目录

[一、服务器规划 1](#_Toc533011379)

[二、安装docker-ce 2](#_Toc533011380)

[三、安装配置etcd 2](#_Toc533011381)

[3.1、etcd简介 2](#_Toc533011382)

[3.2、etcd的应用场景 3](#_Toc533011383)

[3.3、etcd工作原理 3](#_Toc533011384)

[3.4、etcd部署单节点 4](#_Toc533011385)

[四、安装 flannel 软件并设置 docker 参数 4](#_Toc533011386)

[4.1、flannel介绍 4](#_Toc533011387)

[4.2、flannel网络原理 4](#_Toc533011388)

[4.3、flannel部署 5](#_Toc533011389)

[五、配置kubernets的master 7](#_Toc533011390)

[六、部署kubernets的node节点 8](#_Toc533011391)

[6.1、pause容器介绍 8](#_Toc533011392)

[6.2、node部署 8](#_Toc533011393)

[七、部署busybox 10](#_Toc533011394)

[八、部署kubedns 11](#_Toc533011395)

[8.1、kube-dns介绍 11](#_Toc533011396)

[8.2、kube-dns部署 12](#_Toc533011397)

[九、部署kubernetes-dashboard 13](#_Toc533011398)

[9.1、kubernetes-dashboard部署 13](#_Toc533011399)

[9.2、kubernetes-dashboard服务验证 14](#_Toc533011400)

[十、Heapster+InfluxDB+Grafana监控服务搭建 14](#_Toc533011401)

[十一、配置ingress暴露kubernetes-dashboard 16](#_Toc533011402)

[11.1、kubernetes服务暴露方式介绍 16](#_Toc533011403)

[11.2、ingress默认页面和控制器部署 16](#_Toc533011404)

[11.3、kubernetes-dashboard通过ingress暴露服务 17](#_Toc533011405)

[11.4、服务验证和排错 18](#_Toc533011406)

[十二、参考文档 19](#_Toc533011407)

# 一、服务器规划

|  |  |  |  |
| --- | --- | --- | --- |
| Ip | 操作系统 | 节点类型 | 安装组件 |
| 192.168.240.16 | Centos 7.4 | Master | Etcd  Kube-apiserver  Kube-controller-manager  Kube-schedule  Flannel  Docker（主节点可以不装） |
| 192.168.240.18 | Centos 7.4 | Node | Kubelet  Kube-proxy  Docker  Flannel |
| 192.168.240.20 | Centos 7.4 | Node | Kubelet  Kube-proxy  Docker  Flannel |

三台机器关闭selinux，关闭firewall，关闭交换分区（swapoff -a），添加host主机信息。

# 二、安装docker-ce

安装epel源，安装docker-ce，三台都要操作，以下以master为例

[root@k8s-master-240-16 ~]# wget <http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

[root@k8s-master-240-16 ~]# rpm -ivh epel-release-latest-7.noarch.rpm

[root@k8s-master-240-16 ~]# yum makecache

[root@k8s-master-240-16 ~]# yum install -y yum-utils device-mapper-persistent-data lvm2

[root@k8s-master-240-16 ~]# yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

[root@k8s-master-240-16 ~]# yum -y install docker-ce

[root@k8s-master-240-16 ~]# mkdir -p /etc/docker

[root@k8s-master-240-16 ~]# sudo tee /etc/docker/daemon.json <<-'EOF'

> {

> "registry-mirrors": ["https://vk9tdna6.mirror.aliyuncs.com"]

