最新的稳定版 kubernetes1.24.4 安装部署

一、 使用最新稳定版背景 和 contianerd 背景和参考官方说明

最新稳定版背景：

1、 最新版稳定 1.24.4 支持更多新特性，修复新漏洞问题。
2、 从 1.20 开始，就宣布弃用 Dockershim。
3、 新版 1.24 开始正式弃用 Dockershim。
4、 新版支持更多容器，如 podman、docker、contianerd

contianerd 背景：

1、 新版 1.24 开始，K8S 已取消 docker 作为其默认 CRI(运行时)，默认支持 contianerd，无需 Dockershim，而 docker 需要第三方 cri-docker 支持。
2、 containerd 可以很好管理容器，包括容器镜像的传输和存储、容器的执行和管理、存储和网络等。
3、 containerd 是从 docker 中分离出来的一个项目，命令和 docker 类似。
4、 旧的的 k8s 的 kubelet 需要先要通过 dockershim 去调用 docker，最后再通过 docker 去调用 containerd。
5、 使用 containerd 作为 K8S 容器运行时的话，由于 containerd 内置了 CRI（Container Runtime Interface：容器运行时接口）插件，kubelet 可以直接调用 containerd。

参考官方说明:

CRI:https://kubernetes.io/zh-cn/docs/setup/production-environment/container-runtimes/

自 1.24 版起，Dockershim 已从 Kubernetes 项目中移除。

Kubernetes 1.24 支持 CRI:

containerd
RI-O
cri-docker（Docker Engine）
Mirantis Container Runtime

二、 节点规划

服务器列表	最小需要配置	主要服务	运行服务	备注
Control01	cpu2 核/内存 2G/硬盘 50G	k8s-master	etcd	
Control02	cpu2 核/内存 2G/硬盘 50G	k8s-master	etcd	
Control03	cpu2 核/内存 2G/硬盘 51G	k8s-master	etcd	
Node04	cpu4 核/内存 4G/硬盘 50G	k8s-node	。	
Node05	cpu4 核/内存 4G/硬盘 50G	k8s-node		
Node06	cpu4 核/内存 4G/硬盘 50G	k8s-node		

/etc/hosts 配置

```
192.168.80.87 node01  master01  etcd01
192.168.80.88 node02  master02  etcd02
192.168.80.89 node03  master03  etcd03
192.168.80.90 node04  slave01
192.168.80.91 node05  slave02
```

192.168.80.92 node06 slave03

192.168.80.93 node07 base01 harbor gitea jenkins efk prometheus

192.168.80.94 mastervip nodevip

三、 初始化 sysctl ,repo,modules

Sysctl.conf 和 repo 配置(Ansible copy 到每台 k8s 节点)如下：

```
# 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).


#below add by jingge

vm.swappiness = 0
kernel.sysrq = 1

net.ipv4.neigh.default.gc_stale_time = 120

net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.arp_announce = 2
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_announce = 2

net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_max_tw_buckets = 16384
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 8192
net.core.somaxconn = 8192
net.ipv4.tcp_synack_retries = 2

net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
fs.file-max = 655360
#end
```

[yunwei@ansible yum.repos.d]$ cat kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/

enabled=1

gpgcheck=0

repo_gpgcheck=0

gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg

https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

加载模块：

sudo modprobe overlay

sudo modprobe br_netfilter

四、 方案一：使用 cri-docker docker

下载 CRI：https://github.com/Mirantis/cri-dockerd

cri-docker（Docker Engine）配置：

1、 wget https://github.com/Mirantis/cri-dockerd/releases/download/v0.2.2/cri-dockerd-0.2.2.amd64.tgz && tar -zxvf cri-dockerd-0.2.2.amd64.tgz

2、 在 cri-docker.service 修改 --pod-infra-container-image=docker.io/gotok8s/pause:3.7 --network-plugin=cni

3、 cd cri-dockerd && cp -a cri-dockerd /usr/bin/ && ll /usr/bin/cri-dockerd

4、 wget https://github.com/Mirantis/cri-dockerd/blob/master/packaging/systemd/cri-docker.service 并 systemctl 启动

docker 安装：

yum -y install docker-ce 并执行启动

sudo systemctl restart docker

五、 方案二：使用 contianerd

安装方法：yum -y install contianerd

配置 CRI：

注释/etc/containerd/config.toml 中 disabled_plugins = ["cri"]，启用 cri。

添加 [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]

