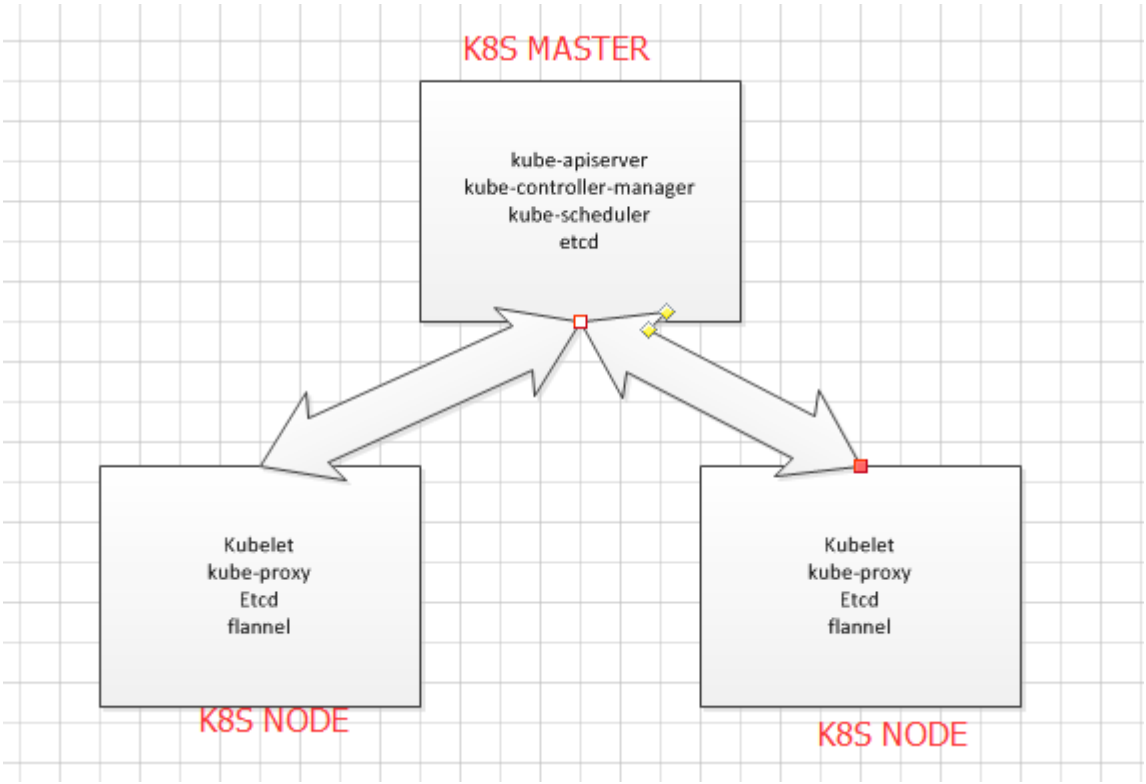
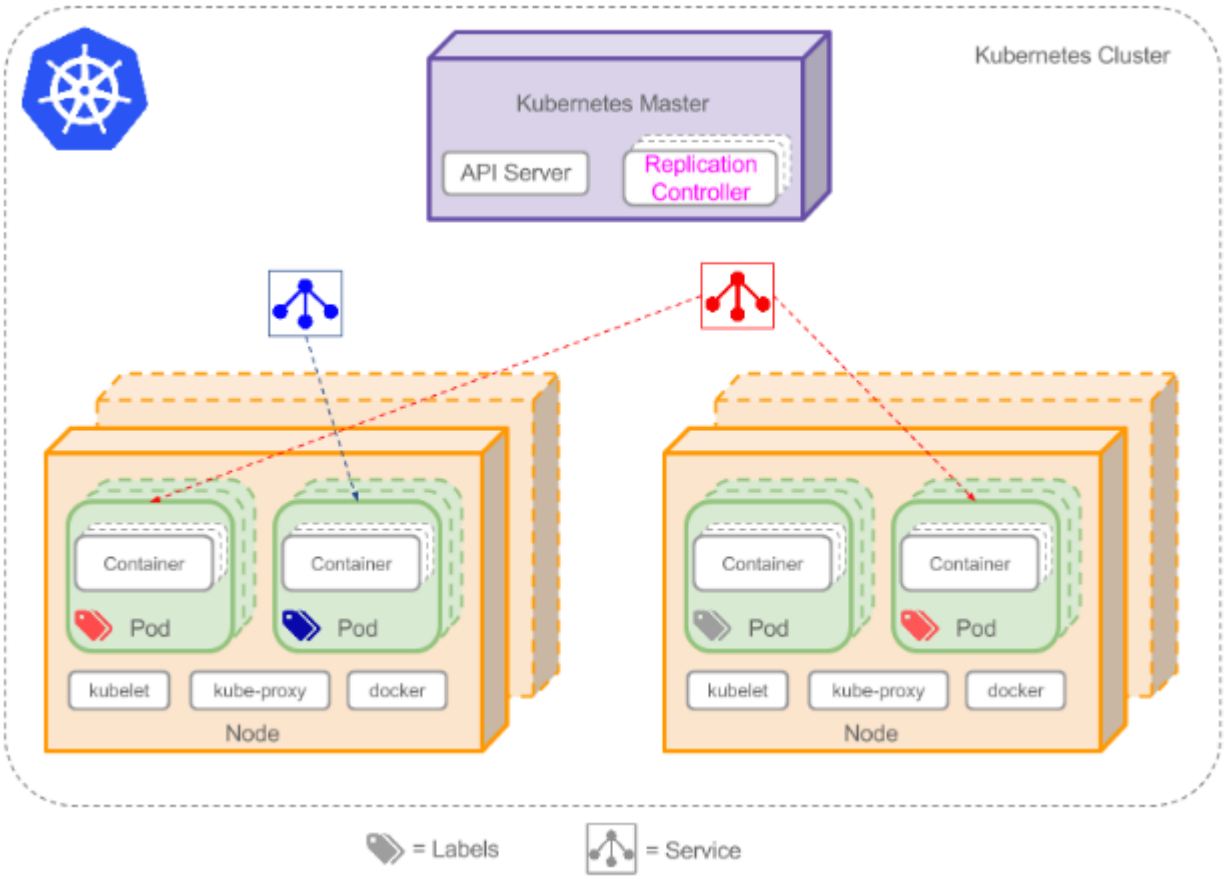
部署规划

