Kubeadm是一个kubernetes官方提供的快速安装和初始化拥有最佳实践（best practice）的kubernetes集群的工具，安装步骤如下:

## 环境准备

以下操作需要在所有节点执行

### 1.1禁用防火墙和swap

禁用防火墙

sudo ufw disable

禁用交换分区

swapoff -a

vim /etc/fstab 注释掉swap分区

free –m 查看分区情况



---注意：这一步完成后重启机器

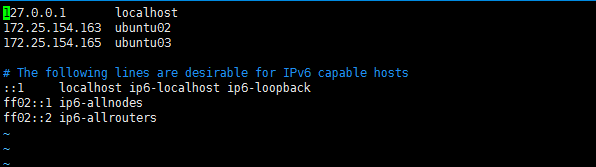
计算机生成了可选文字:
日／etc/fstab:staticfilesysteminformation·
#
#Use'blkid'toprinttheuniversallyuniqueidentifierfora
#device;thismaybeusedwithUUID=asamorerobust树ayto
#thatworksevenifdisksareaddedandremoved.seefstab(5)
name
devices
#
#<filesystem><mountpoint>
/dev/mapper/pcc树docker一vg一root
#/bootwason/dev/sdalduring
UU工O二日口口”二d'.
<type><options>
/ext4
installation
<dump><paSS>
e户广0户S＝二二LJr一七l,00
#/dev/mapper/pccwdocker一vg一swap_1
#/dev/mapper/crypts树aplnones衬ap
none
SW00
/boot
边尸尸
eXtZ
defaults
井
S加二p
5树
忍

sudo iptables -F && sudo iptables -X && sudo iptables -F -t nat && sudo iptables -X -t nat

sudo iptables -P FORWARD ACCEPT

### 1.2 修改hosts文件

将所有节点的主机名和固定IP对应



### 1.3 安装ssh和docker

1、安装openssh-server

sudo apt-get install openssh-server

2、安装docker // 寻找docker docker search kubernetes-dashboard

apt-get update

apt-get install -y docker.io

### 1.4 安装k8s组件

1、在所有节点执行以下脚本，安装所需软件以及替换国内更新源；

apt-get update && apt-get install -y apt-transport-https curl

curl <https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg> | apt-key add -

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb <https://mirrors.aliyun.com/kubernetes/apt/> kubernetes-xenial main

EOF

2、在所有节点安装k8s的必须组件kubeadm, kubectl, and kubelet

apt-get update

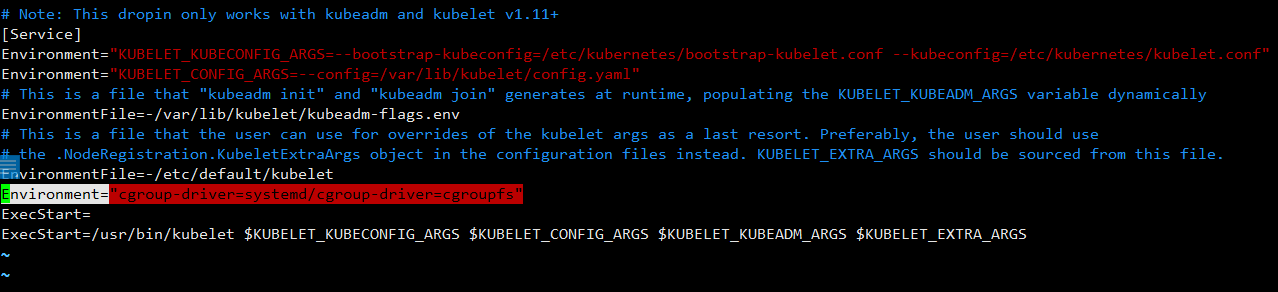
apt-get install -y kubelet kubeadm kubectl

3、在所有节点修改Kubernetes 配置

vim /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

添加如下内容：

Environment="cgroup-driver=systemd/cgroup-driver=cgroupfs"



