

Kubernetes集群部署篇

一、环境准备

1、机器环境前置条件

当前演示准备3台虚拟机环境，或者是3台阿里云服务器都可

k8s-master01: 此机器用来安装k8s-master的操作环境

k8s-node01: 此机器用来安装k8s node节点的环境

k8s-node02: 此机器用来安装k8s node节点的环境

修改网络配置，确保虚拟机网络连通，Xshell连接

节点CPU核数必须是： ≥ 2 核，否则k8s无法启动

DNS网络：最好设置为本地网络连通的DNS,否则网络不通，无法下载一些镜像

linux内核：linux内核必须是4版本以上，因此必须把linux核心进行升级

前置环境准备完成

2、依赖环境安装配置

注：每一台机器都要安装此依赖环境

1、给每一台机器设置主机名

```
hostnamectl set-hostname k8s-master01
hostnamectl set-hostname k8s-node01
hostnamectl set-hostname k8s-node02
```

查看主机名

```
hostname
```

配置IP host映射关系

```
vi /etc/hosts
192.168.66.10 k8s-master01
192.168.66.11 k8s-node01
192.168.66.12 k8s-node02
```

2、安装依赖环境

```
yum install -y conntrack ntpdate ntp ipvsadm ipset jq iptables curl sysstat  
libseccomp wget vim net-tools git iproute lrzsz bash-completion tree bridge-  
utils unzip bind-utils gcc
```

3、安装iptables，启动iptables，设置开机自启，清空iptables规则，保存当前规则到默认规则

关闭防火墙并设置开机禁用防火墙

```
systemctl stop firewalld && systemctl disable firewalld
```

置空iptables

```
yum -y install iptables-services && systemctl start iptables && systemctl enable  
iptables && iptables -F && service iptables save
```

4、关闭selinux

关闭swap分区【虚拟内存】并且永久关闭虚拟内存

```
swapoff -a && sed -i '/ swap / s/^(.*)$/#\1/g' /etc/fstab
```

关闭selinux

```
setenforce 0 && sed -i 's/^SELINUX=.*$/SELINUX=disabled/' /etc/selinux/config
```

5、升级Linux内核为4.4版本

```
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-4.el7.elrepo.noarch.rpm
```

安装内核

```
yum --enablerepo=elrepo-kernel install -y kernel-lt
```

查询已安装的内核

```
rpm -qa | grep kernel
```

查看默认启动项

```
awk -F\' ' $1=="menuentry " {print $2}' /etc/grub2.cfg
```

上面命令中找到新内核的名称，替换后执行下面命令，令开机从新内核启动

```
grub2-set-default 'CentOS Linux (4.4.230-1.el7.elrepo.x86_64) 7 (Core)'
```

重启机器 注意：设置完内核后，需要重启服务器才会生效

```
reboot
```

重启后查询内核

```
uname -r
```

6、调整内核参数，对于k8s

```
cat > kubernetes.conf <<EOF
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1
net.ipv4.ip_forward=1
net.ipv4.tcp_tw_recycle=0
vm.swappiness=0
vm.overcommit_memory=1
vm.panic_on_oom=0
fs.inotify.max_user_instances=8192
fs.inotify.max_user_watches=1048576
fs.file-max=52706963
fs.nr_open=52706963
net.ipv6.conf.all.disable_ipv6=1
net.netfilter.nf_conntrack_max=2310720
EOF
```

将优化内核文件拷贝到/etc/sysctl.d/文件夹下，这样优化文件开机的时候能够被调用

```
cp kubernetes.conf /etc/sysctl.d/kubernetes.conf
```

手动刷新，让优化文件立即生效

```
sysctl -p /etc/sysctl.d/kubernetes.conf
```

上面手动刷新会有这个异常，忽略即可，是因为有一个组件没有加载

```
sysctl: cannot stat /proc/sys/net/netfilter/nf_conntrack_max: No such file or
directory
```