> }

> EOF

[root@k8s-master-240-16 ~]# systemctl daemon-reload

[root@k8s-master-240-16 ~]# systemctl start docker

[root@k8s-master-240-16 ~]# systemctl enable docker

参考：<https://cr.console.aliyun.com/cn-hangzhou/mirrors>

# 三、安装配置etcd

## 3.1、etcd简介

etcd是一个高可用的分布式键值(key-value)数据库。etcd内部采用raft协议作为一致性算法，etcd基于Go语言实现。

etcd是一个服务发现系统，具备以下的特点：

简单：安装配置简单，而且提供了HTTP API进行交互，使用也很简单

安全：支持SSL证书验证

快速：根据官方提供的benchmark数据，单实例支持每秒2k+读操作

可靠：采用raft算法，实现分布式系统数据的可用性和一致性

## 3.2、etcd的应用场景

用于服务发现，服务发现(ServiceDiscovery)要解决的是分布式系统中最常见的问题之一，即在同一个分布式集群中的进程或服务如何才能找到对方并建立连接。

要解决服务发现的问题，需要具备下面三种必备属性。

- 一个强一致性、高可用的服务存储目录。

基于Ralf算法的etcd天生就是这样一个强一致性、高可用的服务存储目录。

一种注册服务和健康服务健康状况的机制。

用户可以在etcd中注册服务，并且对注册的服务配置key TTL，定时保持服务的心跳以达到监控健康状态的效果。

一种查找和连接服务的机制。

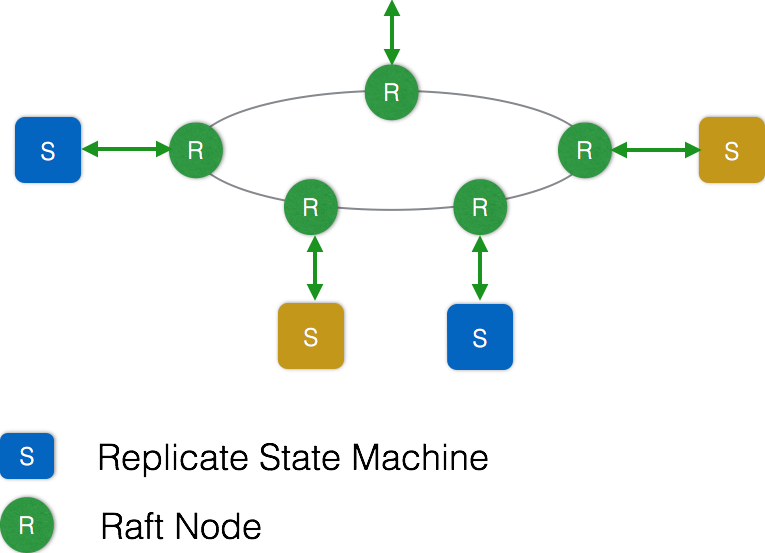
通过在etcd指定的主题下注册的服务业能在对应的主题下查找到。为了确保连接，我们可以在每个服务机器上都部署一个proxy模式的etcd，这样就可以确保访问etcd集群的服务都能够互相连接。

实际上，etcd作为一个受到Zookeeper与doozer启发而催生的项目，除了拥有与之类似的功能外，更具有以下4个特点{![引自[**Docker**](http://lib.csdn.net/base/4)官方文档]}。

* 简单：基于HTTP+JSON的API让你用curl命令就可以轻松使用。
* 安全：可选SSL客户认证机制。
* 快速：每个实例每秒支持一千次写操作。
* 可信：使用Raft算法充分实现了分布式。

## 3.3、etcd工作原理

ETCD使用Raft协议来维护集群内各个节点状态的一致性。简单说，ETCD集群是一个分布式系统，由多个节点相互通信构成整体对外服务，每个节点都存储了完整的数据，并且通过Raft协议保证每个节点维护的数据是一致的。



如图所示，每个ETCD节点都维护了一个状态机，并且，任意时刻至多存在一个有效的主节点。主节点处理所有来自客户端写操作，通过Raft协议保证写操作对状态机的改动会可靠的同步到其他节点。

ETCD工作原理核心部分在于Raft协议。本节接下来将简要介绍Raft协议，具体细节请参考其[论文]。  
Raft协议正如论文所述，确实方便理解。主要分为三个部分：选主，日志复制，安全性。、

参考文档：https://yq.aliyun.com/articles/11035

## 3.4、etcd部署单节点

这里只做单节点etcd，只在master上面安装

[root@k8s-master-240-16 ~]# yum install -y etcd

[root@k8s-master-240-16 ~]# vi /etc/default/etcd

ETCD\_DATA\_DIR="/var/lib/etcd/default"

ETCD\_LISTEN\_CLIENT\_URLS="http://0.0.0.0:2379"

ETCD\_ADVERTISE\_CLIENT\_URLS=http://0.0.0.0:2379

[root@k8s-master-240-16 ~]# sed -i 's/localhost/0.0.0.0/g' /etc/etcd/etcd.conf

[root@k8s-master-240-16 ~]# systemctl start etcd && systemctl enable etcd

Flannel的配置信息全部在Etcd里面记录，往Etcd里面写入下面这个最简单的配置，只指定Flannel能用来分配给每个Docker节点的拟IP地址段：

[root@k8s-master-240-16 ~]# etcdctl --endpoints http://192.168.240.16:2379 set /coreos.com/network/config '{"NetWork":"10.0.0.0/16"}'

