

## 2001.3.25 Kubernetes 集群部署

### Kubernetes 部署方式

官方提供 kubernetes 部署三种方式：

Minikube ( 主要用来体验 k8s 的使用 )

minikube 是一个工具，可以在本地快速运行一个单点的 kubernetes，尝试 kubernetes 或日常开发的用户使用，不能用于生产环境

官方文档：<https://kubernetes.io/docs/setup/minikube/>

### 二进制包

从官方下载发行版的二进制包。手动部署每一个组件，组成 kubernetes 集群，目前企业生产环境中主要使用该方式：

下载地址：

### Kubeadm

Kubeadm 是谷歌推出的一个专门用于快速部署 kubernetes,集群的工具。在集群部署的过程中，可以通过 kubeadm init,来初始化 master 节点，然后使用 kubeadm join 将其他的节点加入到集群中。

Kubeadm 通过简单配置可以快速将一个最小可用的集群运行起来。它在设计之初关注点是快速安装并将集群运行起来，而不是一步步关于各节点环境的准备工作。同样的，kubernetes,集群在使用过程中的各种插件也不是 kubeadm 关注的重点，比如 kubernetes.集群 WEB Dashboard、prometheus 监控集群业务等。kubeadm 应用的目的是作为所有部署的基础，并通过 kubeadm 使得部署 kubernetes 集群更加容易。

Kubeadm 的简单快捷的部署应用到如下三方面

- 新用户可以从 kubeadm 开始快速搭建 Kubernetes 并了解。
- 熟悉 Kubernetes 的用户可以使用 kubeadm 快速搭建集群并测试他们的应用。
- 大型的项目可以将 kubeadm 配合其他的安装工具一起使用，形成一个比较复杂的系统。
- 官方文档：

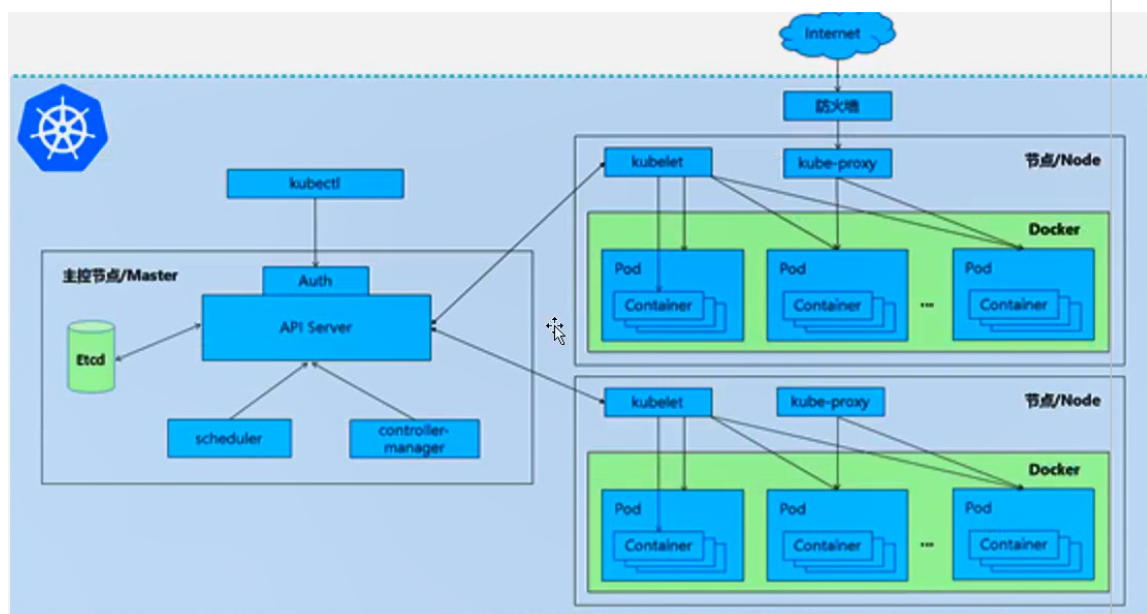
<https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm/>  
<https://kubernetes.io/docs/setup/independent/install-kubeadm/>

## 基于 kubeadm 部署 k8s 集群

### 一，环境准备

操作系统↔	IP 地址↔	主机名↔	组件↔
CentOS7.5↔	192.168.200.111↔	k8s-master↔	kubeadm、kubelet、kubectl、docker-ce↔
CentOS7.5↔	192.168.200.112↔	k8s-node01↔	kubeadm、kubelet、kubectl、docker-ce↔
CentOS7.5↔	192.168.200.113↔	k8s-node02↔	kubeadm、kubelet、kubectl、docker-ce↔