## 二、配置MASTER服务器

以下操作只在master服务器执行。

### 2.1 初始化kubernetes集群

在初始化kubernetes集群之前，由于国内网络不能访问google，需要先从docker hub下载所需的所有image,然后利用docker标记为系统所需的image名称以及版本；***(注意：node节点在添加到集群中时也需要从google下载image，所以以下image也需在node节点通过该方法提前下载)***

----此次测试环境所需的image以及版本号

docker pull mirrorgooglecontainers/kube-apiserver-amd64:v1.12.1

docker pull mirrorgooglecontainers/kube-controller-manager-amd64:v1.12.1

docker pull mirrorgooglecontainers/kube-scheduler-amd64:v1.12.1

docker pull mirrorgooglecontainers/kube-proxy-amd64:v1.12.1

docker pull mirrorgooglecontainers/pause:3.1

docker pull mirrorgooglecontainers/etcd-amd64:3.2.24

docker pull coredns/coredns:1.2.2

----进行标记；

docker tag mirrorgooglecontainers/kube-proxy-amd64:v1.12.1 k8s.gcr.io/kube-proxy:v1.12.1

docker tag mirrorgooglecontainers/kube-apiserver-amd64:v1.12.1 k8s.gcr.io/kube-apiserver:v1.12.1

docker tag mirrorgooglecontainers/kube-controller-manager-amd64:v1.12.1 k8s.gcr.io/kube-controller-manager:v1.12.1

docker tag mirrorgooglecontainers/kube-scheduler-amd64:v1.12.1 k8s.gcr.io/kube-scheduler:v1.12.1

docker tag mirrorgooglecontainers/etcd-amd64:3.2.24 k8s.gcr.io/etcd:3.2.24

docker tag coredns/coredns:1.2.2 k8s.gcr.io/coredns:1.2.2

docker tag mirrorgooglecontainers/pause:3.1 k8s.gcr.io/pause:3.1

----删除不需要的image

docker rmi mirrorgooglecontainers/kube-proxy-amd64:v1.12.1

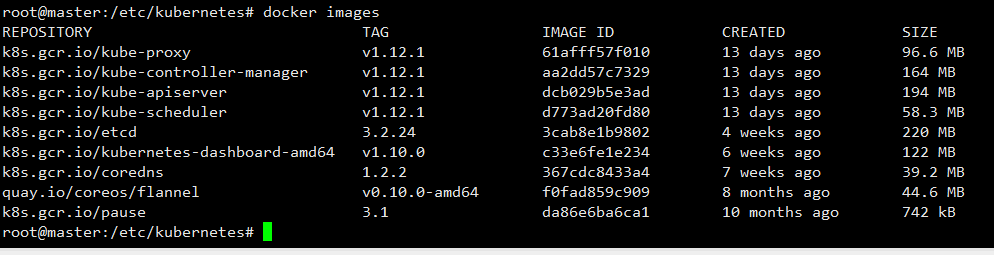
docker rmi mirrorgooglecontainers/kube-apiserver-amd64:v1.12.1

docker rmi mirrorgooglecontainers/kube-controller-manager-amd64:v1.12.1

docker rmi mirrorgooglecontainers/kube-scheduler-amd64:v1.12.1

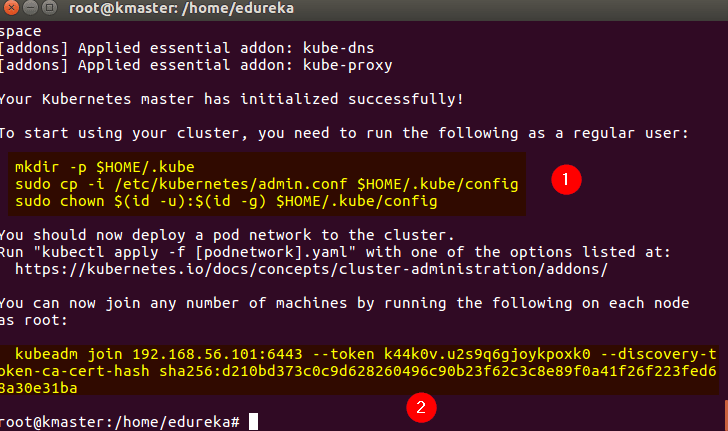
docker rmi mirrorgooglecontainers/etcd-amd64:3.2.24

