openEuler 欧拉 22.03(LTS-SP1)安装 K8S

目录

一、环境要求

需要准备三台 Centos 虚拟机，并且安装好 Docker 组件，具体规格如下所示：

	CPU	内存	硬盘	备注
Master	4C	4G	60G	
Node1	4C	4G	60G	
Node2	4C	4G	60G	

二、欧拉系统安装

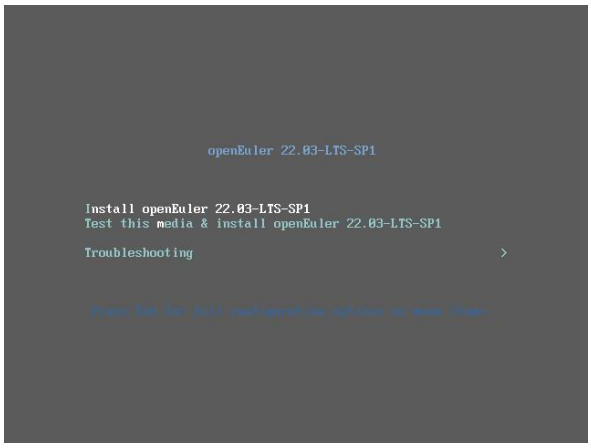
2.1 镜像下载

链接如下所示，下载标准版系统即可

https://www.openeuler.org/zh/download/archive/detail/?version=openEuler%2022.03%20LTS%20SP1

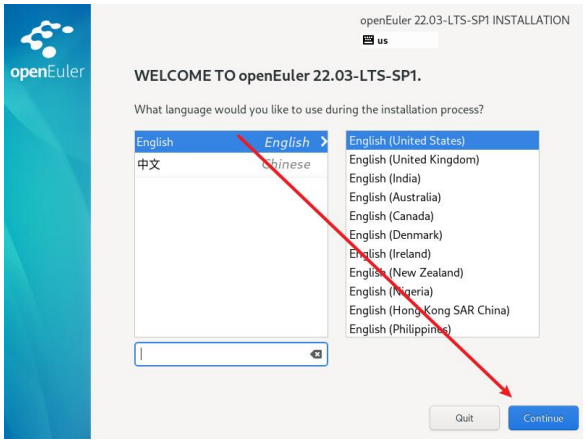
2.2 系统安装

虚机创建完成加电开机后方向键上键选择第一个选项后回车。



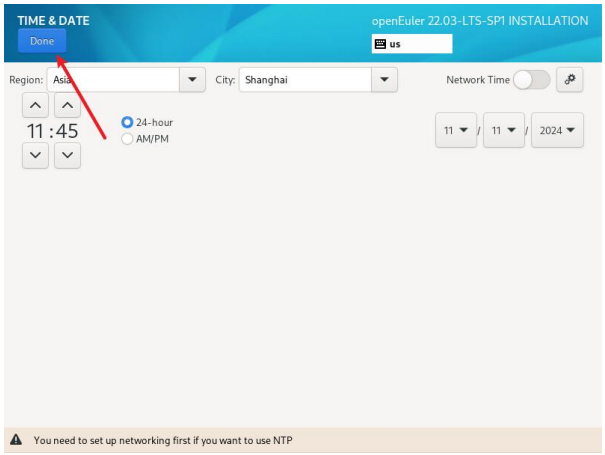
2.3 选择系统语言

选择系统语言为英文



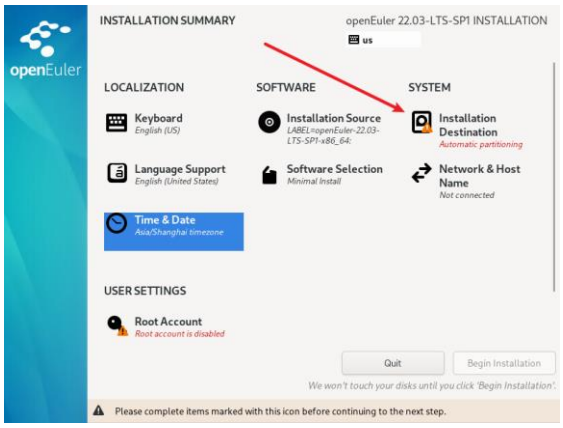
2.4 配置时间

点击 Time&Date，修改时区为亚洲-上海，修改时间为本机时间

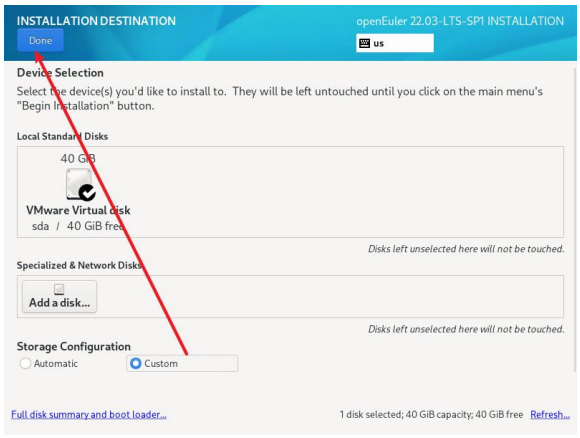


2.5 配置磁盘

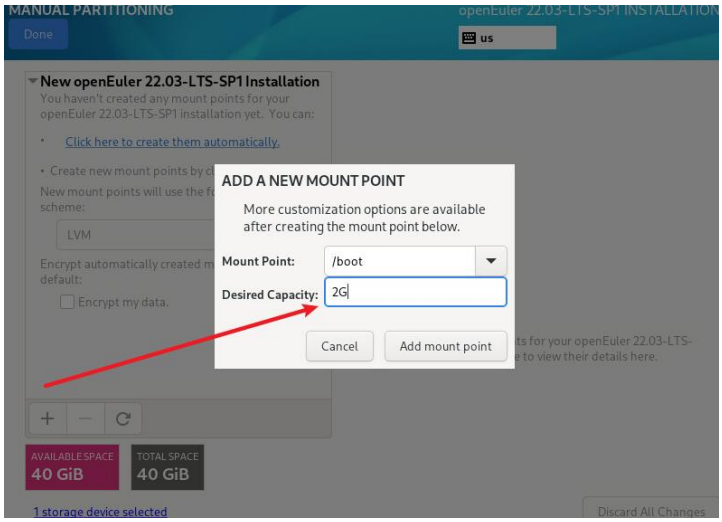
点击 Installation Destination



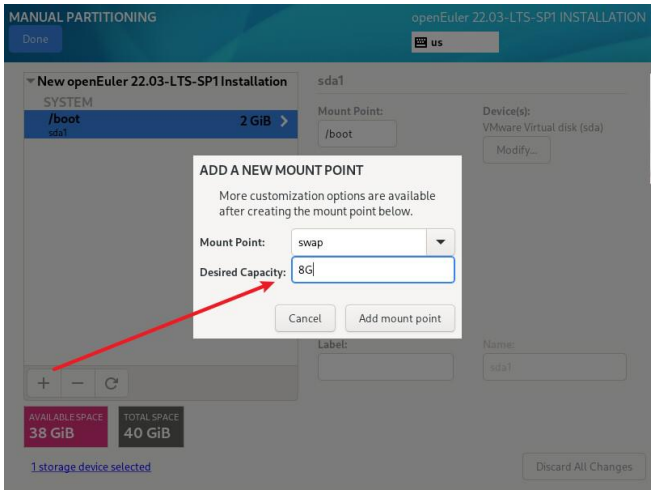
选择 Custom 后点击 Done 进行手动分区



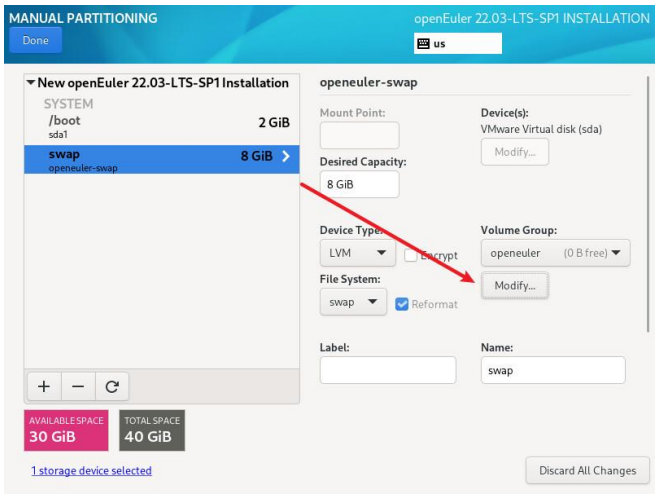
点击“+”号进行手动分区，创建/boot 分区，大小为 2G



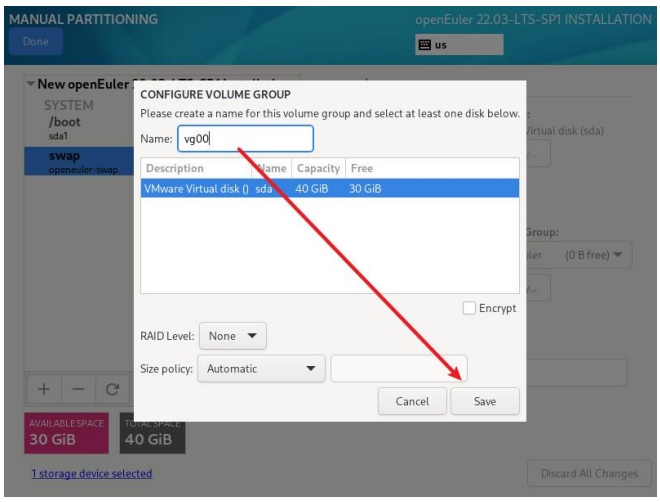
创建 swap 分区，swap 分区根据内存大小来判断，内存小于 4G，设置为内存的 2 倍，内存足够大，可以再设置大一些。