7、调整系统临时区（如果已设置可略过）

设置系统时区为中国/上海

```
timedatectl set-timezone Asia/Shanghai
```

将当前的 UTC 时间写入硬件时钟

```
timedatectl set-local-rtc 0
```

重启依赖于系统时间的服务

```
systemctl restart rsyslog  
systemctl restart crond
```

8、关闭系统不需要的服务

```
systemctl stop postfix && systemctl disable postfix
```

9、设置日志保存方式

创建保存日志的目录

```
mkdir /var/log/journal
```

创建配置文件存放目录

```
mkdir /etc/systemd/journald.conf.d
```

创建配置文件

```
cat > /etc/systemd/journald.conf.d/99-prophet.conf <<EOF  
[Journal]  
Storage=persistent  
Compress=yes  
SyncIntervalSec=5m  
RateLimitInterval=30s  
RateLimitBurst=1000  
SystemMaxUse=10G  
SystemMaxFileSize=200M  
MaxRetentionSec=2week  
ForwardToSyslog=no  
EOF
```

重启systemd journald的配置

```
systemctl restart systemd-journald
```

10、打开文件数调整 (可忽略，不执行)

```
echo "* soft nofile 65536" >> /etc/security/limits.conf  
echo "* hard nofile 65536" >> /etc/security/limits.conf
```

11、kube-proxy 开启 ipvs 前置条件

```
modprobe br_netfilter
cat > /etc/sysconfig/modules/ipvs.modules <<EOF
#!/bin/bash
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack_ipv4
EOF
```

使用lsmod命令查看这些文件是否被引导

```
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash
/etc/sysconfig/modules/ipvs.modules && lsmod | grep -e ip_vs -e
nf_conntrack_ipv4
```

3、docker部署

1、安装依赖

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

添加yum软件源

(配置一个稳定 (stable) 的仓库、仓库配置会保存到/etc/yum.repos.d/docker-ce.repo文件中)

以下两个选一即可

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-
ce.repo
```

```
yum-config-manager --add-repo http://mirrors.aliyun.com/docker-
ce/linux/centos/docker-ce.repo
```

更新Yum安装的相关Docker软件包&安装Docker CE

```
yum update -y && yum install docker-ce
```

测试是否安装成功

```
docker -v
```

yum下载失败解决示例

如果出现下载失败的问题，例如下面这样

```
Error downloading packages:
3:docker-ce-19.03.12-3.el7.x86_64: [Errno 256] No more mirrors to try.
containerd.io-1.2.13-3.2.el7.x86_64: [Errno 256] No more mirrors to try
```

清除yum缓存

```
yum clean all
```

执行上面添加yum软件源的命令切换源

查看仓库配置是否修改生效

```
vi /etc/yum.repos.d/docker-ce.repo
```

重新执行如下更新下载命令

```
yum update -y && yum install docker-ce
```

2、设置docker daemon文件

创建/etc/docker目录

```
mkdir /etc/docker
```

更新daemon.json文件+配置阿里云容器镜像服务

```
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "registry-mirrors": ["https://tda5gnvq.mirror.aliyuncs.com"],
  "log-driver": "json-file",
  "log-opts": {"max-size": "100m"}
}
EOF
```

注意：一定注意编码问题，出现错误：查看命令：`journalctl -amu docker` 即可发现错误

创建，存储docker配置文件

```
mkdir -p /etc/systemd/system/docker.service.d
```

3、重启docker服务并设置docker为开机自启动

```
systemctl daemon-reload && systemctl restart docker && systemctl enable docker
```

docker images 测试一下，注：如果无法启动说明配置文件有问题

4、kubeadm[一键安装k8s]

安装kubernetes的时候，需要安装kubelet, kubeadm等包，但k8s官网给的yum源是

packages.cloud.google.com，国内访问不了，此时我们可以使用阿里云的yum仓库镜像

1、配置阿里云的yum仓库镜像

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=kubernetes
baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
        http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF
```