docker rmi coredns/coredns:1.2.2  
docker rmi mirrorgooglecontainers/pause:3.1



完成以上操作之后，下来开始初始化kubernetes 集群；

**kubeadm init --pod-network-cidr=10.100.0.0/16**  （私网地址，不能和public网重合），初始化完成后，会有如下图所示；



### 2.2 配置kubectl

mkdir -p $HOME/.kube

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

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

### 2.3 创建pod的网络

使用flannel创建pod网络;

先创建kube-flannel-amd64.yml文件，内容如下：

---

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1beta1

metadata:

name: flannel

rules:

- apiGroups:

- ""

resources:

- pods

verbs:

- get

- apiGroups:

- ""

resources:

- nodes

verbs:

- list

- watch

- apiGroups:

- ""

resources:

- nodes/status

verbs:

- patch

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1beta1

metadata:

name: flannel

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: flannel

subjects:

- kind: ServiceAccount

name: flannel

namespace: kube-system

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: flannel

namespace: kube-system

---

kind: ConfigMap

apiVersion: v1

metadata:

name: kube-flannel-cfg

namespace: kube-system

labels:

tier: node

app: flannel

data:

cni-conf.json: |

{

"name": "cbr0",

"plugins": [

{

"type": "flannel",

"delegate": {

"hairpinMode": true,

"isDefaultGateway": true

}

},

{

"type": "portmap",

"capabilities": {

"portMappings": true

}

}

]

}

net-conf.json: |

{

"Network": "10.100.0.0/16",

"Backend": {

"Type": "vxlan"

}

}

---

apiVersion: extensions/v1beta1

kind: DaemonSet

metadata:

name: kube-flannel-ds-amd64

namespace: kube-system

labels:

tier: node

app: flannel

spec:

template:

metadata:

labels:

tier: node

app: flannel

spec:

hostNetwork: true

nodeSelector:

beta.kubernetes.io/arch: amd64

tolerations:

- operator: Exists

effect: NoSchedule

serviceAccountName: flannel

initContainers:

- name: install-cni

image: quay.io/coreos/flannel:v0.10.0-amd64

command:

- cp

args:

- -f

- /etc/kube-flannel/cni-conf.json

- /etc/cni/net.d/10-flannel.conflist

volumeMounts:

- name: cni

mountPath: /etc/cni/net.d

- name: flannel-cfg

mountPath: /etc/kube-flannel/

containers:

- name: kube-flannel

image: quay.io/coreos/flannel:v0.10.0-amd64

command:

- /opt/bin/flanneld

args:

- --ip-masq

- --kube-subnet-mgr

resources:

requests:

cpu: "100m"

memory: "50Mi"

limits:

cpu: "100m"

memory: "50Mi"

securityContext:

privileged: true

env:

- name: POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: POD\_NAMESPACE

valueFrom:

fieldRef:

fieldPath: metadata.namespace

volumeMounts:

- name: run

mountPath: /run

- name: flannel-cfg

mountPath: /etc/kube-flannel/

volumes:

- name: run

hostPath:

path: /run

- name: cni

hostPath:

path: /etc/cni/net.d

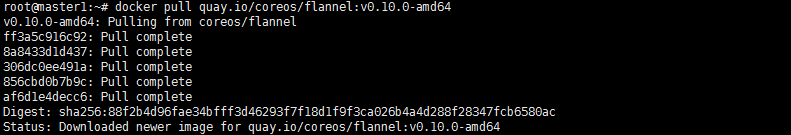
- name: flannel-cfg

configMap:

name: kube-flannel-cfg

注意：kube-flannel-amd64.yml文件中用到了quay.io/coreos/flannel:v0.10.0-amd64，所以可以提前下载

docker pull quay.io/coreos/flannel:v0.10.0-amd64



完成后在使用kubectl 创建flannel网络，命令如下：

**kubectl create -f kube-flannel-amd64.yml**

完成后，查看一下pod的状态；

kubectl get pods --all-namespaces

计算机生成了可选文字:
1root@c一pc：一＃kubectlgetpods一all一namespaces
2NAMESPACENAMEREADY
kube一systemetcd一c一pcl/1
、kube一systemkube一apiserver一c一pcl/1
5kube一systemkube一controller一manager一c一pcl/1
6kube一systemkube一dns一2838158301一vosrqo/3
7kube一systemkube一flannel一ds一vp7mto/2
kube一systemkube一proxy一q12901/1
kube一systemkube一scheduler一c一pcl/1
STATUS
Running
Running
Running
Pendlng
ContainerCreating
Running
Running
RESTARTS
0
0
0
0
0
0
0
AGE
3h
3h
3h
3h
125
3卜
3卜

kubectl get node计算机生成了可选文字:
这个时候开始仓11建pull们annel的image7，稍等一下，
1r00toc一pc：一＃kubectlgetnodes
2NAMESTATUSAGEVERSION
3c一pcReady3hvl.7.0
rootoc一pc：一＃kubectlgetpods一all一namespaces
NAMESPACENAMEREADY
。二kube一systemetcd一c一pcl/1
了kube一systemkube一aplserver一c一pcl/1
压kube一systemkube一controller一manager一c一pcl/l
‘〕kube一systemkube一dns一2838158301一vosrq3/3
一二kube一systemkube一flannel一ds一vp7mtZ/2
工工kube一systemkube一proxy一q12901/1
土二kube一systemkube一scheduler一c一pcl/l
STATUS
Running
Running
Running
Running
Running
Running
Running
RESTARTS
0
0
0
0
0
0
0
AGE
3h
3h
3h
3h
6m
3h
3h

### 2.4 安装dashboard

首先创建kubernetes-dashboard.yaml文件，内容如下：

# Copyright 2017 The Kubernetes Authors.

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

# <http://www.apache.org/licenses/LICENSE-2.0>

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

# ------------------- Dashboard Secret ------------------- #

apiVersion: v1

kind: Secret

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard-certs

namespace: kube-system

type: Opaque

---

# ------------------- Dashboard Service Account ------------------- #

apiVersion: v1

kind: ServiceAccount

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

---

# ------------------- Dashboard Role & Role Binding ------------------- #

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: kubernetes-dashboard-minimal

namespace: kube-system

rules:

# Allow Dashboard to create 'kubernetes-dashboard-key-holder' secret.

- apiGroups: [""]

resources: ["secrets"]

verbs: ["create"]

# Allow Dashboard to create 'kubernetes-dashboard-settings' config map.

- apiGroups: [""]

resources: ["configmaps"]

verbs: ["create"]

# Allow Dashboard to get, update and delete Dashboard exclusive secrets.

- apiGroups: [""]

resources: ["secrets"]

resourceNames: ["kubernetes-dashboard-key-holder", "kubernetes-dashboard-certs"]

verbs: ["get", "update", "delete"]

# Allow Dashboard to get and update 'kubernetes-dashboard-settings' config map.

- apiGroups: [""]

resources: ["configmaps"]

resourceNames: ["kubernetes-dashboard-settings"]

verbs: ["get", "update"]

# Allow Dashboard to get metrics from heapster.

- apiGroups: [""]

resources: ["services"]

resourceNames: ["heapster"]

verbs: ["proxy"]

- apiGroups: [""]

resources: ["services/proxy"]

resourceNames: ["heapster", "http:heapster:", "https:heapster:"]

verbs: ["get"]

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: kubernetes-dashboard-minimal

namespace: kube-system

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: kubernetes-dashboard-minimal

subjects:

- kind: ServiceAccount

name: kubernetes-dashboard

namespace: kube-system

---

# ------------------- Dashboard Deployment ------------------- #

kind: Deployment

apiVersion: apps/v1beta2

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

replicas: 1

revisionHistoryLimit: 10

selector:

matchLabels:

k8s-app: kubernetes-dashboard

template:

metadata:

labels:

k8s-app: kubernetes-dashboard

spec:

containers:

