链滴

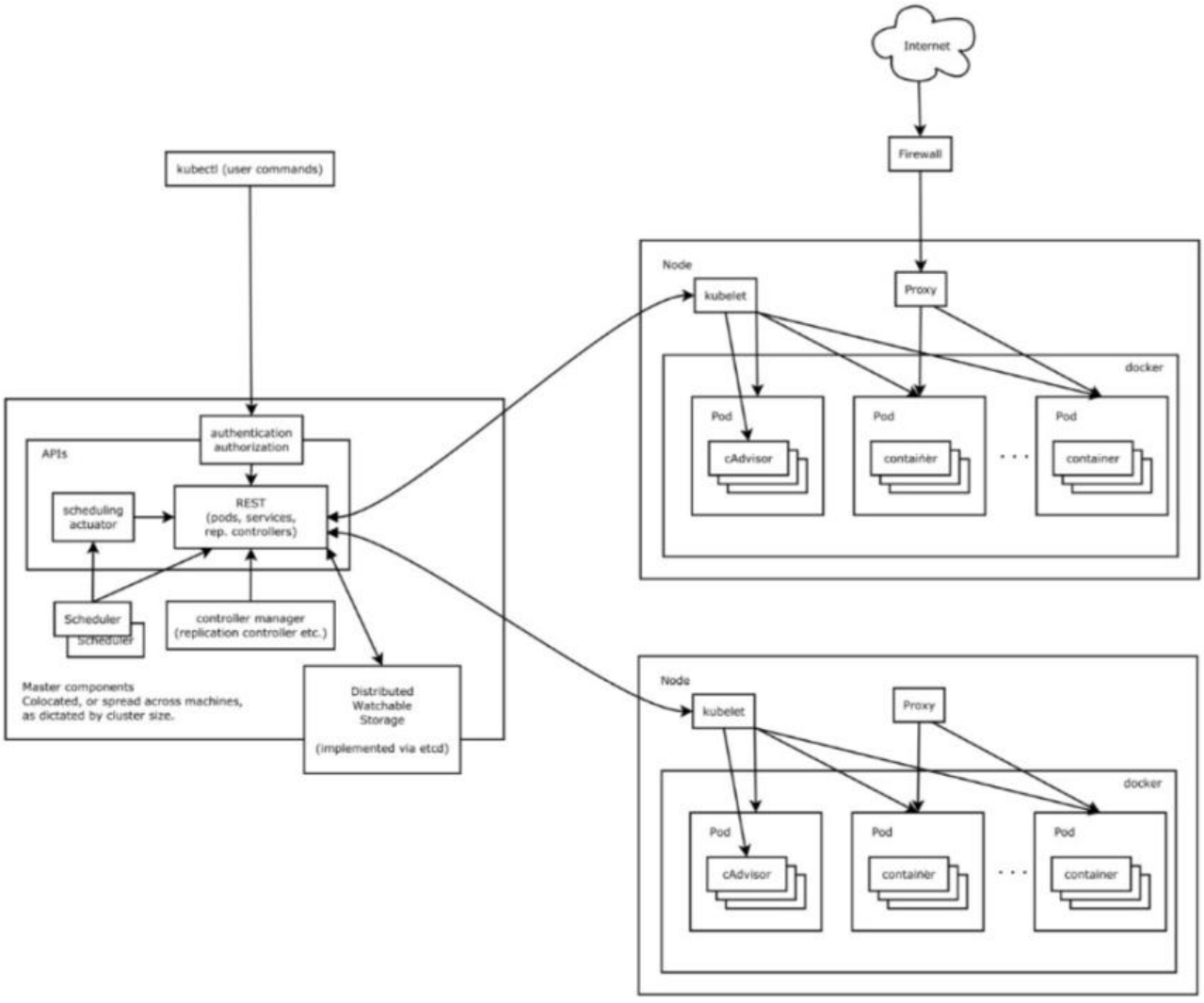
Kubernetes| 使用 Kubeadm 搭建 Kubernetes 集群

作者：jianzh5

原文链接：https://ld246.com/article/1586913078192

来源网站：链滴

Kubernetes架构



原文链接：Kubernetes| 使用 Kubeadm 搭建 Kubernetes 集群

Kubernetes 主要由以下几个核心组件组成：

- etcd 保存了整个集群的状态；
- kube-apiserver 提供了资源操作的唯一入口，并提供认证、授权、访问控制、API 注册和发现等机；
- kube-controller-manager 负责维护集群的状态，比如故障检测、自动扩展、滚动更新等；
- kube-scheduler 负责资源的调度，按照预定的调度策略将 Pod 调度到相应的机器上；
- kubelet 负责维持容器的生命周期，同时也负责 Volume（CVI）和网络（CNI）的管理；
- Container runtime 负责镜像管理以及 Pod 和容器的真正运行（CRI），默认的容器运行时为 Docker；
- kube-proxy 负责为 Service 提供 cluster 内部的服务发现和负载均衡；

今天我们先重点说一下kubelet组件，**kubelet 主要负责同容器运行时（比如 Docker 项目）打交道**而这个交互所依赖的，是一个称作 CRI（Container Runtime Interface）的远程调用接口，这个接定义了容器运行时的各项核心操作，比如：启动一个容器需要的所有参数。

此外，kubelet 还通过 gRPC 协议同一个叫作 Device Plugin 的插件进行交互。这个插件，是 Kubernetes 项目用来管理 GPU 等宿主机物理设备的主要组件，也是基于 Kubernetes 项目进行机器学习练、高性能作业支持等工作必须关注的功能。

而kubelet 的另一个重要功能，则是调用网络插件和存储插件为容器配置网络和持久化存储。这两个件与 kubelet 进行交互的接口，分别是 CNI（Container Networking Interface）和 CSI（Container Storage Interface）。

kubelet 完全是为了实现 Kubernetes 项目对容器的管理能力而实现的一个组件。

了解完Kubernetes架构后，我们今天使用Kubeadm部署一个Kubernetes集群。

使用Kubeadm部署Kubernetes集群很简单，只需要两步操作即可：kubeadm init，kubeadm join
当然在正式安装之前咱们先需要做一下基础准备！

基础环境准备

安装一个Kubernates最小集群需要三台机器，一台Master节点，两台Node节点，机器规划如下：

虚拟机版本 色	主机名	IP
centos7 aster	kubernetes-master	192.168.136.128
centos7 ode	kubernetes-node1	192.168.136.129
centos7 ode	kubernetes-node2	192.168.136.130

- 安装并启动Docker

安装过程略，可参看我之前的Docker系列文章，安装完成后使用 systemctl start docker 命令启动Docker

