--address 不能设置为 127.0.0.1，否则后续 Pods 访问 kubelet 的 API 接口时会失败，因为 Pods 访问的 127.0.0.1 指向自己而不是 kubelet；

如果设置了 --hostname-override 选项，则 kube-proxy 也需要设置该选项，否则会出现找不到 Node 的情况；

--cgroup-driver 配置成 systemd，不要使用 cgroup，否则在 CentOS 系统中 kubelet 讲启动失败。docker 修改 cgroup 启动参数 --exec-opt native.cgroupdriver=systemd

--experimental-bootstrap-kubeconfig 指向 bootstrap kubeconfig 文件，kubelet 使用该文件中的用户名和 token 向 kube-apiserver 发送 TLS Bootstrapping 请求；

管理员通过了 CSR 请求后，kubelet 自动在 --cert-dir 目录创建证书和私钥文件 (kubelet-client.crt 和 kubelet-client.key)，然后写入 --kubeconfig 文件；

建议在 --kubeconfig 配置文件中指定 kube-apiserver 地址，如果未指定 --api-servers 选项，则必须指定 --require-kubeconfig 选项后才从配置文件中读取 kube-apiserver 的地址，否则 kubelet 启动后将找不到 kube-apiserver (日志中提示未找到 API Server），kubectl get nodes 不会返回对应的 Node 信息;

--cluster-dns 指定 kubedns 的 Service IP(可以先分配，后续创建 kubedns 服务时指定该 IP)，--cluster-domain 指定域名后缀，这两个参数同时指定后才会生效；

--cluster-domain 指定 pod 启动时 /etc/resolve.conf 文件中的 search domain ，起初我们将其配置成了 cluster.local.，这样在解析 service 的 DNS 名称时是正常的，可是在解析 headless service 中的 FQDN pod name 的时候却错误，因此我们将其修改为 cluster.local，去掉嘴后面的 ”点号“ 就可以解决该问题，关于 kubernetes 中的域名/服务名称解析请参见我的另一篇文章。

--kubeconfig=/etc/kubernetes/kubelet.kubeconfig 中指定的 kubelet.kubeconfig 文件在第一次启动 kubelet 之前并不存在，请看下文，当通过 CSR 请求后会自动生成 kubelet.kubeconfig 文件，如果你的节点上已经生成了~/.kube/config 文件，你可以将该文件拷贝到该路径下，并重命名为 kubelet.kubeconfig，所有 node 节点可以共用同一个 kubelet.kubeconfig 文件，这样新添加的节点就不需要再创建 CSR 请求就能自动添加到 kubernetes 集群中。同样，在任意能够访问到 kubernetes 集群的主机上使用 kubectl --kubeconfig 命令操作集群时，只要使用~/.kube/config 文件就可以通过权限认证，因为这里面已经有认证信息并认为你是 admin 用户，对集群拥有所有权限。

KUBELET*POD*INFRA_CONTAINER 是基础镜像容器，需要翻墙下载。

--network-plugin=cni 启用 cni 管理 docker 网络

-cni-conf-dir=/etc/cni/net.d/ CNI 配置路径

注意 需要修改 docker cgroup 驱动方式： --exec-opt native.cgroupdriver=systemd

kubelet 依赖启动配置文件 bootstrap.kubeconfig

```
systemctl daemon-reload

systemctl enable kubelet

systemctl start kubelet

systemctl status kubelet
```

通过 kublet 的 TLS 证书请求

kubelet 首次启动时向 kube-apiserver 发送证书签名请求，必须通过后 kubernetes 系统才会将该 Node 加入到集群。

查看未授权的 CSR 请求

```
# kubectl get csr

NAME                                                   AGE       REQUESTOR           CO
NDITION

node-csr-8I8soRqLhxiH2nThkgUsL2oIaKyh15AuNOVgJddWBqA   2s        kubelet-bootstrap
 Pending

node-csr-9byGSZPAX0eT60qME8_2PIZ0Q4GkDTFG-1tvPhVaH40   49d       kubelet-bootstrap
 Approved,Issued

node-csr-DpvCEHT98ARavxjdLpa_yl_aNGddNTAX07MEVSAjnUM   4d        kubelet-bootstrap
 Approved,Issued
```

```
node-csr-nAOtjarW3mJ3boQ3AtaeGCbQYbW_jo8AGscFnk1uxqw   8d        kubelet-bootstrap
 Approved,Issued

node-csr-sgI8CYnTFQZqaZg9wdJP6OabqBiNA0DpZ5Z0wCC14bQ   54d       kubelet-bootstrap
 Approved,Issued
```