注意：所有主机配置推荐 CPU：2C+ Memory：2G+↔



#### 1.1，主机初始化配置

所有主机配置关闭防火墙和 selinux

```
iptables -F
setenforce 0
systemctl stop firewalld
systemctl disable firewalld
systemctl stop NetworkManager
systemctl disable NetworkManager
sed -i '/^SELINUX=/s/enforcing/disabled/' /etc/selinux/config
```

配置主机名并绑定 hosts，不同主机，名称不同

```
[root@k8s-master ~]# cat /etc/hosts
192.168.200.111 k8s-master
192.168.200.112 k8s-node01
192.168.200.113 k8s-node02
[root@k8s-master ~]# scp /etc/hosts 192.168.200.112:/etc/
[root@k8s-master ~]# scp /etc/hosts 192.168.200.113:/etc/
[root@k8s-master ~]# hostname k8s-master 111
[root@k8s-node-1 ~]# hostname k8s-node01 112
[root@k8s-node-2 ~]# hostname k8s-node02 113
```

## 1.2 主机配置初始化 ( 三台主机一起干 )

```
[root@k8s-node02 ~]# yum -y install vim wget net-tools lrzsz
[root@k8s-node02 ~]# swapoff -a          #零时关闭虚拟内存
[root@k8s-node02 ~]# sed -i '/swap/s/^/#/' /etc/fstab #永久关闭虚拟内存
[root@k8s-node02 ~]# vim /etc/sysctl.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
[root@k8s-node02 ~]# modprobe br_netfilter
[root@k8s-node02 ~]# sysctl -p
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
[root@k8s-master ~]# scp /etc/sysctl.conf 192.168.200.112:/etc/
[root@k8s-master ~]# scp /etc/sysctl.conf 192.168.200.113:/etc/
```

## 1.3 部署 docker 环境

三台主机分别部署 docker 环境，因为 k8s 容器需要 docker 的编排

```
[root@k8s-node02 ~]# wget -O /etc/yum.repos.d/CentOS-Media.repo
http://mirrors.aliyun.com/repo/Centos-7.repo
[root@k8s-node02 ~]# yum -y install yum-utils device-mapper-persistent-data
```

使用 YUM 方式安装 docker 时，推荐使用阿里的 YUM 源，阿里的官方开源站点地址是：  
<https://developer.aliyun.com/mirror/>，可以在站点内找到 docker 的源地址

### 三台主机部署 docker ( 这个位置的操作不建议使用 )

```
[root@k8s-node02 ~]#  
yum-config-manager --add https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo  
[root@k8s-node02 ~]# yum -y install docker-ce  
[root@k8s-node02 ~]# systemctl start docker  
[root@k8s-node02 ~]# docker --version  
Docker version 20.10.5, build 55c4c88  
[root@k8s-node02 ~]#  
scp /etc/yum.repos.d/docker-ce.repo 192.168.200.112:/etc/yum.repos.d/  
scp /etc/yum.repos.d/docker-ce.repo 192.168.200.113:/etc/yum.repos.d/
```

### 三台主机部署 docker

```
[root@k8s-master ~]# mkdir docker #每台主机的操作  
[root@k8s-master ~]# cd docker/ #每台主机的操作  
[root@k8s-master ~]# rz /上传 docker 的 rpm 包  
[root@k8s-master ~]# scp -r docker/* 192.168.200.112:/root/docker  
[root@k8s-master ~]# scp -r docker/* 192.168.200.113:/root/docker  
[root@k8s-master ~]# yum -y install *.rpm #注意安装的时候进入到 docker 目录中
```

### 配置镜像加速器 ( 所有主机的操作 )

```
[root@k8s-master ~]# vim /etc/docker/daemon.json #文件没有就创建  
{  
    "registry-mirrors":["https://nyakyfun.mirror.aliyuncs.com"]  
}  
[root@k8s-master ~]# systemctl daemon-reload  
[root@k8s-master ~]# systemctl restart docker  
[root@k8s-master ~]# docker info  
[root@k8s-master ~]# scp /etc/docker/daemon.json 192.168.200.112:/etc/docker/  
[root@k8s-master ~]# scp /etc/docker/daemon.json 192.168.200.113:/etc/docker/  
[root@k8s-master ~]# systemctl daemon-reload  
[root@k8s-master ~]# systemctl restart docker  
[root@k8s-master ~]# docker info
```

注意：三台都需要执行的命令

## 二 . 部署 k8s 集群

### 2.1 , 组件介绍

三个节点都需要安装下面三个组件

- kubeadm：安装工具，是所有的组件都会以容器的方式运行
- kubect：客户连接 k8sPAI 的工具
- kubelet：运行在 node 节点，用来启动容器的工具

