6. 部署kubernetes node节点

kubernetes node 节点包含如下组件：

- Flanneld ：现在需要在serivce配置文件中增加 TLS配置。
- Docker
- kubelet
- kube-proxy

注意：每台 node 上都需要安装 flannel，master 节点上可以不必安装。

步骤简介

1. 确认在上一步中我们安装配置的网络插件flannel已启动且运行正常
2. 安装配置docker后启动
3. 安装配置kubelet、kube-proxy后启动
4. 验证

目录和文件

检查一下三个节点上，经过前几部操作生成的配置文件。

```
[root@vlnx251101 kubernetes]# ls /etc/kubernetes/ssl/
admin-key.pem  admin.pem  ca-key.pem  ca.pem  kubelet.crt
kubelet.key  kube-proxy-key.pem  kube-proxy.pem  kubernetes-
key.pem  kubernetes.pem

[root@vlnx251101 kubernetes]# ls /etc/kubernetes/
bootstrap.kubeconfig  kubelet  kubelet.kubeconfig  kube-
proxy.kubeconfig  ssl  token.csv
```

安装和配置kubelet

kubernets1.8

对于kuberentes1.8集群，必须关闭swap，否则kubelet启动将失败。修改/etc/fstab将，swap系统注释掉。

kubelet启动时向 kube-apiserver发送 TLS bootstrapping请求，需要先将 bootstrap token文件中的 kubelet-bootstrap用户赋予 system:node-bootstrapper cluster 角色(role)，然后kubelet才能有权限创建认证请求 (certificate signing requests)：

```
[root@vlnx251101 kubernetes]# cd /etc/kubernetes/

[root@vlnx251101 kubernetes]# kubectl create
clusterrolebinding kubelet-bootstrap \
--clusterrole=system:node-bootstrapper \
--user=kubelet-bootstrap
```

- --user=kubelet-bootstrap 是在 /etc/kubernetes/token.csv文件中指定的用户名，同时也写入了 /etc/kubernetes/bootstrap.kubeconfig文件；

分发到所有node节点

```
[root@vlnx251101 ~]# scp ~/.kube/config
192.168.251.101:/etc/kubernetes/kubelet.kubeconfig
```

下载最新的kubelet 和kube-proxy二进制文件

参见master

创建kubelet的 service配置文件

文件位置/usr/lib/systemd/system/kubelet.service

```
[root@vlnx251101 kubernetes]# vim
/usr/lib/systemd/system/kubelet.service
```

```
[Unit]
Description=Kubernetes Kubelet Server
Documentation=https://github.com/GoogleCloudPlatform/kubernetes
After=docker.service
Requires=docker.service

[Service]
WorkingDirectory=/var/lib/kubelet
EnvironmentFile=-/etc/kubernetes/config
EnvironmentFile=-/etc/kubernetes/kubelet
ExecStart=/usr/local/bin/kubelet \
          $KUBE_LOGTOSTDERR \
          $KUBE_LOG_LEVEL \
          $KUBELET_API_SERVER \
          $KUBELET_ADDRESS \
          $KUBELET_PORT \
          $KUBELET_HOSTNAME \
          $KUBE_ALLOW_PRIV \
          $KUBELET_POD_INFRA_CONTAINER \
          $KUBELET_ARGS
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

[root@vlnx251101 kubernetes]# mkdir /var/lib/kubelet

kubelet的配置文件 /etc/kubernetes/kubelet。其中的 IP 地址更改为你的每台node 节点的IP地址

```
[root@vlnx251101 kubernetes]# vim /etc/kubernetes/kubelet

##
## kubernetes kubelet (minion) config
#
## The address for the info server to serve on (set to 0.0.0.0
or "" for all interfaces)
KUBELET_ADDRESS="--address=192.168.251.102"
#
## The port for the info server to serve on
#KUBELET_PORT="--port=10250"
#
## You may leave this blank to use the actual hostname
KUBELET_HOSTNAME="--hostname-override=192.168.251.102"
#
## location of the api-server
#KUBELET_API_SERVER="--api-
servers=http://192.168.251.101:8080"
#
## pod infrastructure container
KUBELET_POD_INFRA_CONTAINER="--pod-infra-container-
image=k8s.gcr.io/pause"
#
## Add your own!
KUBELET_ARGS="--cgroup-driver=systemd --cluster-dns=10.254.0.2
--bootstrap-kubeconfig=/etc/kubernetes/bootstrap.kubeconfig --
kubeconfig=/etc/kubernetes/kubelet.kubeconfig --cert-
dir=/etc/kubernetes/ssl --cluster-domain=cluster.local. --
hairpin-mode promiscuous-bridge --serialize-image-pulls=false
--runtime-cgroups=/systemd/system.slice --kubelet-
cgroups=/systemd/system.slice"
```

kubernetes1.8

相对于kubenrete1.6的配置变动：

- 对于kuberentes1.8集群中的kubelet配置，取消了KUBELET_API_SERVER的配置，而改用kubeconfig文件来定义master地址，所以请注释掉KUBELET_API_SERVER配置。

- 如果使用systemd方式启动，则需要额外增加两个参数--runtime-cgroups=/systemd/system.slice --kubelet-cgroups=/systemd/system.slice
- --experimental-bootstrap-kubeconfig 在1.9版本已经变成了--bootstrap-kubeconfig
- --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 将启动失败（保持docker和kubelet中的cgroup driver配置一致即可，不一定非使用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； --require-kubeconfig 在1.10版本被移除，参看PR；

- --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，去掉最后面的 ”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，这里我用的是私有镜像仓库地址，**大家部署的时候需要修改为自己的镜像**。可以直接 docker pull zhaoyonggang/pause 下载。pod-infrastructure镜像是Redhat制作的，大小接近80M，下载比较耗时，其实该镜像并不运行什么具体进程，可以使用Google的pause镜像gcr.io/google_containers/pause-amd64:3.0，这个镜像只有300多K。

启动kublet