这里我用3台服务器搭建一个简单的集群：
192.168.10.147 # master节点(etcd,kubernetes-master)
192.168.10.148 # node节点(etcd,kubernetes-node,docker,flannel)
192.168.10.149 # node节点(etcd,kubernetes-node,docker,flannel)

由于kubernetes的进程较多，每个节点上的进程如图：





安装

- 1、分别先在两个node上安装docker

```
yum update

tee /etc/yum.repos.d/docker.repo <<EOF
[dockerrepo]
name=Docker Repository
baseurl=https://yum.dockerproject.org/repo/main/centos/7/
enabled=1
gpgcheck=1
gpgkey=https://yum.dockerproject.org/gpg
EOF

yum install docker-engine
```

- 2、在master安装

```
yum install kubernetes-master etcd -y
```

- 3、分别在两个node上安装

```
yum install kubernetes-node etcd flannel -y
```

- 4、etcd集群配置

在**master**节点上编辑**etcd**配置文件

```
vi /etc/etcd/etcd.conf

# [member]
ETCD_NAME=etcd1
ETCD_DATA_DIR="/var/lib/etcd/etcd1.etcd"
#ETCD_WAL_DIR=""
#ETCD_SNAPSHOT_COUNT="10000"
#ETCD_HEARTBEAT_INTERVAL="100"
#ETCD_ELECTION_TIMEOUT="1000"
ETCD_LISTEN_PEER_URLS="http://192.168.10.147:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.10.147:2379,http://127.0.0.1:2379"
CD_MAX_SNAPSHOTS="5"
#ETCD_MAX_WALS="5"
#ETCD_CORS=""
#
#[cluster]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.10.147:2380"
# if you use different ETCD_NAME (e.g. test), set ETCD_INITIAL_CLUSTER value for this name, i.e.
"test=http://..."
ETCD_INITIAL_CLUSTER="etcd1=http://192.168.10.147:2380,etcd2=http://192.168.10.148:2380,etcd3=htt
p://192.168.10.149:2380"
#ETCD_INITIAL_CLUSTER_STATE="new"
#ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.10.147:2379"
#ETCD_DISCOVERY=""
#ETCD_DISCOVERY_SRV=""
#ETCD_DISCOVERY_FALLBACK="proxy"
#ETCD_DISCOVERY_PROXY=""
#
#[proxy]
#ETCD_PROXY="off"
#ETCD_PROXY_FAILURE_WAIT="5000"
#ETCD_PROXY_REFRESH_INTERVAL="30000"
#ETCD_PROXY_DIAL_TIMEOUT="1000"
#ETCD_PROXY_WRITE_TIMEOUT="5000"
#ETCD_PROXY_READ_TIMEOUT="0"
#
#[security]
#ETCD_CERT_FILE=""
#ETCD_KEY_FILE=""
#ETCD_CLIENT_CERT_AUTH="false"
#ETCD_TRUSTED_CA_FILE=""
#ETCD_PEER_CERT_FILE=""
#ETCD_PEER_KEY_FILE=""
#ETCD_PEER_CLIENT_CERT_AUTH="false"
#ETCD_PEER_TRUSTED_CA_FILE=""
#
#[logging]
#ETCD_DEBUG="false"
# examples for -log-package-levels etcdserver=WARNING,security=DEBUG

#ETCD_LOG_PACKAGE_LEVELS=""
```