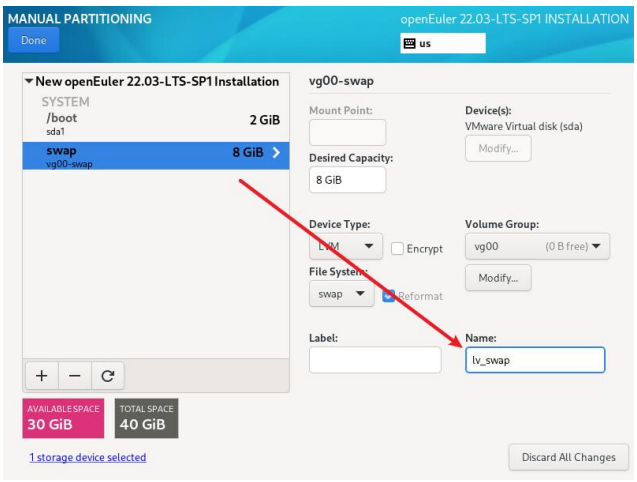
点击 Modify 进行修改卷组



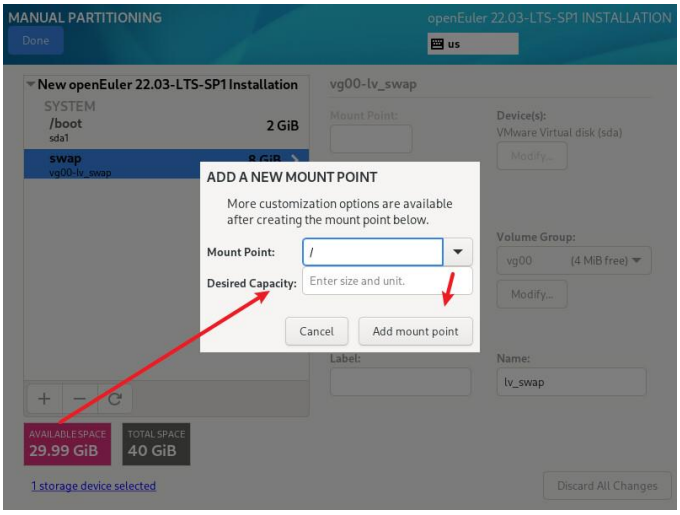
修改卷组名称为 vg00 后点击 save



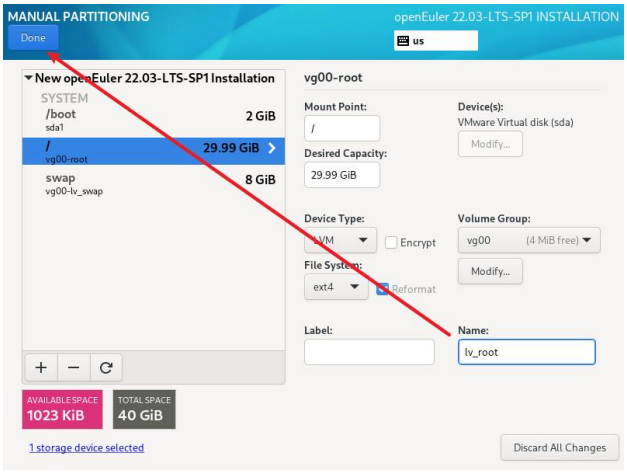
修改名称为 lv_swap



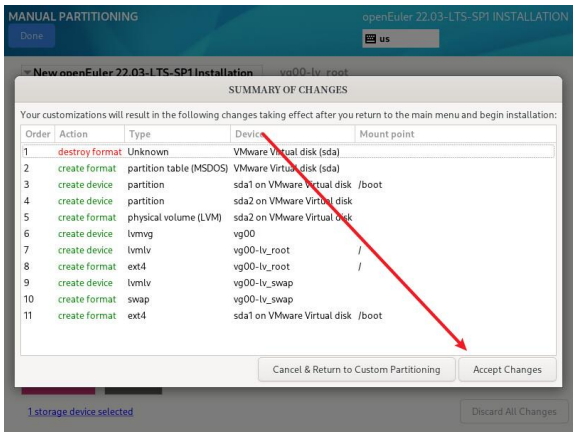
剩余其他空间全部分配给“/”



修改名称为 lv_root 后点击 Done 完成配置

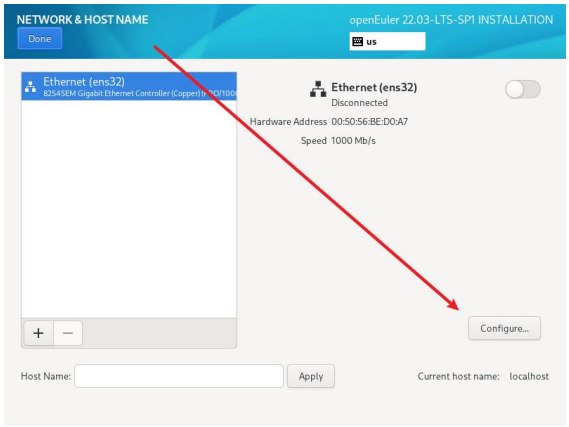


点击接受更改完成磁盘分区配置

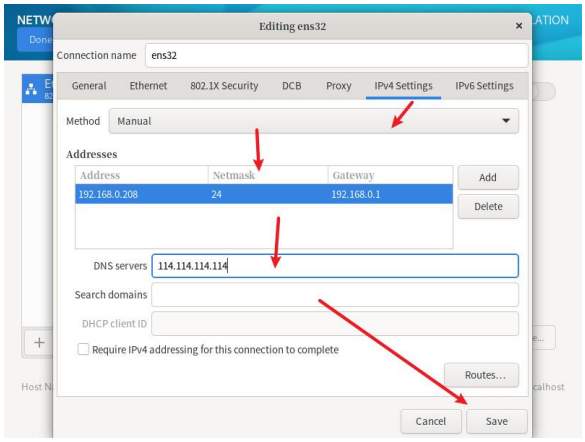


2.6 配置网络

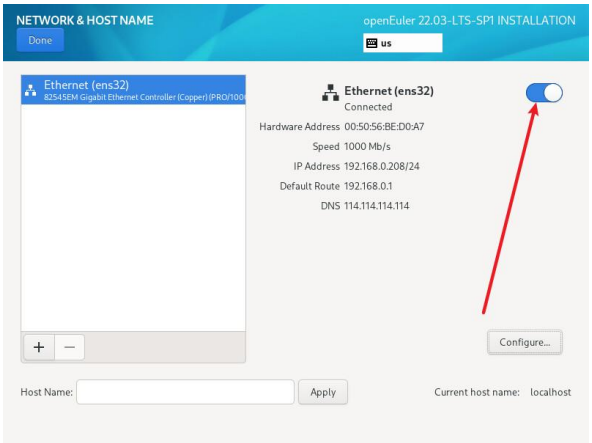
点击 NETWORK&HOSTNAME 后点击 configure 进行配置地址