{"NetWork":"10.0.0.0/16"}

[root@k8s-master-240-16 ~]#

# 四、安装 flannel 软件并设置 docker 参数

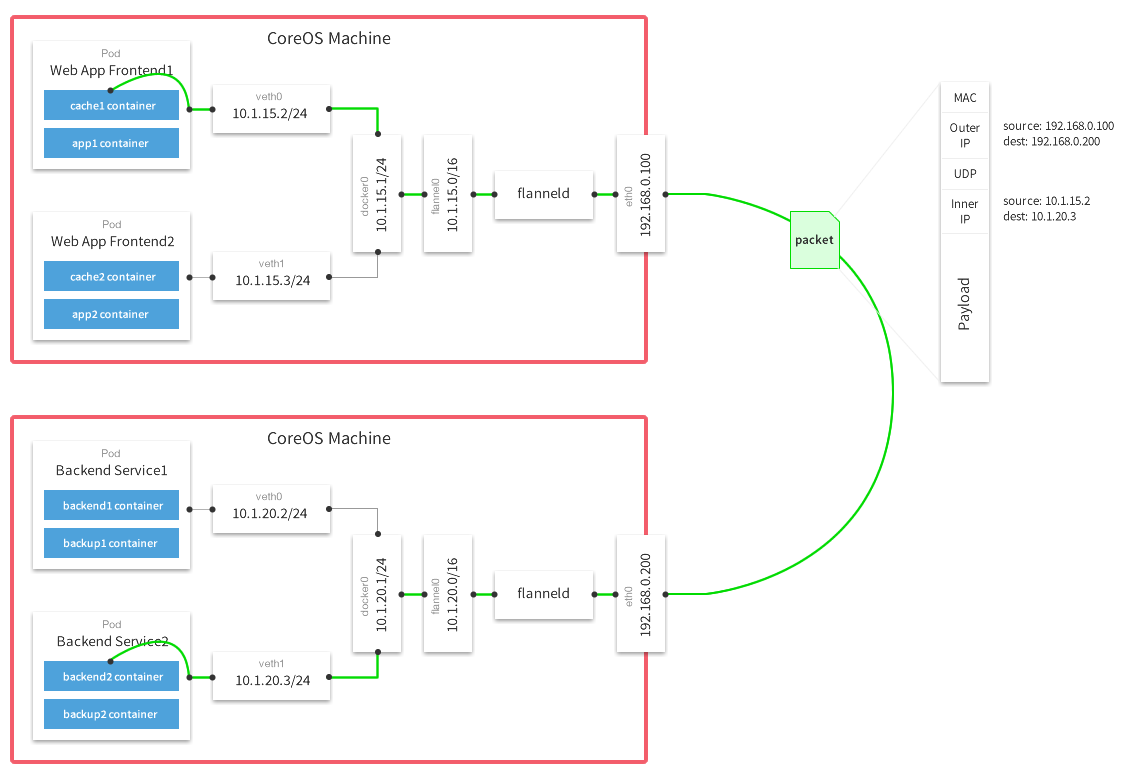
## 4.1、flannel介绍

Flannel是 CoreOS 团队针对 Kubernetes 设计的一个覆盖网络（Overlay Network）工具，其目的在于帮助每一个使用 Kuberentes 的 CoreOS 主机拥有一个完整的子网，简单来说，它的功能是让集群中的不同节点主机创建的Docker容器都具有全集群唯一的虚拟IP地址。

flannel实质上是一种“覆盖网络(overlay network)”，也就是将TCP数据包装在另一种网络包里面进行路由转发和通信，目前已经支持UDP、VxLAN、AWS VPC和GCE路由等数据转发方式。

## 4.2、flannel网络原理

默认的节点间数据通信方式是UDP转发，在Flannel的GitHub页面有如下的一张原理图：



## 4.3、flannel部署

下载flannel（<https://github.com/coreos/flannel/releases>）

[root@k8s-master-240-16 local]# tar -zxvf flannel-v0.10.0-linux-amd64.tar.gz

[root@k8s-master-240-16 local]# cp flanneld mk-docker-opts.sh /usr/local/bin/

[root@k8s-master-240-16 /]# vi /usr/lib/systemd/system/flanneld.service

[Unit]

Description=Flanneld overlay address etcd agent

After=network.target

After=network-online.target

Wants=network-online.target

Before=docker.service

[Service]

Type=notify

EnvironmentFile=/etc/sysconfig/flanneld

ExecStart=/usr/local/bin/flanneld $FLANNEL\_OPTIONS

ExecStartPost=/usr/local/bin/mk-docker-opts.sh -k DOCKER\_NETWORK\_OPTIONS -d /run/flannel/docker

Restart=on-failure

[Install]

WantedBy=multi-user.target

RequiredBy=docker.service

[root@k8s-master-240-16 ~]# vi /usr/lib/systemd/system/docker.service

[Unit]

Description=Docker Application Container Engine

Documentation=https://docs.docker.com

After=network-online.target firewalld.service

Wants=network-online.target

[Service]

Type=notify

# the default is not to use systemd for cgroups because the delegate issues still

# exists and systemd currently does not support the cgroup feature set required

# for containers run by docker

ExecStart=/usr/bin/dockerd $DOCKER\_NETWORK\_OPTIONS

EnvironmentFile=/run/flannel/docker

ExecReload=/bin/kill -s HUP $MAINPID

# Having non-zero Limit\*s causes performance problems due to accounting overhead

# in the kernel. We recommend using cgroups to do container-local accounting.

LimitNOFILE=infinity

LimitNPROC=infinity

LimitCORE=infinity

# Uncomment TasksMax if your systemd version supports it.

# Only systemd 226 and above support this version.

#TasksMax=infinity

TimeoutStartSec=0

# set delegate yes so that systemd does not reset the cgroups of docker containers

Delegate=yes

# kill only the docker process, not all processes in the cgroup

KillMode=process

# restart the docker process if it exits prematurely

Restart=on-failure

StartLimitBurst=3

StartLimitInterval=60s

[Install]

WantedBy=multi-user.target

[root@k8s-master-240-16 ~]# vi /etc/sysconfig/flanneld

# Flanneld configuration options

# etcd url location. Point this to the server where etcd runs

FLANNEL\_ETCD\_ENDPOINTS="http://192.168.240.16:2379"

# etcd config key. This is the configuration key that flannel queries

# For address range assignment

FLANNEL\_ETCD\_KEY="/coreos.com/network"

FLANNEL\_OPTIONS="--etcd-endpoints=http://192.168.240.16:2379 --ip-masq=true"

# Any additional options that you want to pass

#FLANNEL\_OPTIONS=""

[root@k8s-master-240-16 ~]# systemctl daemon-reload

[root@k8s-master-240-16 ~]# systemctl start flanneld

[root@k8s-master-240-16 ~]# systemctl enable flanneld

[root@k8s-master-240-16 ~]# systemctl restart docker ###注意docker网络被重新覆盖了

# 五、配置kubernets的master

从 [CHANGELOG页面](https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG.md) 下载 client 或 server tarball 文件，server 的 tarball kubernetes-server-linux-amd64.tar.gz 已经包含了 client(kubectl) 二进制文件，所以不用单独下载kubernetes-client-linux-amd64.tar.gz文件。

[root@k8s-master-240-16 local]# wget <https://dl.k8s.io/v1.12.4/kubernetes-server-linux-amd64.tar.gz>

