集群环境部署整理

目录

1 集群环境部署

1.1 集群环境的搭建

1.1.1 虚拟机映像下载

https://mirror.iscas.ac.cn/centos/7/isos/x86_64/CentOS-7-x86_64-DVD-2009.iso
此处下载 Linux 虚拟机

1.1.2 主机安装

主要步骤如下
1，参考虚拟机配置此处的 IP 地址要通过前 24 位需要通过查看 VMware 的设置来进行。

作用	IP地址	操作系统	配置
Master	192.168.109.101	Centos7.5 基础设施服务器	2颗CPU 2G内存 50G硬盘
Node1	192.168.109.102	Centos7.5 基础设施服务器	2颗CPU 2G内存 50G硬盘
Node2	192.168.109.103	Centos7.5 基础设施服务器	2颗CPU 2G内存 50G硬盘

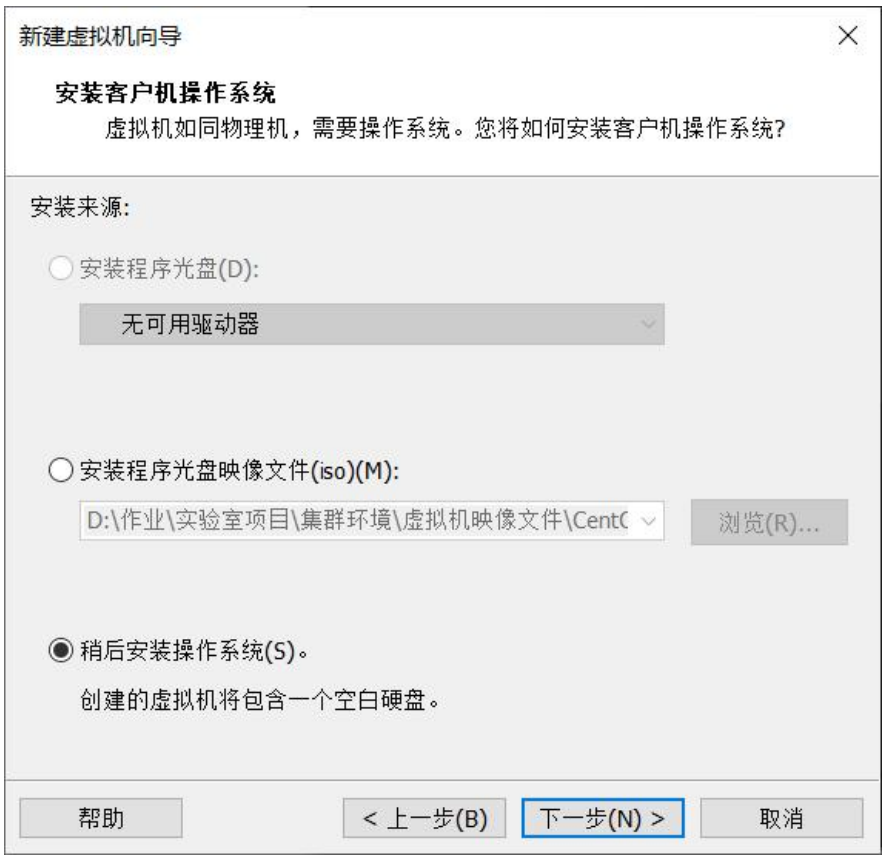
在此处查看 Net 模式的地址 192.168.42.0，前 24 位不动，后面八位可以自定义，建议使用 3 个连续的数字便于记忆。