- name: kubernetes-dashboard

image: k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.0

ports:

- containerPort: 8443

protocol: TCP

args:

- --auto-generate-certificates

# Uncomment the following line to manually specify Kubernetes API server Host

# If not specified, Dashboard will attempt to auto discover the API server and connect

# to it. Uncomment only if the default does not work.

# - --apiserver-host=http://my-address:port

volumeMounts:

- name: kubernetes-dashboard-certs

mountPath: /certs

# Create on-disk volume to store exec logs

- mountPath: /tmp

name: tmp-volume

livenessProbe:

httpGet:

scheme: HTTPS

path: /

port: 8443

initialDelaySeconds: 30

timeoutSeconds: 30

volumes:

- name: kubernetes-dashboard-certs

secret:

secretName: kubernetes-dashboard-certs

- name: tmp-volume

emptyDir: {}

serviceAccountName: kubernetes-dashboard

# Comment the following tolerations if Dashboard must not be deployed on master

tolerations:

- key: node-role.kubernetes.io/master

effect: NoSchedule

---

# ------------------- Dashboard Service ------------------- #

kind: Service

apiVersion: v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

ports:

- port: 443

targetPort: 8443

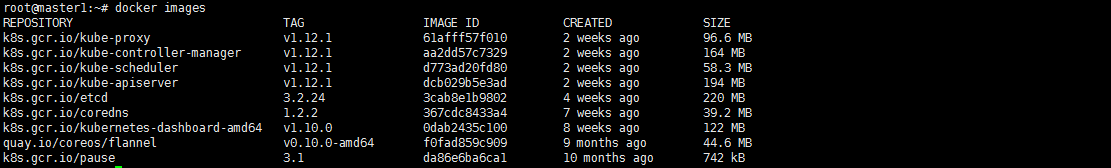
selector:

k8s-app: kubernetes-dashboard

如之前一样，还需要提前下载 k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.0镜像，通过docker pull 下载，然后在进行tag就可以，完成后执行：

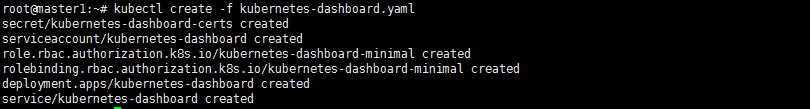
docker pull mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.0  
docker tag mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.0 k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.0

docker rmi mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.0



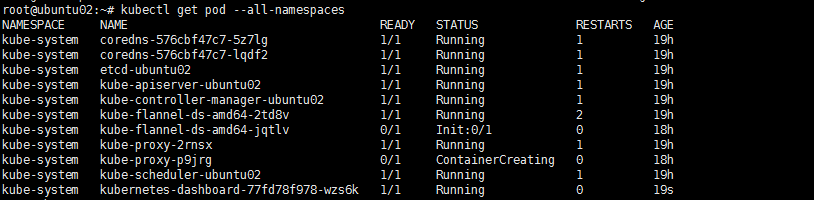
kubectl create -f kubernetes-dashboard.yaml

kubectl delete -f kubernetes-dashboard.yaml

****

kubectl get pod --all-namespaces

创建成功后，查看pod状态，dashboard已经是running状态；



然后执行：

kubectl proxy或kubectl proxy --address=172.25.105.60 --disable-filter=true

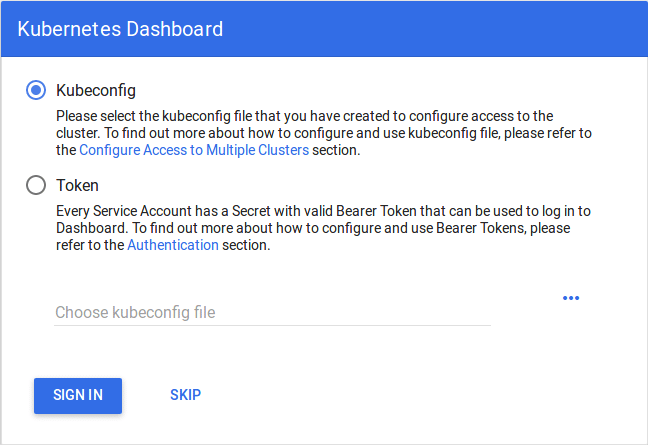
https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2018/05/kubectl-proxy.png