SystemdCgroup = true

启动 sudo systemctl restart contianerd

六、 Node 节点安装 keepalived haproxy

安装：yum clean all && yum -y keepalived haproxy

配置如下图：

```
[yunwei@192_168_80_82 ~]$ cat /etc/keepalived/keepalived.conf.
! Configuration File for keepalived

global_defs {
   notification_email {
     acassen@firewall.loc
     failover@firewall.loc
     sysadmin@firewall.loc
   }
   notification_email_from Alexandre.Cassen@firewall.loc
   smtp_server 192.168.200.1
   smtp_connect_timeout 30
   router_id LVS_DEVEL
   vrrp_skip_check_adv_addr
   vrrp_strict
   vrrp_garp_interval 0
   vrrp_gna_interval 0
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 168
    priority 100
    advert_int 1
    nopreempt
    authentication {
        auth_type PASS
        auth_pass jingge
    }
    virtual_ipaddress {
        192.168.80.94
    }
}
```

```
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
#---------------------------------------------------------------------
# apiserver frontend which proxys to the control plane nodes
#---------------------------------------------------------------------
frontend apiserver
    bind *:6443
    mode tcp
    option tcplog
    default_backend apiserver

#---------------------------------------------------------------------
# round robin balancing for apiserver
#---------------------------------------------------------------------
backend apiserver
    option httpchk GET /healthz
    http-check expect status 200
    mode tcp
    option ssl-hello-chk
    balance     roundrobin
        server  node01  node01:6443 check
        server  node02  node02:6443 check
        server  node03  node03:6443 check
```

启动：systemctl restart keepalived ; systemctl restart haproxy

七、 安装 kubeadm、kubelet 和 kubectl

安装：yum install -y kubectl-1.24.4-0 kubeadm-1.24.4-0 kubelet-1.24.4-0 --disableexcludes=Kubernetes

echo -e 'KUBELET_EXTRA_ARGS=\"--cgroup-driver=systemd\"' >> /etc/sysconfig/kubelet"

启动：sudo systemctl enable --now kubelet （k8s 没安装完，有错误日志提示）

八、 导入官方镜像

Github 下载地址：

https://dl.k8s.io/v1.24.4/kubernetes-server-linux-amd64.tar.gz

解压后导入镜像：

docker image load -i /tmp/kube-apiserver.tar

docker image load -i /tmp/kube-controller-manager.tar

docker image load -i /tmp/ kube-scheduler.tar

docker image load -i /tmp/kube-proxy.tar

tag 镜像：

docker tag k8s.gcr.io/kube-apiserver-amd64:v1.24.4 k8s.gcr.io/kube-apiserver:v1.24.4

docker tag k8s.gcr.io/kube-controller-manager-amd64:v1.24.4 k8s.gcr.io/kube-controller-manager:v1.24.4

docker tag k8s.gcr.io/kube-scheduler-amd64:v1.24.4 k8s.gcr.io/kube-scheduler:v1.24.4

docker tag k8s.gcr.io/kube-proxy-amd64:v1.24.4 k8s.gcr.io/kube-proxy:v1.24.4

九、 Kubeadm 初始化安装

两种安装方式（用一种即可以）：

1、命令直接初始化

kubeadm init --kubernetes-version v1.24.4 --pod-network-cidr “10.64.0.0/12” --service-cidr ”10.128.0.0/12” --control-plane-endpoint ”LOAD_BALANCER_DNS:LOAD_BALANCER_PORT” --upload-certs

2、生成默认配置文件:

kubeadm config print init-defaults > init-defaults.yaml

加 podsubnet 和 controlPlaneEndpoint: mastervip:6443

为了实现 docker 使用的 cgroupdriver 与 kubeproxy 使用的 mode 的一致性，建议修改 kubeadm init 的默认配置文件 添加如下文件内容。

apiVersion: kubeproxy.config.k8s.io/v1alpha1

kind: KubeProxyConfiguration

mode: ipvs

```
apiVersion: kubeadm.k8s.io/v1beta3
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 9600h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 192.168.80.87
  bindPort: 6443
nodeRegistration:
  criSocket: unix:///var/run/containerd/containerd.sock
  imagePullPolicy: IfNotPresent
  name: node01
  taints: null
---
apiServer:
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta3
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controlPlaneEndpoint: mastervip:6443
controllerManager: {}
dns: {}
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: k8s.gcr.io
kind: ClusterConfiguration
kubernetesVersion: 1.24.4
networking:
  dnsDomain: cluster.local
  serviceSubnet: 10.128.0.0/12
  podSubnet: 10.64.0.0/12
scheduler: {}
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
mode: ipvs
```

