kubernetes高可用部署文档

参考：https://www.kubernetes.org.cn/4948.html

1.13.0版本的kubeadm有了一个较大的发展，kubeadm配置文件的API已经进入了beta阶段。
并且kubeadm已经实验性的启用了直接通过kubeadm join命令构建HA master的功能，很大程度上简化了HA master的构建过程。
另一方面，heapster项目已经被废弃并从1.13.0版本开始停止支持，未来建议使用metrics和Prometheus代替。

本系列从1.13.0版本开始将直接使用kubeadm join实现HA master，也不再提供heapster插件，而是使用metrics替代之。
本方案中升级内核是必要的，为私有仓库单独准备一台机器也是必要的，如果私自跳过步骤或调整结构，个人对脚本运行的结果不做保证

本文中的自动化部署脚本可以在Lentil1016/kubeadm-ha找到，欢迎Star/Fork/提issue和PR。
在我的环境上进行示例自动化部署的录像可以在该链接查看

集群方案：

- 发行版：CentOS 7
- 容器运行时：Docker-18.09.2-ce（应用yum安装的版本，建议用18.06的版本）
- 内核： 4.20.10-1.el7.elrepo.x86_64
- 版本：Kubernetes: 1.13.0
- 网络方案：Calico（更换网络插件，而非以前的flannel）
- kube-proxy mode：IPVS
- master高可用方案：keepalived LVS（使用keepalived+LVS，作为高可用及负载均衡方案）
- DNS插件：CoreDNS
- metrics插件：metrics-server
- 界面：kubernetes-dashboard
- ingress控制器：traefik

Kubernetes集群搭建

集群结构摘要

此处为举列说明，假定各个机器的主机信息以及IP分布如下，需要额外说明的是，由于私有仓库需要占用80端口，与ingress controller冲突，因此为私有仓库单独准备一台机器是必要的：

Host Name	Role	IP
registry	image registry	192.168.255.160
master01	master	192.168.255.161
master02	master	192.168.255.162
master03	master	192.168.255.163
-	Virtual IP	192.168.255.164
node01	worker	192.168.255.165

node02	worker	192. 168. 255. 166

进行系统配置

rpm

- selinux
- SwapKubernetes 1.8
- linux swapswappiness
- L2iptablesFORWARDCNINetwork Plugin Requirements
- issue#19
- IPVS

```
#
# Selinux/firewalld
systemctl stop firewalld
systemctl disable firewalld
setenforce 0
sed -i "s/SELINUX=enforcing/SELINUX=disabled/g" /etc/selinux/config

#
swapoff -a
yes | cp /etc/fstab /etc/fstab_bak
cat /etc/fstab_bak |grep -v swap > /etc/fstab

# IPTablescore
echo """
vm.swappiness = 0
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
""" > /etc/sysctl.conf
sysctl -p

#
yum install -y ntpdate
ntpdate -u ntp.api.bz

#
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-2.el7.elrepo.noarch.rpm
yum --enablerepo=elrepo-kernel install kernel-ml-devel kernel-ml -y

# 4.14
grub2-editenv list

#
reboot

# IPVS
uname -a
cat > /etc/sysconfig/modules/ipvs.modules <<EOF
#!/bin/bash
ipvs_modules="ip_vs ip_vs_lc ip_vs_wlc ip_vs_rr ip_vs_wrr ip_vs_lblc
ip_vs_lblcr ip_vs_dh ip_vs_sh ip_vs_fo ip_vs_nq ip_vs_sed ip_vs_ftp
nf_conntrack"
for kernel_module in \${ipvs_modules}; do
 /sbin/modinfo -F filename \${kernel_module} > /dev/null 2>&1
 if [ $? -eq 0 ]; then
 /sbin/modprobe \${kernel_module}
 fi
done
EOF
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash
/etc/sysconfig/modules/ipvs.modules && lsmod | grep ip_vs
```

sysctl -pcentos7bridge-nf-call-ip6tablesNo such file or directory

KubernetesMacuuidHostnameMacuuid

```
# UUIDMac
cat /sys/class/dmi/id/product_uuid
ip link
```