2，创建虚拟机

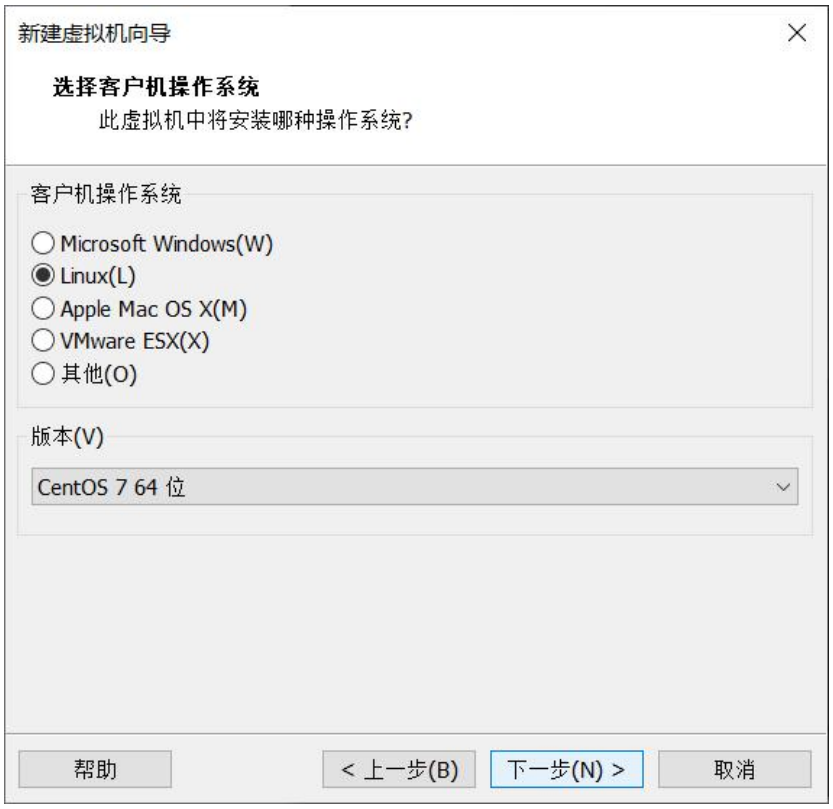
（1）点击 VMWare 创建虚拟机，并选择自定义创建，并选择下一步

（2）硬件兼容性选择默认

（3）选择稍后安装操作系统

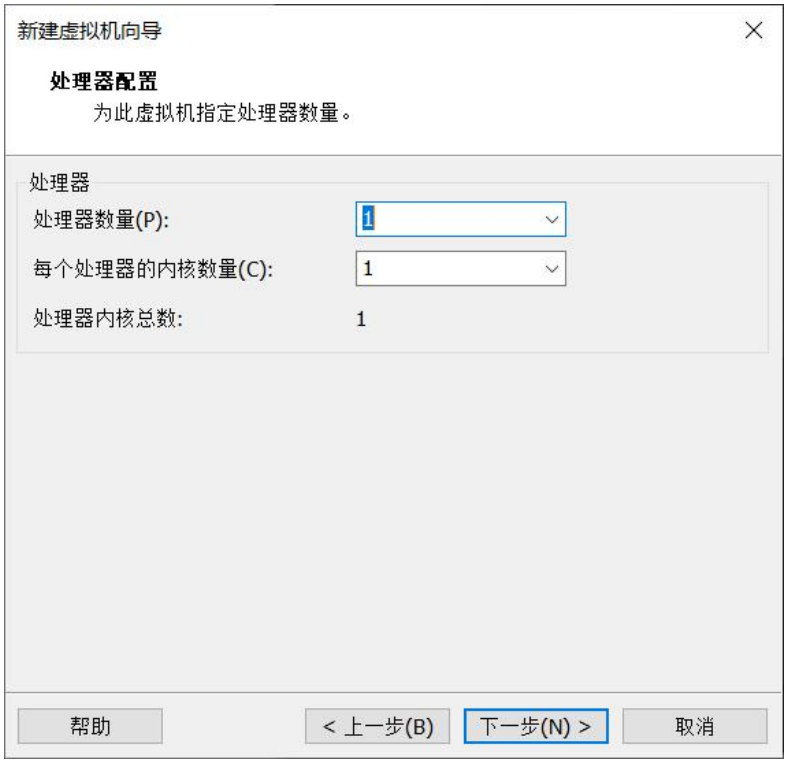


（4）选择如图所示对应的版本号



（5）选择存储位置

（6）选择处理机数量

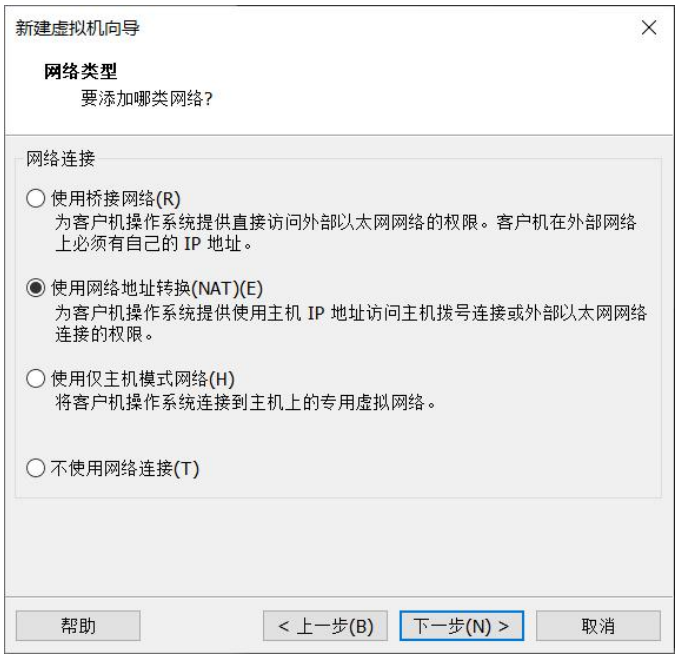


此处 master 节点使用 1 个处理器，4 个内核。node 工作节点使用 1 个处理器，

两个内核。

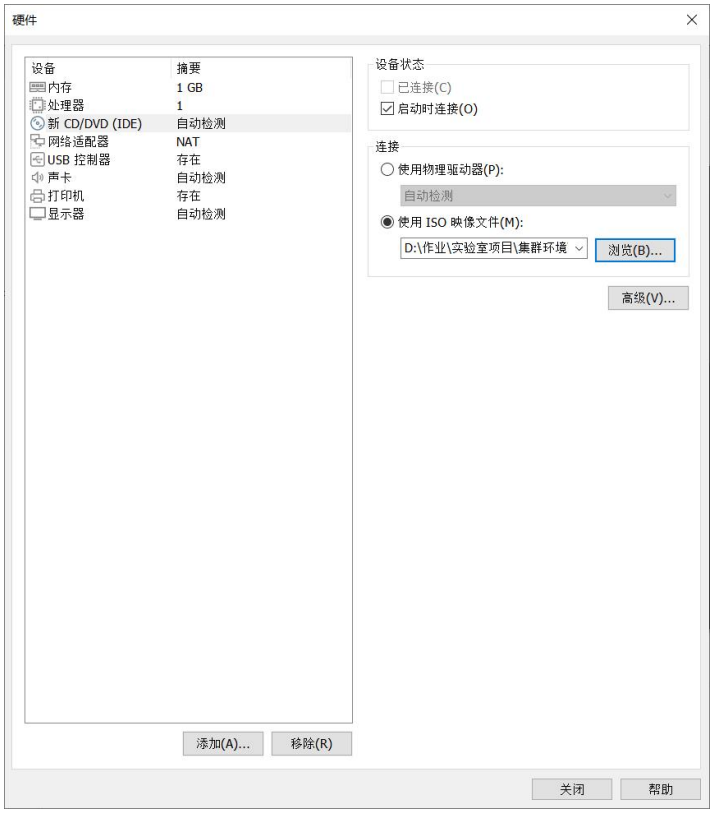
（7）内存选择
master 节点选用 8GB,node 节点选用 4GB

（8）选择添加网络类型，选择默认的 NAT



（9）之后一直保存默认进行下一步即可。

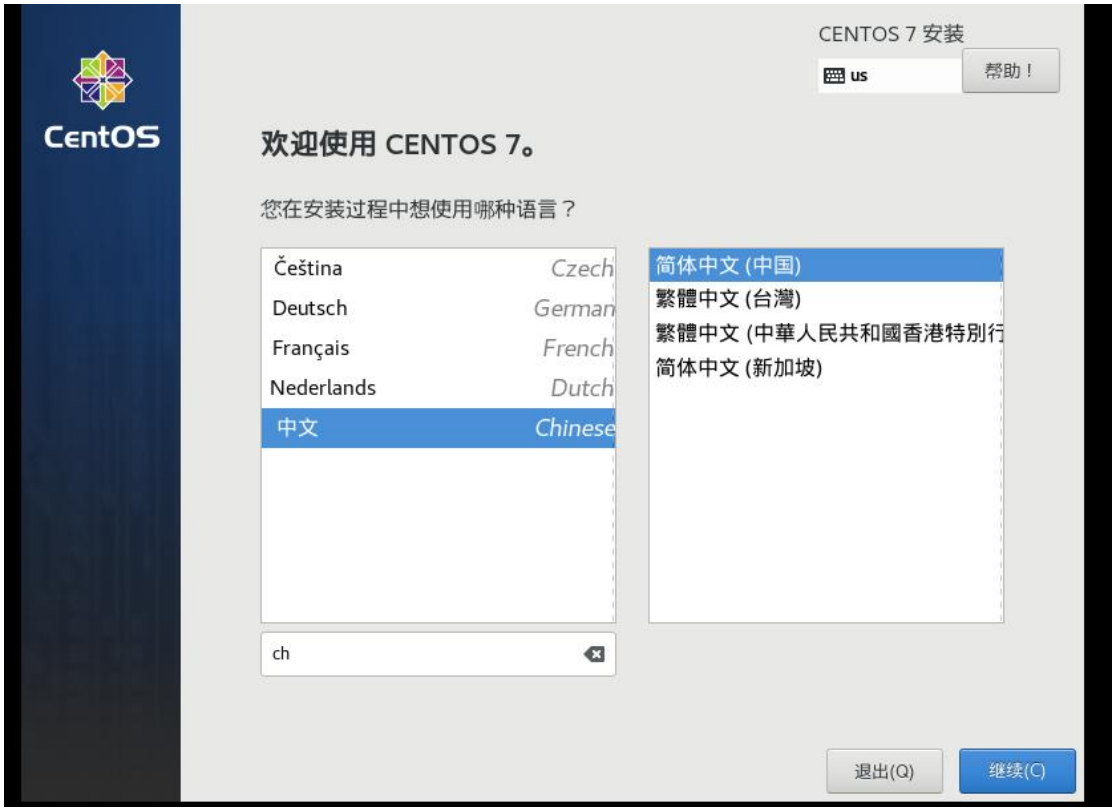
（10）最后在此处，点击自定义硬件，并将虚拟机映像加入即可完成创建。



3，虚拟机配置

进入虚拟机安装页面后进行如下配置