镜像下载:

ansible k8s -uroot -m shell -a "docker pull k8s.gcr.io/pause:3.7 ;docker pull k8s.gcr.io/etcd:3.5.3-0; docker pull k8s.gcr.io/coredns/coredns:v1.8.6"

初始化安装:

kubeadm init --config "/root/init-defaults.yaml" --upload-certs --v=5

```
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 the control-plane node running the following command on each as root:

  kubeadm join mastervip:6443 --token abcdef.0123456789abcdef \
        --discovery-token-ca-cert-hash sha256:508078d51b7bcd2dbf3dd4620547a02cb83a6f40b05d8be9db585630f5964ff1 \
        --control-plane --certificate-key 3c8de57bcd226f7b1c9e2ced0931c7392b7769e6bd85b3ea9119977e4f1f1384

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join mastervip:6443 --token abcdef.0123456789abcdef \
        --discovery-token-ca-cert-hash sha256:508078d51b7bcd2dbf3dd4620547a02cb83a6f40b05d8be9db585630f5964ff1

[root@node01 ~]#
```

出现这个，表示初始化 安装成功

十、 加入 control 节点 和 worker 节点

Control01 和 control02 节点执行：

kubeadm join mastervip:6443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash

sha256:508078d51b7bcd2dbf3dd4620547a02cb83a6f40b05d8be9db585630f5964ff1 \

--control-plane --certificate-key

3c8de57bcd226f7b1c9e2ced0931c7392b7769e6bd85b3ea9119977e4f1f1384

```
This node has joined the cluster and a new control plane instance was created:

* Certificate signing request was sent to apiserver and approval was received.
* The Kubelet was informed of the new secure connection details.
* Control plane label and taint were applied to the new node.
* The Kubernetes control plane instances scaled up.
* A new etcd member was added to the local/stacked etcd cluster.

To start administering your cluster from this node, 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

Run 'kubectl get nodes' to see this node join the cluster.
```

三台 worker 节点执行：

kubeadm join mastervip:6443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash

sha256:508078d51b7bcd2dbf3dd4620547a02cb83a6f40b05d8be9db585630f5964ff1

```
[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@node04 ~]#
```

十一、 添加集群网络组件 calico

没有安装网络组件时：

```
[root@node01 ~]# kubectl get node
NAME     STATUS     ROLES           AGE     VERSION
node01   NotReady   control-plane   134m    v1.24.4
node02   NotReady   control-plane   4m30s   v1.24.4
node03   NotReady   control-plane   4m20s   v1.24.4
node04   NotReady   <none>          8m11s   v1.24.4
node05   NotReady   <none>          7m52s   v1.24.4
[root@node01 ~]#
```

安装方法：

curl https://docs.projectcalico.org/manifests/calico.yaml -O

sed -ie "s/# - name: CALICO_IPV4POOL_CIDR/- name: CALICO_IPV4POOL_CIDR/g" calico.yaml

sed -ie 's?# value: "192.168.0.0/16"? value: "192.168.0.0/16"?g' calico.yaml

POD_CIDR="10.64.0.0/12" && sed -i -e "s?192.168.0.0/16?$POD_CIDR?g" calico.yaml

在 Deployment 修改 replicas 数量，1 到 3 个数量。

安装以后：

```
[yunwei@192_168_80_82 ~]$ kubectl get node
NAME     STATUS   ROLES           AGE    VERSION
node01   Ready    control-plane   174m   v1.24.4
node02   Ready    control-plane   44m    v1.24.4
node03   Ready    control-plane   44m    v1.24.4
node04   Ready    <none>          48m    v1.24.4
node05   Ready    <none>          48m    v1.24.4
node06   Ready    <none>          31m    v1.24.4
```

十二、 验证测试群

创建 deployment，测试访问正常，集群 OK

kubectl create deployment mynginx --image=nginx --replicas=2

kubectl expose deployment mynginx --port=80 --target-port=80

kubectl get svc

curl 10.142.137.79

```
[root@node01 ~]# curl 10.142.137.79
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
```