2、安装kubeadm、kubelet、kubectl

```
yum install -y kubeadm-1.15.1 kubelet-1.15.1 kubectl-1.15.1
```

将kubelet设置为开机自启动，并启动 kubelet

```
systemctl enable kubelet && systemctl start kubelet
```

二、集群安装

1、各节点安装基础镜像

展示出安装k8s需要哪些基础镜像

```
kubeadm config images list
```






kubeadm 初始化k8s集群的时候，会从gce Google云中下载（pull）相应的镜像，

当然可以选择执行 docker pull 根据上面命令展示出来的镜像，挨个拉取镜像

但是镜像相对比较大，下载可能会失败，所以导入下载好的镜像包到本地docker镜像仓库

注：下面的步骤在master主机执行即可

1、rz上传依赖镜像 tar包

 docker-compose	2020/7/15 11:03	文件	11,463 KB
 flannel.tar	2020/7/22 11:01	好压 TAR 压缩文件	52,487 KB
 harbor-offline-installer-v1.2.0.tgz	2020/7/15 11:57	好压 TGZ 压缩文件	478,957 KB
 kubeadm-basic.images.tar.gz	2020/7/15 12:21	好压 GZ 压缩文件	235,607 KB
 kube-flannel.yml	2020/7/22 19:27	YML 文件	14 KB

解压镜像

```
tar -zxvf kubeadm-basic.images.tar.gz
```

解压结束后如图所示，当然可以也使用docker load -i xxx.tar挨个把tar包的镜像导入到本地仓库

```
[root@k8s-master01 ~]# cd kubeadm-basic.images/
[root@k8s-master01 kubeadm-basic.images]# ll
total 815744
-rw-r--r-- 1 root root 208394752 Aug  5  2019 apiserver.tar
-rw-r--r-- 1 root root 40542720 Aug  5  2019 coredns.tar
-rw-r--r-- 1 root root 258365952 Aug  5  2019 etcd.tar
-rw-r--r-- 1 root root 160290304 Aug  5  2019 kubec-con-man.tar
-rw-r--r-- 1 root root 754176 Aug  5  2019 pause.tar
-rw-r--r-- 1 root root 84282368 Aug  5  2019 proxy.tar
-rw-r--r-- 1 root root 82675200 Aug  5  2019 scheduler.tar
```

2、编写脚本文件，导入镜像包到本地docker镜像仓库

任意目录下创建sh脚本文件

```
touch image-load.sh
```

vi image-load.sh 导入下面脚本代码

注意 /root/kubeadm-basic.images 是你本机镜像解压的目录位置

```
#!/bin/bash
ls /root/kubeadm-basic.images > /tmp/images-list.txt
cd /root/kubeadm-basic.images
for i in $(cat /tmp/images-list.txt)
do
    docker load -i $i
done
rm -rf /tmp/images-list.txt
```

修改权限，可执行权限

```
chmod 755 image-load.sh
```

运行脚本文件，开始执行镜像导入

```
./image-load.sh
```


查看本地仓库镜像

```
docker images
```

3、传输文件及镜像到其他node节点

注：其他机器也需要安装K8S的环境，所以传输镜像tar包及脚本文件到其他node节点

注：在当前master主机执行

传递到node01节点

```
scp -r image-load.sh kubeadm-basic.images root@k8s-node01:/root/
```

传递到node02节点

```
scp -r image-load.sh kubeadm-basic.images root@k8s-node02:/root/
```

