30分钟教你部署K8S集群.md

# 一、环境准备

PS:每台机器都要执行

## 1、机器环境

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

k8s-master: 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

## 2、依赖环境

安装依赖环境，注意：每一台机器都需要安装此依赖环境  
[root@localhost ~]# 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 yum-utils device-mapper-persistent-data lvm2  
  
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(停止并禁用)  
[root@localhost ~]# systemctl stop firewalld && systemctl disable firewalld  
  
置空iptables   
[root@localhost ~]# yum -y install iptables-services && systemctl start iptables && systemctl enable iptables && iptables -F && service iptables save  
  
关闭selinux  
关闭swap分区【虚拟内存】并且永久关闭虚拟内存  
[root@localhost ~]# swapoff -a && sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab  
关闭selinux  
[root@localhost ~]# setenforce 0 && sed -i 's/^SELINUX=.\*/SELINUX=disabled/' /etc/selinux/config  
  
升级Linux内核版本  
安装源  
centos8执行  
[root@localhost ~]# yum install https://www.elrepo.org/elrepo-release-8.el8.elrepo.noarch.rpm -y  
centos7执行  
[root@localhost ~]# yum install https://www.elrepo.org/elrepo-release-7.el7.elrepo.noarch.rpm -y  
  
列出所有可用内核  
[root@localhost ~]# yum --enablerepo=elrepo-kernel list|grep kernel  
  
升级内核最新版  
[root@localhost ~]# yum --enablerepo=elrepo-kernel install -y kernel-lt  
  
查询已安装的内核  
[root@localhost ~]# rpm -qa | grep kernel  
查看默认启动项  
[root@localhost ~]# awk -F\' '$1=="menuentry " {print $2}' /etc/grub2.cfg   
CentOS Linux (4.4.227-1.el7.elrepo.x86\_64) 7 (Core)  
CentOS Linux (3.10.0-1127.el7.x86\_64) 7 (Core)  
CentOS Linux (0-rescue-415ea3f02941f246af981f6f14e92f33) 7 (Core)  
  
默认启动的顺序是从0开始（CentOS Linux (3.10.0-1127.el7.x86\_64) 7），新内核是从头插入，所以需要选择0  
[root@localhost ~]# grub2-set-default 0  
  
重启  
[root@localhost ~]# reboot  
使用新的内核，重启后使用的内核版本:  
[root@localhost ~]# uname -r   
4.4.227-1.el7.elrepo.x86\_64  
  
删除旧内核  
[root@localhost ~]# rpm -qa | grep kernel  
kernel-3.10.0-1127.10.1.el7.x86\_64  
kernel-3.10.0-1127.el7.x86\_64  
kernel-3.10.0-1062.el7.x86\_64  
kernel-tools-3.10.0-1127.10.1.el7.x86\_64  
kernel-headers-3.10.0-1127.10.1.el7.x86\_64  
kernel-lt-4.4.227-1.el7.elrepo.x86\_64  
kernel-tools-libs-3.10.0-1127.10.1.el7.x86\_64  
[root@localhost ~]# yum remove kernel-3.10.0-1127.10.1.el7.x86\_64 kernel-3.10.0-1127.el7.x86\_64 kernel-3.10.0-1062.el7.x86\_64  
  
优化系统内核参数（对于k8s）  
[root@localhost ~]# cat > /etc/sysctl.d/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  
  
  
手动刷新，让优化文件立即生效  
[root@localhost ~]# sysctl -p /etc/sysctl.d/kubernetes.conf  
#注意：可能会报错(最后的一个配置net.netfilter.nf\_conntrack\_max=2310720)  
# sysctl: cannot stat /proc/sys/net/netfilter/nf\_conntrack\_max: 没有那个文件或目录  
# 也许没有加载ip\_conntrack.尝试：lsmod |grep conntrack   
# 如果这是空的,请加载：modprobe ip\_conntrack  
# 或者尝试net.nf\_conntrack\_max代替net.netfilter.nf\_conntrack\_max = xxxx  
  
调整系统临时区 --- 如果已经设置时区，可略过  
设置系统时区为中国/上海  
[root@localhost ~]# timedatectl set-timezone Asia/Shanghai  
#将当前的 UTC 时间写入硬件时钟  
[root@localhost ~]# timedatectl set-local-rtc 0  
#重启依赖于系统时间的服务  
[root@localhost ~]# systemctl restart rsyslog  
[root@localhost ~]# systemctl restart crond

## 3、docker部署

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  
[root@localhost ~]# yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo  
  