（1）选择简体中文，点击继续



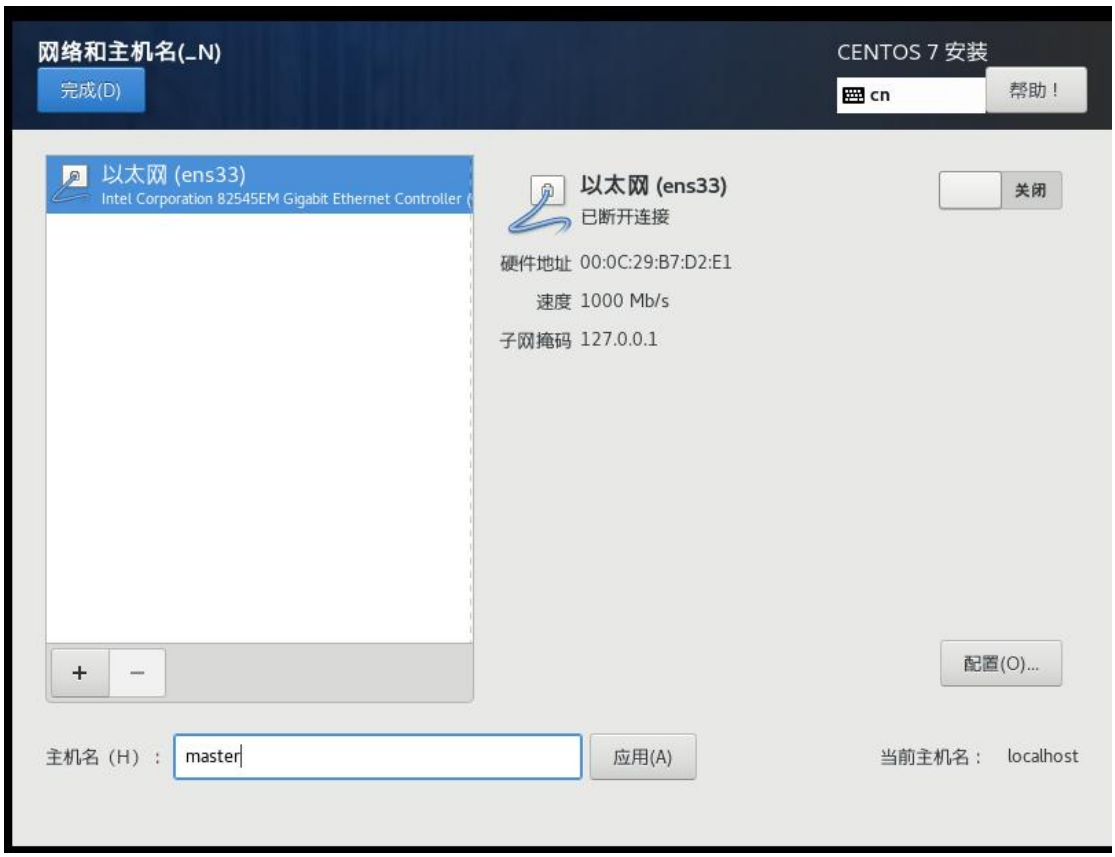
（2）点击软件选择

（3）选择基础设施服务器，点击完成

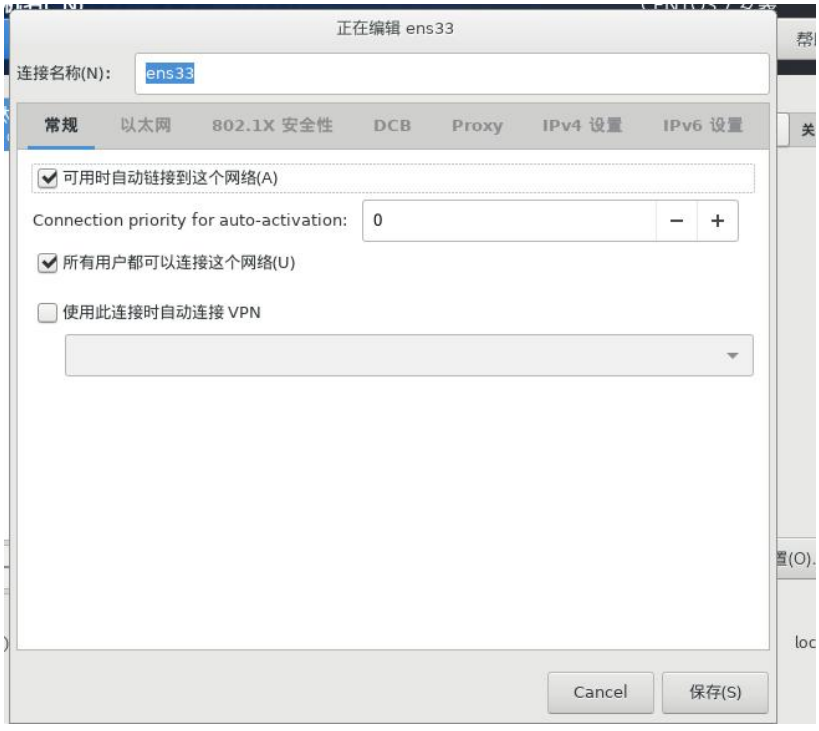
（4）点击一下安装位置，并点击完成，可以解锁开始安装按钮

（5）最后最重点的是，点击网络和主机名来配置

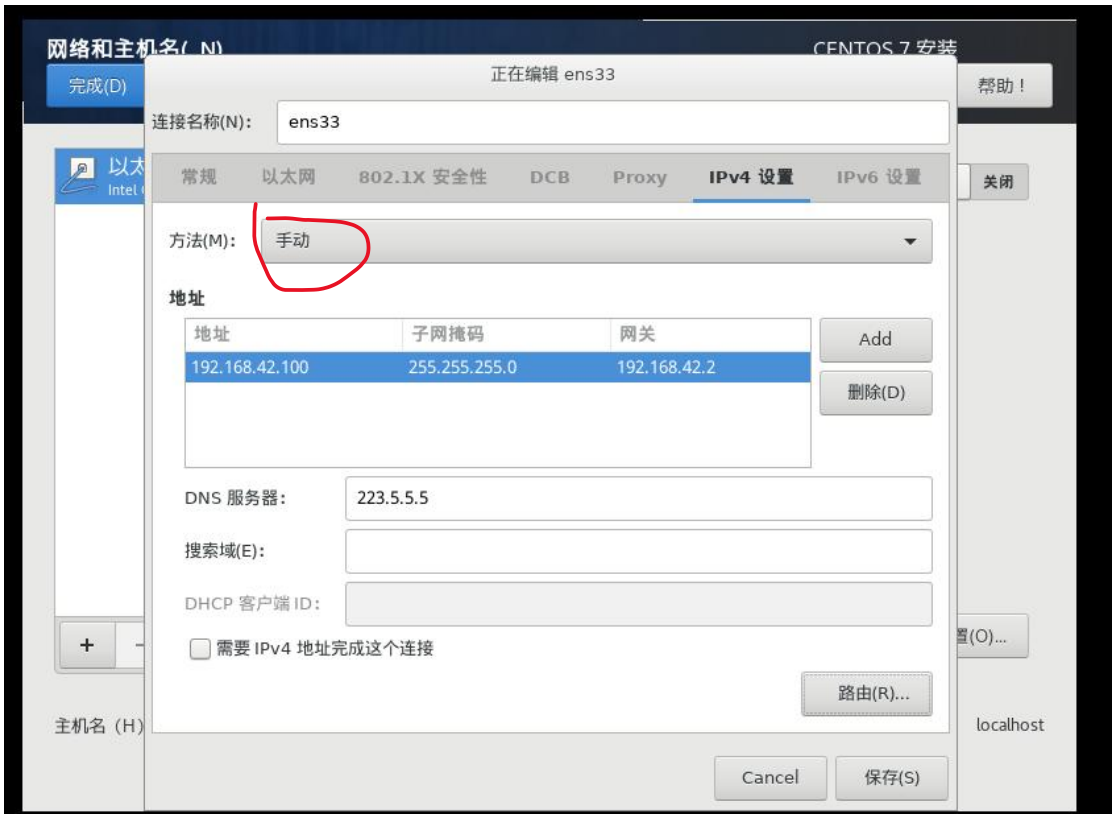
（6）此处主机名选择 master（其余 node1,node2 同理），并点击应用（**不要忘记**）



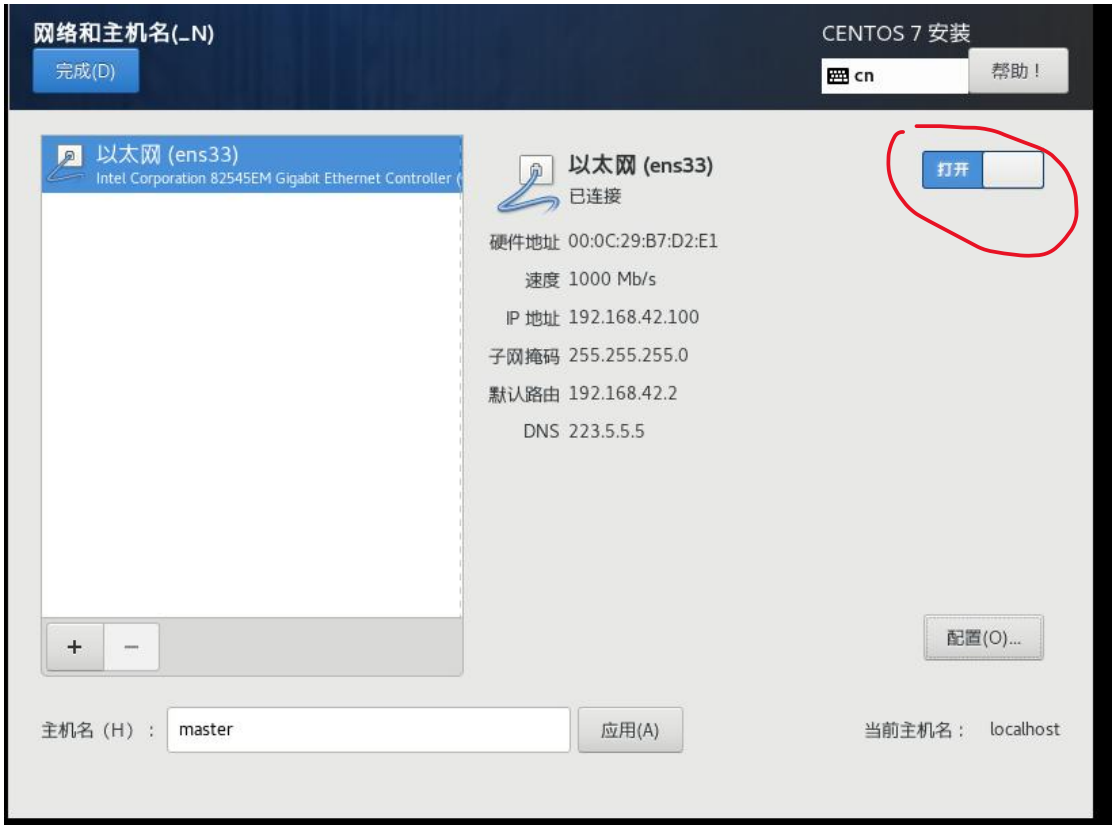
（7）点击配置后，点击常规
此处可用时自动链接到这个网络一定要勾选



（8）点击 Ipv4 配置，网络地址按照之间所使用的方法来进行，其余的可以参考下面配置，方法一定要选择为手动，点击保存即可。



（9）最后一定要回归到该界面检查以太网是否打开



（10）点击完成即可开始安装。
其余的工作节点 node1 与 node2 可以按照上述方式来进行同样的配置即可。
需要修改的就是处理器的核数，IP 地址及主机名。