[root@k8s-master-240-16 local]# tar -zxvf kubernetes-server-linux-amd64.tar.gz

[root@k8s-master-240-16 local]# cd kubernetes

[root@k8s-master-240-16 kubernetes]# cp -r server/bin/{kube-apiserver,kube-controller-manager,kube-scheduler,kubectl,kube-proxy,kubelet} /usr/local/bin/

部署api-server

上传部署脚本，脚本包含增加kube-api的配置文件，启动文件，启动kube-api。



[root@k8s-master-240-16 kubernetes]# mkdir -pv /etc/kubernetes/

[root@k8s-master-240-16 kubernetes]# ./apiserver.sh 192.168.240.16 <http://192.168.240.16:2379>

[root@k8s-master-240-16 kubernetes]# systemctl status kube-apiserver ###验证下状态，查看是否启动

部署scheduler

上传部署脚本，脚本包含增加scheduler的配置文件，启动文件，启动scheduler。



[root@k8s-master-240-16 kubernetes]# ./scheduler.sh 192.168.240.16

[root@k8s-master-240-16 kubernetes]# systemctl status kube-scheduler ###验证下状态，查看是否启动

部署controller-manager

上传部署脚本，脚本包含增加scheduler的配置文件，启动文件，启动controller-manager。



[root@k8s-master-240-16 kubernetes]# ./controller-manager.sh 192.168.240.16

[root@k8s-master-240-16 kubernetes]# systemctl status kube-controller-manager ###验证下状态，查看是否启动

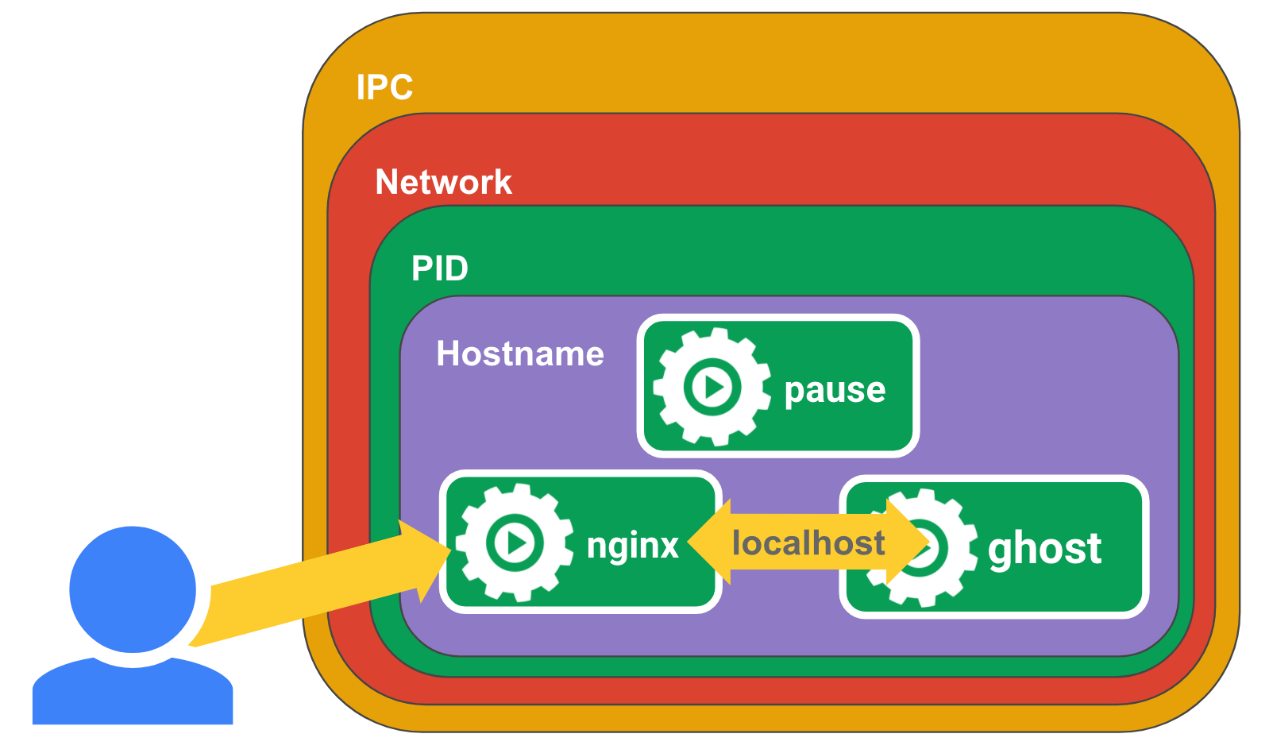
# 六、部署kubernets的node节点

## 6.1、pause容器介绍

Pause容器，又叫Infra容器，kubernetes中的pause容器主要为每个业务容器提供以下功能：

* 在pod中担任Linux命名空间共享的基础；
* 启用pid命名空间，开启init进程。

在[The Almighty Pause Container](https://www.ianlewis.org/en/almighty-pause-container)这篇文章中做出了详细的说明，pause容器的作用可以从这个例子中看出，首先见下图：



## 6.2、node部署

以node1为例，node2使用同样的方法

[root@k8s-node1-240-18 local]# tar -zxvf kubernetes-server-linux-amd64.tar.gz

[root@k8s-node1-240-18 local]# cd kubernetes

[root@k8s-node1-240-18 kubernetes]# cp -r ./server/bin/{kube-proxy,kubelet} /usr/local/bin/

部署kubelet

**创建 kubelet.service**

[root@k8s-node1-240-18 /]# vi /usr/lib/systemd/system/kubelet.service

[Unit]