查看可用版本  
[root@localhost ~]# yum list docker-ce --showduplicates | sort -r  
  
更新Yum&安装Docker CE （不加版本默认最新版）  
[root@localhost ~]# yum update -y && yum install docker-ce -y  
  
重启docker服务,设置开机自启动  
[root@localhost ~]# systemctl restart docker && systemctl enable docker  
[root@localhost system]# docker --version  
Docker version 19.03.11, build 42e35e61f3  
  
更新daemon.json文件   
[root@localhost ~]# cat > /etc/docker/daemon.json <<EOF  
{  
"registry-mirrors": ["https://b9pmyelo.mirror.aliyuncs.com"],  
"exec-opts": ["native.cgroupdriver=systemd"],  
"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   
  
重启  
[root@localhost ~]# systemctl restart docker

## 4、安装kubeadm、kubelet、kubectl

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，我们使用阿里云的yum仓库镜像。  
[root@localhost ~]# 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  
  
安装kubeadm、kubelet、kubectl  
列出所有版本  
[root@localhost ~]# yum list kubeadm --showduplicates | sort -r  
安装  
[root@localhost ~]# yum install -y kubeadm-1.18.2 kubelet-1.18.2 kubectl-1.18.2  
  
启动kubelet，添加自启  
[root@localhost ~]# systemctl enable kubelet && systemctl start kubelet

kubeadm可以一键部署集群

## 5、复制虚拟机

master和node的都需要以上的环境，直接复制虚拟机，选择任意一个为master，其他为node

# 二、集群部署

## 1、配置主机名

master设置主机名：  
[root@localhost ~]# hostnamectl set-hostname k8s-master  
在master添加hosts：  
[root@localhost ~]# cat >> /etc/hosts << EOF  
10.20.33.80 k8s-master  
10.20.33.12 k8s-node01  
EOF  
  
node设置主机名：  
[root@localhost ~]# hostnamectl set-hostname k8s-node01

如果有多个node，每个node都需要配置自己的主机名（不能相同），然后在master的/etc/hosts添加配置，让master知道每个node的ip和对应的主机名

## 2、 部署Kubernetes Master

在Master执行  
[root@localhost ~]# ip a  
获取master ip 10.20.33.115   
  
获取默认配置，并输出到kubeadm-config.yaml  
[root@localhost ~]# kubeadm config print init-defaults > kubeadm-config.yaml  
W0617 15:35:43.386146 13616 configset.go:202] WARNING: kubeadm cannot validate component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]  
[root@localhost ~]# ls  
kubeadm-config.yaml  
在kubeadm-config.yaml基础上做修改，或者直接执行下边的命令手动写入文件  
  
#修改初始化配置文件kubeadm-config.yaml  
cat << EOF>kubeadm-config.yaml  
apiVersion: kubeadm.k8s.io/v1beta2  
bootstrapTokens:  
- groups:  
 - system:bootstrappers:kubeadm:default-node-token  
 token: abcdef.0123456789abcdef  
 ttl: 24h0m0s  
 usages:  
 - signing  
 - authentication  
kind: InitConfiguration  
localAPIEndpoint:  
#前边获取的master本机ip  
 advertiseAddress: 10.20.33.115  
 bindPort: 6443  
nodeRegistration:  
 criSocket: /var/run/dockershim.sock  
 #本机节点name  
 name: k8s-master  
 taints:  
 - effect: NoSchedule  
 key: node-role.kubernetes.io/master  
---  
apiServer:  
 timeoutForControlPlane: 4m0s  
apiVersion: kubeadm.k8s.io/v1beta2  
certificatesDir: /etc/kubernetes/pki  
clusterName: kubernetes  
controllerManager: {}  
dns:  
 type: CoreDNS  
etcd:  
 local:  
 dataDir: /var/lib/etcd  
#镜像仓库地址，默认k8s.gcr.io，修改为阿里云的  
imageRepository: registry.aliyuncs.com/google\_containers  
kind: ClusterConfiguration  
#修改成kubectl一致的版本  
kubernetesVersion: v1.18.2  
networking:  
 dnsDomain: cluster.local  
 serviceSubnet: 10.96.0.0/12  
 # 指定flannel模型通信 pod网段地址,此网段和flannel网段一致   
 podSubnet: 10.244.0.0/16  
scheduler: {}  
EOF  
  
  
进行初始化，CPU核心数量必须大于1核，否则无法执行成功  
[root@localhost ~]# kubeadm init --config=kubeadm-config.yaml   
W0725 12:24:53.652756 2081 configset.go:202] WARNING: kubeadm cannot validate component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]  
[init] Using Kubernetes version: v1.18.2  
[preflight] Running pre-flight checks  
error execution phase preflight: [preflight] Some fatal errors occurred:  
 [ERROR NumCPU]: the number of available CPUs 1 is less than the required 2  
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`  
To see the stack trace of this error execute with --v=5 or higher  
  