配置 IPv4 地址类型为手动，新增 IP 地址、掩码、网关、DNS 后点击 save 进行保存。

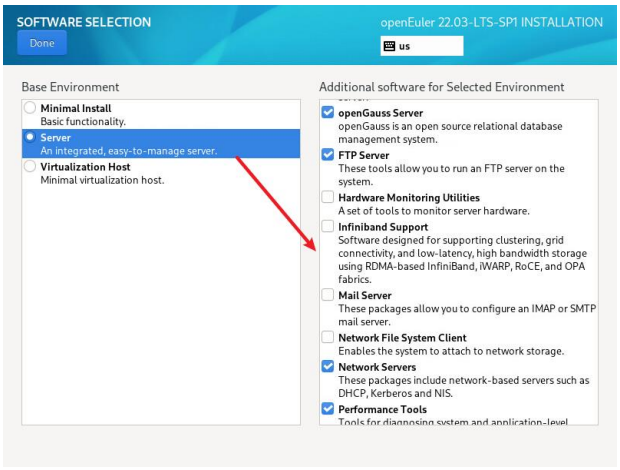


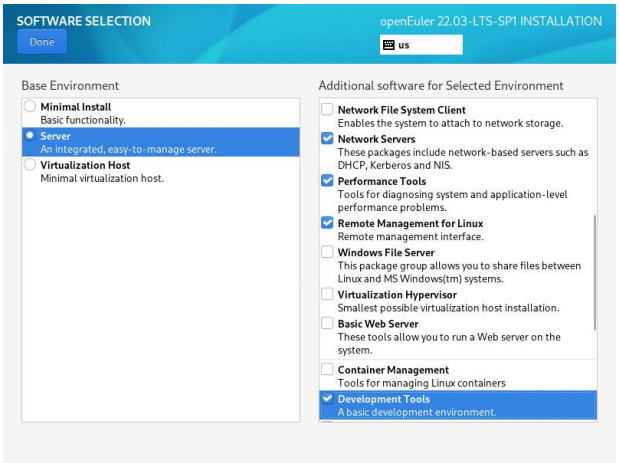
完成网卡配置后需要开启网卡开关，然后点击 Done 完成配置。



2.7 软件选择

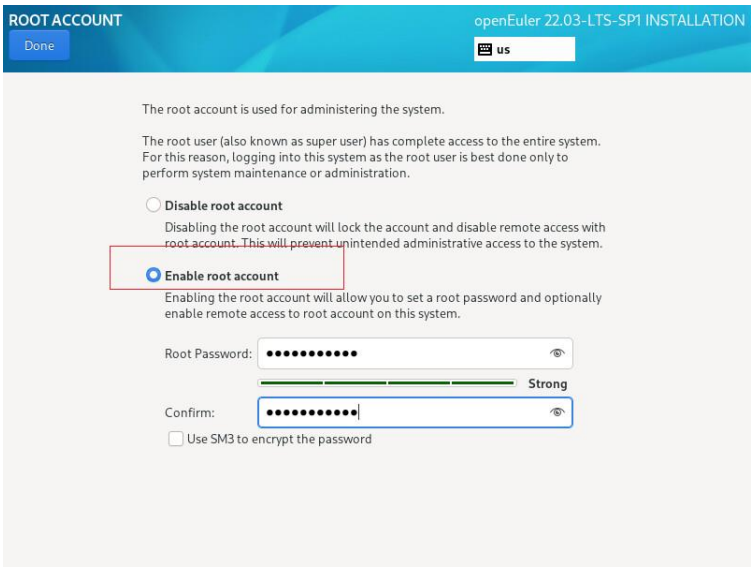
点击 SOFTWARE SELECTION 选择服务器环境，右侧软件勾选如下所示内容



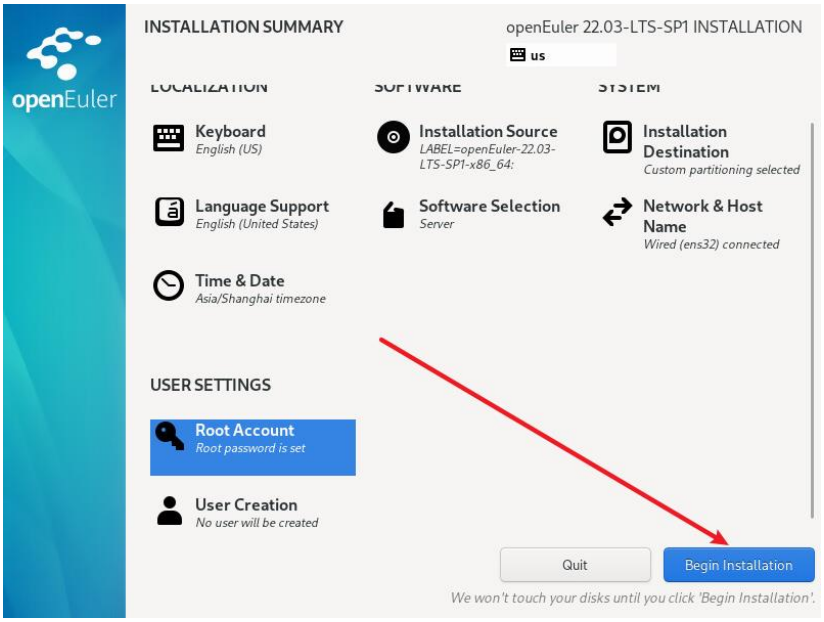


2.8 设置 root 密码

点击 ROOT ACCOUNT 后选择 enable root account，设置 root 密码后点击 DONE 完成配置



设置完成后点击 Begin Installation 完成虚机设置，完成安装后点击 reboot system 进行重启系统。



三、基础配置

3.1 系统版本信息

输入 cat /etc/os-release 查看当前安装的系统版本否为 22.03(LTS-SP1)

```
[root@Master ~]# cat /etc/os-release
NAME="openEuler"
VERSION="22.03 (LTS-SP1)"
ID="openEuler"
VERSION_ID="22.03"
PRETTY_NAME="openEuler 22.03 (LTS-SP1)"
ANSI_COLOR="0;31"

[root@Master ~]#
```

3.2 修改网卡 IP 地址

欧拉 22.03 版本修改网卡接口 IP 方式和 Centos 基本一致，先输入 ifconfig 或 ip add 查看当前的网卡编号

```
[root@localhost ~]# ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.205  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::250:56ff:febe:e471  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:be:e4:71  txqueuelen 1000  (Ethernet)
        RX packets 567  bytes 47958 (46.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 247  bytes 35848 (35.0 KiB)
        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 324  bytes 62911 (61.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 324  bytes 62911 (61.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:34:bc:47  txqueuelen 1000  (Ethernet)
        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
```

然后输入 vi /etc/sysconfig/network-scripts/ifcfg-ens160 进行修改 IP，修改完成后按“ESC”键，然后按“:”键再输入 wq 进行保存退出。

```
[root@Master ~]#
[root@Master ~]# vi /etc/sysconfig/network-scripts/ifcfg-ens160
```

修改完 IP 后需要输入 nmcli c reload 和 nmclic c up ens160 进行

应用新的配置。

```
[root@localhost ~]#
[root@localhost ~]# nmcli c reload
[root@localhost ~]#
[root@localhost ~]# nmcli c up ens160
```

3.3 修改主机名称和主机映射

在 Master 和 Node 节点上输入 vi /etc/hostname 后，分别修改虚机的名称，修改后重启虚机重新连接 SSH 窗口后查看名字是否正常。

```
[root@Master ~]# cat /etc/hostname
Master
[root@Master ~]#
```

在 Master 和 Node 节点上输入 vi /etc/hosts 后，将虚机名称与实际的 IP 地址进行一对一配置。

```
[root@Master ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.0.205 Master
192.168.0.206 Node-1
192.168.0.207 Node-2
[root@Master ~]#
```

配置完成后需要进行 ping 测试是否正常通信。

```
[root@Master ~]# ping Node-1
\PING Node-1 (192.168.0.206) 56(84) bytes of data.
64 bytes from Node-1 (192.168.0.206): icmp_seq=1 ttl=64 time=0.900 ms
64 bytes from Node-1 (192.168.0.206): icmp_seq=2 ttl=64 time=0.412 ms
^C
--- Node-1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.412/0.656/0.900/0.244 ms
[root@Master ~]# ping Node-2
PING Node-2 (192.168.0.207) 56(84) bytes of data.
64 bytes from Node-2 (192.168.0.207): icmp_seq=1 ttl=64 time=0.885 ms
64 bytes from Node-2 (192.168.0.207): icmp_seq=2 ttl=64 time=0.325 ms
^C
--- Node-2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.325/0.605/0.885/0.280 ms
[root@Master ~]#
```

3.4 关闭防火墙

在 Master 和 Node 节点上输入 systemctl disable firewalld 和

systemctl stop firewalld 关闭防火墙，然后输入 firewall-cmd --state 查看防火墙状态是否为 not running

```
[root@Master ~]#
[root@Master ~]# systemctl disable firewalld
Removed /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
Removed /etc/systemd/system/multi-user.target.wants/firewalld.service.
[root@Master ~]#
[root@Master ~]# systemctl stop firewalld
[root@Master ~]#
[root@Master ~]# firewall-cmd --state
not running
[root@Master ~]#
```

3.5 关闭 selinux

在 Master 和 Node 节点上输入 vi /etc/sysconfig/selinux 后将 SELINUX 的状态修改为 disabled

```
[root@Node-2 ~]# cat /etc/sysconfig/selinux

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3.6 配置时间同步

在 Master 和 Node 节点上输入 dnf install ntpdate 安装 ntpdate 服务

```
[root@Master ~]#
[root@Master ~]# dnf install ntpdate
Last metadata expiration check: 0:21:49 ago on Wed 06 Nov 2024 10:19:30 AM CST.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version                 Repository      Size
================================================================================
Installing:
 ntp                     x86_64              4.2.8p15-13.oe2203sp1   update         619 k
Installing dependencies:
 ntp-help                noarch              4.2.8p15-13.oe2203sp1   update         1.3 M
Installing weak dependencies:
 ntpstat                 noarch              0.6-4.oe2203sp1         OS              11 k

Transaction Summary
================================================================================
Install  3 Packages

Total download size: 1.9 M
Installed size: 3.9 M
Is this ok [y/N]:
```

在 Master 和 Node 节点上输入 ntpdate time1.aliyun.com 和阿里云的 NTP 同步一次时间

```
[root@Master ~]#
[root@Master ~]# ntpdate time1.aliyun.com
 6 Nov 10:55:54 ntpdate[8137]: step time server 203.107.6.88 offset +4.464857 sec
[root@Master ~]#
[root@Master ~]#
```

在 Master 和 Node 节点上输入 crontab -e 然后输入 0 0 * * * ntpdate time1.aliyun.com 配置每天 0 点自动和阿里云 NTP 服务器同步时间。

```
0 0 * * * ntpdate time1.aliyun.com
~
~
```

3.7 配置内核转发和网桥过滤

在 Master 和 Node 节点上输入 vi /etc/sysctl.conf 然后将 net.ipv4.ip_forwar=0 改为 1 用来开启内核路由转发