- 使用命令将docker服务设置开机启动

systemctl enable docker

- 查看docker 版本，确保各个节点安装的docker版本一致
- 关闭禁用各节点的防火墙

停止防火墙：systemctl stop firewalld.service

禁用防火墙：systemctl disable firewalld.service

查看防火墙状态：systemctl list-unit-files|grep firewalld.service

- 关闭各节点的selinux

编辑 /etc/selinux/config 文件并设置 SELINUX 的值为 disabled

- 关闭各节点的swap

如果不关闭kubernetes运行会出现错误， 即使安装成功了，node重启后也会出现kubernetes serve 运行错误。

sudo swapoff -a

- 给各节点添加kubernetes的yum源

进入目录 cd /etc/yum.repos.d/

```
[root@kubernetes-master yum.repos.d]# cat > kubernetes.repo << EOF
> [kubernetes]
> name=Kubernetes Repo
> baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
> gpgcheck=1
> gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
> enable=1
> EOF
```

内容如下，大家可以直接复制粘贴。

```
[kubernetes]
name=Kubernetes Repo
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
enable=1
EOF
```

- 查看yum源是否可用

yum repolist

```
[root@kubernetes-master yum.repos.d]# yum repolist
已加载插件：fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
kubernetes
kubernetes/primary
kubernetes
源标识                                                源名称
base/7/x86_64                                         CentOS-7 - Base
docker-ce-stable/x86_64                               Docker CE Stable - x86_64
extras/7/x86_64                                       CentOS-7 - Extras
kubernetes                                            Kubernetes Repo
updates/7/x86_64                                      CentOS-7 - Updates
repolist: 12,776
[root@kubernetes-master yum.repos.d]#
```