其他节点依次执行sh脚本，导入镜像

```
./image-load.sh
```

2、部署Kubernetes

通过部署yaml文件，来达到一键式部署

注：下面的步骤在master主机执行即可

1、拉取yaml资源配置文件

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

2、修改yaml资源文件

注：需要修改的部分如下图所示

```
localAPIEndpoint:
  advertiseAddress: 192.168.52.105 # 注意：修改配置文件的IP地址
kubernetesVersion: v1.15.1 #注意：修改版本号，必须和kubect1版本保持一致
networking:
  # 指定flannel模型通信 pod网段地址,此网段和flannel网段一致，不用变
  podSubnet: "10.244.0.0/16"
  serviceSubnet: "10.96.0.0/12"
  #添加如下内容指定使用ipvs网络进行通信，注意缩进
  ---
  apiVersion: kubeproxy.config.k8s.io/v1alpha1
  kind: kubeProxyConfiguration
  featureGates:
```

```
SupportIPVSProxyMode: true
mode: ipvs
```

3、初始化主节点，开始部署

```
kubeadm init --config=kubeadm-config.yaml --experimental-upload-certs | tee
kubeadm-init.log
```

注意：执行此命令，CPU核心数量必须大于1核，否则无法执行成功

kubernetes主节点初始化成功后，如下所示：

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, 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

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/

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

kubeadm join 192.168.52.105:6443 --token abcdef.0123456789abcdef \
--discovery-token-ca-cert-hash sha256:ea317791c651a7da3f27cac1bbe36bbfa4bc4a64b37ba56d4e9ee3a5c8368f28
```

4、按照K8S的提示执行如下命令

创建目录，保存连接配置缓存，认证文件

```
mkdir -p $HOME/.kube
```

拷贝集群管理配置文件

```
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

授权给配置文件

```
chown $(id -u):$(id -g) $HOME/.kube/config
```

查询node节点

```
kubectl get node
```






发现已经可以成功查询node节点信息了，但是节点的状态却是NotReady,不是Running的状态。原因是此时我们使

用ipvs+flannel的方式进行网络通信，但是flannel网络插件还没有部署，因此节点状态此时为NotReady

3、部署flannel网络插件

同样，拉取国外镜像较慢且容易失败，所以使用本地下载好的flannel镜像，将flannel.tar包上传，

注意：主从节点均需要上传并执行后续的导入和打标签操作

	docker-compose	2020/7/15 11:03	文件	11,463 KB
	flannel.tar	2020/7/22 11:01	好压 TAR 压缩文件	52,487 KB
	harbor-offline-installer-v1.2.0.tgz	2020/7/15 11:57	好压 TGZ 压缩文件	478,957 KB
	kubeadm-basic.images.tar.gz	2020/7/15 12:21	好压 GZ 压缩文件	235,607 KB
	kube-flannel.yml	2020/7/22 19:27	YML 文件	14 KB

将镜像tar包导入到本地镜像库

```
docker load -i flannel.tar
```

给镜像打标签，在本地仓库生成新的v1版本镜像






```
docker tag quay-mirror.qiniu.com/coreos/flannel:v0.12.0-amd64 flannel:v1
```

注：docker images 查看主从机器

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
flannel	v1	4e9f801d2217	4 months ago	52.7MB
quay-mirror.qiniu.com/coreos/flannel	v0.12.0-amd64	4e9f801d2217	4 months ago	52.7MB
k8s.gcr.io/kube-apiserver	v1.15.1	68c3eb07bfc3	12 months ago	207MB
k8s.gcr.io/kube-controller-manager	v1.15.1	d75082f1d121	12 months ago	159MB
k8s.gcr.io/kube-proxy	v1.15.1	89a062da739d	12 months ago	82.4MB
k8s.gcr.io/kube-scheduler	v1.15.1	b0b3c4c404da	12 months ago	81.1MB
k8s.gcr.io/coredns	1.3.1	eb516548c180	18 months ago	40.3MB
k8s.gcr.io/etcd	3.3.10	2c4adeb21b4f	20 months ago	258MB
k8s.gcr.io/pause	3.1	da86e6ba6ca1	2 years ago	742kB