```
[root@Master ~]# cat /etc/sysctl.conf
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
kernel.sysrq=0
net.ipv4.ip_forward=1
net.ipv4.conf.all.send_redirects=0
net.ipv4.conf.default.send_redirects=0
net.ipv4.conf.all.accept_source_route=0
net.ipv4.conf.default.accept_source_route=0
net.ipv4.conf.all.accept_redirects=0
net.ipv4.conf.default.accept_redirects=0
net.ipv4.conf.all.secure_redirects=0
net.ipv4.conf.default.secure_redirects=0
net.ipv4.icmp_echo_ignore_broadcasts=1
net.ipv4.icmp_ignore_bogus_error_responses=1
net.ipv4.conf.all.rp_filter=1
net.ipv4.conf.default.rp_filter=1
net.ipv4.tcp_syncookies=1
kernel.dmesg_restrict=1
net.ipv6.conf.all.accept_redirects=0
net.ipv6.conf.default.accept_redirects=0
[root@Master ~]#
```

在 Master 和 Node 节点上输入

cat > /etc/sysctl.d/k8s.conf <<EOF

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

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

EOF

用于添加网桥过滤和内核转发配置文件

```
[root@Master ~]# cat > /etc/sysctl.d/k8s.conf  <<EOF
> net.bridge.bridge-nf-call-ip6tables = 1
> net.bridge.bridge-nf-call-iptables = 1
> EOF
[root@Master ~]#
[root@Master ~]#
```

在 Master 和 Node 节点上输入 modprobe br_netfilter 用来加载 br_netfilter 模块

```
[root@Node-2 ~]#
[root@Node-2 ~]# modprobe br_netfilter
[root@Node-2 ~]#
[root@Node-2 ~]#
```

在 Master 和 Node 节点上输入 lsmod | grep br_netfilter 查看是否完成加载

```
[root@Master ~]# lsmod | grep br_netfilter
br_netfilter           32768  0
bridge                270336  1 br_netfilter
[root@Master ~]#
```

在 Master 和 Node 节点上输入 sysctl -p 使用默认配置文件生效

```
[root@Master ~]# sysctl -p
/proc/sys/kernel/sysrq = 0
/proc/sys/net/ipv4/ip_forward = 1
/proc/sys/net/ipv4/conf/all/send_redirects = 0
/proc/sys/net/ipv4/conf/default/send_redirects = 0
/proc/sys/net/ipv4/conf/all/accept_source_route = 0
/proc/sys/net/ipv4/conf/default/accept_source_route = 0
/proc/sys/net/ipv4/conf/all/accept_redirects = 0
/proc/sys/net/ipv4/conf/default/accept_redirects = 0
/proc/sys/net/ipv4/conf/all/secure_redirects = 0
/proc/sys/net/ipv4/conf/default/secure_redirects = 0
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts = 1
/proc/sys/net/ipv4/icmp_ignore_bogus_error_responses = 1
/proc/sys/net/ipv4/conf/all/rp_filter = 1
/proc/sys/net/ipv4/conf/default/rp_filter = 1
/proc/sys/net/ipv4/tcp_syncookies = 1
/proc/sys/kernel/dmesg_restrict = 1
/proc/sys/net/ipv6/conf/all/accept_redirects = 0
/proc/sys/net/ipv6/conf/default/accept_redirects = 0
[root@Master ~]#
```

在 Master 和 Node 节点上同时还需要输入 sysctl -p /etc/sysctl.d/k8s.conf 使配置生效

```
[root@Node-2 ~]#
[root@Node-2 ~]# sysctl -p /etc/sysctl.d/k8s.conf
/proc/sys/net/bridge/bridge-nf-call-ip6tables = 1
/proc/sys/net/bridge/bridge-nf-call-iptables = 1
[root@Node-2 ~]#
[root@Node-2 ~]#
```

在 Master 和 Node 节点上配置 br_netfilter 开机自动启动功能

输入 vi /etc/sysconfig/modules/brnetfilter.modules 后输入如下信息

```
[root@Master ~]# vi /etc/sysconfig/modules/brnetfilter.modules
[root@Master ~]#
[root@Master ~]# cat /etc/sysconfig/modules/brnetfilter.modules
#!/bin/bash
modprobe br_netfilter
[root@Master ~]#
[root@Master ~]#
```

在 Master 和 Node 节点上配置 brnetfilter.modules 可执行权限

chmod +x /etc/sysconfig/modules/brnetfilter.modules

```
[root@Master ~]#
[root@Master ~]# chmod +x /etc/sysconfig/modules/brnetfilter.modules
[root@Master ~]#
[root@Master ~]#
```

3.8 安装 ipset 和 ipvsadm

在 Master 和 Node 节点上输入 yum -y install ipset ipvsadm 进行安装组件

```
[root@Node-2 ~]#
[root@Node-2 ~]#
[root@Node-2 ~]# yum -y install ipset ipvsadm
Last metadata expiration check: 1:29:32 ago on Wed 06 Nov 2024 10:36:57 AM CST.
Package ipset-7.15-2.oe2203sp1.x86_64 is already installed.
Dependencies resolved.
================================================================================
```

输入 vi /etc/sysconfig/modules/ipvs.modules 后输入以下内容

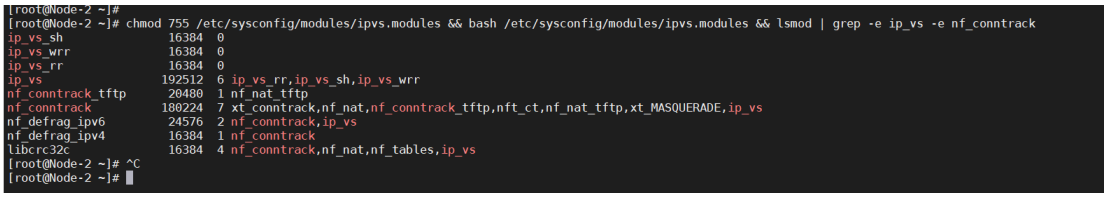
```
#!/bin/bash
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack
```

```
[root@Node-2 ~]# vi /etc/sysconfig/modules/ipvs.modules
[root@Node-2 ~]#
[root@Node-2 ~]# cat /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
[root@Node-2 ~]#
[root@Node-2 ~]#
```

在 Master 和 Node 节点上输入以下命令对 ipvs-modules 文件授权并启动该配置，查看进程是否如下所示。

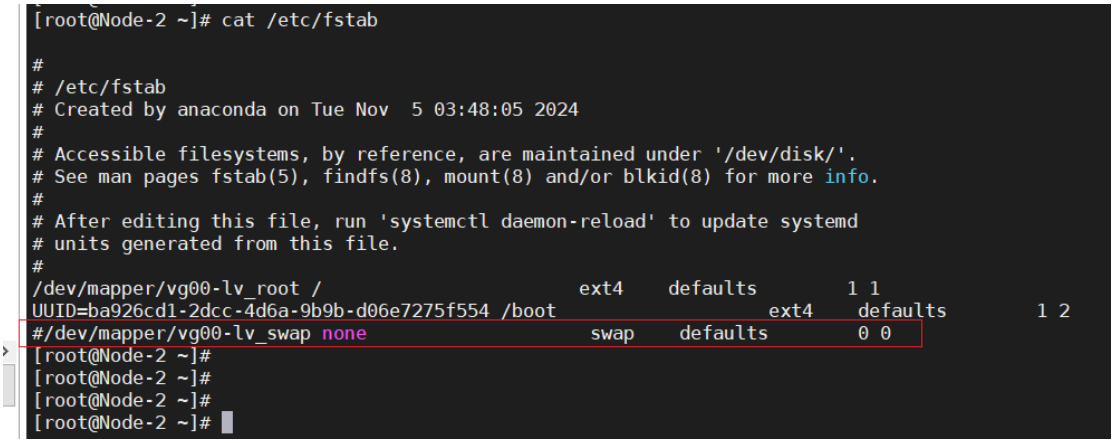
```
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash
```

/etc/sysconfig/modules/ipvs.modules && lsmod | grep -e ip_vs -e nf_conntrack



3.9 关闭 SWAP 分区

在 Master 和 Node 节点上输入 vi /etc/fstab 注释掉如下图所示的行，完成后需要 reboot 重启系统。



四、docker 工具安装

4.1 查看当前系统中的 docker 版本

输入 dnf list | grep docker 查看当前系统中是否存在 docker18.09 版本