在其它主机访问如下地址：

<http://172.25.105.60:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/>

在master主机访问如下地址：

<http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/>



不要关闭该页面，重新打开一个命令终端，执行以下所有命令，获取Token;

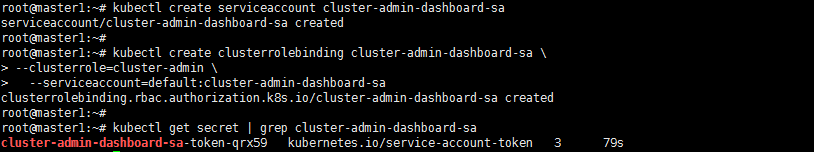
$ kubectl create serviceaccount cluster-admin-dashboard-sa

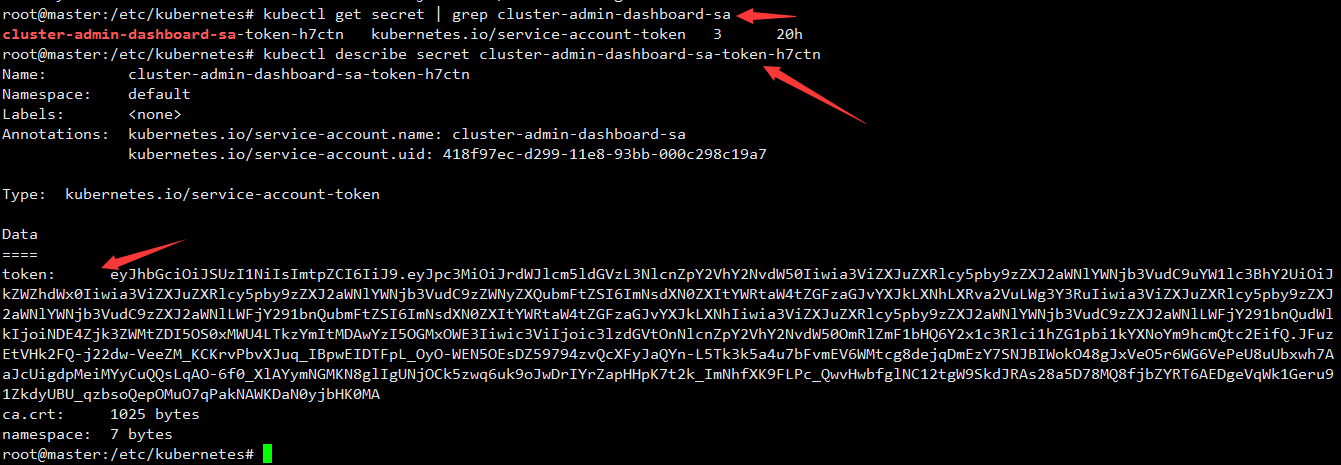
$ kubectl create clusterrolebinding cluster-admin-dashboard-sa \

--clusterrole=cluster-admin \

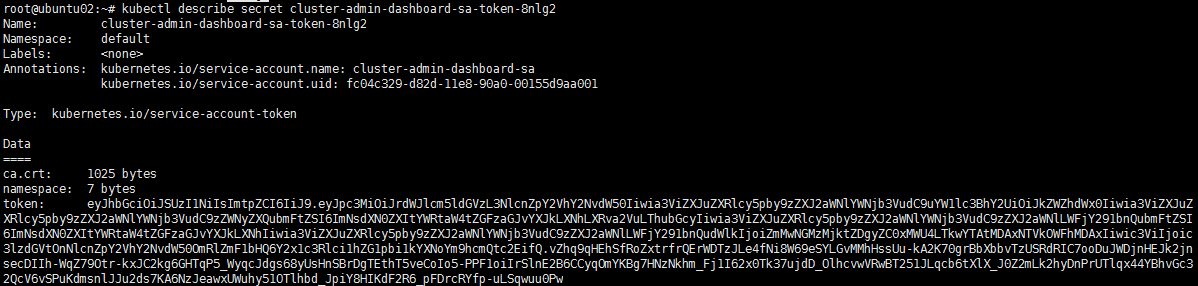
--serviceaccount=default:cluster-admin-dashboard-sa

