      1/1       Running   0          3m
kube-system   kube-apiserver-master1            1/1       Running   0          3m
kube-system   kube-controller-manager-master1   1/1       Running   0          3m
kube-system   kube-dns-86f4d74b45-5nrb5         0/3       Pending   0          4m
kube-system   kube-proxy-ktxmb                  1/1       Running   0          4m
kube-system   kube-scheduler-master1            1/1       Running   0          3m
```

部署flannel网络，可以去https://github.com/coreos/flannel中找到

kube-flannel.yml文件，可以git clone到本地之后再上传到服务器上。



上传到master服务器



kubernetes-dashboard.yaml是后面创建dashboard需要的yaml文件，可以直接wegt获取

```
https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml
```

它的github链接为：https://github.com/kubernetes/dashboard

然后部署flannel网络，需要执行：

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

网络就绪后，节点的状态会变为ready，类似这样

```
[root@master1 kubernetes1.10]# kubectl get node
NAME      STATUS    ROLES     AGE       VERSION
master1   Ready     master    18m       v1.10.1
```

4.5 node节点加入集群

在Node节点执行，如下命令：

```
kubeadm join 192.168.1.106:6443 --token wct45y.tq23fogetd7rp3ck --discovery-token-ca-cert-hash sha256:c267e2423dba21fdf6fc9c07e3b3fa17884c4f24f0c03f2283a230c70b07772f
```

执行完成之后然后再master节点上执行kubectl get nodes，就可以看到类似：

```
[root@master1 kubernetes1.10]# kubectl get nodes
NAME      STATUS    ROLES     AGE       VERSION
master1   Ready     master    31m       v1.10.1
node1     Ready     <none>    44s       v1.10.1
```

代表节点加入了k8s集群，在k8s-node1和k8s-node2上执行完成之后，最终结果如图：

```
[root@k8s-master ~]# kubectl get nodes
NAME          STATUS   ROLES    AGE     VERSION
k8s-master    Ready    master   3h59m   v1.14.0
k8s-node1     Ready    <none>   3h57m   v1.14.0
k8s-node2     Ready    <none>   3h57m   v1.14.0
[root@k8s-master ~]#
```

表示整个Kubernetes-1.14.0集群就安装好了。

4.6 部署k8s ui界面，dashboard

首先修改kubernetes-dashboard.yaml，在最尾部部分修改：

```
# ------------------- Dashboard Service ------------------- #

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
spec:
  ports:
    - port: 443
      targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
```

这里添加

```
# 添加映射到虚拟机的端口,k8s只支持30000以上的端口
      nodePort: 30001
```

变成类似：

```yaml
kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
spec:
  # 添加Service的type为NodePort
  type: NodePort
  ports:
    - port: 443
      targetPort: 8443
      # 添加映射到虚拟机的端口,k8s只支持30000以上的端口
      nodePort: 30001
  selector:
    k8s-app: kubernetes-dashboard
```

然后执行

```
kubectl apply -f kubernetes-dashboard.yaml
```

接着就可通过在浏览器输入IP:端口就可以访问k8s的dashboard了

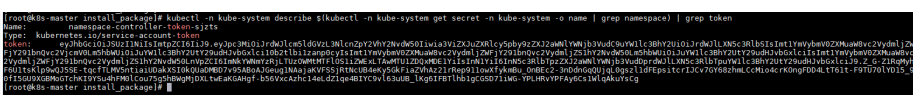
```
https://<node-ip>:<node-port>
```

这里对应应该是192.168.1.106:30001（master IP:上面添加的nodePort）

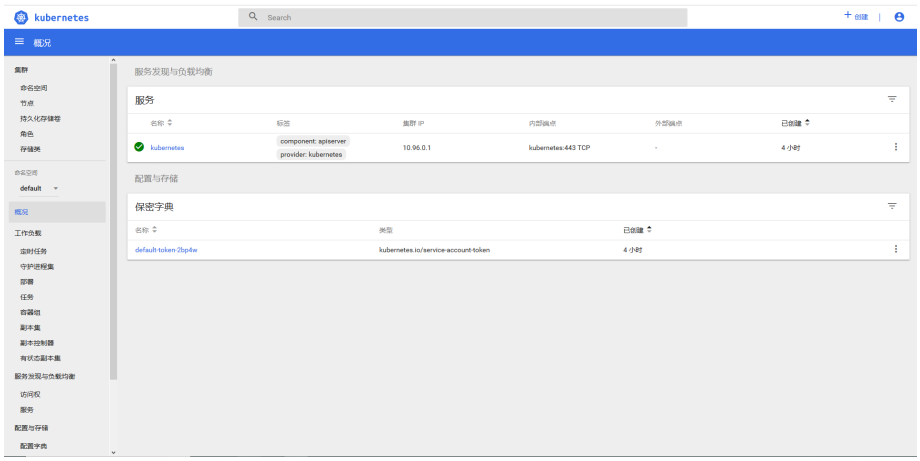
打开的过程中需要进行认证，可以选择“令牌”

令牌可以通过如下命令获得：

```
kubectl -n kube-system describe $(kubectl -n kube-system
get secret -n kube-system -o name | grep namespace) | grep
token
```



接着就可以看到kubernetes dashboard界面了



4.7 其他设置

a master节点默认不可部署pod

执行类似如下命令，可以在 kubectl edit node master1中taint配置参数下查到

```
root@master1:/var/lib/kubelet# kubectl taint node master1
node-role.kubernetes.io/master- node "master1" untainted
```

b 清理系统，重新搭建需要执行kubeadm reset

```
kubeadm reset
```

c 查看详细的pod信息

```
kubectl get pods -o wide
```

5 参考链接

5.1 centos7.3 kubernetes/k8s 1.10 离线安装

https://www.jianshu.com/p/9c7e1c957752

5.2 kubernetes1.13安装dashboard

https://blog.csdn.net/fanren224/article/details/86610466

5.3 [kubernetes1.9安装dashboard，以及token认证问题]

https://segmentfault.com/a/1190000013681047