```
[root@Master ~]# dnf list | grep docker
pcp-pmda-docker.x86_64                  5.3.7-2.oe2203sp1                   @anaconda
docker-client-java.noarch               8.11.7-2.oe2203sp1                  everything
docker-client-java.src                  8.11.7-2.oe2203sp1                  source
docker-compose.noarch                   1.22.0-4.oe2203sp1                  everything
docker-compose.src                      1.22.0-4.oe2203sp1                  source
docker-engine.src                       2:18.09.0-340.oe2203sp1             update-source
docker-engine.x86_64                    2:18.09.0-340.oe2203sp1             update
docker-engine-debuginfo.x86_64          2:18.09.0-340.oe2203sp1             update
docker-engine-debugsource.x86_64        2:18.09.0-340.oe2203sp1             update
docker-runc.src                         1.1.3-30.oe2203sp1                  update-source
docker-runc.x86_64                      1.1.3-30.oe2203sp1                  update
pcp-pmda-docker.x86_64                  5.3.7-4.oe2203sp1                   update
podman-docker.noarch                    1:0.10.1-12.oe2203sp1               everything
python-docker.src                       5.0.3-1.oe2203sp1                   source
python-docker-help.noarch               5.0.3-1.oe2203sp1                   everything
python-docker-pycreds.src               0.4.0-2.oe2203sp1                   source
python-dockerpty.src                    0.4.1-3.oe2203sp1                   source
python-dockerpty-help.noarch            0.4.1-3.oe2203sp1                   everything
python3-docker.noarch                   5.0.3-1.oe2203sp1                   everything
python3-docker-pycreds.noarch           0.4.0-2.oe2203sp1                   everything
python3-dockerpty.noarch                0.4.1-3.oe2203sp1                   everything
[root@Master ~]#
```

4.2 安装 docker-engine

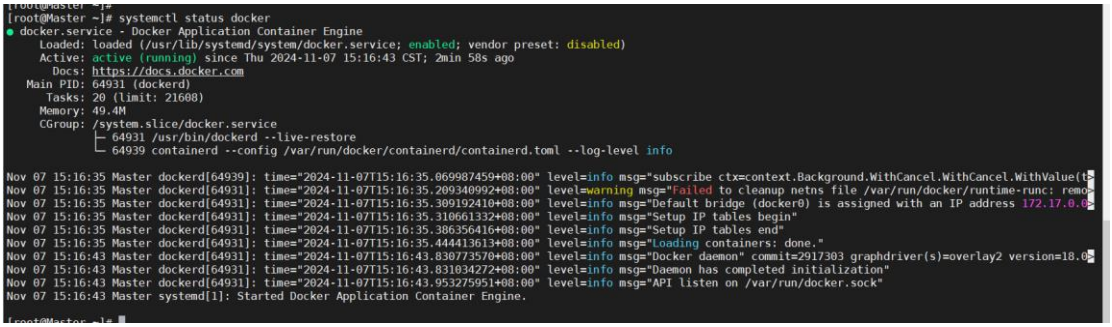
在 Master 和 Node 输入 dnf install docker 进行安装 docker-engine

```
[root@Master ~]# dnf install docker
Last metadata expiration check: 1:16:16 ago on Thu 07 Nov 2024 01:59:53 PM CST.
Dependencies resolved.
================================================================================
 Package              Architecture        Version                   Repository      Size
================================================================================
Installing:
 docker-engine        x86_64              2:18.09.0-340.oe2203sp1   update          39 M

Transaction Summary
================================================================================
Install  1 Package
```

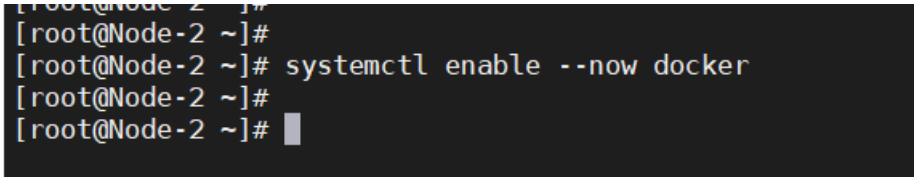
输入 docker version 进行验证是否完成安装 docker-engine

```
[root@Master ~]# docker version
Client:
 Version:           18.09.0
 EulerVersion:      18.09.0.340
 API version:       1.39
 Go version:        go1.17.3
 Git commit:        2917303
 Built:             Tue Aug  6 03:51:50 2024
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.09.0
  EulerVersion:     18.09.0.340
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.17.3
  Git commit:       2917303
  Built:            Tue Aug  6 03:51:11 2024
  OS/Arch:          linux/amd64
  Experimental:     false
[root@Master ~]#
```

输入 systemctl status docker 查看 docker 进程是否启动。



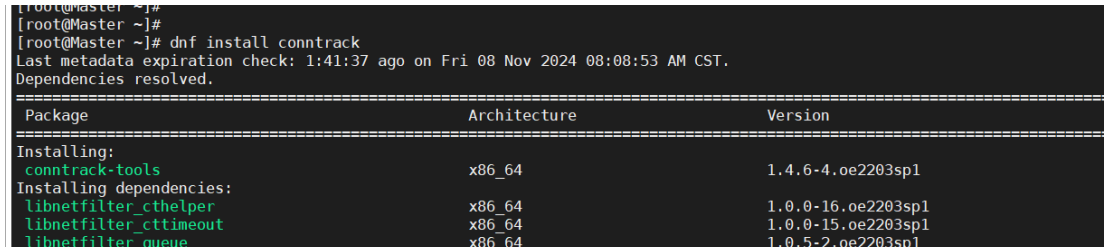
在 Master 和 Node 输入 systemctl enable --now docker 设置 docker 开机自启动



五、K8S 软件安装

5.1 安装 K8S 依赖

在 Master 和 Node 节点输入 dnf install conntrack 安装 K8S 依赖。



5.2 安装 kubeadm、kubelet 等组件

在 Master 节点输入以下内容进行安装组件

dnf install -y kubernetes-kubeadm kubernetes-kubelet kubernetes-master

```
[root@Master ~]#
[root@Master ~]# dnf install -y kubernetes-kubeadm kubernetes-kubelet kubernetes-master
Last metadata expiration check: 1:50:15 ago on Fri 08 Nov 2024 08:08:53 AM CST.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version                 Repository
================================================================================
Installing:
 kubernetes-kubeadm      x86_64              1.20.2-16.oe2203sp1     EPOL
 kubernetes-kubelet      x86_64              1.20.2-16.oe2203sp1     EPOL
 kubernetes-master       x86_64              1.20.2-16.oe2203sp1     EPOL
Installing dependencies:
 kubernetes-client       x86_64              1.20.2-16.oe2203sp1     EPOL

Transaction Summary
================================================================================
Install  4 Packages
```

在 Node 节点上输入以下内容进行安装组件

dnf install -y kubernetes-kubeadm kubernetes-kubelet kubernetes-node

```
[root@Node-1 ~]#
[root@Node-1 ~]#
[root@Node-1 ~]# dnf install -y kubernetes-kubeadm kubernetes-kubelet kubernetes-node
Last metadata expiration check: 2:28:24 ago on Fri 08 Nov 2024 07:33:11 AM CST.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version
================================================================================
Installing:
 kubernetes-kubeadm      x86_64              1.20.2-16.oe2203sp1
 kubernetes-kubelet      x86_64              1.20.2-16.oe2203sp1
 kubernetes-node         x86_64              1.20.2-16.oe2203sp1
Installing dependencies:
 kubernetes-client       x86_64              1.20.2-16.oe2203sp1
 socat                   x86_64              1.7.3.2-8.oe2203sp1
```

5.3 启动 kubelet 组件服务

在 Master 和 Node 节点上输入 systemctl enable kubelet 启动 kubelet 服务

```
[root@Master ~]#
[root@Master ~]# systemctl enable kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@Master ~]#
```

在 Master 上使用命令查看阿里云镜像仓库中是否存在相关镜像

kubeadm config images list --kubernetes-version=v1.20.2 --image-repository registry.aliyuncs.com/google_containers

```
[root@Master ~]#
[root@Master ~]# kubeadm config images list --kubernetes-version=v1.20.2 --image-repository registry.aliyuncs.com/google_containers
registry.aliyuncs.com/google_containers/kube-apiserver:v1.20.2
registry.aliyuncs.com/google_containers/kube-controller-manager:v1.20.2
registry.aliyuncs.com/google_containers/kube-scheduler:v1.20.2
registry.aliyuncs.com/google_containers/kube-proxy:v1.20.2
registry.aliyuncs.com/google_containers/pause:3.2
registry.aliyuncs.com/google_containers/etcd:3.4.13-0
registry.aliyuncs.com/google_containers/coredns:1.7.0
[root@Master ~]#
```

在 Master 上使用命令安装相关镜像。

kubeadm config images pull --kubernetes-version=v1.20.2 --

image-repository registry.aliyuncs.com/google_containers

```
[root@Master ~]#
[root@Master ~]#
[root@Master ~]# kubeadm config images pull --kubernetes-version=v1.20.2 --image-repository registry.aliyuncs.com/google_containers
[config/images] Pulled registry.aliyuncs.com/google_containers/kube-apiserver:v1.20.2
[config/images] Pulled registry.aliyuncs.com/google_containers/kube-controller-manager:v1.20.2
[config/images] Pulled registry.aliyuncs.com/google_containers/kube-scheduler:v1.20.2
[config/images] Pulled registry.aliyuncs.com/google_containers/kube-proxy:v1.20.2
[config/images] Pulled registry.aliyuncs.com/google_containers/pause:3.2
[config/images] Pulled registry.aliyuncs.com/google_containers/etcd:3.4.13-0
[config/images] Pulled registry.aliyuncs.com/google_containers/coredns:1.7.0
[root@Master ~]#
[root@Master ~]# ^C
[root@Master ~]# ^C
[root@Master ~]#
```

输入 docker images 查看是否下载完成 7 个镜像