做好上面的准备工作后，我们来安装Kubeadm。

Kubeadm 安装

Master安装

- 修改master主机名为kubernetes-master

hostnamectl set-hostname kubernetes-master

- 卸载原kubeadm（若有）：

yum remove -y kubelet kubeadm kubectl

- 安装kubeadm：

yum install -y kubelet kubeadm kubectl

- 重启 docker，并启动 kubelet

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

- 查看kubelet状态：

systemctl status kubelet

如果此时执行 service status kubelet 命令，将得到 kubelet 启动失败的错误提示，请忽略此错误，为必须完成后续步骤中 kubeadm init 的操作，kubelet 才能正常启动

- 生成kubeadm配置文件kubeadm.yml

进入文件夹 cd /app/k8s，执行命令生成配置文件

kubeadm config print init-defaults --kubeconfig ClusterConfiguration > kubeadm.yml

在文件夹下，会生成一个kubeadm.yml文件，需要对kubeadm.yml进行修改。

```
apiVersion: kubeadm.k8s.io/v1beta2
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 192.168.136.128
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: kubernetes-master
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
---
apiServer:
```

原文链接：Kubernetes| 使用 Kubeadm 搭建 Kubernetes 集群

```
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta2
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controllerManager: {}
dns:
  type: CoreDNS
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: k8s.gcr.io
kind: ClusterConfiguration
kubernetesVersion: v1.18.0
networking:
  dnsDomain: cluster.local
  podSubnet: "192.168.0.0/16"
  serviceSubnet: 10.96.0.0/12
scheduler: {}
```

主要做三处修改：

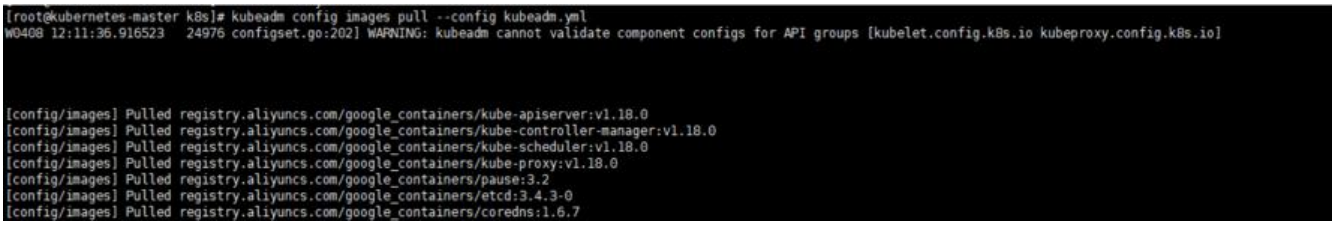
修改imageRepository为registry.aliyuncs.com/google_containers 阿里镜像源；

修改kubernetesVersion，我们使用v1.18.0作为kubernetes版本；

修改podSubnet，配置成Calico默认网段

- 查看并拉取镜像

```
# 查看所需镜像列表
kubeadm config images list --config kubeadm.yml
# 拉取镜像
kubeadm config images pull --config kubeadm.yml
```