十三、 安装图形界面（可完成大部分命令行管理操作）

参考官网安装参考官网：

https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard

安装步骤：

wget

https://raw.githubusercontent.com/kubernetes/dashboard/v2.6.1/aio/deploy/recommended.yaml

Kubectl apply -f recommended.yaml

图形界面 dashboard:

1、Creating a Service Account 和 Creating a ClusterRoleBinding（如下内容保存为 yaml 并执行）

apiVersion: v1

```
  kind: ServiceAccount
  metadata:
    name: admin-user
    namespace: kubernetes-dashboard
  ---
  apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRoleBinding
  metadata:
    name: admin-user
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: cluster-admin
  subjects:
  - kind: ServiceAccount
    name: admin-user
    namespace: kubernetes-dashboard
```

2、Getting a Bearer Token

kubectl -n kubernetes-dashboard create token admin-user 命令输出的 token 保存用于登录

3、Remove the admin ServiceAccount and ClusterRoleBinding（如取消执行，这步一般无需操作执行）

kubectl -n kubernetes-dashboard delete serviceaccount admin-user

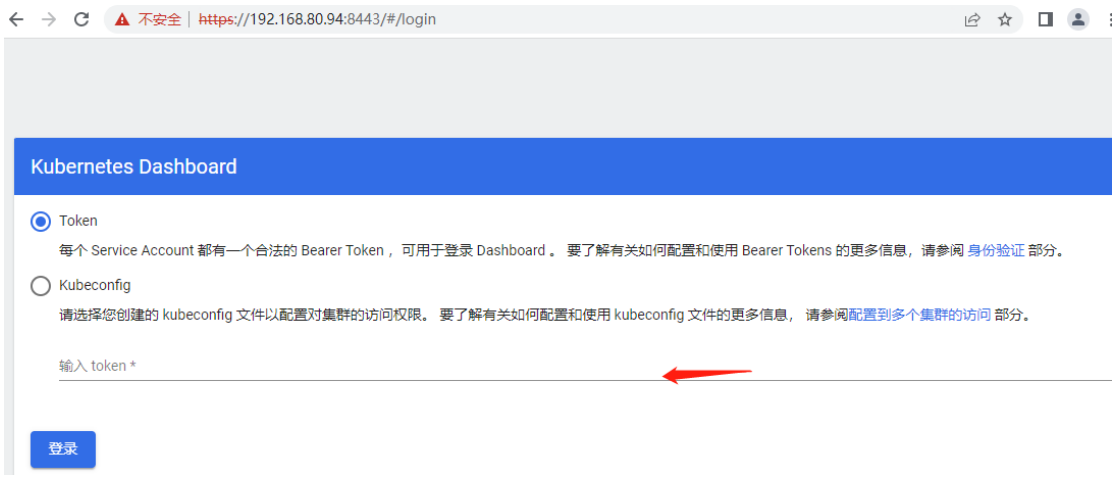
kubectl -n kubernetes-dashboard delete clusterrolebinding admin-user

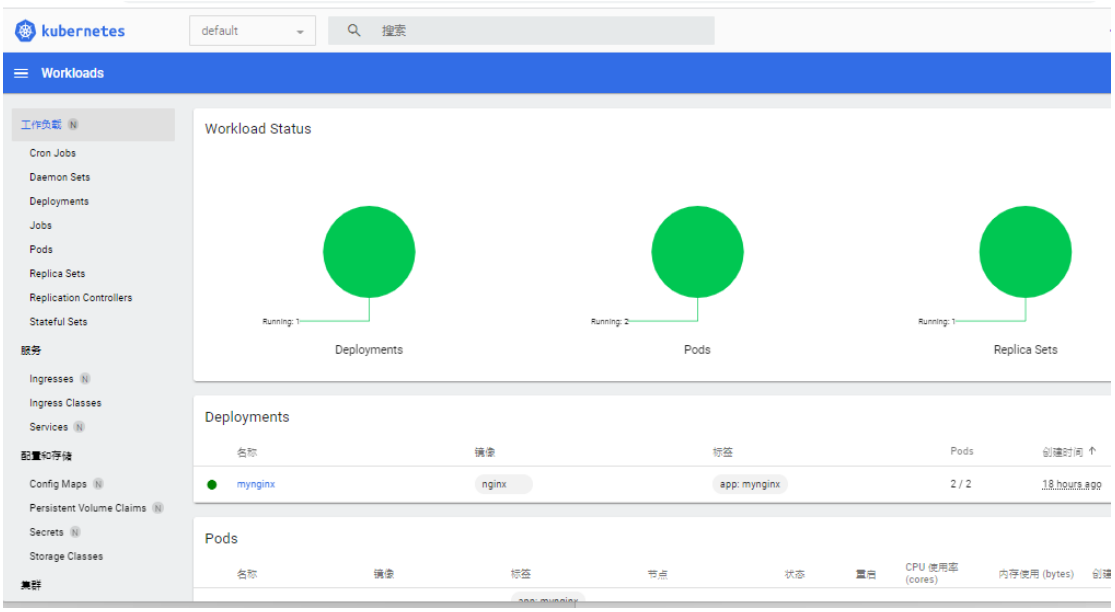
4、开放对外端口 （官网用的是 kubectl proxy，我习惯采用 nodport）