安装配置Docker

Docker1.13iptables filterFOWARDKubernetesNodePoddockeriptables

```
# docker
# docker
yum install -y yum-utils device-mapper-persistent-data lvm2
yum-config-manager \
 --add-repo \
 https://download.docker.com/linux/centos/docker-ce.repo

yum makecache fast
yum install -y docker-ce

# systemctlDocker
sed -i "13i ExecStartPost=/usr/sbin/iptables -P FORWARD ACCEPT"
/usr/lib/systemd/system/docker.service

# docker
systemctl daemon-reload
systemctl enable docker
systemctl start docker
```

安装私有镜像库

dockerK8sinsecure-registryhosts

```
# http
# Docker
mkdir -p /etc/docker
echo -e '{\n"insecure-registries":["k8s.gcr.io", "gcr.io", "quay.io"]\n}' >
/etc/docker/daemon.json
systemctl restart docker

# registryIP
REGISTRY_HOST="192.168.255.160"
# Hosts
yes | cp /etc/hosts /etc/hosts_bak
cat /etc/hosts_bak|grep -vE '(gcr.io|harbor.io|quay.io)' > /etc/hosts
echo """
$REGISTRY_HOST gcr.io harbor.io k8s.gcr.io quay.io """ >> /etc/hosts
```

https://pan.baidu.com/s/1ZdmgnrYGVobc22FX__vYwg 69gq registryregistry

```
# registry
docker load < k8s-repo-1.13.0
docker run --restart=always -d -p 80:5000 --name repo
harbor.io:1180/system/k8s-repo:v1.13.0
```

```
$ docker run -it harbor.io:1180/system/k8s-repo:v1.13.0 list

k8s.gcr.io/coredns:1.2.6 k8s.gcr.io/etcd:3.2.24
k8s.gcr.io/kube-apiserver:v1.13.0
k8s.gcr.io/kube-controller-manager:v1.13.0
k8s.gcr.io/kube-proxy:v1.13.0
k8s.gcr.io/kube-scheduler:v1.13.0 k8s.gcr.io/pause:3.1
k8s.gcr.io/traefik:1.7.5
k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.0
gcr.io/kubernetes-helm/tiller:v2.12.0
k8s.gcr.io/addon-resizer:1.8.4
k8s.gcr.io/metrics-server-amd64:v0.3.1
quay.io/calico/cni:v3.3.2
quay.io/calico/node:v3.3.2
quay.io/calico/typha:v3.3.2
```

安装配置kubernetes

基本安装

首先下载链接：链接：https://pan.baidu.com/s/1t3EWAt4AET7JaIVIbz-zHQ 提取码：djnf ，并放置在k8s各个master和worker主机上

```
# master & workerkubernetes
yum install -y socat keepalived ipvsadm
cd /path/to/downloaded/file
tar -xzvf k8s-v1.13.0-rpms.tgz
cd k8s-v1.13.0
rpm -Uvh * --force
systemctl enable kubelet
kubeadm version -o short
```

配置免密码登陆

```
# master01ssh
ssh-keygen
#
cat ~/.ssh/id_rsa.pub
#
$ cat ~/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDSkRerh8iYvC7e0u9xs2hW/bXN9Jz8k+DdC4Rrl322IJ
xlj3/lMn8ROfYIfrmbEkNrgyhg3W/0N1W6+54locecrHC2gIvh62bdEHLFUetnU902ykeO1XR5
lvfO3TOgGgwC05x+wFdYhGSoJGytFDvCKytAxjgS7xVraYCjtviHv9aTUXamnTJg7vvT5QViLG
2qUz+O3WLUDRHB9Lu+nErdHXIEtMtCsnTJWXgBsAw6IarMQ+12UtUXr5kW8itWD8EGLMYxzrKv
Bqr+H18R50DOuDlZGDETEkHGGZwUfGRYnRf8NfNZfFzPUohxnL6Gbr8z2evW2daWJChE29r+le
K9 root@master01
```

master01 master02 master03root ~/.ssh/authorized_keys **master01**

master01master01 master02 master03()