```
[root@Master ~]#
[root@Master ~]# docker images
REPOSITORY                                                        TAG                 IMAGE ID            CREATED             SIZE
registry.aliyuncs.com/google_containers/kube-proxy                v1.20.2             43154ddb57a8        3 years ago         118MB
registry.aliyuncs.com/google_containers/kube-apiserver            v1.20.2             a8c2fdb8bf76        3 years ago         122MB
registry.aliyuncs.com/google_containers/kube-controller-manager   v1.20.2             a27166429d98        3 years ago         116MB
registry.aliyuncs.com/google_containers/kube-scheduler            v1.20.2             ed2c44fbdd78        3 years ago         46.4MB
registry.aliyuncs.com/google_containers/etcd                      3.4.13-0            0369cf4303ff        4 years ago         253MB
registry.aliyuncs.com/google_containers/coredns                   1.7.0               bfe3a36ebd25        4 years ago         45.2MB
registry.aliyuncs.com/google_containers/pause                     3.2                 80d28bedfe5d        4 years ago         683kB
[root@Master ~]#
```

5.4 初始化 Master 集群

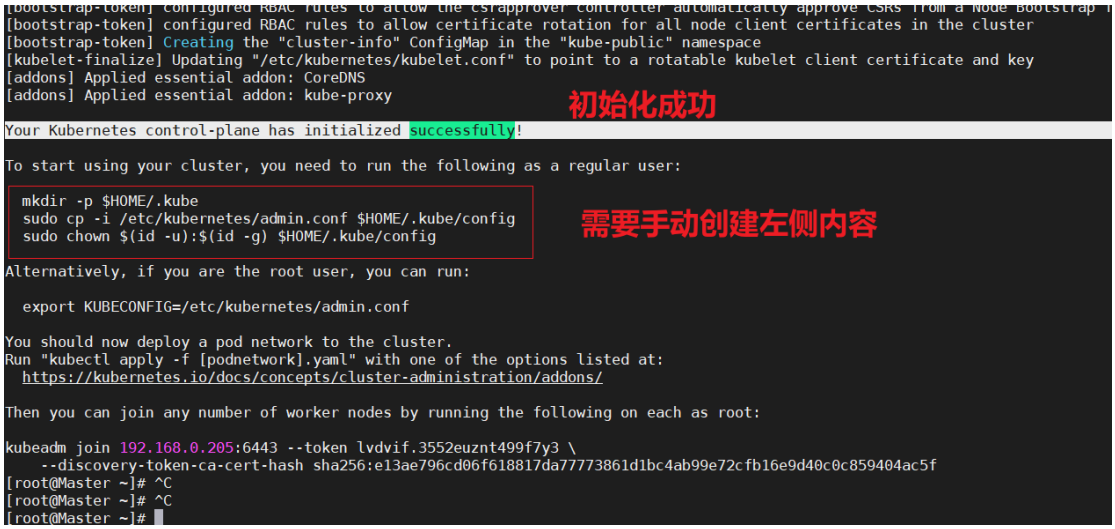
在 Master 上输入以下命令进行初始化集群，其中 apiserver-advertise-address 为 Master 虚机的 IP 地址。

kubeadm init --apiserver-advertise-address=192.168.0.205 --image-repository registry.aliyuncs.com/google_containers --kubernetes-version v1.20.2 --service-cidr=10.1.0.0/16 --pod-network-cidr=10.244.0.0/16

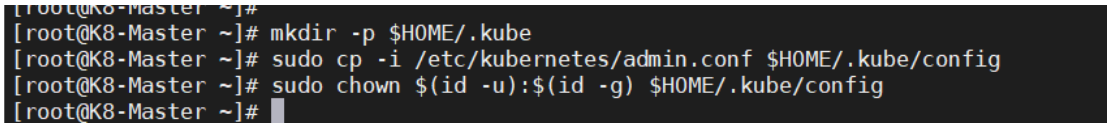
```
[root@Master ~]#
[root@Master ~]#
[root@Master ~]# kubeadm init --apiserver-advertise-address=192.168.0.205 --image-repository registry.aliyuncs.com/google_containers --kubernetes-version v1.20.2 --service
-cidr=10.1.0.0/16 --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.20.2
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at https://kubernetes
.io/docs/setup/cri/
        [WARNING FileExisting-socat]: socat not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
```

初始化完成后需要查看是否提示

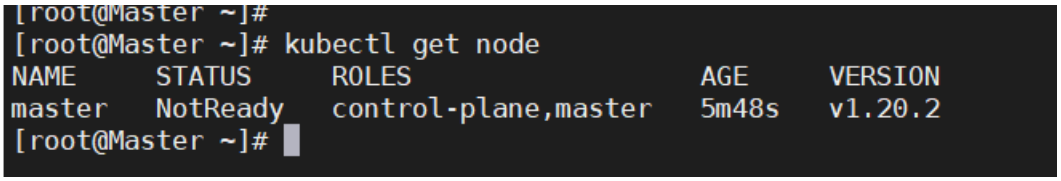
Your Kubernetes control-plane has initialized successfully!



按照部署成功后的提示需要在 Master 上创建以下目录和文件



使用 kubectl get node 查看 Master 节点状态



5.5 Node 节点加入 Master 集群

在 5.4 中输入初始化 Master 集群后，在最后面会自动生成 Node 节点加入 Master 集群的命令，只需要在 Node 节点上粘贴对应命令即可。

```
[addons] Applied essential addon: CoreDNS
[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

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 192.168.0.205:6443 --token lvdvif.3552euznt499f7y3 \
    --discovery-token-ca-cert-hash sha256:e13ae796cd06f618817da77773861d1bc4ab99e72cfb16e9d40c0c859404ac5f
[root@Master ~]# ^C
[root@Master ~]# ^C
[root@Master ~]# ^C
[root@Master ~]#
[root@Master ~]#
[root@Master ~]#
[root@Master ~]#
[root@Master ~]#
[root@Master ~]#  mkdir -p $HOME/.kube
```

在 Node 节点上粘贴如上信息。

```
[root@Node-1 ~]#
[root@Node-1 ~]# kubeadm join 192.168.0.205:6443 --token lvdvif.3552euznt499f7y3 \
>     --discovery-token-ca-cert-hash sha256:e13ae796cd06f618817da77773861d1bc4ab99e72cfb16e9d40c0c859404ac5f
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/
cri/
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[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] Starting the kubelet
[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.

[root@Node-1 ~]#
```

在 Master 节点上输入 kubectl get node 查看节点是否加入集群

```
[root@Master ~]# kubectl get node
NAME     STATUS     ROLES                  AGE   VERSION
master   NotReady   control-plane,master   15m   v1.20.2
node-1   NotReady   <none>                 80s   v1.20.2
node-2   NotReady   <none>                 32s   v1.20.2
[root@Master ~]#
```

5.6 下载 Calico 网络插件

在 Master 上输入以下命令进行下载 Calico 插件。

wget https://docs.projectcalico.org/v3.19/manifests/calico.yaml

```
[root@Master ~]#
[root@Master ~]# wget https://docs.projectcalico.org/v3.19/manifests/calico.yaml
--2024-11-08 11:33:29--  https://docs.projectcalico.org/v3.19/manifests/calico.yaml
Resolving docs.projectcalico.org (docs.projectcalico.org)... 52.74.232.59, 52.220.155.145, 2406:da18:b3d:e202::65, ...
Connecting to docs.projectcalico.org (docs.projectcalico.org)|52.74.232.59|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 189916 (185K) [text/yaml]
Saving to: 'calico.yaml'

calico.yaml                         100%[=============================================================================>] 185.46K   462KB/s    in 0.4s

2024-11-08 11:33:31 (462 KB/s) - 'calico.yaml' saved [189916/189916]

[root@Master ~]#
[root@Master ~]# ls
anaconda-ks.cfg  calico.yaml
[root@Master ~]#
```

也可以使用如下附件，附件已完成地址修改和注释删除

calico.yaml

下载完成后可以输入 ls 查看是否存在 calico 的 yaml 文件

使用命令 vi calico.yaml 进行修改 yaml 文件，进入文件后先按“:”键，然后输入 set nu 后回车。

```
            "type": "calico",
            "log_level": "info",
            "log_file_path": "/var/log/calico/cni/cni.log",
            "datastore_type": "kubernetes",
            "nodename": "__KUBERNETES_NODE_NAME__",
            "mtu": __CNI_MTU__,
            "ipam": {
                "type": "calico-ipam"
            },
            "policy": {
                "type": "k8s"
            },
            "kubernetes": {
                "kubeconfig": "__KUBECONFIG_FILEPATH__"
            }
:set nu
```

回车后再按“:”键，然后输入 3684 后回车跳转至 3684 行。按“i”键后将 name 和 value 前面的注释删除，然后将 value 的地址修改为 10.244.0.0/16，配置结束后保存退出。

```
3679                  key: veth_mtu
3680            # The default IPv4 pool to create on startup if none exists. Pod IPs will be
3681            # chosen from this range. Changing this value after installation will have
3682            # no effect. This should fall within `--cluster-cidr`.
3683            - name: CALICO_IPV4POOL_CIDR
3684              value: "10.244.0.0/16"
3685            # Disable file logging so `kubectl logs` works.
3686            - name: CALICO_DISABLE_FILE_LOGGING
3687              value: "true"
```

在 Master 上输入 kubectl create -f calico.yaml 文件进行应用 yaml 文件