注意：后面步骤在master主机执行即可

传入本地的yml文件 (修改为flannel:v1版本的)**

	docker-compose	2020/7/15 11:03	文件	11,463 KB
	flannel.tar	2020/7/22 11:01	好压 TAR 压缩文件	52,487 KB
	harbor-offline-installer-v1.2.0.tgz	2020/7/15 11:57	好压 TGZ 压缩文件	478,957 KB
	kubeadm-basic.images.tar.gz	2020/7/15 12:21	好压 GZ 压缩文件	235,607 KB
	kube-flannel.yml	2020/7/22 19:27	YML 文件	14 KB

yml部署flannel

```
kubectl create -f kube-flannel.yml
```

执行成功以后 `kubectl get node` 查看，可能仍然是NotReady状态，需要一些时间

可以通过 `kubectl get pod -n kube-system` 查看kube-system已经运行的系统pod里flannel是否运行

```
[root@k8s-master01 ~]# kubectl get pod -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE
coredns-5c98db65d4-h44zn           1/1     Running   0           24m
coredns-5c98db65d4-vhp6j           1/1     Running   0           24m
etcd-k8s-master01                   1/1     Running   0           23m
kube-apiserver-k8s-master01         1/1     Running   0           23m
kube-controller-manager-k8s-master01 1/1     Running   0           23m
kube-flannel-ds-amd64-plfh2         1/1     Running   0           4m12s
kube-proxy-6vrh6                    1/1     Running   0           24m
kube-scheduler-k8s-master01         1/1     Running   0           23m
```

查看时若处于Ready状态，说明网络组件已经部署成功

```
[root@k8s-master01 ~]# kubectl get node
NAME             STATUS    ROLES    AGE   VERSION
k8s-master01    Ready    master   20m   v1.15.1
```

附录：kubectl get pod -n kube-system 解释

所有k8s的资源都是通过镜像的方式做部署的，运行的模式就是通过pod体现，这些pod就是k8s的系统资源的pod

-n表示namespace 表示在kube-system这个命名空间下的pod的信息

可以追加 `-o wide` 查看pod详细信息

```
kubectl get pod -n kube-system -o wide
```

附录：发现通过flannel部署的pod都出现pending,ImagePullBackOff类似问题

查询一个pod的详细日志信息，找到错误原因（一般为镜像拉去失败）

```
kubectl describe pod kube-flannel-ds-amd64-jd67h -n kube-system
```

如果kube-flannel.yml文件修改过，则使用下面的命令来更新并部署

```
kubectl apply -f kube-flannel.yml
```

4、其他node节点加入

构建kubernetes主节点成功，会产生一个日志文件kubeadm-init.log，内容如下图所示：

（命令中指定了日志输出文件“tee kubeadm-init.log”）

加入主节点以及其余工作节点，执行安装日志中的命令即可

查看日志文件

```
cat kubeadm-init.log
```

下图红色部分给出的命令即是把其他节点加入进来的命令

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, 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

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/

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

kubeadm join 192.168.52.105:6443 --token abcdef.0123456789abcdef \
--discovery-token-ca-cert-hash sha256:ea317791c651a7da3f27cac1bbe36bbfa4bc4a64b37ba56d4e9ee3a5c8368f28
[root@k8s-master01 ~]#
```

复制命令到其他几个node节点进行执行即可(注意是在其他从机服务器上执行)

执行完毕，可以在**master主机**查看 `kubectl get node`

发现还有节点处于NotReady状态，是因为这些节点pod容器还处于初始化的状态，需要时间

当其他node节点加入进来且处于Ready状态，则K8S集群搭建成功

```
[root@k8s-master01 ~]# kubectl get node
NAME           STATUS    ROLES    AGE   VERSION
k8s-master01   Ready     master   105m  v1.15.1
k8s-node01     Ready     <none>   57m   v1.15.1
k8s-node02     Ready     <none>   57m   v1.15.1
```

-----完结撒花🌸🌸、(°▽°)ノ🌸-----