通过 CSR 请求

```
kubectl certificate approve node-csr-8I8soRqLhxiH2nThkgUsL2oIaKyh15AuNOVgJddWBqA
```

查看 通过的 node

```
kubectl  get node

NAME             STATUS    AGE       VERSION

172.16.200.206   Ready     11m       v1.7.6

172.16.200.209   Ready     49d       v1.7.6

172.16.200.216   Ready     4d        v1.7.6
```

自动生成了 kubelet.kubeconfig 文件和公私钥

```
ls -l /etc/kubernetes/kubelet.kubeconfig
```

> 注意：假如你更新 kubernetes 的证书，只要没有更新 token.csv，当重启 kubelet 后，该 node 就会自动加入到 kuberentes 集群中，而不会重新发送 certificaterequest，也不需要在 master 节点上执行 kubectl certificate approve 操作。前提是不要删除 node 节点上的/etc/kubernetes/ssl/kubelet* 和/etc/kubernetes/kubelet.kubeconfig 文件。否则 kubelet 启动时会提示找不到证书而失败。

配置 kube-proxy

创建 kube-proxy 的 service 配置文件

文件路径/usr/lib/systemd/system/kube-proxy.service

```
cat  > /usr/lib/systemd/system/kube-proxy.service << EOF

[Unit]

Description=Kubernetes Kube-Proxy Server

Documentation=https://github.com/GoogleCloudPlatform/kubernetes
```

```
After=network.target



[Service]

EnvironmentFile=-/etc/kubernetes/config

EnvironmentFile=-/etc/kubernetes/proxy

ExecStart=/usr/local/kubernetes/server/bin/kube-proxy \

        $KUBE_LOGTOSTDERR \

        $KUBE_LOG_LEVEL \

        $KUBE_MASTER \

        $KUBE_PROXY_ARGS

Restart=on-failure

LimitNOFILE=65536



[Install]

WantedBy=multi-user.target

EOF
```

kube-proxy 配置文件/etc/kubernetes/proxy

```
cat  > /etc/kubernetes/proxy << EOF

###

# kubernetes proxy config



# default config should be adequate



# Add your own!

KUBE_PROXY_ARGS="--bind-address=172.16.200.100 --hostname-override=172.16.200.100
--kube-api-burst=50 --kube-api-qps=20  --master=http://172.16.200.100:8080
--kubeconfig=/etc/kubernetes/kube-proxy.kubeconfig --cluster-cidr=10.254.0.0/16
--log-dir=/data/logs/kubernetes/ --v=2 --logtostderr=false"

EOF
```

--hostname-override 参数值必须与 kubelet 的值一致，否则 kube-proxy 启动后会找不到该 Node，从而不会创建任何 iptables 规则；

kube-proxy 根据 --cluster-cidr 判断集群内部和外部流量，指定 --cluster-cidr 或 --masquerade-all 选项后 kube-proxy 才会对访问 Service IP 的请求做 SNAT；

--kubeconfig 指定的配置文件嵌入了 kube-apiserver 的地址、用户名、证书、秘钥等请求和认证信息；

- 预定义的 RoleBinding cluster-admin 将 User system:kube-proxy 与 Role system:node-proxier 绑定，该 Role 授予了调用 kube-apiserver Proxy 相关 API 的权限；

启动 kube-proxy

```
systemctl daemon-reload

systemctl enable kube-proxy

systemctl start kube-proxy

systemctl status kube-proxy
```

创建资源对象

1. 根据yaml 配置文件一次性创建service、rc

kubectl create -f my-service.yaml -f my-rc.yaml

2. 查看资源对象
   - 查看所有pod 列表

     ```
     kubectl get pod -n <namespace>
     ```

   - 查看RC和service 列表

     ```
     kubectl get rc,svc
     ```

3. 描述资源对象
   - 显示Node的详细信息

     ```
     kubectl describe node <node-name>
     ```

   - 显示Pod的详细信息

     ```
     kubectl describe pod <pod-name>
     ```

4. 删除资源对象

   - 基于pod.yaml 定义的名称删除pod

     ```
     kubectl delete -f pod.yaml
     ```

   - 删除所有包含某个label的pod 和service

     ```
     kubectl delete pod,svc -l name=<label-name>
     ```

   - 删除所有Pod

     ```
     kubectl delete pod --all
     ```

5. 执行容器的命令

   - 执行pod 的date 命令

     ```
     kubectl exec <pod-name> -- date
     ```

   - 通过bash 获得pod中某个容器的TTY，相当于登陆容器

     ```
     kubectl exec -it <pod-name> -c <container-name> -- bash
     ```

6. 查看容器的日志

   ```
   kubectl logs <pod-name>
   ```