```
[root@Master ~]# kubectl create -f calico.yaml
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/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/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
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
serviceaccount/calico-node created
deployment.apps/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
poddisruptionbudget.policy/calico-kube-controllers created
[root@Master ~]#
```

如果应用过程中有报错需要撤销 yaml 文件的应用，需要输入 kubectl delete -f https://docs.projectcalico.org/v3.19/manifests/calico.yaml

输入 kubectl get pods -n kube-system 查看是否存在 calico 组件信息。

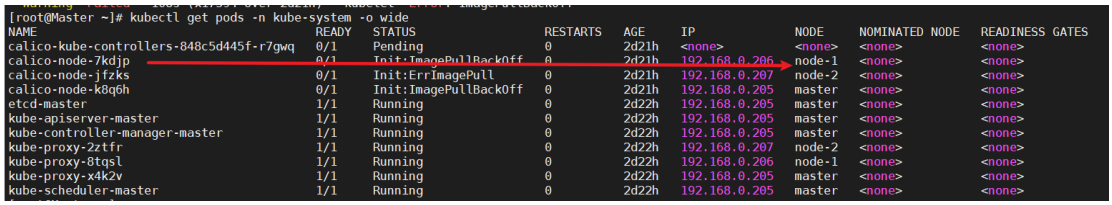
```
[root@Master ~]# kubectl get pods -n kube-system
NAME                                       READY   STATUS                  RESTARTS   AGE
calico-kube-controllers-848c5d445f-r7gwq   0/1     Pending                 0          115s
calico-node-7kdjp                          0/1     Init:ImagePullBackOff   0          114s
calico-node-jfzks                          0/1     Init:ImagePullBackOff   0          114s
calico-node-k8q6h                          0/1     Init:ImagePullBackOff   0          114s
coredns-7f89b7bc75-9mghl                   0/1     Pending                 0          51m
coredns-7f89b7bc75-d4nfr                   0/1     Pending                 0          51m
etcd-master                                1/1     Running                 0          51m
kube-apiserver-master                      1/1     Running                 0          51m
kube-controller-manager-master             1/1     Running                 0          51m
kube-proxy-2ztfr                           1/1     Running                 0          36m
kube-proxy-8tqsl                           1/1     Running                 0          37m
kube-proxy-x4k2v                           1/1     Running                 0          51m
kube-scheduler-master                      1/1     Running                 0          51m
[root@Master ~]#
```

注：coredns 组件状态为 Pengding 是因为 calico 组件没有正常 running，当 calico 组件正常 running 后，coredns 组件自动恢复正常。

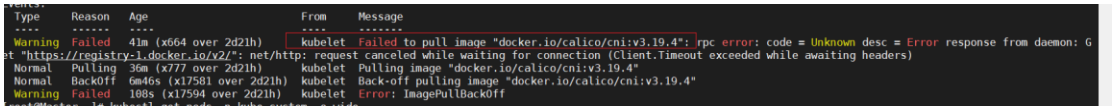
5.6.1 Calico 组件加载失败处理

如上所示，calico-node 组件的状态全部处于

Init:ImagePullBackOff 状态，以 calico-node-7kdjp 为例，输入 kubectl get pods -n kube-system -o wide 查看该组件在 Node-1 节点上。



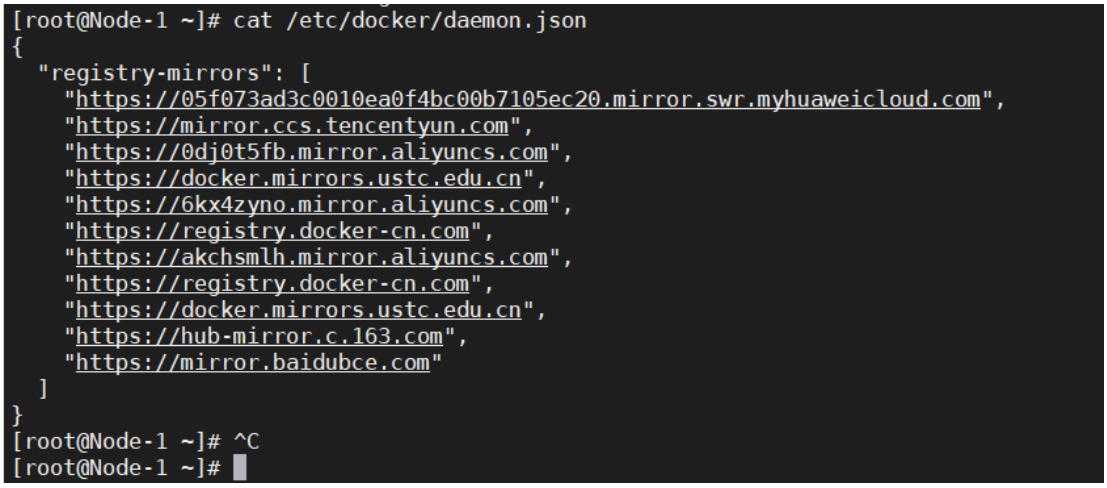
输入以下命令查看 calico-node-7kdjp 组件加载失败的原因 kubectl describe pod calico-node-7kdjp -n kube-system 可以看到失败的原因是无法自动拉取到镜像



登录 Node-1 节点虚机，输入 vi /etc/docker/daemon.json 输入以下内容

```
{
  "registry-mirrors": [

"https://05f073ad3c0010ea0f4bc00b7105ec20.mirror.swr.myhuaweicloud.com",
      "https://mirror.ccs.tencentyun.com",
      "https://0dj0t5fb.mirror.aliyuncs.com",
      "https://docker.mirrors.ustc.edu.cn",
      "https://6kx4zyno.mirror.aliyuncs.com",
      "https://registry.docker-cn.com",
```

```
    "https://akchsmlh.mirror.aliyuncs.com",
    "https://registry.docker-cn.com",
    "https://docker.mirrors.ustc.edu.cn",
    "https://hub-mirror.c.163.com",
    "https://mirror.baidubce.com"
  ]
}
```

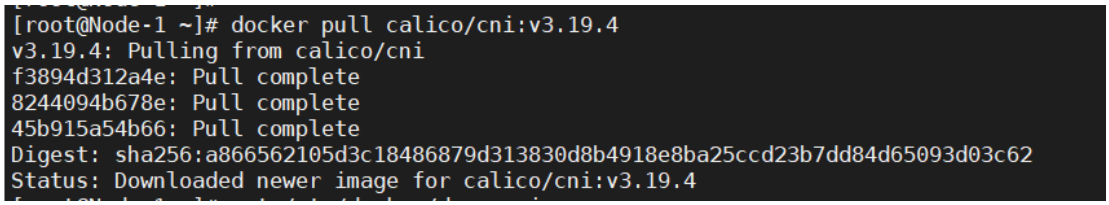


保存退出后输入以下两条命令进行重启 docker 进程

systemctl daemon-reload

systemctl restart docker

在 Node-1 节点上输入 docker pull calico/cni:v3.19.4 手动拉取 calico 镜像



在 Master 上输入 kubectl get pods -n kube-system -o

wide 可以看到 calico-node-7kdjp 组件状态已经是 Running 状态（需要等待 1-2 分钟启动时间）

剩下的 calico-node-jfzks 和 calico-node-k8q6h 处理方式一致。

```
[root@Master ~]# kubectl get pods -n kube-system -o wide
NAME                                READY   STATUS                  RESTARTS   AGE    IP               NODE     NOMINATED NODE   READINESS GATES
calico-kube-controllers-848c5d445f-r7gwq   1/1     Running                 0          2d22h  10.244.84.129    node-1   <none>           <none>
calico-node-7kdjp                   1/1     Running                 0          2d22h  192.168.0.206    node-1   <none>           <none>
calico-node-jfzks                   0/1     Init:ImagePullBackOff   0          2d22h  192.168.0.207    node-2   <none>           <none>
calico-node-k8q6h                   0/1     Init:ImagePullBackOff   0          2d22h  192.168.0.205    master   <none>           <none>
etcd-master                         1/1     Running                 0          2d23h  192.168.0.205    master   <none>           <none>
kube-apiserver-master               1/1     Running                 0          2d23h  192.168.0.205    master   <none>           <none>
kube-controller-manager-master      1/1     Running                 0          2d23h  192.168.0.205    master   <none>           <none>
kube-proxy-2ztfr                    1/1     Running                 0          2d23h  192.168.0.207    node-2   <none>           <none>
kube-proxy-8tqsl                    1/1     Running                 1          2d23h  192.168.0.206    node-1   <none>           <none>
kube-proxy-x4k2v                    1/1     Running                 0          2d23h  192.168.0.205    master   <none>           <none>
kube-scheduler-master               1/1     Running                 0          2d23h  192.168.0.205    master   <none>           <none>
[root@Master ~]#
```

确保所有组件状态启动正常