在**node1**上编辑配置文件

```
vi /etc/etcd/etcd.conf

# [member]
ETCD_NAME=etcd2
ETCD_DATA_DIR="/var/lib/etcd/etcd2"
#ETCD_WAL_DIR=""
#ETCD_SNAPSHOT_COUNT="10000"
#ETCD_HEARTBEAT_INTERVAL="100"
#ETCD_ELECTION_TIMEOUT="1000"
ETCD_LISTEN_PEER_URLS="http://192.168.10.148:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.10.148:2379,http://127.0.0.1:2379"
#ETCD_MAX_SNAPSHOTS="5"
#ETCD_MAX_WALS="5"
#ETCD_CORS=""
#
#[cluster]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.10.148:2380"
# if you use different ETCD_NAME (e.g. test), set ETCD_INITIAL_CLUSTER value for this name, i.e.
"test=http://..."
ETCD_INITIAL_CLUSTER="etcd1=http://192.168.10.147:2380,etcd2=http://192.168.10.148:2380,etcd3=htt
p://192.168.10.149:2380"
#ETCD_INITIAL_CLUSTER_STATE="new"
#ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.10.148:2379"
#ETCD_DISCOVERY=""
#ETCD_DISCOVERY_SRV=""
#ETCD_DISCOVERY_FALLBACK="proxy"
#ETCD_DISCOVERY_PROXY=""
#
#[proxy]
#ETCD_PROXY="off"
#ETCD_PROXY_FAILURE_WAIT="5000"
#ETCD_PROXY_REFRESH_INTERVAL="30000"
#ETCD_PROXY_DIAL_TIMEOUT="1000"
#ETCD_PROXY_WRITE_TIMEOUT="5000"
#ETCD_PROXY_READ_TIMEOUT="0"
#
#[security]
#ETCD_CERT_FILE=""
#ETCD_KEY_FILE=""
#ETCD_CLIENT_CERT_AUTH="false"
#ETCD_TRUSTED_CA_FILE=""
#ETCD_PEER_CERT_FILE=""
#ETCD_PEER_KEY_FILE=""
#ETCD_PEER_CLIENT_CERT_AUTH="false"
#ETCD_PEER_TRUSTED_CA_FILE=""
#
#[logging]
#ETCD_DEBUG="false"
# examples for -log-package-levels etcdserver=WARNING,security=DEBUG
#ETCD_LOG_PACKAGE_LEVELS=""
```