Description=Kubernetes Kubelet Server

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

After=docker.service

Requires=docker.service

[Service]

#手动创建此目录

WorkingDirectory=-/var/kubeletwork

EnvironmentFile=-/etc/kubernetes/kubelet

ExecStart=/usr/local/bin/kubelet $KUBELET\_ARGS

Restart=on-failure

[Install]

WantedBy=multi-user.target

**创建 kube-proxy.service**

[root@k8s-node1-240-18 /]# vi /usr/lib/systemd/system/kube-proxy.service  
[Unit]

Description=Kubernetes Kube-Proxy Server

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

After=network.service

Requires=network.service

[Service]

EnvironmentFile=-/etc/kubernetes/proxy

ExecStart=/usr/local/bin/kube-proxy $KUBE\_PROXY\_ARGS

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

**创建 kubelet.kubeconfig**

[root@k8s-node1-240-18 /]# vi /etc/kubernetes/kubelet.kubeconfig

apiVersion: v1

kind: Config

clusters:

- cluster:

server: http://192.168.240.16:8080/

name: local

contexts:

- context:

cluster: local

name: local

current-context: local

创建/etc/kubernetes/kubelet

[root@k8s-node1-240-18 /]# vi /etc/kubernetes/kubelet

KUBELET\_ARGS="--kubeconfig=/etc/kubernetes/kubelet.kubeconfig --hostname-override=192.168.240.18 --logtostderr=true --log-dir=/var/log/kubernets/log --v=2 --address=192.168.240.18 --port=10250 --fail-swap-on=false --pod-infra-container-image=registry.cn-hangzhou.aliyuncs.com/google\_containers/pause:3.1"

【注】：此处hostname-override和address为node1节点的ip，此处的镜像为pause为pod管理的基础镜像，此处使用阿里的镜像下载（<https://cr.console.aliyun.com/cn-hangzhou/images>）

创建/etc/kubernetes/proxy

[root@k8s-node1-240-18 /]# vi /etc/kubernetes/proxy

KUBE\_PROXY\_ARGS="--master=http://192.168.240.16:8080 --hostname-override=k8s-node1-240-18 --v=2 --logtostderr=true --log-dir=/var/log/kubernets/log"

【注】此处master为主节点的ip，hostname-override为node1的主机名

配置开机启动

[root@k8s-node1-240-18 /]# systemctl daemon-reload

[root@k8s-node1-240-18 /]# systemctl enable kubelet

[root@k8s-node1-240-18 /]# systemctl enable kube-proxy

[root@k8s-node1-240-18 /]# systemctl start kubelet

[root@k8s-node1-240-18 /]# systemctl start kube-proxy

[root@k8s-node1-240-18 /]# systemctl status kubelet ###验证启动状态

[root@k8s-node1-240-18 /]# systemctl status kube-proxy ###验证启动状态

**查看集群状态**

[root@k8s-master-240-16 kubernetes]# kubectl get node

NAME STATUS ROLES AGE VERSION

192.168.240.18 Ready <none> 8m30s v1.12.4

192.168.240.20 Ready <none> 20s v1.12.4

[root@k8s-master-240-16 kubernetes]#

[root@k8s-master-240-16 kubernetes]# kubectl get componentstatus

NAME STATUS MESSAGE ERROR

controller-manager Healthy ok

scheduler Healthy ok

etcd-0 Healthy {"health": "true"}

[root@k8s-master-240-16 kubernetes]#

参考文档：<https://blog.csdn.net/a735131232/article/details/83352517>

参考文档：<https://jimmysong.io/kubernetes-handbook/practice/node-installation.html>

# 七、部署busybox

[root@k8s-master-240-16 k8s]# mkdir -pv busybox

上传busybox的yaml文件



[root@k8s-master-240-16 busybox]# kubectl create -f busybox.yaml

Error from server (ServerTimeout): error when creating "busybox.yaml": No API token found for service account "default", retry after the token is automatically created and added to the service account

解决办法：

[root@k8s-master-240-16 busybox]# vi /etc/kubernetes/kube-apiserver

KUBE\_ADMISSION\_CONTROL="--admission-control=" 改为

KUBE\_ADMISSION\_CONTROL="--admission-control=NamespaceLifecycle,NamespaceExists,LimitRanger,ResourceQuota"

[root@k8s-master-240-16 busybox]# systemctl restart kube-apiserver.service

[root@k8s-master-240-16 busybox]# systemctl status kube-apiserver.service

[root@k8s-master-240-16 busybox]# vi /lib/systemd/system/kube-apiserver.service

ExecStart 最后增加${KUBE\_ADMISSION\_CONTROL}

参考文档：<https://www.cnblogs.com/menkeyi/p/7096605.html>

[root@k8s-master-240-16 busybox]# systemctl restart kube-apiserver.service

[root@k8s-master-240-16 busybox]# kubectl create -f busybox.yaml

验证busybox容器是否起来

[root@k8s-master-240-16 busybox]# kubectl get pod -o wide busybox

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE

busybox 1/1 Running 0 2m31s 10.0.94.2 192.168.240.18 <none>

[root@k8s-master-240-16 busybox]#

# 八、部署kubedns

## 8.1、kube-dns介绍

kube-dns主要实现K8S集群内部服务发现以及集群内部通过服务名进行访问。kubernetes集群中的服务都会有一个只用于内部访问的ClusterIP

kube-dns可以实现集群内部的服务名与ClusterIP的解析，实现服务之间通过服务名的访问，方便部署管理

Kubernetes DNS pod 中包括 3 个容器：

* kubedns：kubedns 进程监视 Kubernetes master 中的 Service 和 Endpoint 的变化，并维护内存查找结构来服务DNS请求。
* dnsmasq：dnsmasq 容器添加 DNS 缓存以提高性能。
* sidecar：sidecar 容器在执行双重健康检查（针对 dnsmasq 和 kubedns）时提供单个健康检查端点（监听在10054端口）。