[root@localhost ~]# kubeadm init --config=kubeadm-config.yaml   
W0617 15:48:20.619712 13686 configset.go:202] WARNING: kubeadm cannot validate component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]  
[init] Using Kubernetes version: v1.18.2  
[preflight] Running pre-flight checks  
[preflight] Pulling images required for setting up a Kubernetes cluster  
[preflight] This might take a minute or two, depending on the speed of your internet connection  
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'  
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"  
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"  
[kubelet-start] Starting the kubelet  
[certs] Using certificateDir folder "/etc/kubernetes/pki"  
[certs] Generating "ca" certificate and key  
[certs] Generating "apiserver" certificate and key  
[certs] apiserver serving cert is signed for DNS names [localhost kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.20.33.115]  
[certs] Generating "apiserver-kubelet-client" certificate and key  
[certs] Generating "front-proxy-ca" certificate and key  
[certs] Generating "front-proxy-client" certificate and key  
[certs] Generating "etcd/ca" certificate and key  
[certs] Generating "etcd/server" certificate and key  
[certs] etcd/server serving cert is signed for DNS names [localhost localhost] and IPs [10.20.33.115 127.0.0.1 ::1]  
[certs] Generating "etcd/peer" certificate and key  
[certs] etcd/peer serving cert is signed for DNS names [localhost localhost] and IPs [10.20.33.115 127.0.0.1 ::1]  
[certs] Generating "etcd/healthcheck-client" certificate and key  
[certs] Generating "apiserver-etcd-client" certificate and key  
[certs] Generating "sa" key and public key  
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"  
[kubeconfig] Writing "admin.conf" kubeconfig file  
[kubeconfig] Writing "kubelet.conf" kubeconfig file  
[kubeconfig] Writing "controller-manager.conf" kubeconfig file  
[kubeconfig] Writing "scheduler.conf" kubeconfig file  
[control-plane] Using manifest folder "/etc/kubernetes/manifests"  
[control-plane] Creating static Pod manifest for "kube-apiserver"  
[control-plane] Creating static Pod manifest for "kube-controller-manager"  
W0617 15:48:27.092221 13686 manifests.go:225] the default kube-apiserver authorization-mode is "Node,RBAC"; using "Node,RBAC"  
[control-plane] Creating static Pod manifest for "kube-scheduler"  
W0617 15:48:27.093584 13686 manifests.go:225] the default kube-apiserver authorization-mode is "Node,RBAC"; using "Node,RBAC"  
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"  
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s  
[apiclient] All control plane components are healthy after 17.002816 seconds  
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace  
[kubelet] Creating a ConfigMap "kubelet-config-1.18" in namespace kube-system with the configuration for the kubelets in the cluster  
[upload-certs] Skipping phase. Please see --upload-certs  
[mark-control-plane] Marking the node localhost as control-plane by adding the label "node-role.kubernetes.io/master=''"  
[mark-control-plane] Marking the node localhost as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]  
[bootstrap-token] Using token: abcdef.0123456789abcdef  
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles  
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes  
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials  
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token  
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster  
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace  
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key  
[addons] Applied essential addon: CoreDNS  
[addons] Applied essential addon: kube-proxy  
  
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 10.20.33.115:6443 --token abcdef.0123456789abcdef \  
 --discovery-token-ca-cert-hash sha256:5a373d36b57e15736fb02b377d2f11f99451160d6c48a72a99ee709d2fac975d   
  
  
通过一份配置文件而不是使用命令行参数来配置 kubeadm init 命令是可能的，但是一些更加高级的功能只能够通过配置文件设定。这份配置文件通过 --config 选项参数指定（也可以通过命令行不用配置文件的方式初始化集群，2选1即可，推荐配置文件方式）  
[root@localhost ~]# kubeadm init \  
 --apiserver-advertise-address=10.20.33.115 \  
 --image-repository registry.aliyuncs.com/google\_containers \  
 --kubernetes-version v1.18.2 \  
 --service-cidr=10.96.0.0/12 \  
 --pod-network-cidr=10.244.0.0/16  
   
apiserver-advertise-address是master的ip  
image-repository是镜像仓库地址，这里指定阿里云镜像仓库地址

## 3、保存配置

创建目录，保存连接配置缓存，认证文件   
[root@localhost ~]# mkdir -p $HOME/.kube   
  
拷贝集群管理配置文件   
[root@localhost ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config   
  