- master初始化

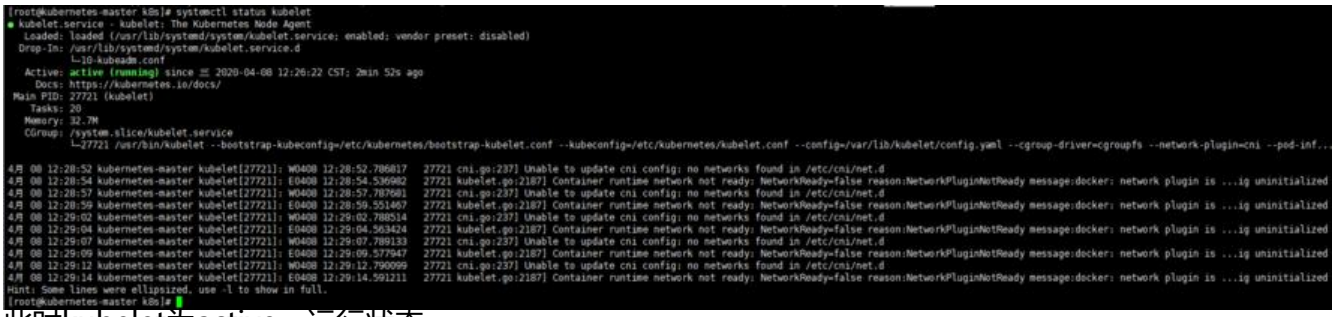
执行以下命令初始化主节点，该命令指定了初始化时需要使用的配置文件，其中添加 --upload-certs 参数可以在后续执行加入节点时自动分发证书文件。追加的 tee kubeadm-init.log 用以输出日志。

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

```
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 192.168.136.128:6443 --token abcdef.0123456789abcdef \
    --discovery-token-ca-cert-hash
sha256:a3ce56bb691c996e2b842ccfc08dbec834e295f30e582be0ca258500e02f49cc
```

- 再次查看kebelet状态

systemctl status kubelet



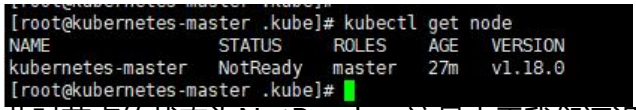
此时kubelet为active，运行状态。

- 配置kubelet

```
rm -rf /root/.kube/
mkdir /root/.kube/
cp -i /etc/kubernetes/admin.conf /root/.kube/config
```

- 验证是否安装成功

kubectl get node



```
[root@kubernetes-master .kube]# kubectl get node
NAME                STATUS     ROLES    AGE   VERSION
kubernetes-master   NotReady   master   27m   v1.18.0
[root@kubernetes-master .kube]#
```

此时节点的状态为NotReady，这是由于我们还没部署任何网络插件，是正常的。

Node 节点安装

Node节点只需要在安装docker的基础上安装 kubeadm组件即可。

- 修改主机名

hostnamectl set-hostname kubernetes-node1

- 卸载原kubeadm（若有）：

yum remove -y kubelet kubeadm kubectl

- 安装kubeadm：

yum install -y kubeadm

- 获取join命令（在Master节点执行）

kubeadm token create --print-join-command

kubeadm join 192.168.136.128:6443 --token bk1hs7.bxxz26xkzamtpn64   --discovery-token-a-cert-hash sha256:a3ce56bb691c996e2b842ccfc08dbec834e295f30e582be0ca258500e02f49c

Node2节点安装过程略。。。

完成两个节点的join后可以在master节点查看节点状态，kubectl get nodes -o wide

```
[root@kubernetes-master .kube]# kubectl get nodes -o wide
NAME                STATUS     ROLES    AGE     VERSION   INTERNAL-IP       EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION                CONTAINER-RUNTIME
kubernetes-master   NotReady   master   4h2m    v1.18.0   192.168.136.128   <none>        CentOS Linux 7 (Core)   3.10.0-1062.18.1.el7.x86_64   docker://19.3.5
kubernetes-node1    NotReady   <none>   5m50s   v1.18.0   192.168.136.129   <none>        CentOS Linux 7 (Core)   3.10.0-1062.4.3.el7.x86_64    docker://19.3.5
localhost.localdomain   NotReady   <none>   8s      v1.18.0   192.168.136.130   <none>        CentOS Linux 7 (Core)   3.10.0-1062.4.3.el7.x86_64    docker://19.3.5
[root@kubernetes-master .kube]#
```

至此 Node节点都已经加入Master。

安装网络插件Calico

在Master节点使用命令kubectl apply -f https://docs.projectcalico.org/v3.13/manifests/calico.yml 进行Calico网络插件的安装。