在**node2**上编辑配置文件

```
vi /etc/etcd/etcd.conf

# [member]
ETCD_NAME=etcd3
ETCD_DATA_DIR="/var/lib/etcd/etcd3"
#ETCD_WAL_DIR=""
#ETCD_SNAPSHOT_COUNT="10000"
#ETCD_HEARTBEAT_INTERVAL="100"
#ETCD_ELECTION_TIMEOUT="1000"
ETCD_LISTEN_PEER_URLS="http://192.168.10.149:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.10.149:2379,http://127.0.0.1:2379"
#ETCD_MAX_SNAPSHOTS="5"
#ETCD_MAX_WALS="5"
#ETCD_CORS=""
#
#[cluster]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.10.149:2380"
# if you use different ETCD_NAME (e.g. test), set ETCD_INITIAL_CLUSTER value for this name, i.e.
"test=http://..."
ETCD_INITIAL_CLUSTER="etcd1=http://192.168.10.147:2380,etcd2=http://192.168.10.148:2380,etcd3=htt
p://192.168.10.149:2380"
#ETCD_INITIAL_CLUSTER_STATE="new"
#ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.10.149:2379"
#ETCD_DISCOVERY=""
#ETCD_DISCOVERY_SRV=""
#ETCD_DISCOVERY_FALLBACK="proxy"
#ETCD_DISCOVERY_PROXY=""
#
#[proxy]
#ETCD_PROXY="off"
#ETCD_PROXY_FAILURE_WAIT="5000"
#ETCD_PROXY_REFRESH_INTERVAL="30000"
#ETCD_PROXY_DIAL_TIMEOUT="1000"
#ETCD_PROXY_WRITE_TIMEOUT="5000"
#ETCD_PROXY_READ_TIMEOUT="0"
#
#[security]
#ETCD_CERT_FILE=""
#ETCD_KEY_FILE=""
#ETCD_CLIENT_CERT_AUTH="false"
#ETCD_TRUSTED_CA_FILE=""
#ETCD_PEER_CERT_FILE=""
#ETCD_PEER_KEY_FILE=""
#ETCD_PEER_CLIENT_CERT_AUTH="false"
#ETCD_PEER_TRUSTED_CA_FILE=""
#
#[logging]
#ETCD_DEBUG="false"
# examples for -log-package-levels etcdserver=WARNING,security=DEBUG
#ETCD_LOG_PACKAGE_LEVELS=""
```

> 针对几个URLS做下简单的解释：
> [member]
> ETCD_NAME ：ETCD的节点名
> ETCD_DATA_DIR：ETCD的数据存储目录
> ETCD_SNAPSHOT_COUNTER：多少次的事务提交将触发一次快照
> ETCD_HEARTBEAT_INTERVAL：ETCD节点之间心跳传输的间隔，单位毫秒
> ETCD_ELECTION_TIMEOUT：该节点参与选举的最大超时时间，单位毫秒
> ETCD_LISTEN_PEER_URLS：该节点与其他节点通信时所监听的地址列表，多个地址使用逗号隔开，其格式可以划分为scheme://IP:PORT，这里的scheme可以是http、https
> ETCD_LISTEN_CLIENT_URLS：该节点与客户端通信时监听的地址列表
> [cluster]
> ETCD_INITIAL_ADVERTISE_PEER_URLS：该成员节点在整个集群中的通信地址列表，这个地址用来传输集群数据的地址。因此这个地址必须是可以连接集群中所有的成员的。
> ETCD_INITIAL_CLUSTER：配置集群内部所有成员地址，其格式为：
> ETCD_NAME=ETCD_INITIAL_ADVERTISE_PEER_URLS，如果有多个使用逗号隔开
> ETCD_ADVERTISE_CLIENT_URLS：广播给集群中其他成员自己的客户端地址列表

至此**etcd**集群就部署完了，然后每个节点上启动

```
systemctl start kube-apiserver
```

验证：

```
[root@k8s1 ~]# etcdctl cluster-health
member 35300bfb5308e02c is healthy: got healthy result from http://192.168.10.147:2379
member 776c306b60e6f972 is healthy: got healthy result from http://192.168.10.149:2379
member a40f86f061be3fbe is healthy: got healthy result from http://192.168.10.148:2379
```

- 5、kubernetes master安装

修改**apiserver**配置文件

```
[root@k8s1 ~]# vi /etc/kubernetes/apiserver
###
# kubernetes system config
#
# The following values are used to configure the kube-apiserver
#

# The address on the local server to listen to.
# KUBE_API_ADDRESS="--insecure-bind-address=127.0.0.1"
KUBE_API_ADDRESS="--address=0.0.0.0"

# The port on the local server to listen on.
KUBE_API_PORT="--port=8080"

# Port minions listen on
KUBELET_PORT="--kubelet-port=10250"

# Comma separated list of nodes in the etcd cluster
KUBE_ETCD_SERVERS="--etcd-
servers=http://192.168.10.147:2379,http://192.168.10.148:2379,http://192.168.10.149:2379"

# Address range to use for services
KUBE_SERVICE_ADDRESSES="--service-cluster-ip-range=10.254.0.0/16"

# default admission control policies
KUBE_ADMISSION_CONTROL="--admission-
control=NamespaceLifecycle,NamespaceExists,LimitRanger,SecurityContextDeny,ServiceAccount,Resourc
eQuota"

# Add your own!
KUBE_API_ARGS=""
```