授权给配置文件   
[root@localhost ~]# chown $(id -u):$(id -g) $HOME/.kube/config

如果不保存配置，执行kubectl命令会报错

## 4、 网络插件flannel（CNI）

[root@localhost ~]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

这个地址现在访问不到了，需要先下载kube-flannel.yml

https://github.com/coreos/flannel/tree/master/Documentation

然后执行

[root@k8s-master ~]# kubectl create -f kube-flannel.yml

## 5、Node加入Kubernetes集群

在Node执行

向集群添加新节点，执行在kubeadm init成功后最后输出的kubeadm join命令，即master初始化成功会在最后生成token,node节点加入集群要使用这个命令

[root@localhost ~]# kubeadm join 10.20.33.115:6443 --token abcdef.0123456789abcdef \  
 --discovery-token-ca-cert-hash sha256:5a373d36b57e15736fb02b377d2f11f99451160d6c48a72a99ee709d2fac975d   
   
加入成功会有以下显示   
W0617 18:57:28.643045 3348 join.go:346] [preflight] WARNING: JoinControlPane.controlPlane settings will be ignored when control-plane flag is not set.  
[preflight] Running pre-flight checks  
 [WARNING Hostname]: hostname "k8s-node01" could not be reached  
 [WARNING Hostname]: hostname "k8s-node01": lookup k8s-node01 on 202.106.0.20:53: no such host  
[preflight] Reading configuration from the cluster...  
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'  
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.18" 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] Starting the kubelet  
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...  
  
This 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 control-plane to see this node join the cluster.  
  
  
在master执行kubectl get nodes  
[root@k8s-master ~]# kubectl get node  
NAME STATUS ROLES AGE VERSION  
k8s-master Ready master 5m21s v1.18.2  
k8s-node01 Ready <none> 4m41s v1.18.2  
k8s-node02 Ready <none> 4m44s v1.18.2  
  
更详细查看命令，可以看见初始化节点所属节点  
查询工作空间中pod容器的详细信息  
[root@k8s-master ~]# kubectl get pod -n kube-system -o wide  
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES  
coredns-7ff77c879f-264fv 1/1 Running 0 9h 10.244.0.2 k8s-master <none> <none>  
coredns-7ff77c879f-wjnp7 1/1 Running 0 9h 10.244.0.3 k8s-master <none> <none>  
etcd-k8s-master 1/1 Running 2 9h 10.20.33.80 k8s-master <none> <none>  
kube-apiserver-k8s-master 1/1 Running 2 9h 10.20.33.80 k8s-master <none> <none>  
kube-controller-manager-k8s-master 1/1 Running 2 9h 10.20.33.80 k8s-master <none> <none>  
kube-flannel-ds-amd64-54b5b 1/1 Running 0 4m47s 10.20.33.80 k8s-master <none> <none>  
kube-flannel-ds-amd64-6g2s7 1/1 Running 0 4m47s 10.0.2.15 k8s-node01 <none> <none>  
kube-proxy-mvp4j 1/1 Running 2 9h 10.0.2.15 k8s-node01 <none> <none>  
kube-proxy-xr76t 1/1 Running 2 9h 10.20.33.80 k8s-master <none> <none>  
kube-scheduler-k8s-master 1/1 Running 2 9h 10.20.33.80 k8s-master <none> <none>

## 6. 测试kubernetes集群

在Kubernetes集群中创建一个pod，验证是否正常运行：

[root@k8s-master ~]# kubectl create deployment nginx --image=nginx  
[root@k8s-master ~]# kubectl expose deployment nginx --port=80 --type=NodePort  
[root@k8s-master ~]# kubectl get pod,svc  
NAME READY STATUS RESTARTS AGE  
pod/nginx-f89759699-xx4jv 1/1 Running 0 20m  
  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 9h  
service/nginx NodePort 10.106.216.150 <none> 80:31880/TCP 19m

svc中有一个service/nginx，PORT（s）代表端口映射，就是把svc的80端口映射到物理机（node节点）的31880端口上，访问http://nodeIP:31880/，出现nginx欢迎页代表集群搭建成功

## 7. 部署 Dashboard

### 获取kubernetes-dashboard.yaml

[root@k8s-master ~]# wget https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml

### 修改kubernetes-dashboard.yaml

找到文件中对应的位置，做出修改

修改前：

kind: Service  
apiVersion: v1  
metadata:  
 labels:  
 k8s-app: kubernetes-dashboard  
 name: kubernetes-dashboard  
 namespace: kubernetes-dashboard  
spec:  
 ports:  
 - port: 443  
 targetPort: 8443  
 selector:  
 k8s-app: kubernetes-dashboard