### 2.2 配置阿里云 yum 源

推荐使用阿里云的 yum 源安装 ( 下载阿里云的 yum 源 )

```
[root@k8s-master yum.repos.d]# wget http://mirrors.aliyun.com/repo/Centos-7.repo
```

```
[root@k8s-master yum.repos.d]# vim /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
       https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
[root@k8s-master]#
scp /etc/yum.repos.d/kubernetes.repo 192.168.200.112:/etc/yum.repos.d/
scp /etc/yum.repos.d/kubernetes.repo 192.168.200.113:/etc/yum.repos.d/
[root@k8s-node01 yum.repos.d]# ls
backup CentOS-Media.repo epel-release-latest-7.noarch.rpm kubernetes.repo
[root@k8s-master yum.repos.d]#
scp /etc/yum.repos.d/Centos-7.repo 192.168.200.112:/etc/yum.repos.d/
scp /etc/yum.repos.d/Centos-7.repo 192.168.200.113:/etc/yum.repos.d/
```

## 2.3 安装 kubelet kubeadm kubectl

所有主机的配置

```
[root@k8s-master]# yum -y install kubelet kubeadm kubectl
[root@k8s-master ~]# systemctl enable kubelet
```

Kubelet 刚安装结束后通过 `systemctl start kubelet` 方式无法启动的，需要加入节点后或者初始化为 master 后才可以启动成功

如果在执行命令过程中出现索引 gpg 检查失败的情况，请使用 `yum install -y --nogpgcheck kubelet kubeadm kubectl` 来安装

### 2.4 配置 init-config.yaml ( master 111 的操作 )

Kubeadm 提供了很多配置项, Kubeadm 配置在 Kubernetes 集群中是存储在 ConfigMap 中的，也可将这些配置写入配置文件，方便管理复杂的配置项。Kubeadm 配内容是通过 `kubeadm config` 命令写入配置文件的。

在 master 节点安装，master 定于为 192.168.200.111，通过如下指令创建默认的 `init.config.yaml` 文件：

```
[root@k8s-master ~]# kubeadm config print init-defaults > init-config.yaml
[root@k8s-master ~]# vim init-config.yaml
12 advertiseAddress: 192.168.200.111      #定义 master 的 IP 地址
16 name: k8s-master                       #master 主机名
31 dataDir: /var/lib/etcd                 #存储位置
32 imageRepository: registry.aliyuncs.com/google_containers #镜像下载地址
38 podSubnet: 10.244.0.0/16               #手动加入。给 pod 授予一个 IP 地址
```

## 2.5 安装 master 节点

拉取所需镜像（这个步骤可以不做，因为 master 初始化时会自动下载，但是很慢。所以我们分开操作，而且不好排错）

```
[root@k8s-master ~]# kubeadm config images list --config init-config.yaml
registry.aliyuncs.com/google_containers/kube-apiserver:v1.20.0
registry.aliyuncs.com/google_containers/kube-controller-manager:v1.20.0
registry.aliyuncs.com/google_containers/kube-scheduler:v1.20.0
registry.aliyuncs.com/google_containers/kube-proxy:v1.20.0
registry.aliyuncs.com/google_containers/pause:3.2
registry.aliyuncs.com/google_containers/etcd:3.4.13-0
registry.aliyuncs.com/google_containers/coredns:1.7.

[root@k8s-master ~]# rz
//上传镜像包 k8s Software/Kubeadm yaml and images/master
[root@k8s-master ~]# cd master/
[root@k8s-master master]# ls | while read line; do docker load < $line; done #导入镜像
[root@k8s-master master]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
registry.aliyuncs.com/google_containers/kube-proxy		v1.20.0	10cc881966cf	3 months ago 118MB
registry.aliyuncs.com/google_containers/kube-controller-manager	v1.20.0		b9fa1895dcaa	3 months ago 116MB
registry.aliyuncs.com/google_containers/kube-scheduler	v1.20.0		3138b6e3d471	3 months ago 46.4MB
registry.aliyuncs.com/google_containers/kube-apiserver	v1.20.0		ca9843d3b545	3 months ago 122MB
registry.aliyuncs.com/google_containers/etcd	3.4.13-0		0369cf4303ff	7 months ago 253MB
registry.aliyuncs.com/google_containers/coredns	1.7.0		bfe3a36ebd25	9 months ago 45.2MB
registry.aliyuncs.com/google_containers/pause	3.2		80d28bedfe5d	13 months ago 683kB

安装 master 节点（初始化）

```
[root@k8s-master ~]# kubeadm init --config=init-config.yaml //初始化安装 k8s
```

根据提示操作：

Kubectl 默认会在执行的用户的家目录下面的 .kube 目录下寻找 config 文件。这里是在初始化时[kubeconfig]步骤生成的 admin.conf 拷贝到.kube/config 下

执行三条命令（这是在初始化时系统提示要执行的命令）

```
[root@k8s-master ~]# mkdir -p $HOME/.kube
[root@k8s-master ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-master ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```