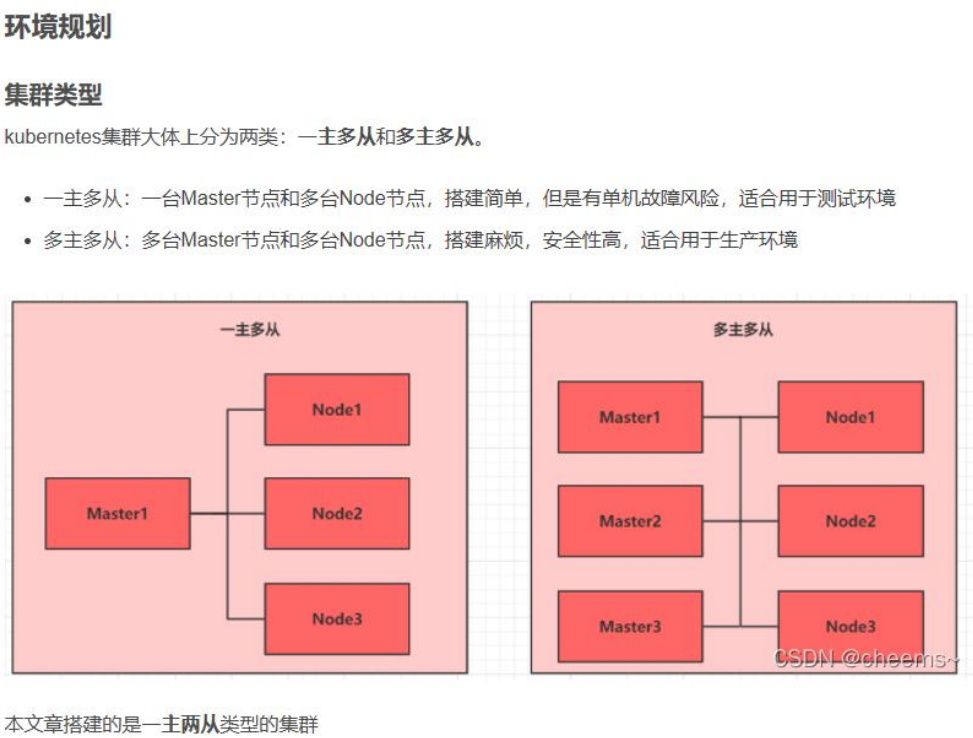
（11）**此处安装完成后，一定一定要保存快照，不然后续出问题会导致该部分需要重做。**

容易出现的问题：
1，虚拟机的网络地址一定要查看 VMware 的配置，且是下方的 NAT 地址，不要搞错。
2，一定要在常规处选择可用时自动链接，特别容易遗漏。
3，最后要检查以太网的连接正常打开

参考文章：
https://www.bilibili.com/video/BV1Qv41167ck?p=5&vd_source=1f2ad75c0cef9ebeb2ad36a3c472f77f

1.1.3 Kubernetes—集群环境搭建

1，环境规划
具体见下图



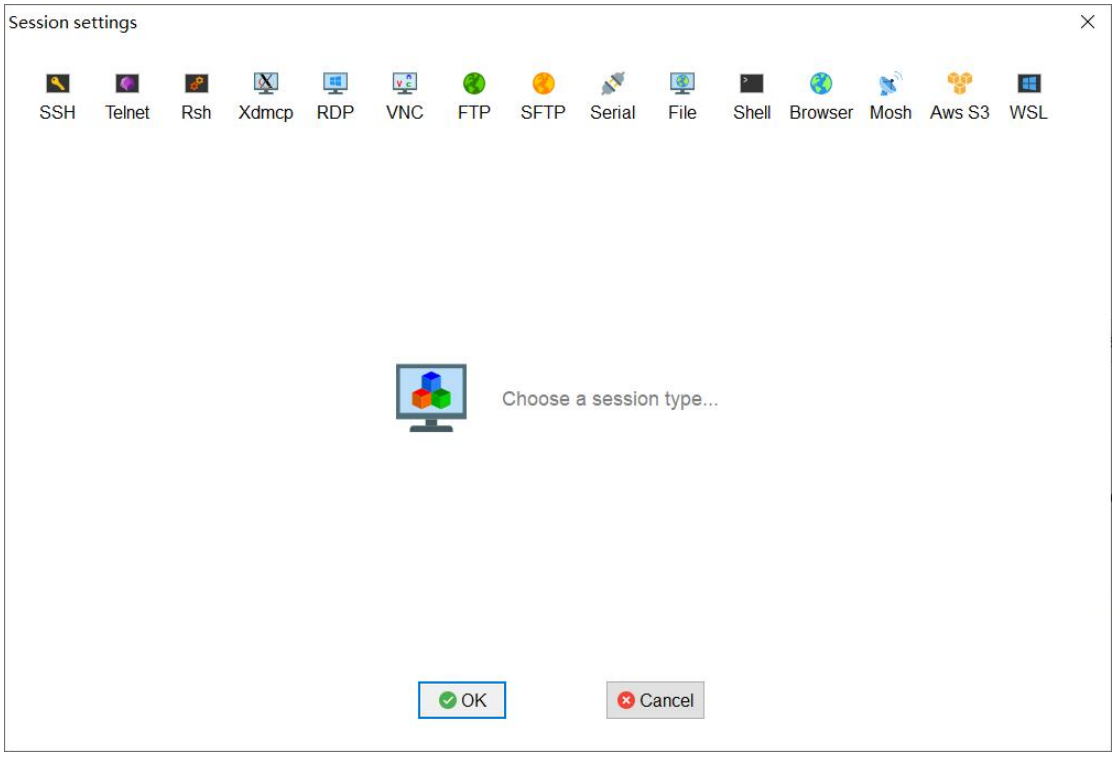
主要目标
然后在每台服务器中分别安装 docker、kubeadm、kubelet、kubectl 程序

2，工具准备

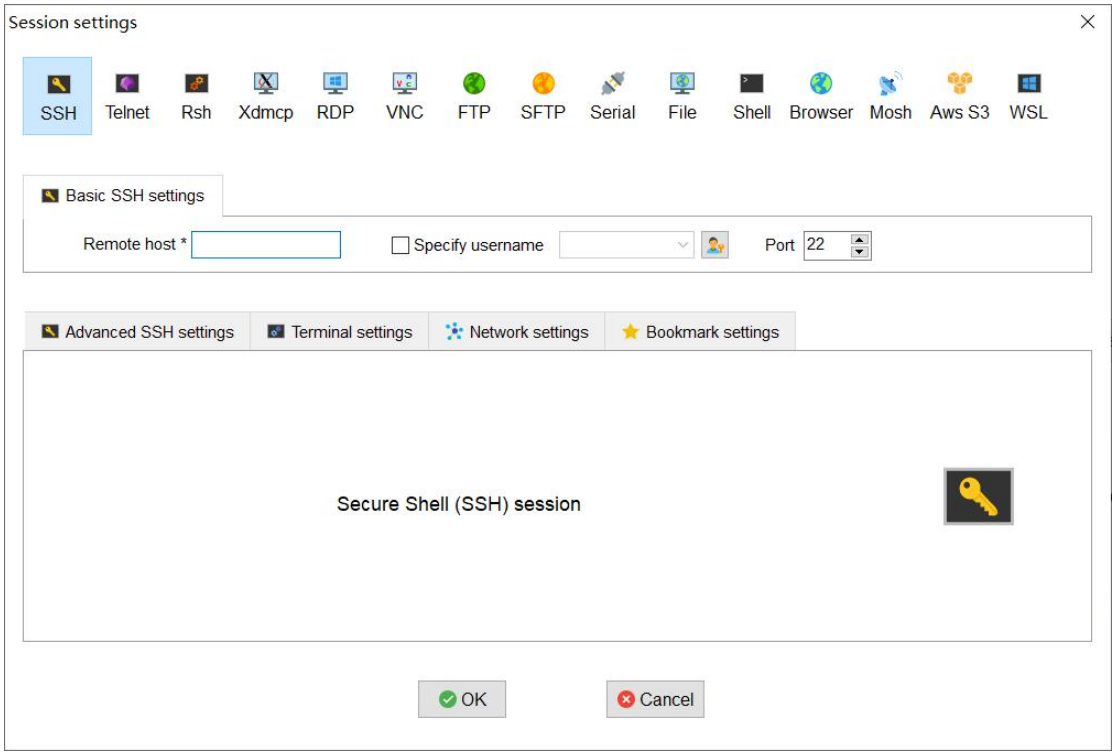
主要的准备工具有 MobaXterm,也可以使用其他工具，但不建议直接在虚拟机上进行操作很容易出现问题。