```
[root@vlnx251101 kubernetes]# systemctl daemon-reload
; systemctl enable kubelet ; systemctl start kubelet
; systemctl status kubelet
```

通过kublet 的TLS证书请求

```
[root@vlnx251101 ~]# kubectl get node --
server=192.168.251.101:8080
NAME              STATUS    ROLES     AGE       VERSION
192.168.251.101   Ready     <none>    1h        v1.11.2
192.168.251.102   Ready     <none>    1h        v1.11.2
192.168.251.103   Ready     <none>    1h        v1.11.2
```

自动生成了`kubelet kubeconfig`文件和公私钥

```
[root@vlnx251101 kubernetes]# ls -l
/etc/kubernetes/kubelet.kubeconfig
-rw-------. 1 root root 2282 Oct 15 12:06
/etc/kubernetes/kubelet.kubeconfig

[root@vlnx251101 kubernetes]# ls -l
/etc/kubernetes/ssl/kubelet*
-rw-r--r--. 1 root root 1050 Oct 15 12:06
/etc/kubernetes/ssl/kubelet-client.crt
-rw-------. 1 root root  227 Oct 15 12:01
/etc/kubernetes/ssl/kubelet-client.key
-rw-r--r--. 1 root root 1119 Oct 15 12:06
/etc/kubernetes/ssl/kubelet.crt
-rw-------. 1 root root 1679 Oct 15 12:06
/etc/kubernetes/ssl/kubelet.key
```

注意：如果启动kubelet的时候见到证书相关的报错，有个trick可以解决这个问题，可以将master节点上的 `~/.kube/config` 文件（该文件在安装kubectl命令行工具这一步

中将会自动生成）拷贝到node节点的 /etc/kubernetes/kubelet.kubeconfig 位置，这样就不需要通过CSR，当kubelet启动后就会自动加入的集群中

配置kube-proxy

安装conntrack

```
[root@vlnx251101 ~]# yum install -y conntrack-tools
```

创建kube-proxy 的service配置文件

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

```
[root@vlnx251101 kubernetes]# vim
/usr/lib/systemd/system/kube-proxy.service

[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/bin/kube-proxy \
          $KUBE_LOGTOSTDERR \
          $KUBE_LOG_LEVEL \
          $KUBE_MASTER \
          $KUBE_PROXY_ARGS
Restart=on-failure
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
```

kube-proxy 配置文件/etc/kubernetes/proxy。

```
[root@vlnx251101 kubernetes]# vim /etc/kubernetes/proxy

###
# kubernetes proxy config
# default config should be adequate
# Add your own!
KUBE_PROXY_ARGS="--bind-address=192.168.251.102 --hostname-override=192.168.251.102 --kubeconfig=/etc/kubernetes/kube-proxy.kubeconfig --cluster-cidr=10.254.0.0/16"
```

- **--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 判断进群内部和外部流量，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、密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

```
[root@vlnx251101 kubernetes]# systemctl daemon-reload ; systemctl enable kube-proxy ; systemctl start kube-proxy ; systemctl status kube-proxy
```

验证测试

创建一个niginx 的service试一下集群是否可用。

```
[root@vlnx251101 ~]# kubectl run nginx --replicas=2 --
labels="run=load-balancer-example" --image=nginx  --port=80

[root@vlnx251101 ~]# kubectl get deployment
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx     2         2         2            0           50s

[root@vlnx251101 ~]# kubectl expose deployment nginx --
type=NodePort --name=example-service

[root@vlnx251101 ~]# kubectl describe svc example-service
Name:                     example-service
Namespace:                default
Labels:                   run=load-balancer-example
Annotations:              <none>
Selector:                 run=load-balancer-example
Type:                     NodePort
IP:                       10.254.154.234
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31053/TCP
Endpoints:                172.30.72.2:80,172.30.96.2:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>


[root@vlnx251101 ~]# curl 10.254.154.234:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

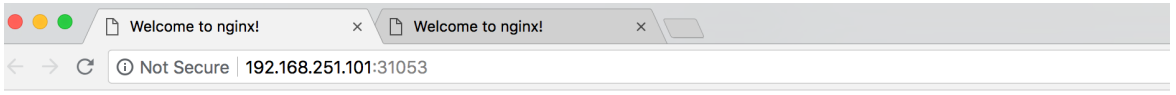
```
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully
installed and
working. Further configuration is required.</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>
</body>
</html>
```

访问

http://192.168.251.101:31053/

http://192.168.251.103:31053/

都可以得到nginx的页面。



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.