。

## 8.2、kube-dns部署

上官方网址下载需要的yaml部署文件：<https://github.com/kubernetes/kubernetes/tree/release-1.8/cluster/addons/dns>

在master节点上传yaml文件：

在node节点上传用到的镜像文件，并生成镜像

导入镜像

[root@k8s-node1-240-18 /]# docker load -i exechealthz-amd64.tar.gz

[root@k8s-node1-240-18 /]# docker load -i kubedns-amd64.tar.gz

[root@k8s-node1-240-18 /]# docker load -i kube-dnsmasq-amd64.tar.gz

查看镜像

[root@k8s-node1-240-18 /]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

registry.cn-hangzhou.aliyuncs.com/mmmgk/exechealthz-amd64 1.2 8ce7b1ef48a5 2 days ago 8.37MB

busybox latest 59788edf1f3e 2 months ago 1.15MB

registry.cn-hangzhou.aliyuncs.com/google\_containers/pause 3.1 da86e6ba6ca1 12 months ago 742kB

hub.c.163.com/allan1991/kubedns-amd64 1.9 26cf1ed9b144 2 years ago 47MB

hub.c.163.com/allan1991/kube-dnsmasq-amd64 1.4 3ec65756a89b 2 years ago 5.13MB

hub.c.163.com/allan1991/exechealthz-amd64 1.2 93a43bfb39bf 2 years ago 8.37MB

[root@k8s-node1-240-18 /]#

修改yaml文件

[root@k8s-master-240-16 kubedns]# vi kubedns-deployment.yaml

args:

- --kube-master-url=http://192.168.240.16:8080

部署kube-dns服务

[root@k8s-master-240-16 kubedns]# kubectl create -f kubedns-svc.yaml

[root@k8s-master-240-16 kubedns]# kubectl create -f kubedns-deployment.yaml

验证kubedns pod

[root@k8s-master-240-16 kubedns]# kubectl get pod --namespace=kube-system

NAME READY STATUS RESTARTS AGE

kube-dns-665845c89f-7wqzf 3/3 Running 0 32s

kube-dns-665845c89f-gvql4 3/3 Running 0 32s

kube-dns-665845c89f-nc2fj 3/3 Running 0 32s

[root@k8s-master-240-16 kubedns]#

故障排查：

[root@k8s-master-240-16 kubedns]# kubectl describe pod kube-dns-6676767856-rqwdf

[root@k8s-master-240-16 kubedns]# kubectl logs kube-dns-6676767856-kns86 -n kube-system

Error from server (BadRequest): a container name must be specified for pod kube-dns-6676767856-kns86, choose one of: [kubedns dnsmasq healthz]

# 九、部署kubernetes-dashboard

## 9.1、kubernetes-dashboard部署

上传kubernetes-dashboard的yaml文件



[root@k8s-master-240-16 k8s]# mkdir -pv kubernetes-dashboard

修改yaml文件

spec:

containers:

- name: kubernetes-dashboard

image: registry.cn-hangzhou.aliyuncs.com/kubernete/kubernetes-dashboard-amd64:v1.10.0

args:

- --apiserver-host=http://192.168.240.16:8080

拉取kubernetes-dashboard需要用到的镜像

[root@k8s-node1-240-18 /]# docker pull registry.cn-hangzhou.aliyuncs.com/kubernete/kubernetes-dashboard-amd64:v1.10.0

部署kubernetes-dashboard

[root@k8s-master-240-16 kubernetes-dashboard]# kubectl create -f kubernetes-dashboard.yaml

## 9.2、kubernetes-dashboard服务验证

验证服务

[root@k8s-master-240-16 kubernetes-dashboard]# kubectl get svc --namespace=kube-system

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kube-dns ClusterIP 10.10.10.2 <none> 53/UDP,53/TCP 15m

kubernetes-dashboard NodePort 10.10.10.111 <none> 80:30479/TCP 49s ##30479是node暴露出来的port

[root@k8s-master-240-16 kubernetes-dashboard]# kubectl get pod --namespace=kube-system