修改后

kind: Service  
apiVersion: v1  
metadata:  
 labels:  
 k8s-app: kubernetes-dashboard  
 name: kubernetes-dashboard  
 namespace: kubernetes-dashboard  
spec:  
 #修改type  
 type: NodePort  
 ports:  
 - port: 443  
 targetPort: 8443  
 #对外暴露端口，可通过https://nodeIP/nodePort访问dashboard  
 nodePort: 30001  
 selector:  
 k8s-app: kubernetes-dashboard

### 启动 Dashboard

[root@k8s-master ~]# kubectl apply -f kubernetes-dashboard.yaml  
  
或者  
[root@k8s-master ~]# kubectl create -f kubernetes-dashboard.yaml   
namespace/kubernetes-dashboard created  
serviceaccount/kubernetes-dashboard created  
service/kubernetes-dashboard created  
secret/kubernetes-dashboard-certs created  
secret/kubernetes-dashboard-csrf created  
secret/kubernetes-dashboard-key-holder created  
configmap/kubernetes-dashboard-settings created  
role.rbac.authorization.k8s.io/kubernetes-dashboard created  
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created  
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created  
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created  
deployment.apps/kubernetes-dashboard created  
service/dashboard-metrics-scraper created  
deployment.apps/dashboard-metrics-scraper created

出现这样的显示，说明dashboard启动成功

### 访问 Dashboard

访问 https://nodeIP:nodePort

一定要访问https,ip是node的ip，不能是master的ip

成功应该出现登录页

### 创建 Dashboard 的用户

创建配置文件  
cat<<EOF>account.yaml  
# Create Service Account  
apiVersion: v1  
kind: ServiceAccount  
metadata:  
 name: admin-user  
 namespace: kube-system  
---  
# Create ClusterRoleBinding  
apiVersion: rbac.authorization.k8s.io/v1beta1  
kind: ClusterRoleBinding  
metadata:  
 name: admin-user  
roleRef:  
 apiGroup: rbac.authorization.k8s.io  
 kind: ClusterRole  
 name: cluster-admin  
subjects:  
- kind: ServiceAccount  
 name: admin-user  
 namespace: kube-system  
EOF  
  
  
创建用户：  
[root@k8s-master ~]# kubectl create -f account.yaml   
serviceaccount/admin-user created  
clusterrolebinding.rbac.authorization.k8s.io/admin-user created

或者：创建service account并绑定默认cluster-admin管理员集群角色：

[root@k8s-master ~]# kubectl create serviceaccount dashboard-admin -n kube-system  
  
绑定默认角色  
[root@k8s-master ~]# kubectl create clusterrolebinding dashboard-admin --clusterrole=cluster-admin --serviceaccount=kube-system:dashboard-admin

### 查看登录 Dashboard 的Token

[root@k8s-master ~]# kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')  
Name: admin-user-token-szhxg  
Namespace: kube-system  
Labels: <none>  
Annotations: kubernetes.io/service-account.name: admin-user  
 kubernetes.io/service-account.uid: d1809632-ceb1-4078-9b68-25c3c6fcd7f8  
  
Type: kubernetes.io/service-account-token  
  
Data  
====  
  
ca.crt: 1025 bytes  
namespace: 11 bytes  
token: eyJhbGciOiJSUzI1NiIsImtpZCI6IlI2akpMOXFSSjRoS3pmWWU4Y2JwZUdXUDZWeUNGYXVBTmdPU0dmcnNKdncifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLXN6aHhnIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJkMTgwOTYzMi1jZWIxLTQwNzgtOWI2OC0yNWMzYzZmY2Q3ZjgiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.iu5sTg6jVVluneE9zEBsI6kpJg0gW230b\_daV24v9NsgXGZs7\_UBPHJ2dfVNw9Kk7z5vzXKGaudwSpSKr0JrSSgRlrH4gjt3kehCgVLKpe0NeVvMgoRuhtIMZ2O3vyB-fKQBhN4uEJcmvd\_sz4qNoLmojFZvs12BcjuaedhBQ0hLu0mjxBQU4\_xlvjxJh\_\_i6J-d5WRRSBN8HkaIdBEwDDq8QMiNxHqxs-KDcCVR4ev02RkUTsbMa674zSoxPj-hN2A\_2nXDhVah6B\_AAn\_1\_uPvjIcYbmOCXSrT1\_Ffug6hFwXr2TzgqbwU7cmQghsnzTV1U2qjpNh5vn3zvlkTTg

这个token就是网页端用来登录Dashboard 的Token