## Kubeadm init 初始化主要执行了一下操作

[init]: 指定版本进行初始化操作↵  
[preflight]: 初始化前的检查和下载所需要的 Docker 镜像文件↵  
[kubelet-start]: 生成 kubelet 的配置文件” /var/lib/kubelet/config.yaml”, 没有这个文件 kubelet 无法启动, 所以初始化之前的 kubelet 实际上启动失败。↵  
[certificates]: 生成 Kubernetes 使用的证书, 存放在/etc/kubernetes/pki 目录中。↵  
[kubeconfig]: 生成 Kubeconfig 文件, 存放在/etc/kubernetes 目录中, 组件之间通信需要使用对应文件。↵  
[control-plane]: 使用/etc/kubernetes/manifest 目录下的 YAML 文件, 安装 Master 组件。↵  
[etcd]: 使用/etc/kubernetes/manifest/etcd.yaml 安装 Etcd 服务。↵  
[wait-control-plane]: 等待 control-plan 部署的 Master 组件启动。↵  
[apiclient]: 检查 Master 组件服务状态。↵  
[uploadconfig]: 更新配置↵  
[kubelet]: 使用 configMap 配置 kubelet。↵  
[patchnode]: 更新 CNI 信息到 Node 上, 通过注释的方式记录。↵  
[mark-control-plane]: 为当前节点打标签, 打了角色 Master, 和不可调度标签, 这样默认就不会使用 Master 节点来运行 Pod。↵  
[bootstrap-token]: 生成 token 记录下来, 后边使用 kubeadm join 往集群中添加节点时会用到↵  
[addons]: 安装附加组件 CoreDNS 和 kube-proxy↵  
↵  
Kubeadm 通过初始化安装是不包括网络插件的, 也就是说初始化之后是不具备相关网络功能的, 比如 k8s-master 节点上查看节点信息都是 “Not Ready” 状态、Pod 的 CoreDNS 无法提供服务等。↵

## 2.6 安装 node 节点

根据 master 安装时的提示信息

```
kubeadm join 192.168.200.111:6443 --token abcdef.0123456789abcdef \
--discovery-token-ca-cert-hash
sha256:0728040530b33d2d8c74d06193aa3c3623604e0cb562bdf932f97def561b8646
```

这是安装 master 时的提示, 把命令直接复制粘贴到 node 节点

**注意:** --token abcdef.0123456789abcdef 表示为一个证书, 但是这个证书有效期只有二十四小时, 如果证书失效, master 节点就会个 node 节点断开。这里要注意一下, 不要用在生产环境

master 查看 node 节点的状态

```
[root@k8s-master ~]# kubectl get nodes #这条命令也是 master 安装时提示的
NAME      STATUS    ROLES    AGE   VERSION
k8s-master NotReady  control-plane,master 18m   v1.20.5
k8s-node01 NotReady  <none>    106s  v1.20.5
k8s-node02 NotReady  <none>    102s  v1.20.5
```

## 2.7 安装 flannel

Master 节点 NotReady 的原因就是因为没有使用任何的网络插件, 此时 Node 和 Master 的连接还不正常。目前最流行的 Kubernetes 网络插件有 Flannel、Calico、Canal、Weave 这里

选择使用 flannel。

所有主机：

Master 上传 kube-flannel.yml，所有主机上传 flannel\_v0.12.0-amd64.tar

如果没有 kube-flannel.yml 包。就上 github 中找

```
[root@k8s-master ~]# rz /flannel_v0.12.0-amd64.tar /kube-flannel.yml
```

```
[root@k8s-master ~]# docker load < flannel_v0.12.0-amd64.tar
```

```
256a7af3acb1: Loading layer 5.844MB/5.844MB
```

```
d572e5d9d39b: Loading layer 10.37MB/10.37MB
```

```
57c10be5852f: Loading layer 2.249MB/2.249MB
```

```
7412f8eefb77: Loading layer 35.26MB/35.26MB
```

```
05116c9ff7bf: Loading layer 5.12kB/5.12kB
```

```
Loaded image: quay.io/coreos/flannel:v0.12.0-amd64
```

把 flannel\_v0.12.0-amd64.tar 包传给 node 主机

两台 Node 节点也作导入

```
[root@k8s-master ~]# scp flannel_v0.12.0-amd64.tar 192.168.200.112:/root
```

```
[root@k8s-master ~]# scp flannel_v0.12.0-amd64.tar 192.168.200.113:/root
```

```
[root@k8s-node02 ~]# docker load < flannel_v0.12.0-amd64.tar
```

master 主机应用 kube-flannel.yml 文件

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

```
podsecuritypolicy.policy/psp.flannel.unprivileged created
```

```
Warning: rbac.authorization.k8s.io/v1beta1 ClusterRole is deprecated in v1.17+, unavailable in v1.22+; use rbac.authorization.k8s.io/v1 ClusterRole
```

```
clusterrole.rbac.authorization.k8s.io/flannel created
```

```
Warning: rbac.authorization.k8s.io/v1beta1 ClusterRoleBinding is deprecated in v1.17+, unavailable in v1.22+; use rbac.authorization.k8s.io/v1 ClusterRoleBinding
```

```
clusterrolebinding.rbac.authorization.k8s.io/flannel created
```

```
serviceaccount/flannel created
```

```
configmap/kube-flannel-cfg created
```

```
daemonset.apps/kube-flannel-ds-amd64 created
```

```
daemonset.apps/kube-flannel-ds-arm64 created
```

```
daemonset.apps/kube-flannel-ds-arm created
```

```
daemonset.apps/kube-flannel-ds-ppc64le created
```

```
daemonset.apps/kube-flannel-ds-s390x created
```

```
[root@k8s-master ~]# kubectl get nodes #查看状态
```