重点注意：MobaXterm 在复制多行或者大量的命令时候，很容易出现首行丢失一定指令的情况。所以在复制多行一定要记得查看首行是否出现错误。

（1）点击左上角 session，出现如下界面，并点击 SSH



（2）输入之前的地址及自己的账号密码来新建会话



（3）点击 MultiExec 模式

则可以在三台主机上同时进行命令行操作。

（4）命令行一些常用的命令

:q 是退出

:wq 是保存后退出

i 表示进入编辑模式

3，主机初始化

（1）检查操作系统版本

```
# 此方式下安装 kubernetes 集群要求 Centos 版本要在 7.5 或之上
[root@master ~]# cat /etc/redhat-release
CentOS Linux release 7.5.1804 (Core)
```

（2）主机名解析，主要是为了方便后面的集群调用

```
# 编辑三台服务器的/etc/hosts 文件
vi /etc/hosts
# 添加下面内容,注意如果你的 ip 跟我不一样，对应的改一下即可
192.168.42.100  master
192.168.42.101  node1
192.168.42.102  node2
```

（3）时间同步，使用 date 来查看并验证

```
# 启动 chronyd 服务
[root@master ~]# systemctl start chronyd
# 设置 chronyd 服务开机自启
[root@master ~]# systemctl enable chronyd
# chronyd 服务启动稍等几秒钟，就可以使用 date 命令验证时间了
[root@master ~]# date
2023 年 07 月 09 日 星期日 15:24:30 CST
```

（4）禁用 iptables 和 firewalld 服务，主要避免与系统规则的混淆

```
# 1 关闭 firewalld 服务
[root@master ~]# systemctl stop firewalld
[root@master ~]# systemctl disable firewalld
# 2 关闭 iptables 服务
[root@master ~]# systemctl stop iptables
[root@master ~]# systemctl disable iptables
```

（5）禁用 selinux

selinux 是 linux 系统下的一个安全服务，如果不关闭它，在安装集群中会产生各种各样的奇葩问题

```
# 编辑 /etc/selinux/config 文件，修改 SELINUX 的值为 disabled
# 注意修改完毕之后需要重启 linux 服务，稍后重启
vi /etc/selinux/config
# 找到 SELINUX=enforcing，改成 SELINUX=disabled
# 测试是否关闭，显示 Disabled 为关闭成功，即重启之后才会成功
[root@master ~]# getenforce
Disabled
```

（6）禁用 swap 分区

swap 分区指的是虚拟内存分区，它的作用是在物理内存使用完之后，将磁盘空间虚拟成内存来使用。启用 swap 设备会对系统的性能产生非常负面的影响，因此 kubernetes 要求每个节点都要禁用 swap 设备。但是如果因为某些原因确实不能关闭 swap 分区，就需要在集群安装过程中通过明确的参数进行配置说明。

```
# 编辑分区配置文件/etc/fstab，注释掉 swap 分区一行
# 注意修改完毕之后需要重启 linux 服务
vi /etc/fstab
# 将下面这一行，可以看到是 swap，前面加个#，注释掉
/dev/mapper/centos-swap swap                    swap    defaults        0 0
# 变成下面这样子，就完成了，稍后再重启
# /dev/mapper/centos-swap swap                    swap    defaults        0
0
```

重点：此处与教程不同,参考文章中 etc 前面少了一个/注意修改

（7）修改 linux 的内核参数

```
# 修改 linux 的内核参数，添加网桥过滤和地址转发功能
# 编辑/etc/sysctl.d/kubernetes.conf 文件
vi /etc/sysctl.d/kubernetes.conf
#添加如下配置:
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
# 重新加载配置
[root@master ~]# sysctl -p
# 加载网桥过滤模块
[root@master ~]# modprobe br_netfilter
# 查看网桥过滤模块是否加载成功
[root@master ~]# lsmod | grep br_netfilter
br_netfilter          22256  0
bridge               146976  1 br_netfilter
```

此处同样注意 etc 前面需要有/

（8）配置 ipvs 功能

在 kubernetes 中 service 有两种代理模型，一种是基于 iptables 的，一种是基于 ipvs 的。两者比较的话，ipvs 的性能明显要高一些，但是如果要使用它，需要手动载入 ipvs 模块

```
# 1 安装 ipset 和 ipvsadm
[root@master ~]# yum install ipset ipvsadmin -y

# 2 添加需要加载的模块写入脚本文件
[root@master ~]# cat <<EOF >  /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_ipv4
EOF

# 3 为脚本文件添加执行权限
[root@master ~]# chmod +x /etc/sysconfig/modules/ipvs.modules

# 4 执行脚本文件
[root@master ~]# /bin/bash /etc/sysconfig/modules/ipvs.modules

# 5 查看对应的模块是否加载成功
```

```
[root@master ~]# lsmod | grep -e ip_vs -e nf_conntrack_ipv4
nf_conntrack_ipv4      15053  9
nf_defrag_ipv4         12729  1 nf_conntrack_ipv4
ip_vs_sh               12688  0
ip_vs_wrr              12697  0
ip_vs_rr               12600  0
ip_vs                 141432  6 ip_vs_rr,ip_vs_sh,ip_vs_wrr
nf_conntrack                                          133053         9
ip_vs,nf_nat,nf_nat_ipv4,nf_nat_ipv6,xt_conntrack,nf_nat_masquerade_ipv
4,nf_nat_masquerade_ipv6,nf_conntrack_ipv4,nf_conntrack_ipv6
libcrc32c              12644  4 xfs,ip_vs,nf_nat,nf_conntrack
```

（9）重启服务器

上面步骤完成之后，需要重新启动 linux 系统

```
[root@master ~]# reboot
```

按 R 即可完成重连

4，安装 docker

三台主机均需要安装 docker，因此需要使用之前的多种执行模式，节约时间

注意：这种模式下有时候第一台主机的复制粘贴会出现问题，记得每次都要检查一下

```
# 1 切换镜像源,换成阿里的
[root@master                              ~]#                              wget
https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo       -O
/etc/yum.repos.d/docker-ce.repo

# 2 安装 docker，我这里直接安装最新版的，不多 bb
[root@master ~]# yum install docker -y

# 3 修改配置文件，Docker 在默认情况下使用的 Cgroup Driver 为 cgroupfs
# 而 kubernetes 推荐使用 systemd 来代替 cgroupfs
[root@master ~]# cd /etc/docker
[root@master ~]# cat <<EOF >  /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "registry-mirrors": ["https://kn0t2bca.mirror.aliyuncs.com"]
}
EOF

# 注意了，这里直接启动 docker 会报错，因为 docker.service 里有一条配置
# 和刚才添加的"exec-opts"冲突了，下面是解决方案

# 4 解决报错
```

```
# vi /lib/systemd/system/docker.service
# 找到并删除下面这句话，保存退出，即可解决
# --exec-opt native.cgroupdriver=cgroupfs \

# 5 启动 docker
[root@master ~]# systemctl daemon-reload
[root@master ~]# systemctl restart docker
[root@master ~]# systemctl enable docker

# 6 检查 docker 状态和版本，能检查到就说明你安装启动好了
[root@master ~]# docker version
[root@master ~]# systemctl status docker
```

5，安装 K8s 的三个组件（同样三台均要）

```
# 由于 kubernetes 的镜像源在国外，速度比较慢，这里切换成国内的镜像源
# 编辑/etc/yum.repos.d/kubernetes.repo
vi /etc/yum.repos.d/kubernetes.repo

#添加下面的配置
[kubernetes]
name=Kubernetes
baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x
86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
       http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

# 安装 kubeadm、kubelet 和 kubectl
[root@master ~]# yum install kubeadm kubelet kubectl -y

# 配置 kubelet 的 cgroup
# 编辑/etc/sysconfig/kubelet
vi /etc/sysconfig/kubelet
# 添加下面的配置
KUBELET_CGROUP_ARGS="--cgroup-driver=systemd"
KUBE_PROXY_MODE="ipvs"

# 设置 kubelet 开机自启
[root@master ~]# systemctl enable kubelet
```