$ kubectl get secret | grep cluster-admin-dashboard-sa



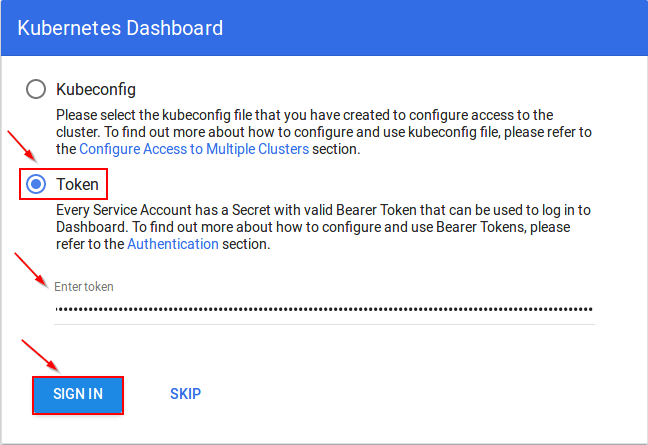


kubectl describe secret cluster-admin-dashboard-sa-token-8nlg2

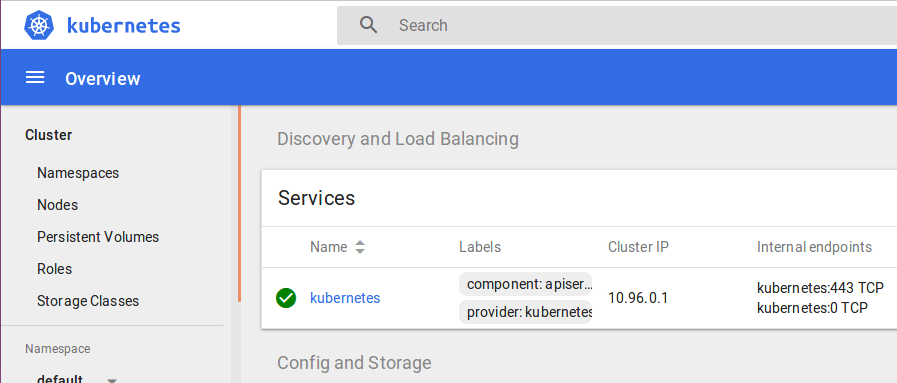


token: eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9..vZhq9qHEhSfRoZxtrfrQErWDTzJLe4fNi8W69eSYLGvMMhHssUu-kA2K70grBbXbbvTzUSRdRIC7ooDuJWDjnHEJk2jnsecDIIh-WqZ79Otr-kxJC2kg6GHTqP5\_WyqcJdgs68yUsHnSBrDgTEthT5veCoIo5-PPF1oiIrSlnE2B6CCyqOmYKBg7HNzNkhm\_Fj1I62x0Tk37ujdD\_OlhcvwVRwBT251JLqcb6tXlX\_J0Z2mLk2hyDnPrUTlqx44YBhvGc32QcV6vSPuKdmsnlJJu2ds7KA6NzJeawxUWuhy51OTlhbd\_JpiY8HIKdF2R6\_pFDrcRYfp-uLSqwuu0Pw