NAME	STATUS	ROLES	AGE	VERSION
k8s-master	Ready	control-plane,master	41m	v1.20.5
k8s-node01	Ready	<none>	24m	v1.20.5
k8s-node02	Ready	<none>	24m	v1.20.5



## 2.8 节点管理命令

以下命令无需执行，仅作为了解

重置 master 和 node 配置

```
[root@k8s-master ~]# kubeadm reset
```

删除 node 配置

```
[root@k8s-master ~]# kubectl delete node 192.168.200.111 是不是 IP 地址取决于
```

kubectl get nodes 命令的结果

以下命令无需执行，仅作为了解

重置 master 和 node 配置

```
[root@k8s-master ~]# kubeadm reset
```

删除 node 配置

```
[root@k8s-master ~]# kubectl delete node 192.168.200.112
```

```
[root@k8s-node01 ~]# docker rm -f $(docker ps -aq)
```

```
[root@k8s-node01 ~]# systemctl stop kubelet
```

```
[root@k8s-node01 ~]# rm -rf /etc/kubernetes/*
```

```
[root@k8s-node01 ~]# rm -rf /var/lib/kubelet/*
```

## 三 . 安装 Dashboard UI

部署 Dashboard

dashboard 的 github 仓库地址: <https://github.com/kubernetes/dashboard>

代码仓库当中，有给出安装示例的相关部署文件，我们可以直接获取之后，直接部署即可

```
[root@k8s-master ~]# kubectl get pods -n kube-system #查看 k8s 组件的状态 ( 这条命令仅供查看 )
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-7f89b7bc75-6tvnf	1/1	Running	0	95m
coredns-7f89b7bc75-pg5nt	1/1	Running	0	95m
etcd-k8s-master	1/1	Running	0	95m
kube-apiserver-k8s-master	1/1	Running	0	95m
kube-controller-manager-k8s-master	1/1	Running	0	95m
kube-flannel-ds-amd64-krk5n	1/1	Running	0	55m
kube-flannel-ds-amd64-ksmvz	1/1	Running	0	55m

kube-flannel-ds-amd64-zgvcz	1/1	Running	0	55m
kube-proxy-blcrq	1/1	Running	0	95m
kube-proxy-tqnk4	1/1	Running	0	79m
kube-proxy-v2xlf	1/1	Running	0	79m
kube-scheduler-k8s-master	1/1	Running	0	95m

Master 上传 recommended.yaml 文件 ( 正常操作 )

```
[root@k8s-master ~]# rz /recommended.yaml
[root@k8s-master ~]# rz /metrics-scrapers_v1.0.4.tar
[root@k8s-master ~]# rz /dashboard_v2.0.0.tar
[root@k8s-master ~]# docker load < metrics-scrapers_v1.0.4.tar //镜像导入
[root@k8s-master ~]# docker load < dashboard_v2.0.0.tar
[root@k8s-master ~]#
scp metrics-scrapers_v1.0.4.tar dashboard_v2.0.0.tar 192.168.200.112:/root
scp metrics-scrapers_v1.0.4.tar dashboard_v2.0.0.tar 192.168.200.113:/root
两台 node 节点也作镜像导入
```

## 3.2 开放端口设置

在默认情况下, dashboard 并不对外开放访问端口, 这里简化操作, 直接使用 nodePort. 的方式将其端口暴露出来, 修改 service 部分的定义:

```
[root@k8s-master ~]# vim recommended.yaml
40 type: NodePort //NodePort 的方式开放一个端口
44 nodePort: 32443 //暴露的端口
164 name: cluster-admin //配置超级管理员权限
[root@k8s-master ~]# kubectl apply -f recommended.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
```

查看 dashboard 状态, 必须确保是 Running 的状态, 才能保证配置是可以使用的