6，准备集群镜像

```
# 在安装 kubernetes 集群之前，必须要提前准备好集群需要的镜像，所需镜像可以通过下面命令查看
[root@master ~]# kubeadm config images list
k8s.gcr.io/kube-apiserver:v1.23.4
k8s.gcr.io/kube-controller-manager:v1.23.4
k8s.gcr.io/kube-scheduler:v1.23.4
k8s.gcr.io/kube-proxy:v1.23.4
k8s.gcr.io/pause:3.6
k8s.gcr.io/etcd:3.5.1-0
k8s.gcr.io/coredns/coredns:v1.8.6
# 下载镜像
# 此镜像在 kubernetes 的仓库中,由于网络原因,无法连接，下面提供了一种替代方案
# 下载国内的镜像然后给镜像改名，如果你看到此博文安装时
# list 里面的版本与我不同，直接修改下面的 image 的版本即可，其他不要动
# 直接在 bash 里面复制粘贴下面的语句即可
images=(
    kube-apiserver:v1.23.4
    kube-controller-manager:v1.23.4
    kube-scheduler:v1.23.4
    kube-proxy:v1.23.4
    pause:3.6
    etcd:3.5.1-0
    coredns:v1.8.6
)
for imageName in ${images[@]} ; do
    docker pull registry.cn-hangzhou.aliyuncs.com/google_containers/$imageName
    docker tag registry.cn-hangzhou.aliyuncs.com/google_containers/$imageName k8s.gcr.io/$imageName
    docker rmi registry.cn-hangzhou.aliyuncs.com/google_containers/$imageName
done
# 注意 coredns 组件路径是 k8s.gcr.io/coredns/coredns:v1.8.6，而不是 k8s.gcr.io/coredns:v1.8.6
# 所以这一个手动修改一下
docker tag k8s.gcr.io/coredns:v1.8.6 k8s.gcr.io/coredns/coredns:v1.8.6
docker rmi k8s.gcr.io/coredns:v1.8.6
```

7，master 初始化（只需要在 master 节点进行即可）此处的地址记得修改为自定义的 master 地址

```
# 创建集群
# 第一个参数是 k8s 的版本，第二个是 pod 的网段，第三个是 service 的网段，第四个是
```

```
master 的地址
[root@master ~]# kubeadm init \
    --kubernetes-version=v1.23.4 \
    --pod-network-cidr=10.244.0.0/16 \
    --service-cidr=10.96.0.0/12 \
    --apiserver-advertise-address=192.168.109.100

# 出现下面这句话就说明安装成功了
Your Kubernetes control-plane has initialized successfully !

#注意安装完成后的提示信息最后几行，有个 token，node 加入集群时会用到
kubeadm join 192.168.109.100:6443 --token jh2hn8.hvebbwqlp9rqd5ri \
    --discovery-token-ca-cert-hash
sha256:d4c63e4449affefd5755ce880bfbd3ae66cc2f8ac8b9b28408f038c622d53000

# 创建必要文件
[root@master ~]# mkdir -p $HOME/.kube
[root@master ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@master ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

8，node 加入集群（该操作只在 node 主机下进行）

```
# 将 node 节点加入集群
[root@master ~]# kubeadm join 192.168.109.100:6443 --token
jh2hn8.hvebbwqlp9rqd5ri \
    --discovery-token-ca-cert-hash
sha256:d4c63e4449affefd5755ce880bfbd3ae66cc2f8ac8b9b28408f038c622d53000

# 查看集群状态 此时的集群状态为 NotReady，这是因为还没有配置网络插件
[root@master ~]# kubectl get nodes
NAME     STATUS     ROLES    AGE     VERSION
master   NotReady   master   6m43s   v1.17.4
node1    NotReady   <none>   22s     v1.17.4
node2    NotReady   <none>   19s     v1.17.4
```

9，安装网络插件（只在 master 主机上进行）

```
# 获取 fannel 的配置文件
[root@master ~]# wget
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k
ube-flannel.yml

# 使用配置文件启动 fannel
[root@master ~]# kubectl apply -f kube-flannel.yml
```

```
# 稍等片刻，再次查看集群节点的状态
[root@master ~]# kubectl get nodes
NAME     STATUS   ROLES                  AGE     VERSION
master   Ready    control-plane,master   4h49m   v1.23.4
node1    Ready    <none>                 4h48m   v1.23.4
node2    Ready    <none>                 4h47m   v1.23.4
```

10，命令补全工具安装

```
[root@master ~]# yum -y install bash-completion
[root@master ~]# source /usr/share/bash-completion/bash_completion
[root@master ~]# source <(kubectl completion bash)
[root@master ~]# echo "source <(kubectl completion bash)" >> ~/.bashrc
```

11，环境测试

接下来在 kubernetes 集群中部署一个 nginx 程序，测试下集群是否在正常工作。

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

# 暴露端口
[root@master ~]# kubectl expose deployment nginx --port=80 --type=NodePort
service/nginx exposed

# 查看服务状态
[root@master ~]# kubectl get pods,service
NAME                         READY   STATUS    RESTARTS   AGE
pod/nginx-85b98978db-sh6cs   1/1     Running   0          38s

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
service/kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP
4h55m
service/nginx        NodePort    10.108.112.185   <none>
80:31325/TCP   10s
```

12，测试环境，访问 nginx 服务则出现如下结果，则表示成功

13，较易错点

（1）注意路径，参考文章中可能会有/etc 与 etc 路径错误的问题，一定要仔细查看。

（2）版本问题，安装安装 kubectl 和 kubeadm，不要装最新版本的，有极大概率会出现版本不兼容问题，导致后续安装失败。推荐都是用 1.23.4 版本，测试可以成功兼容。

（3）此处完成后注意一定保存镜像。该部分可能是出现错误最多的部分，做完后一定注意保存。

参考文章：

https://blog.csdn.net/qq_42956653/article/details/123183215

2 在集群上安装部署 istio1.14

2.1 版本对照表

版本	目前支持	发行日期	停止维护	支持的 Kubernetes 版本	未测试，可能支持的 Kubernetes 版本
master	否，仅限开发	-	-	-	-
1.15	是	2022 年 8 月 31 日	~ 2023 年 3 月（预期）	1.22, 1.23, 1.24, 1.25	1.16, 1.17, 1.18, 1.19, 1.20, 1.21
1.14	是	2022 年 5 月 24 日	~ 2023 年 1 月（预期）	1.21, 1.22, 1.23, 1.24	1.16, 1.17, 1.18, 1.19, 1.20
1.13	是	2022 年 2 月 11 日	~ 2022 年 10 月（预期）	1.20, 1.21, 1.22, 1.23	1.16, 1.17, 1.18, 1.19
1.12	是	2021 年 11 月 18 日	2022 年 7 月 12 日	1.19, 1.20, 1.21, 1.22	1.16, 1.17, 1.18
1.11	否	2021 年 8 月 12 日	2022 年 3 月 25 日	1.18, 1.19, 1.20, 1.21, 1.22	1.16, 1.17
1.10	否	2021 年 5 月 18 日	2022 年 1 月 7 日	1.18, 1.19, 1.20, 1.21	1.16, 1.17, 1.22
1.9	否	2021 年 2 月 9 日	2021 年 10 月 8 日	1.17, 1.18, 1.19, 1.20	1.15, 1.16
1.8	否	2020 年 11 月 10 日	2021 年 5 月 12 日	1.16, 1.17, 1.18, 1.19	1.15
1.7	否	2020 年 8 月 21 日	2021 年 2 月 25 日	1.16, 1.17, 1.18	1.15
1.6 及更早	否	-	-	-	-

2.2 安装 Istio

（1）下载 Istio CLI（istioctl）、安装清单、示例和工具
此处为下载最新版本

```
[root@k8scloude1 ~]# curl -L https://istio.io/downloadIstio | sh -
```

此处为下载指定版本

```
[root@k8scloude1 ~]# curl -L https://istio.io/downloadIstio | ISTIO_VERSION=1.14.3 TARGET_ARCH=x86_64 sh -
```

（2）下载完之后添加至 path(此处一定要注意重点查看路径)

```
root@k8scloude1 bin]# pwd /root/istio-1.14.3/bin
[root@k8scloude1 bin]# ls istioctl #临时生效
[root@k8scloude1 bin]# export PATH=/root/istio-1.14.3/bin:$PATH #永久生效
```

```
[root@k8scloude1 bin]# vim /etc/profile.d/istioctl.sh
[root@k8scloude1 bin]# cat /etc/profile.d/istioctl.sh export ISTIOCTL_HOME="/root/istio-1.14.3" export PATH="$ISTIOCTL_HOME/bin:$PATH"
[root@k8scloude1 bin]# source /etc/profile.d/istioctl.sh
```

（3）查看版本

```
[root@k8scloude1 bin]# istioctl version no running Istio pods in "istio-system" 1.14.3
```