然后复制上面的token,粘贴到之前的登陆界面的token中；



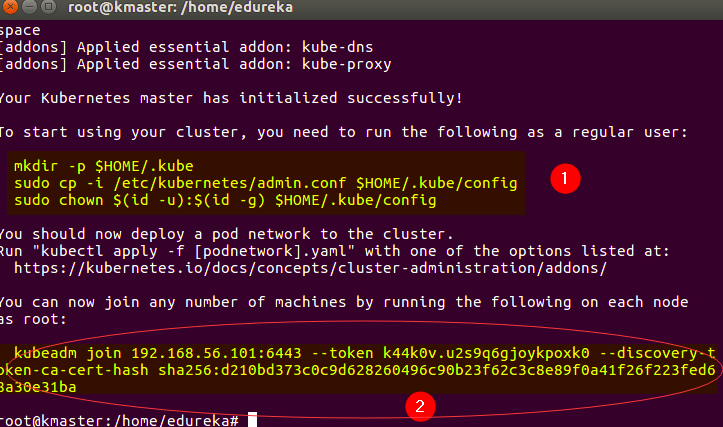
登陆成功dashboard



## 三、添加node到master

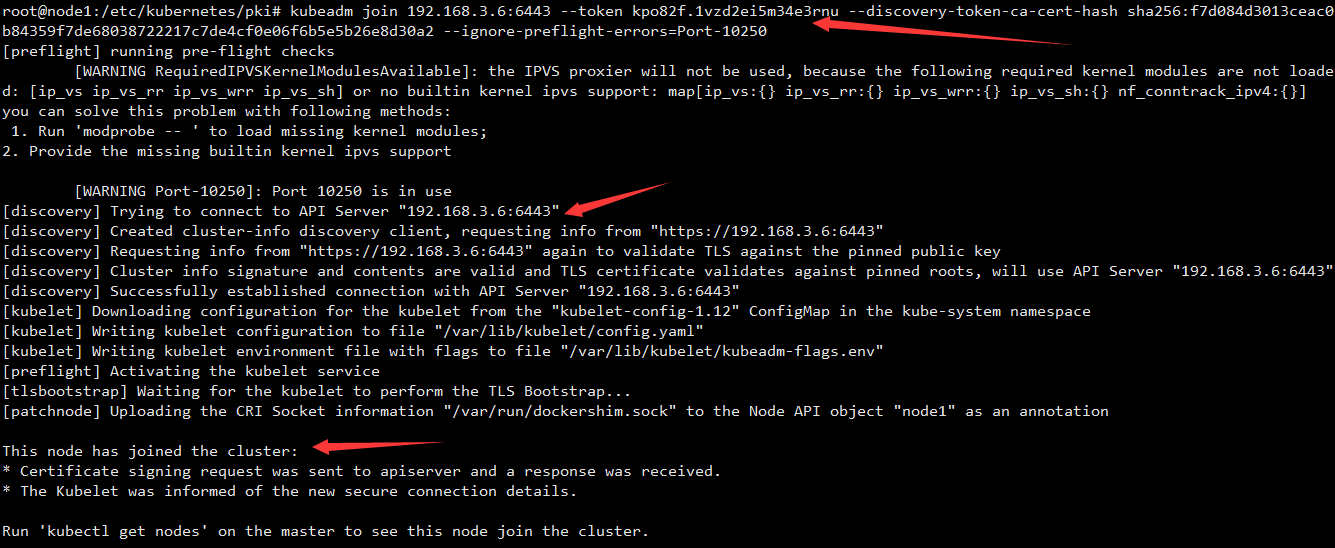
### 3.1 添加node

添加节点到master很简单，只需要一条命令即可完成，之前在master节点初始化kubernetes的时候，会有这样一个输出提示；



（截图仅供参考）

在node节点执行以下命令，即可将该节点添加到master服务器（注意：192.168.3.6是master节点的地址）；

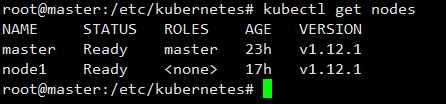


kubeadm join 172.25.105.60:6443 --token 50qncl.75mr2hwylra6ovlj --discovery-token-ca-cert-hash sha256:2d46c4fdb774fef76326c3397b07c8ea1db1927860295ceeeece9cca17109721

### 3.2 查看集群node状态

在master节点执行

$ kubectl get nodes



确认k8s集群pods运行状态

kubectl get pods -n kube-system