```
[root@k8s-master ~]# kubectl get pods -n kubernetes-dashboard
```

NAME	READY	STATUS	RESTARTS	AGE
dashboard-metrics-scraper-7b59f7d4df-gcg46	1/1	Running	0	4m11s
kubernetes-dashboard-74d688b6bc-gclnp	1/1	Running	0	4m11s

Web 页面访问：https://192.168.200.111:32443



#### 您的连接不是私密连接

攻击者可能会试图从 192.168.200.111 窃取您的信息（例如：密码、通讯内容或信用卡信息）。了解详情

NET::ERR\_CERT\_AUTHORITY\_INVALID



如果您想获得 Chrome 最高级别的安全保护，请[开启增强型保护](#)

点击高级进入页面

高级

返回安全连接

#### Kubernetes 仪表盘

##### ☒ Token

每个 Service Account 都有一个 valid Bearer Token，可用于登录 Dashboard。要了解有关如何配置和使用 Bearer Tokens 的更多信息，请参阅 [Authentication](#) 部分。

##### ☐ Kubeconfig

请选择您创建的kubeconfig文件以配置对集群的访问权限。要了解有关如何配置和使用kubeconfig文件的更多信息，请参阅[Configure Access to Multiple Clusters](#) 部分。

输入 token \*

登录

可以看到出现如上图画面，需要我们输入一个 kubeconfig,文件或者一个 token。事实上在安装 dashboard 时,也为我们默认创建好了一个 serviceaccount,为 kubernetes-dashboard，并为其生成好了 token，我们可以通过如下指令获取该 sa 的 token:

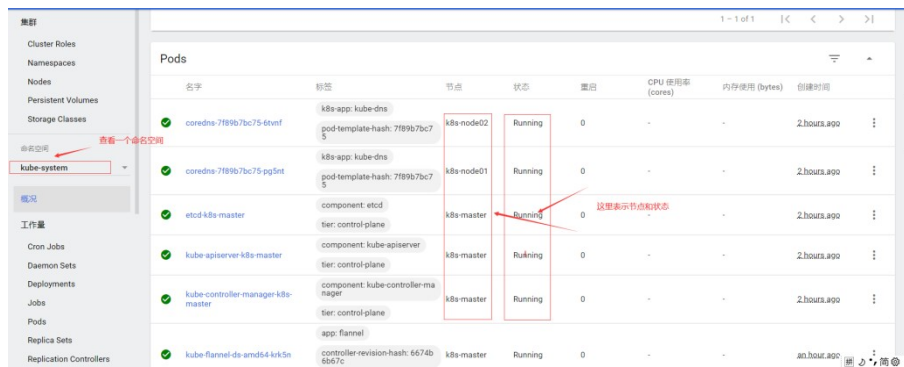
Master 主机获取一个验证码使用验证码登入

```
[root@k8s-master ~]# kubectl describe secret -n kubernetes-dashboard $(kubectl get secret -n kubernetes-dashboard |grep kubernetes-dashboard-token | awk '{print $1}') |grep token |awk '{print $2}'
```

使用这个验证码在 Token 登入

```
eyJhbGciOiJIUzI1NiIsImtpZCI6IktZbC00aE1lbTBWMkQTEp2dm5TSnA1N0JLS05sNU1yRFk2NGpuUS1seDgifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWVudC9uYW1lc3BhY2UiOiJrdWJlcm5ldGVzLWVRhc2hib2FyZCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWJyY291bnQvc2VjcmV0Lm5hbWUiOiJrdWJlcm5ldGVzLWVRhc2hib2FyZC10b2t1bi1uNDVwdyIsImt1YmVybmV0ZXMuaW8vc2VydmljZWJyY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJrdWJlcm5ldGVzLWVRhc2hib2FyZCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWJyY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6IjRkNGFkNWE0LTdhZmMtNDUwYi05ZTcwLTBkNDQxNGNiODY1MyIsInN1YiI6InN5c3RlbnRpdzZXJ2aWNlYWVudC9uYW1lc3BhY2UiOiJrdWJlcm5ldGVzLWVRhc2hib2FyZDprdWJlcm5ldGVzLWVRhc2hib2FyZCJ9.whVj9-0ckRBr381_WM72nfBlgwzAuPrsehAbLc8qFbODmNQPgDvkYBLDCA7LDbkMxt5W0Lmw1uUczTytS
```

QotjWSFMCAyI-KD8M-iQLNpc6MwAtsA4nQ6YhZEXGi0nJs9dLpg7HiWCbhA-wic19dDcuKkny6OFF7UwzQNOiW8-4MHUoh2GkTsDh0GdKK1387dOMV4BUgrQ8zCQEda8wSMHI\_opaFuay7y72KzWFcyeKNcPBN0d4fSKm7TC1Xo8jUxwxTSLEDBLhCiAaJJH4pw\_g0mBGq2jdLNH\_M\_OCwQBMSoHuj8qJDuoNphiQerAYkc56LZyA5WlmyFvZhw



名字	标签	节点	状态	重启	CPU 使用率 (cores)	内存使用 (bytes)	创建时间
coredns-789b7bc75-6tncf	k8s-app: kube-dns pod-template-hash: 7f89b7bc7	k8s-node02	Running	0	-	-	2 hours ago
coredns-789b7bc75-pg9nt	k8s-app: kube-dns pod-template-hash: 7f89b7bc7	k8s-node01	Running	0	-	-	2 hours ago
etcd-k8s-master	component: etcd tier: control-plane	k8s-master	Running	0	-	-	2 hours ago
kube-apiserver-k8s-master	component: kube-apiserver tier: control-plane	k8s-master	Running	0	-	-	2 hours ago
kube-controller-manager-k8s-master	component: kube-controller-manager tier: control-plane	k8s-master	Running	0	-	-	2 hours ago
kube-flannel-ds-amd64-kk5n	app: flannel controller-revision-hash: 6674b6b67c	k8s-master	Running	0	-	-	2 hours ago

## 四 . 安装 metrics-server

### 4.1 在 node 节点上下载镜像 ( 两台 node 的操作 )