（4）了解内置组件

标注 ✓ 的组件安装在每个配置文件中：

	default	demo	minimal	remote	empty	preview
核心组件						
istio-egressgateway		✓				
istio-ingressgateway	✓	✓				✓
istiod	✓	✓	✓			✓

推荐用于生产部署的配置文件是 default 配置文件。

我们将安装 demo 配置文件，因为它包含所有的核心组件，启用了跟踪和日志记录，便于学习不同的 Istio 功能。

（5）部署 Istio Operator

```
[root@k8scloude1 bin]# istioctl operator init
Installing operator controller in namespace: istio-operator using image: docker.io/istio/operator:1.14.3 Operator controller will watch namespaces: istio-system ✔ Istio operator installed ✔ Installation complete
```

（6）创建配置文件

此处同样注意记录路径！

```
#创建目录 istioyaml，用来专门存放 yaml 文件
[root@k8scloude1 ~]# mkdir istioyaml

[root@k8scloude1 ~]# cd istioyaml/

[root@k8scloude1 istioyaml]# vim istio-demo-profile.yaml

#profile: demo 表示使用 demo 配置文件安装 istio
[root@k8scloude1 istioyaml]# cat istio-demo-profile.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: istio-system
```

```
---
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
metadata:
  namespace: istio-system
  name: demo-istio-install
spec:
  profile: demo
```

（7）创建资源

```
[root@k8scloude1 istioyaml]# kubectl apply -f istio-demo-profile.yaml
Warning: resource namespaces/istio-system is missing the
kubectl.kubernetes.io/last-applied-configuration annotation which is
required by kubectl apply. kubectl apply should only be used on resources
created declaratively by either kubectl create --save-config or kubectl
apply. The missing annotation will be patched automatically.
namespace/istio-system configured
istiooperator.install.istio.io/demo-istio-install created

#可以查看 istio-system 命名空间下的所有资源
[root@k8scloude1 istioyaml]# kubectl get all -o wide -n istio-system
NAME                                    READY   STATUS    RESTARTS   AGE
IP              NODE         NOMINATED NODE   READINESS GATES
pod/istio-egressgateway-58949b7c84-k7v6f       1/1         Running     0
3m38s   10.244.112.173   k8scloude2   <none>           <none>
pod/istio-ingressgateway-75bc568988-9j4wv      1/1         Running     0
3m38s   10.244.251.238   k8scloude3   <none>           <none>
pod/istiod-84d979766b-kz5sd                       1/1       Running    0
4m20s   10.244.112.130   k8scloude2   <none>           <none>

NAME                           TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)
AGE     SELECTOR
service/istio-egressgateway     ClusterIP        10.102.56.241    <none>
80/TCP,443/TCP
3m35s   app=istio-egressgateway,istio=egressgateway
service/istio-ingressgateway          LoadBalancer          10.107.131.65
192.168.110.190
15021:30093/TCP,80:32126/TCP,443:30293/TCP,31400:30628/TCP,15443:30966/
TCP   3m35s   app=istio-ingressgateway,istio=ingressgateway
service/istiod                    ClusterIP       10.103.37.59     <none>
15010/TCP,15012/TCP,443/TCP,15014/TCP
```

```
4m21s   app=istiod,istio=pilot

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
CONTAINERS    IMAGES                          SELECTOR
deployment.apps/istio-egressgateway      1/1        1                  1
3m39s             istio-proxy             docker.io/istio/proxyv2:1.14.3
app=istio-egressgateway,istio=egressgateway
deployment.apps/istio-ingressgateway     1/1        1                  1
3m39s             istio-proxy             docker.io/istio/proxyv2:1.14.3
app=istio-ingressgateway,istio=ingressgateway
deployment.apps/istiod                      1/1       1                1
4m21s   discovery     docker.io/istio/pilot:1.14.3     istio=pilot

NAME                                           DESIRED   CURRENT   READY
AGE     CONTAINERS    IMAGES                          SELECTOR
replicaset.apps/istio-egressgateway-58949b7c84    1         1         1
3m39s             istio-proxy             docker.io/istio/proxyv2:1.14.3
app=istio-egressgateway,istio=egressgateway,pod-template-hash=58949b7c8
4
replicaset.apps/istio-ingressgateway-75bc568988   1         1         1
3m39s             istio-proxy             docker.io/istio/proxyv2:1.14.3
app=istio-ingressgateway,istio=ingressgateway,pod-template-hash=75bc568
988
replicaset.apps/istiod-84d979766b                 1         1         1
4m21s           discovery                  docker.io/istio/pilot:1.14.3
istio=pilot,pod-template-hash=84d979766b
```

（8）检查安装状态

```
[root@k8scloude1 istioyaml]# kubectl get pod -o wide -n istio-system
NAME                                    READY   STATUS    RESTARTS   AGE
IP              NODE         NOMINATED NODE   READINESS GATES
istio-egressgateway-58949b7c84-k7v6f        1/1           Running      0
9m20s   10.244.112.173   k8scloude2   <none>           <none>
istio-ingressgateway-75bc568988-9j4wv       1/1           Running      0
9m20s   10.244.251.238   k8scloude3   <none>           <none>
istiod-84d979766b-kz5sd                  1/1     Running   0          10m
10.244.112.130   k8scloude2   <none>           <none>
```

3，可能易错点

（1）路径问题，根据参考文章来进行实际操作时候，需要尤其注意的是路径问题。观察每个命令是在哪一级目录来操作的，避免出现错误。

（2）修改文件问题，一般使用 vim 来进行，注意不要修改错误，使用 i 表示插入，:wq 表示保存并退出，:q 表示直接退出。

（3）有些过程出现错误或者没有预期输出，可能先不管，先进行后续的操作。后续操作得出预期输出在，则表明还是可以正常进行。

4,参考文章

https://www.cnblogs.com/renshengdezheli/p/16836404.html

3 Online-boutique 部署

3.1 环境准备

验证集群与 Istio 的版本

```
[root@k8scloude1 ~]# istioctl version
client version: 1.14.3
control plane version: 1.14.3
data plane version: 1.14.3 (1 proxies)

[root@k8scloude1 ~]# kubectl get nodes
NAME         STATUS   ROLES                  AGE    VERSION
k8scloude1   Ready    control-plane,master   283d   v1.21.9
k8scloude2   Ready    <none>                 283d   v1.21.9
k8scloude3   Ready    <none>                 283d   v1.21.9
```

创建命名空间，并使用 sidecar 自动注入（此处的命名空间可以自己来指定）

```
#创建命名空间 online-boutique
[root@k8scloude1 ~]# kubectl create ns online-boutique
namespace/online-boutique created

#切换命名空间
[root@k8scloude1 ~]# kubens online-boutique
Context "kubernetes-admin@kubernetes" modified.
Active namespace is "online-boutique".

#让命名空间 online-boutique 启用 sidecar 自动注入
[root@k8scloude1 ~]# kubectl label ns online-boutique
istio-injection=enabled
namespace/online-boutique labeled

[root@k8scloude1 ~]# kubectl get ns -l istio-injection --show-labels
NAME              STATUS   AGE   LABELS
online-boutique                          Active                          16m
istio-injection=enabled,kubernetes.io/metadata.name=online-boutique
```

3.2 具体部署

（1）克隆仓库代码

```
#安装 git
[root@k8scloude1 ~]# yum -y install git

#查看 git 版本
[root@k8scloude1 ~]# git version
git version 1.8.3.1

#创建 online-boutique 目录，项目放在该目录下
[root@k8scloude1 ~]# mkdir online-boutique

[root@k8scloude1 ~]# cd online-boutique/

[root@k8scloude1 online-boutique]# pwd
/root/online-boutique

#git 克隆代码
[root@k8scloude1 online-boutique]# git clone
https://github.com/GoogleCloudPlatform/microservices-demo.git
正克隆到 'microservices-demo'...
remote: Enumerating objects: 8195, done.
remote: Counting objects: 100% (332/332), done.
remote: Compressing objects: 100% (167/167), done.
remote: Total 8195 (delta 226), reused 241 (delta 161), pack-reused 7863
接收对象中: 100% (8195/8195), 30.55 MiB | 154.00 KiB/s, done.
处理 delta 中: 100% (5823/5823), done.

[root@k8scloude1 online-boutique]# ls
microservices-demo
```

此处可能会出现网络问题，git 一直拉取不到远程的代码，可以切换网络或者从本地电脑拖动文件进去

（2）进入配置文件目录

```
[root@k8scloude1 online-boutique]# cd microservices-demo/

[root@k8scloude1 microservices-demo]# ls
cloudbuild.yaml          CODEOWNERS              docs    istio-manifests
kustomize  pb         release       SECURITY.md    src
CODE_OF_CONDUCT.md  CONTRIBUTING.md  hack  kubernetes-manifests  LICENSE
```

```
README.md  renovate.json  skaffold.yaml  terraform

[root@k8scloude1 microservices-demo]# cd release/

[root@k8scloude1 release]# ls
istio-manifests.yaml  kubernetes-manifests.yaml
```