安装完成后我们可以使用命令kubectl get pods -n kube-system -o wide 查看pod状态。

```
[root@kubernetes-master ~]# kubectl get pods -n kube-system -o wide
NAME                                       READY   STATUS    RESTARTS   AGE     IP                NODE                NOMINATED NODE   READINESS GATES
calico-kube-controllers-5b8b769fcd-2bv99   1/1     Running   1          5d7h    192.168.237.6     kubernetes-master   <none>           <none>
calico-node-dlhmb                          1/1     Running   0          5m17s   192.168.136.130   kubernetes-node2    <none>           <none>
calico-node-ntq5n                          1/1     Running   0          5m53s   192.168.136.129   kubernetes-node1    <none>           <none>
calico-node-xg8s7                          1/1     Running   1          5d7h    192.168.136.128   kubernetes-master   <none>           <none>
coredns-7ff77c879f-qwmnl                   1/1     Running   1          6d9h    192.168.237.5     kubernetes-master   <none>           <none>
coredns-7ff77c879f-txnlc                   1/1     Running   1          6d9h    192.168.237.4     kubernetes-master   <none>           <none>
etcd-kubernetes-master                     1/1     Running   4          6d9h    192.168.136.128   kubernetes-master   <none>           <none>
kube-apiserver-kubernetes-master           1/1     Running   4          6d9h    192.168.136.128   kubernetes-master   <none>           <none>
kube-controller-manager-kubernetes-master  1/1     Running   4          6d9h    192.168.136.128   kubernetes-master   <none>           <none>
kube-proxy-5p9qd                           1/1     Running   0          5m53s   192.168.136.129   kubernetes-node1    <none>           <none>
kube-proxy-956p9                           1/1     Running   0          5m17s   192.168.136.130   kubernetes-node2    <none>           <none>
kube-proxy-g2bzm                           1/1     Running   4          6d9h    192.168.136.128   kubernetes-master   <none>           <none>
kube-scheduler-kubernetes-master           1/1     Running   6          6d9h    192.168.136.128   kubernetes-master   <none>           <none>
```

再次查看node节点状态kubectl get nodes -o wide，发现处于Ready状态

```
[root@kubernetes-master k8s]# kubectl get nodes -o wide
NAME                STATUS   ROLES    AGE     VERSION   INTERNAL-IP       EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION                CONTAINER-RUNTIME
kubernetes-master   Ready    master   6d9h    v1.18.0   192.168.136.128   <none>        CentOS Linux 7 (Core)   3.10.0-1062.18.1.el7.x86_64   docker://19.3.5
kubernetes-node1    Ready    <none>   9m57s   v1.18.0   192.168.136.129   <none>        CentOS Linux 7 (Core)   3.10.0-1062.4.3.el7.x86_64    docker://19.3.5
kubernetes-node2    Ready    <none>   9m21s   v1.18.0   192.168.136.130   <none>        CentOS Linux 7 (Core)   3.10.0-1062.4.3.el7.x86_64    docker://19.3.5
[root@kubernetes-master k8s]#
```

这样我们整个Kubernetes集群已经搭建完成，大家可以开始部署你的服务应用了！

重新加入节点

若节点需要重新加入节点可以按照如下步骤进行：

- 先在node节点执行 kubeadm reset -f 命令，重置kubeadm
- 在Master节点删除原节点

kubectl delete node kubernetes-node1

- 在Master节点获取join命令

kubeadm token create --print-join-command

- 在Node节点执行命令重新加入集群

kubeadm join 192.168.136.128:6443 --token bk1hs7.bxxz26xkzamtpn64 --discovery-token-ca-cert-hash sha256:a3ce56bb691c996e2b842ccfc08dbec834e295f30e582be0ca258500e02f49cc

安装错误

在安装Node节点时可能会出现如下的错误

[ERROR FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables contents are not set to 1

解决方法：

echo "1" >/proc/sys/net/bridge/bridge-nf-call-iptables

重启Master节点后执行kubectl 相关命令出现如下错误，则很可能是没关闭swap导致kubelet无法正启动，可以使用free -m 命令查看

若swap所在行不为0则表示未关闭swap，可以使用swapoff -a命令关闭，然后使用命令systemctl start kubelet 重新启动kubelet 。