NAME READY STATUS RESTARTS AGE

kube-dns-665845c89f-7wqzf 3/3 Running 0 15m

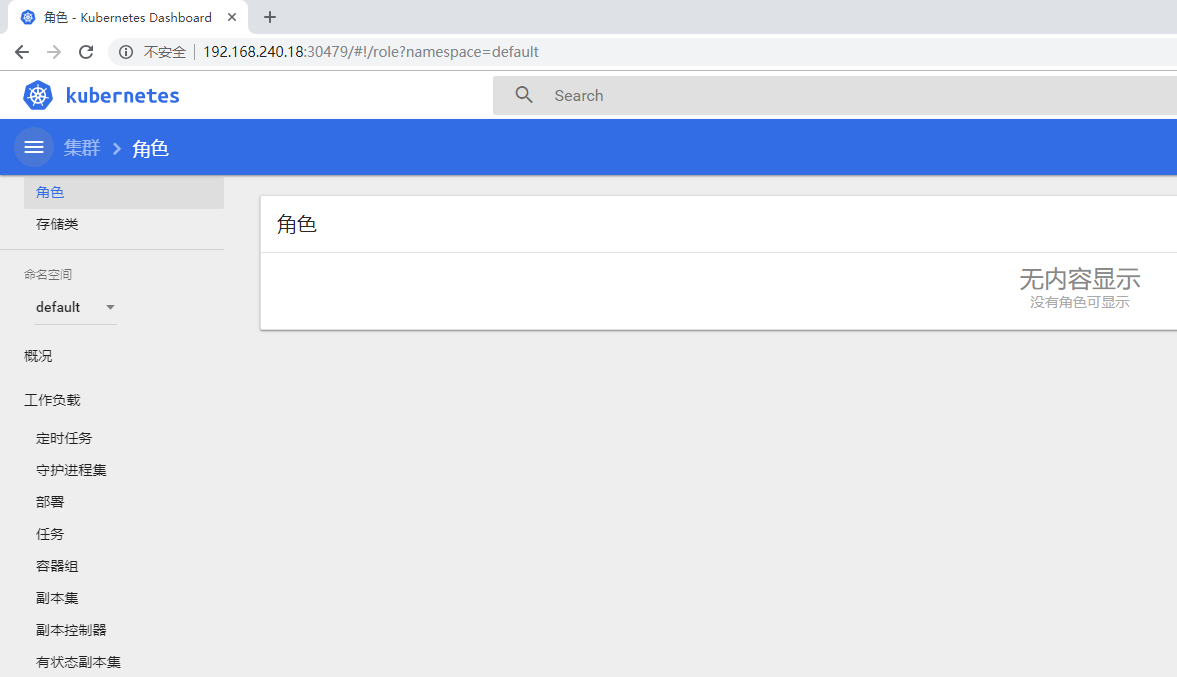
kube-dns-665845c89f-gvql4 3/3 Running 0 15m

kube-dns-665845c89f-nc2fj 3/3 Running 0 15m

kubernetes-dashboard-b7c6f7846-k5hhf 1/1 Running 0 58s

[root@k8s-master-240-16 kubernetes-dashboard]#

访问测试



# 十、Heapster+InfluxDB+Grafana监控服务搭建

[root@k8s-master-240-16 k8s]# git clone <https://github.com/kubernetes/heapster.git>

[root@k8s-master-240-16 k8s]# cd heapster/deploy/kube-config/influxdb/

[root@k8s-master-240-16 influxdb]# ls

grafana.yaml heapster.yaml influxdb.yaml

grafana.yaml中镜像改为：registry.cn-hangzhou.aliyuncs.com/k8s-kernelsky/heapster-grafana-amd64:v5.0.4

influxdb.yaml中镜像改为：registry.cn-hangzhou.aliyuncs.com/cbec-k8s/heapster-influxdb-amd64:v1.3.3

heapster.yaml中镜像改为：registry.cn-hangzhou.aliyuncs.com/k8s-images1/heapster-amd64:v1.5.4

- --source=kubernetes:https://kubernetes.default ##注释此行

- --source=kubernetes:http://192.168.240.16:8080?inClusterConfig=false&useServiceAccount=true&auth= #增加此行

拉取镜像

[root@k8s-node1-240-18 /]# docker pull registry.cn-hangzhou.aliyuncs.com/k8s-kernelsky/heapster-grafana-amd64:v5.0.4

[root@k8s-node1-240-18 /]# docker pull registry.cn-hangzhou.aliyuncs.com/cbec-k8s/heapster-influxdb-amd64:v1.3.3

[root@k8s-node2-240-20 /]# docker pull registry.cn-hangzhou.aliyuncs.com/k8s-images1/heapster-amd64:v1.5.4

部署服务：

[root@k8s-master-240-16 influxdb]# kubectl create -f influxdb.yaml

[root@k8s-master-240-16 influxdb]# kubectl create -f grafana.yaml

[root@k8s-master-240-16 influxdb]# kubectl create -f heapster.yaml

查看pod状态

[root@k8s-master-240-16 influxdb]# kubectl get pod --namespace=kube-system -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE

heapster-5cc8695598-fbrns 1/1 Running 0 73s 10.0.94.7 192.168.240.18 <none>

kube-dns-665845c89f-7wqzf 3/3 Running 0 57m 10.0.94.3 192.168.240.18 <none>

kube-dns-665845c89f-gvql4 3/3 Running 0 57m 10.0.29.2 192.168.240.20 <none>

kube-dns-665845c89f-nc2fj 3/3 Running 0 57m 10.0.29.3 192.168.240.20 <none>

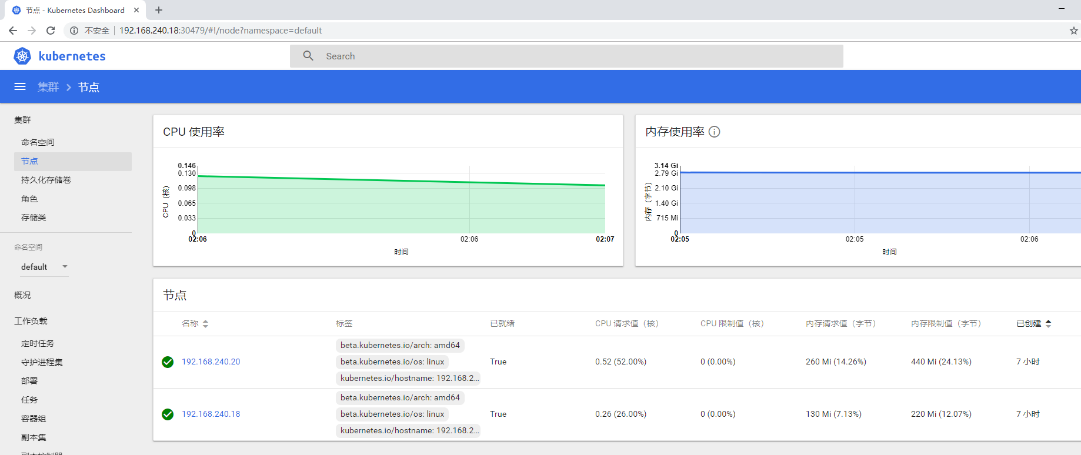
kubernetes-dashboard-b7c6f7846-k5hhf 1/1 Running 0 42m 10.0.94.4 192.168.240.18 <none>