kubectl edit -n kubernetes-dashboard svc/kubernetes-dashboard

删除 ClusterIP 参数 ， 修改 type=NodePort,添加 nodePort 8443

登录 https://192.168.80.94:8443





十四、 截图 docker 模式 和 contianerd 模式区别

对 kubectl 命令来说没有任何区别，都是调用 CRI

```
[yunwei@192_168_80_82 ~]$ kubectl get node -o wide
NAME     STATUS   ROLES           AGE    VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION                CONTAINER-RUNTIME
node01   Ready    control-plane   174m   v1.24.4   192.168.80.87    <none>        CentOS Linux 7 (Core)   3.10.0-1160.76.1.el7.x86_64   containerd://1.6.8
node02   Ready    control-plane   44m    v1.24.4   192.168.80.88    <none>        CentOS Linux 7 (Core)   3.10.0-1160.76.1.el7.x86_64   containerd://1.6.8
node03   Ready    control-plane   44m    v1.24.4   192.168.80.89    <none>        CentOS Linux 7 (Core)   3.10.0-1160.76.1.el7.x86_64   containerd://1.6.8
node04   Ready    <none>          48m    v1.24.4   192.168.80.90    <none>        CentOS Linux 7 (Core)   3.10.0-1160.76.1.el7.x86_64   containerd://1.6.8
node05   Ready    <none>          48m    v1.24.4   192.168.80.91    <none>        CentOS Linux 7 (Core)   3.10.0-1160.76.1.el7.x86_64   containerd://1.6.8
node06   Ready    <none>          31m    v1.24.4   192.168.80.92    <none>        CentOS Linux 7 (Core)   3.10.0-1160.76.1.el7.x86_64   containerd://1.6.8
```

Docker 模式如下图

```
[root@t01 ~]# kubectl get node -o wide
NAME   STATUS   ROLES           AGE   VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE                         KERNEL-VERSION                 CONTAINER-RUNTIME
t01    Ready    control-plane   62d   v1.24.2   192.168.56.101   <none>        Rocky Linux 8.6 (Green Obsidian)   4.18.0-372.9.1.el8.x86_64   docker://20.10.17
t02    Ready    control-plane   62d   v1.24.2   192.168.56.102   <none>        Rocky Linux 8.6 (Green Obsidian)   4.18.0-372.9.1.el8.x86_64   docker://20.10.17
t03    Ready    control-plane   62d   v1.24.2   192.168.56.103   <none>        Rocky Linux 8.6 (Green Obsidian)   4.18.0-372.9.1.el8.x86_64   docker://20.10.17
t04    Ready    <none>          62d   v1.24.2   192.168.56.104   <none>        Rocky Linux 8.6 (Green Obsidian)   4.18.0-372.9.1.el8.x86_64   docker://20.10.17
t05    Ready    <none>          62d   v1.24.2   192.168.56.105   <none>        Rocky Linux 8.6 (Green Obsidian)   4.18.0-372.9.1.el8.x86_64   docker://20.10.17
t06    Ready    <none>          62d   v1.24.2   192.168.56.106   <none>        Rocky Linux 8.6 (Green Obsidian)   4.18.0-372.9.1.el8.x86_64   docker://20.10.17
[root@t01 ~]#
```

十五、 常见问题

错误： * spec.ports[0].nodePort: Invalid value: 80: provided port is not in the valid range. The range of valid ports is 30000-32767

解决： vim /etc/kubernetes/manifests/kube-apiserver.yaml 添加

- --service-node-port-range=1-65535