部署HA Master

HA Mastermaster-1IP

```
# HA master
cd ~/

# (CIDRIP)
echo """
CP0_IP=192.168.255.161
CP1_IP=192.168.255.162
CP2_IP=192.168.255.163
VIP=192.168.255.164
NET_IF=ens33
CIDR=10.244.0.0/16
""" > ./cluster-info

bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Lentil1016/kubeadm-ha/1.13.0/kubeha-gen.
sh)"
# 210
```

可以在该链接查看安装全过程的录像，安装结束后会打印出如下的信息，最后一行为加入集群的命令。

```
$ kubectl  get node -o wide
NAME       STATUS   ROLES    AGE   VERSION   INTERNAL-IP
EXTERNAL-IP   OS-IMAGE               KERNEL-VERSION
CONTAINER-RUNTIME
master01   Ready    master   22h   v1.13.0   192.168.255.161
<none>        CentOS Linux 7 (Core)   4.20.10-1.el7.elrepo.x86_64
docker://18.6.0
master02   Ready    master   22h   v1.13.0   192.168.255.162
<none>        CentOS Linux 7 (Core)   4.20.10-1.el7.elrepo.x86_64
docker://18.6.0
master03   Ready    master   22h   v1.13.0   192.168.255.163
<none>        CentOS Linux 7 (Core)   4.20.10-1.el7.elrepo.x86_64
docker://18.6.0
```

加入work node

join commandkubeadm

```
## token $ kubeadm token create --print-join-command
kubeadm join 192.168.255.164:6443 --token f18mw2.kfhnjrhx48ymjztr
--discovery-token-ca-cert-hash
sha256:41974beb6e69b75eaacbaccc6d6bd6b874b5e03a0cb3559ebebd552d433534d0
## node kubernetes
$ kubeadm join 192.168.255.164:6443 --token f18mw2.kfhnjrhx48ymjztr
```

```
--discovery-token-ca-cert-hash
sha256:41974beb6e69b75eaacbaccc6d6bd6b874b5e03a0cb3559ebebd552d433534d0
[preflight] Running pre-flight checks
 [WARNING SystemVerification]: this Docker version is not on the list of
validated versions: 18.09.2. Latest validated version: 18.06
[discovery] Trying to connect to API Server "192.168.255.164:6443"
[discovery] Created cluster-info discovery client, requesting info from
"https://192.168.255.164:6443"
[discovery] Requesting info from "https://192.168.255.164:6443" again to
validate TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS
certificate validates against pinned roots, will use API Server
"192.168.255.164:6443"
[discovery] Successfully established connection with API Server
"192.168.255.164:6443"
[join] Reading configuration from the cluster...
[join] FYI: You can look at this config file with 'kubectl -n kube-system
get cm kubeadm-config -oyaml'
[kubelet] Downloading configuration for the kubelet from the
"kubelet-config-1.13" ConfigMap in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock"
to the Node API object "node01" as an annotationThis 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 master to see this node join the cluster.
##
$ kubectl  get node -o wide
NAME       STATUS   ROLES    AGE   VERSION   INTERNAL-IP
EXTERNAL-IP   OS-IMAGE               KERNEL-VERSION
CONTAINER-RUNTIME
master01   Ready    master   22h   v1.13.0   192.168.255.161
<none>        CentOS Linux 7 (Core)   4.20.10-1.el7.elrepo.x86_64
docker://18.6.0
master02   Ready    master   22h   v1.13.0   192.168.255.162
<none>        CentOS Linux 7 (Core)   4.20.10-1.el7.elrepo.x86_64
docker://18.6.0
master03   Ready    master   22h   v1.13.0   192.168.255.163
<none>        CentOS Linux 7 (Core)   4.20.10-1.el7.elrepo.x86_64
docker://18.6.0
node01     Ready    <none>   17h   v1.13.0   192.168.255.165
<none>        CentOS Linux 7 (Core)   4.20.10-1.el7.elrepo.x86_64
docker://18.6.0
node02     Ready    <none>   17h   v1.13.0   192.168.255.166
<none>        CentOS Linux 7 (Core)   4.20.10-1.el7.elrepo.x86_64
docker://18.6.0
```