配置**controller-manager** 暂时不做修改

```
[root@k8s1 etcd]# vi /etc/kubernetes/controller-manager
###
# The following values are used to configure the kubernetes controller-manager

# defaults from config and apiserver should be adequate

# Add your own!
KUBE_CONTROLLER_MANAGER_ARGS=""
```

启动**Master**上的三个服务

```
systemctl start kube-apiserver
systemctl start kube-controller-manager
systemctl start kube-scheduler
systemctl enable kube-apiserver
systemctl enable kube-controller-manager
systemctl enable kube-scheduler
```

- 6、kubernetes node安装部署

修改节点**config**配置文件

```
[root@k8s1 ~]# vi /etc/kubernetes/config
###
# kubernetes system config
#
# The following values are used to configure various aspects of all
# kubernetes services, including
#
#   kube-apiserver.service
#   kube-controller-manager.service
#   kube-scheduler.service
#   kubelet.service
#   kube-proxy.service
# logging to stderr means we get it in the systemd journal
KUBE_LOGTOSTDERR="--logtostderr=true"

# journal message level, 0 is debug
KUBE_LOG_LEVEL="--v=0"

# Should this cluster be allowed to run privileged docker containers
KUBE_ALLOW_PRIV="--allow-privileged=false"

# How the controller-manager, scheduler, and proxy find the apiserver
KUBE_MASTER="--master=http://192.168.10.147:8080"
~
```

修改**kubelet**配置

```
[root@k8s1 ~]# vi /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=127.0.0.1"

# 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=192.168.10.148"

# location of the api-server
KUBELET_API_SERVER="--api-servers=http://192.168.10.147:8080"

# pod infrastructure container
KUBELET_POD_INFRA_CONTAINER="--pod-infra-container-image=registry.access.redhat.com/rhel7/pod-
infrastructure:latest"

# Add your own!
KUBELET_ARGS=""
```

分别启动**kubernetes node**服务

```
systemctl start kubelet
systemctl start kube-proxy
systemctl enable kubelet
systemctl enable kube-proxy
```

网络配置

这里网络部分是以插件的形式配置在kubernetes集群中，这里选用flannel。

- 1、安装flannel

上述步骤已经在node上安装
yum install flannel -y

- 2、配置flannel

```
[root@k8s1 ~]# vi /etc/sysconfig/flanneld

FLANNEL_ETCD_KEY="/atomic.io
# Flanneld configuration options

# etcd url location.  Point this to the server where etcd runs
FLANNEL_ETCD="http://192.168.10.147:2379"

# etcd config key.  This is the configuration key that flannel queries
# For address range assignment
FLANNEL_ETCD_KEY="/coreos.com/network"

# Any additional options that you want to pass
#FLANNEL_OPTIONS=""
```

- 3、为flannel创建分配的网络

```
# 只在master上etcd执行
etcdctl mk /coreos.com/network/config '{"Network": "10.1.0.0/16"}'
# 若要重新建，先删除
etcdctl rm /coreos.com/network/ --recursive
```

> 重置docker0网桥的配置
> 删除docker启动时默认创建的docker0网桥，flannel启动时会获取到一个网络地址，并且配置docker0的IP地址，作为该网络的网关地址，如果此时docker0上配置有IP地址，那么flannel将会启动失败。
> `ip link del docker0`

检查

在master上执行下面，检查kubernetes的状态

```
[root@k8s1 ~]# kubectl get nodes
NAME             STATUS    AGE
192.168.10.148   Ready     3h
192.168.10.149   Ready     3h
```

在master上执行下面，检查etcd的状态

```
[root@k8s1 ~]# etcdctl  member list
35300bfb5308e02c: name=etcd1 peerURLs=http://192.168.10.147:2380
clientURLs=http://192.168.10.147:2379
776c306b60e6f972: name=etcd3 peerURLs=http://192.168.10.149:2380
clientURLs=http://192.168.10.149:2379
a40f86f061be3fbe: name=etcd2 peerURLs=http://192.168.10.148:2380
clientURLs=http://192.168.10.148:2379
```

> centos7查看日志命令： journalctl -xe 或者 systemctl status flanneld.service flanneld对应改成你的项目