```
[root@Master ~]# kubectl get pods -n kube-system -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP               NODE     NOMINATED NODE   READINESS GATES
calico-kube-controllers-848c5d445f-f6c7h   1/1     Running   0          15m     10.244.219.67    master   <none>           <none>
calico-node-cknnd                   1/1     Running   0          15m     192.168.0.205    master   <none>           <none>
calico-node-xpt82                   1/1     Running   0          3m57s   192.168.0.207    node-2   <none>           <none>
calico-node-zjj4x                   1/1     Running   0          4m39s   192.168.0.206    node-1   <none>           <none>
coredns-7f89b7bc75-mkrgw            1/1     Running   0          16m     10.244.219.65    master   <none>           <none>
coredns-7f89b7bc75-njckg            1/1     Running   0          16m     10.244.219.66    master   <none>           <none>
etcd-master                         1/1     Running   0          16m     192.168.0.205    master   <none>           <none>
kube-apiserver-master               1/1     Running   0          16m     192.168.0.205    master   <none>           <none>
kube-controller-manager-master      1/1     Running   0          16m     192.168.0.205    master   <none>           <none>
kube-proxy-c8c88                    1/1     Running   0          3m57s   192.168.0.207    node-2   <none>           <none>
kube-proxy-gwxjr                    1/1     Running   0          4m39s   192.168.0.206    node-1   <none>           <none>
kube-proxy-txvlp                    1/1     Running   0          16m     192.168.0.205    master   <none>           <none>
kube-scheduler-master               1/1     Running   0          16m     192.168.0.205    master   <none>           <none>
[root@Master ~]#
```

5.7 查看 K8s 集群状态

在 Master 输入 kubectl get node 查看节点状态是否为 Ready

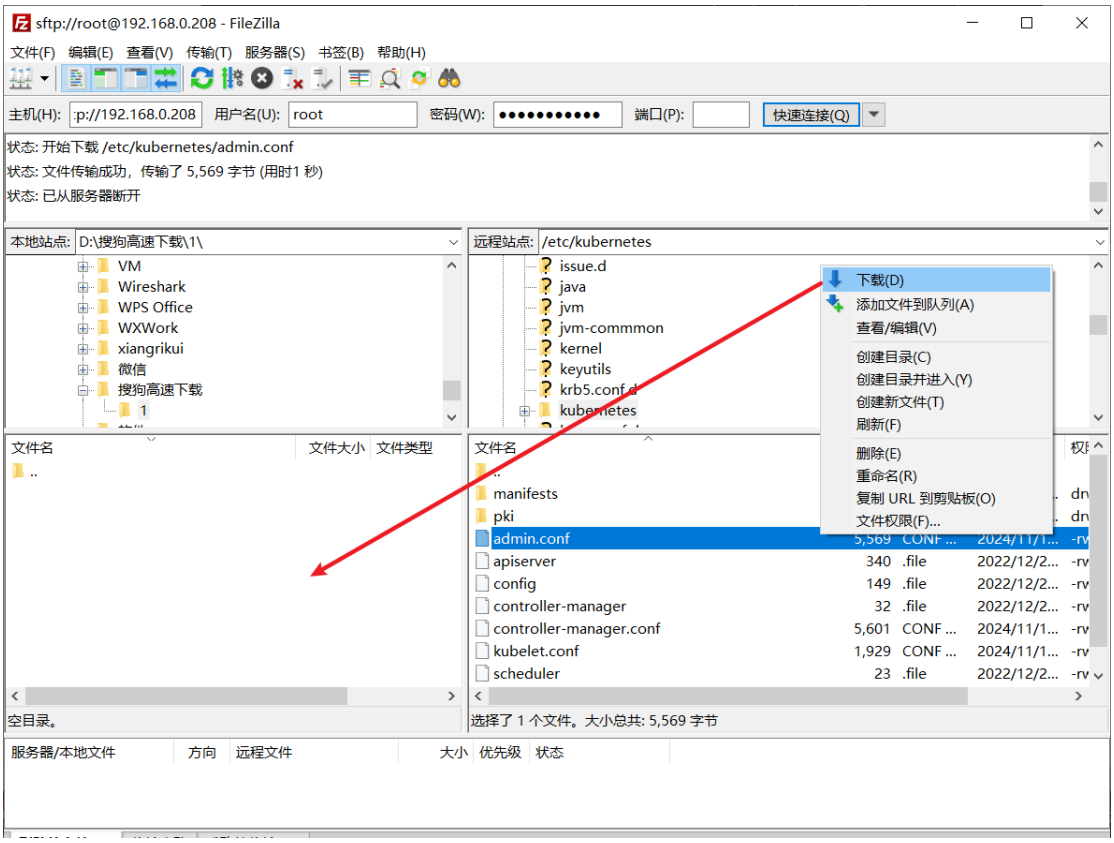
```
[root@Master ~]# kubectl get node
NAME     STATUS   ROLES                  AGE     VERSION
master   Ready    control-plane,master   18m     v1.20.2
node-1   Ready    <none>                 6m27s   v1.20.2
node-2   Ready    <none>                 5m45s   v1.20.2
[root@Master ~]#
```

5.8 Node 节点 K8S 命令报错处理

Node 节点输入 kubectl get node 或其他命令查看状态时会报错如下信息

```
[root@K8-Node1 ~]# kubectl get node
The connection to the server localhost:8080 was refused - did you specify the right host or port?
[root@K8-Node1 ~]# kubectl get all
The connection to the server localhost:8080 was refused - did you specify the right host or port?
[root@K8-Node1 ~]# kubectl get all
The connection to the server localhost:8080 was refused - did you specify the right host or port?
[root@K8-Node1 ~]# ^C
```

此报错原因是因为 Node 节点上没有 Master 的 amin.conf 环境变量，该变量位于/etc/kubernetes/路径下，使用 FTP 工具登录 Master 节点后台在对应的路径下手动下载 admin.conf 文件，然后上传到 Node 节点



在 Node 节点上输入 cd /etc/ kubernetes 然后输入 ls 查看是否存在 admin.conf 文件

```
[root@K8-Node1 ~]# cd /etc/kubernetes/
[root@K8-Node1 kubernetes]# ls
admin.conf  config  kubelet  kubelet.conf  kubelet.kubeconfig  manifests  pki  proxy
[root@K8-Node1 kubernetes]#
```

在 Node 节点上分别输入以下两条命令

echo export KUBECONFIG=/etc/kubernetes/admin.conf >>

~/.bash_profile

source ~/.bash_profile

```
[root@K8-Node1 kubernetes]# echo export KUBECONFIG=/etc/kubernetes/admin.conf >> ~/.bash_profile
[root@K8-Node1 kubernetes]#
[root@K8-Node1 kubernetes]# source ~/.bash_profile
```

完成配置后 Node 节点可以正常使用 kubectl get pod 命令

```
[root@K8-Node1 kubernetes]# kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
nginx-6799fc88d8-4w98h   1/1     Running   0          24m
[root@K8-Node1 kubernetes]#
```

六、测试发布 Nginx 应用

6.1 创建 nginx deployment

在 Master 上输入 kubectl create deployment nginx --image=nginx

```
[root@K8-Master ~]# kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
[root@K8-Master ~]#
```

6.2 创建 nginx service

在 Master 上输入 kubectl expose deployment nginx --port=80 --target-port=80 --type=NodePort

```
[root@K8-Master ~]# kubectl expose  deployment nginx --port=80 --target-port=80 --type=NodePort
service/nginx exposed
```

6.3 查看 IP 和端口

在 Master 上输入 kubectl get pods --all-namespaces -o wide 查看 Nginx 是否正常运行。

```
[root@K8-Master ~]# kubectl get pods --all-namespaces -o wide
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE     IP               NODE         NOMINATED NODE   READINESS GATES
default       nginx-6799fc88d8-zjc4c                     1/1     Running   0          4m12s   10.244.224.66    k8-node2     <none>           <none>
kube-system   calico-kube-controllers-848c5d445f-6t6wc   1/1     Running   0          93m     10.244.172.194   k8-node1     <none>           <none>
kube-system   calico-node-4fvwg                          1/1     Running   0          93m     192.168.0.208    k8-master    <none>           <none>
kube-system   calico-node-hp285                          1/1     Running   0          93m     192.168.0.211    k8-node2     <none>           <none>
kube-system   calico-node-vv5nm                          1/1     Running   0          93m     192.168.0.209    k8-node1     <none>           <none>
kube-system   coredns-7f89b7bc75-664pb                   1/1     Running   0          97m     10.244.172.193   k8-node1     <none>           <none>
kube-system   coredns-7f89b7bc75-bmx2n                   1/1     Running   0          97m     10.244.172.195   k8-node1     <none>           <none>
kube-system   etcd-k8-master                             1/1     Running   0          97m     192.168.0.208    k8-master    <none>           <none>
kube-system   kube-apiserver-k8-master                   1/1     Running   0          97m     192.168.0.208    k8-master    <none>           <none>
kube-system   kube-controller-manager-k8-master          1/1     Running   0          97m     192.168.0.208    k8-master    <none>           <none>
kube-system   kube-proxy-2gnnf                           1/1     Running   0          97m     192.168.0.208    k8-master    <none>           <none>
kube-system   kube-proxy-c476w                           1/1     Running   0          95m     192.168.0.211    k8-node2     <none>           <none>
kube-system   kube-proxy-dsv82                           1/1     Running   0          96m     192.168.0.209    k8-node1     <none>           <none>
kube-system   kube-scheduler-k8-master                   1/1     Running   0          97m     192.168.0.208    k8-master    <none>           <none>
[root@K8-Master ~]#
```

在 Master 上输入 kubectl get service 查看访问的端口为 32111

```
[root@K8-Master ~]# kubectl get service
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.1.0.1       <none>        443/TCP        97m
nginx        NodePort    10.1.131.46    <none>        80:32111/TCP   14s
[root@K8-Master ~]#
```

6.4 访问 WEB 页面

在浏览器输入任意 Master/Node 的 ip 加端口号访问页面