（3）拉取镜像

```
[root@k8scloude1 release]# ls
istio-manifests.yaml  kubernetes-manifests.yaml

[root@k8scloude1 release]# vim kubernetes-manifests.yaml

#可以看到安装此项目需要 13 个镜像，gcr.io 表示是 Google 的镜像
[root@k8scloude1 release]# grep image kubernetes-manifests.yaml
       image:
gcr.io/google-samples/microservices-demo/emailservice:v0.4.0
         image:
gcr.io/google-samples/microservices-demo/checkoutservice:v0.4.0
       image:
gcr.io/google-samples/microservices-demo/recommendationservice:v0.4.0
         image: gcr.io/google-samples/microservices-demo/frontend:v0.4.0
       image:
gcr.io/google-samples/microservices-demo/paymentservice:v0.4.0
       image:
gcr.io/google-samples/microservices-demo/productcatalogservice:v0.4.0
       image:
gcr.io/google-samples/microservices-demo/cartservice:v0.4.0
       image: busybox:latest
       image:
gcr.io/google-samples/microservices-demo/loadgenerator:v0.4.0
       image:
gcr.io/google-samples/microservices-demo/currencyservice:v0.4.0
       image:
gcr.io/google-samples/microservices-demo/shippingservice:v0.4.0
       image: redis:alpine
       image: gcr.io/google-samples/microservices-demo/adservice:v0.4.0

[root@k8scloude1 release]# grep image kubernetes-manifests.yaml | uniq |
wc -l
13

#在 k8s 集群的 worker 节点提前下载镜像，以 k8scloude2 为例
```

```
#  把  gcr.io  换  为  gcr.lank8s.cn  ,  比  如
gcr.io/google-samples/microservices-demo/emailservice:v0.4.0  换  为
gcr.lank8s.cn/google-samples/microservices-demo/emailservice:v0.4.0
[root@k8scloude2          ~]#          docker          pull
gcr.lank8s.cn/google-samples/microservices-demo/emailservice:v0.4.0
。。。。。。
其他那些镜像就按照此方法下载......
。。。。。。
[root@k8scloude2          ~]#          docker          pull
gcr.lank8s.cn/google-samples/microservices-demo/adservice:v0.4.0

#镜像下载之后，使用 sed 把 kubernetes-manifests.yaml 文件中的 gcr.io 修改为
gcr.lank8s.cn
[root@k8scloude1   release]#   sed   -i   's/gcr.io/gcr.lank8s.cn/'
kubernetes-manifests.yaml

#此时 kubernetes-manifests.yaml 文件中的镜像就全被修改了
[root@k8scloude1 release]# grep image kubernetes-manifests.yaml
       image:
gcr.lank8s.cn/google-samples/microservices-demo/emailservice:v0.4.0
         image:
gcr.lank8s.cn/google-samples/microservices-demo/checkoutservice:v0.4.0
       image:
gcr.lank8s.cn/google-samples/microservices-demo/recommendationservice:v
0.4.0
         image:
gcr.lank8s.cn/google-samples/microservices-demo/frontend:v0.4.0
       image:
gcr.lank8s.cn/google-samples/microservices-demo/paymentservice:v0.4.0
       image:
gcr.lank8s.cn/google-samples/microservices-demo/productcatalogservice:v
0.4.0
       image:
gcr.lank8s.cn/google-samples/microservices-demo/cartservice:v0.4.0
       image: busybox:latest
       image:
gcr.lank8s.cn/google-samples/microservices-demo/loadgenerator:v0.4.0
       image:
gcr.lank8s.cn/google-samples/microservices-demo/currencyservice:v0.4.0
       image:
gcr.lank8s.cn/google-samples/microservices-demo/shippingservice:v0.4.0
       image: redis:alpine
       image:
gcr.lank8s.cn/google-samples/microservices-demo/adservice:v0.4.0
```

```
#istio-manifests.yaml 文件没有镜像
[root@k8scloude1 release]# vim istio-manifests.yaml
[root@k8scloude1 release]# grep image istio-manifests.yaml
```

（4）在 online-boutique 命名空间创建 k8s 资源

```
[root@k8scloude1 release]# kubectl apply -f /root/online-boutique/microservices-demo/release/kubernetes-manifests.yaml -n online-boutique
```

此处路径一定要选择绝对路径，可以使用 pwd 来查看

（5）检查所有 pod 均在运行

此处重点需要与文章不同，通过名称来查（由于前面所进行的操作不一样）

```
kubectl get pod -n online-boutique
```

（6）创建 Istio 资源（注意此时的路径需要改变）

```
[root@k8scloude1 microservices-demo]# pwd
/root/online-boutique/microservices-demo

[root@k8scloude1 microservices-demo]# ls istio-manifests/
allow-egress-googleapis.yaml  frontend-gateway.yaml  frontend.yaml

[root@k8scloude1 microservices-demo]# kubectl apply -f ./istio-manifests
serviceentry.networking.istio.io/allow-egress-googleapis created
serviceentry.networking.istio.io/allow-egress-google-metadata created
gateway.networking.istio.io/frontend-gateway created
virtualservice.networking.istio.io/frontend-ingress created
virtualservice.networking.istio.io/frontend created
```

（7）得到入口网关的 IP 地址并打开服务

此处的前两步可以不产生效果，不影响最后的端口暴露

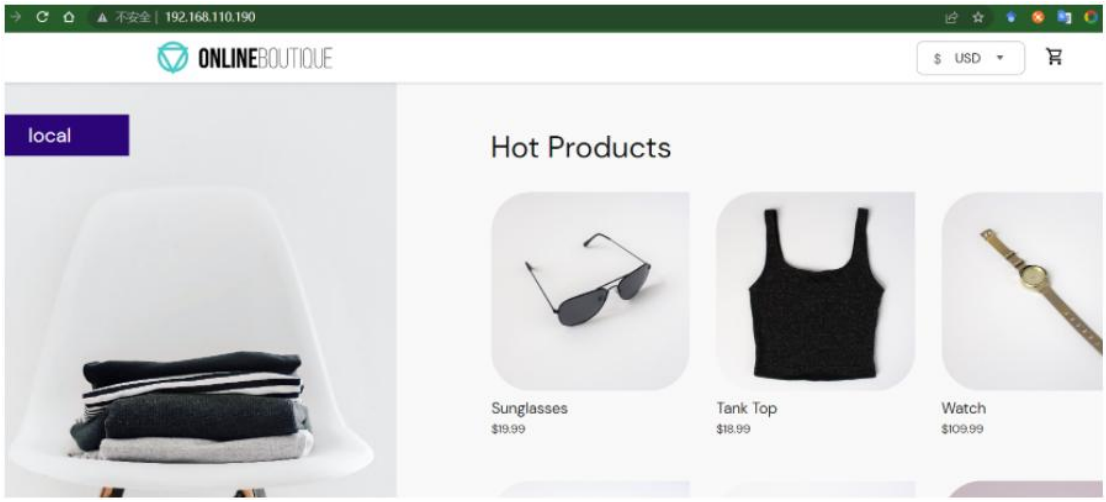
最后通过 get service 得到端口后，通过 master 主机 IP +端口访问即可

```
[root@k8scloude1 microservices-demo]# INGRESS_HOST="$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')"

[root@k8scloude1 microservices-demo]# echo "$INGRESS_HOST"
192.168.110.190

[root@k8scloude1 microservices-demo]# kubectl get service -n istio-system istio-ingressgateway -o wide
NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP
```

```
PORT(S)
AGE   SELECTOR
istio-ingressgateway    LoadBalancer    10.107.131.65    192.168.110.190
15021:30093/TCP,80:32126/TCP,443:30293/TCP,31400:30628/TCP,15443:30966/
TCP   27d   app=istio-ingressgateway,istio=ingressgateway
```

注意该处不用关闭 frontend-external 服务

最终效果如下图即可完成



总结易错点：

1，版本问题：目前直接从 github 上拉取的为 0.8.0 版本，相应的镜像是存在一定问题的。且拉取之前的版本会与当前的配置文件不符合，从而导致报错。这种情况的解决往往是从本地将旧版本的 online-boutique 传上去，使用旧的镜像版本即可。

2，该处有些步骤也得不到预期的输出，尤其是在最后可以直接通过端口访问，没有必要直接通过域名。

参考文章

https://www.cnblogs.com/renshengdezheli/p/16841875.html#51-%E9%83%A8%E7%BD%B2-online-boutique-%E5%BA%94%E7%94%A8