```
[root@k8s-node01 ~]# rz //metrics-server-amd64_v0.3.6.tar 包
[root@k8s-node01 ~]# scp metrics-server-amd64_v0.3.6.tar 192.168.200.113:/root
[root@k8s-node01 ~]# docker load < metrics-server-amd64_v0.3.6.tar
[root@k8s-node02 ~]# docker load < metrics-server-amd64_v0.3.6.tar
932da5156413: Loading layer 3.062MB/3.062MB
7bf3709d22bb: Loading layer 38.13MB/38.13MB
Loaded image: bluersw/metrics-server-amd64:v0.3.6
[root@k8s-node01 ~]#
docker tag bluersw/metrics-server-amd64:v0.3.6 k8s.gcr.io/metrics-server-amd64:v0.3.6
```

### 4.2 修改 kubernetes apiserver 启动参数 ( master )

在 kube-apiserver 项中添加如下配置选项修改后 apiserver 会自动重启

```
[root@k8s-master ~]# vim /etc/kubernetes/manifests/kube-apiserver.yaml
44 --enable-aggregator-routing=true
```

### 4.3 master 上进行部署

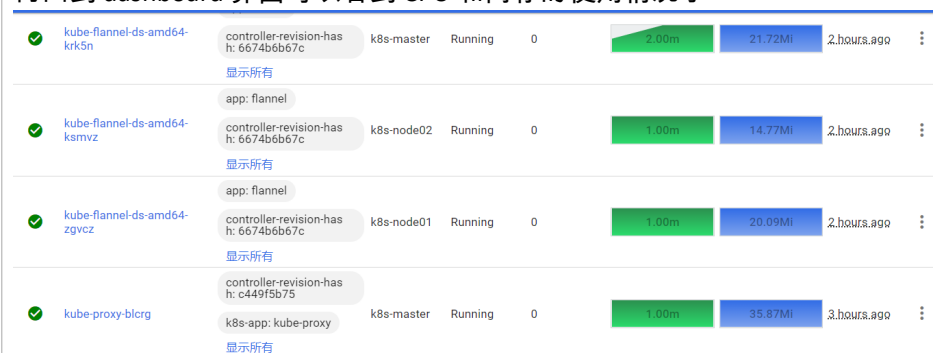
```
[root@k8s-master ~]# rz //components.yaml
[root@k8s-master ~]# kubectl apply -f components.yaml
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
Warning: apiregistration.k8s.io/v1beta1 APIService is deprecated in v1.19+, unavailable in v1.22+;
use apiregistration.k8s.io/v1 APIService
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
serviceaccount/metrics-server created
deployment.apps/metrics-server created
```

```
service/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
```

等待 1-2 分钟查看结果

```
[root@k8s-master ~]# kubectl top nodes
NAME          CPU(cores) CPU%  MEMORY(bytes) MEMORY%
k8s-master    65m       3%   1115Mi       64%
k8s-node01    22m       1%   780Mi        45%
k8s-node02    26m       1%   652Mi        37%
```

再回到 dashboard 界面可以看到 CPU 和内存的使用情况了



## 五 . 应用部署测试

下面我们部署一个简单的 Nginx WEB 服务，该容器运行时会监听 80 端口。

Kubernetes 支持两种方式创建资源：

( 1 ) 用 kubectl 命令直接创建，在命令行中通过参数指定资源的属性。此方式简单直观，比较适合临时测试或实验使用。( 不建议使用 )

```
kubectl run net-test --image=alpine --replicas=2 sleep 36000
```

( 2 ) 通过配置文件和 kubectl create/apply 创建。在配置文件中描述了应用的信息和需要达到的预期状态。

以 Deployment YAML 方式创建 nginx 服务

创建 deployment

```
[root@k8s-master ~]# rz /nginx-deployment.yaml
apiVersion: apps/v1 #apiVersion 是当前配置格式的版本
kind: Deployment #kind 是要创建的资源类型，这里是 Deployment
metadata: #metadata 是该资源的元数据，name 是必须的元数据项
  name: nginx-deployment
  labels:
    app: nginx
spec: #spec 部分是该 Deployment 的规则说明
  replicas: 3 #replicas 指定副本数量，默认为 1
  selector:
    matchLabels:
      app: nginx
  template: #template 定义 Pod 的模板，这是配置的重要部分
    metadata: #metadata 定义 Pod 的元数据，至少要填一个 label，label 的 key 和 value 可以任意指定
      labels:
        app: nginx
    spec: #spec 描述的是 Pod 的规则，此部分定义 pod 中每一个容器的属性，name 和 image 是必需的
      containers:
        - name: nginx
          image: nginx:1.19.4
          ports:
            - containerPort: 80
```

## 两台 Node 节点上传 nginx 软件包

```
[root@k8s-node01 ~]# rz //nginx-1.19.tar
[root@k8s-node01 ~]# docker load < nginx-1.19.tar
[root@k8s-node01 ~]# docker images
```

REPOSITORY	SIZE	TAG
nginx		latest
ago	133MB	
registry.aliyuncs.com/google_containers/kube-proxy	118MB	v1.20.0
ago	45.2MB	
registry.aliyuncs.com/google_containers/coredns		1.7.0
ago		
kubernetesui/dashboard		v2.0.0
ago	222MB	
kubernetesui/metrics-scraper		v1.0.4

```
[root@k8s-node01 ~]# docker tag nginx nginx:1.19.4 //两台 node 主机修改镜像名
[root@k8s-node01 ~]# docker images
```

## 创建 nginx-deployment 应用

```
[root@k8s-master ~]# kubectl create -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
```

## 查看 deployment 状态

```
[root@k8s-master ~]# kubectl get deployment
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx-deployment	3/3	3	3	88s

查看具体某个 pod 的状态信息

```
[root@k8s-master ~]# kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-deployment-7947dc656-2q7wt	1/1	Running	0	2m30s
nginx-deployment-7947dc656-gql6m	1/1	Running	0	2m30s
nginx-deployment-7947dc656-j55z9	1/1	Running	0	2m30s

查看具体某个 pod 的状态信息

```
[root@k8s-master ~]# kubectl describe pod nginx-deployment-7947dc656-2q7wt
```

这里需要指定 ID 号

```
Name:          nginx-deployment-7947dc656-2q7wt
Namespace:     default
Priority:       0
Node:          k8s-node02/192.168.200.113
Start Time:    Fri, 26 Mar 2021 19:11:34 +0800
Labels:        app=nginx
               pod-template-hash=7947dc656
Annotations:   <none>
Status:        Running
IP:            10.244.2.8
IPs:           10.244.2.8
Controlled By: ReplicaSet/nginx-deployment-7947dc656
Containers:
  nginx:
    Container ID:  docker://9498f31cb6ea55499a03cc243d0892c19f9731bc162bfea5d39789934a81f18c
    Image:         nginx:1.19.4
    Image ID:      docker://sha256:ae2feff98a0cc5095d97c6c283dcd33090770c76d63877caa99aefbbe4343bdd
    Port:          80/TCP
    Host Port:     0/TCP
    State:         Running
      Started:     Fri, 26 Mar 2021 19:11:35 +0800
      Ready:       True
      Restart Count: 0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-6lrcm (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-6lrcm:
    Type:          Secret (a volume populated by a Secret)
    SecretName:    default-token-6lrcm
```

查看 pod 的位置信息

```
[root@k8s-master ~]# kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMIN
nginx-deployment-7947dc656-2q7wt	1/1	Running	0	9m2s	10.244.2.8	k8s-node02	<none>
>	<none>						
nginx-deployment-7947dc656-gql6m	1/1	Running	0	9m2s	10.244.1.7	k8s-node01	<none>
>	<none>						
nginx-deployment-7947dc656-j55z9	1/1	Running	0	9m2s	10.244.1.6	k8s-node01	<none>
>	<none>						

直接访问一个 nginx 服务

```
[root@k8s-master ~]# curl http://10.244.2.8 : 这里需要指定一个 server 的 IP 地址
```



```
body {  
  width: 35em;  
  margin: 0 auto;  
  font-family: Tahoma, Verdana, Arial, sans-serif;  
}
```

删除一个 pod

```
[root@k8s-master ~]# kubectl delete pods nginx-deployment-7947dc656-gql6m
```

这里需要指定一个 pod 的名字，但是这里 pod 不会被删掉，因为系统会再次帮你启动一个 pod

想要删除 pod 就要直接删除一个 .yaml 文件。

例如：删除一个 nginx-deployment.yaml 文件

```
[root@k8s-master ~]# kubectl delete -f nginx-deployment.yaml
```

```
deployment.apps "nginx-deployment" deleted
```

```
[root@k8s-master ~]# kubectl get po
```

```
No resources found in default namespace
```