monitoring-grafana-7c775d7595-rp69q 1/1 Running 0 8m50s 10.0.94.6 192.168.240.18 <none>

monitoring-influxdb-7c65494b98-jpqdv 1/1 Running 0 11m 10.0.94.5 192.168.240.18 <none>

[root@k8s-master-240-16 influxdb]#

查看面板监控信息



# 十一、配置ingress暴露kubernetes-dashboard

## 11.1、kubernetes服务暴露方式介绍

Kubernetes关于服务的暴露主要是通过NodePort方式，通过绑定minion主机的某个端口,然后进行pod的请求转发和负载均衡，但这种方式下缺陷是

* Service可能有很多个，如果每个都绑定一个node主机端口的话，主机需要开放外围一堆的端口进行服务调用，管理混乱
* 无法应用很多公司要求的防火墙规则

理想的方式是通过一个外部的负载均衡器，绑定固定的端口，比如80,然后根据域名或者服务名向后面的Service ip转发,Nginx很好的解决了这个需求，但问题是如果有新的服务加入，如何去修改Nginx的配置，并且加载这些配置？ Kubernetes给出的方案就是Ingress,Ingress包含了两大主件Ingress Controller和Ingress.

* Ingress解决的是新的服务加入后，域名和服务的对应问题，基本上是一个ingress的对象，通过yaml进行创建和更新进行加载。
* Ingress Controller是将Ingress这种变化生成一段Nginx的配置，然后将这个配置通过Kubernetes API写到Nginx的Pod中，然后reload.

## 11.2、ingress默认页面和控制器部署

[root@k8s-master-240-16 k8s]# mkdir -pv ingress

上传ingress默认的后端页面yaml文件，主要实现**解析不到的URL就转发到默认后端页面**



上传**Ingress Controller的yaml文件**

****

[root@k8s-master-240-16 ingress]# ll

total 8

-rw-r--r-- 1 root root 1188 Feb 15 2018 default-http-backend.yaml

-rw-r--r-- 1 root root 1557 Feb 15 2018 ingress-controller.yaml

[root@k8s-master-240-16 ingress]#

修改ingress-controller.yaml文件

[root@k8s-master-240-16 ingress]# vi ingress-controller.yaml

value: http://192.168.240.16:8080

拉取默认后端镜像

[root@k8s-node1-240-18 /]# docker pull registry.cn-hangzhou.aliyuncs.com/google\_containers/defaultbackend:1.0

[root@k8s-node1-240-18 /]# docker pull registry.cn-hangzhou.aliyuncs.com/google\_containers/nginx-ingress-controller:0.8.3

部署默认页面和ingress- controller

[root@k8s-master-240-16 ingress]# kubectl create -f default-http-backend.yaml

[root@k8s-master-240-16 ingress]# kubectl create -f ingress-controller.yaml

[root@k8s-master-240-16 ingress]# kubectl get pod --namespace=kube-system -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE

default-http-backend-657f96f6bd-svwf6 1/1 Running 0 5m14s 10.0.29.4 192.168.240.20 <none>

heapster-5cc8695598-fbrns 1/1 Running 0 23m 10.0.94.7 192.168.240.18 <none>

kube-dns-665845c89f-7wqzf 3/3 Running 0 80m 10.0.94.3 192.168.240.18 <none>

kube-dns-665845c89f-gvql4 3/3 Running 0 80m 10.0.29.2 192.168.240.20 <none>

kube-dns-665845c89f-nc2fj 3/3 Running 0 80m 10.0.29.3 192.168.240.20 <none>

kubernetes-dashboard-b7c6f7846-k5hhf 1/1 Running 0 65m 10.0.94.4 192.168.240.18 <none>

monitoring-grafana-7c775d7595-rp69q 1/1 Running 0 31m 10.0.94.6 192.168.240.18 <none>

monitoring-influxdb-7c65494b98-jpqdv 1/1 Running 0 33m 10.0.94.5 192.168.240.18 <none>

nginx-ingress-controller-dsxb9 1/1 Running 0 2m40s 10.0.29.5 192.168.240.20 <none>

nginx-ingress-controller-ll2ws 1/1 Running 0 2m40s 10.0.94.8 192.168.240.18 <none>

[root@k8s-master-240-16 ingress]#

## 11.3、kubernetes-dashboard通过ingress暴露服务

[root@k8s-master-240-16 kubernetes-dashboard]# pwd

/usr/local/k8s/kubernetes-dashboard

[root@k8s-master-240-16 kubernetes-dashboard]# cat ingress-kubernetes-dashboard.yaml

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: kubernetes-dashboard

labels:

app: kubernetes-dashboard

annotations: kubernetes-dashboard

spec:

rules:

- host: kubernetes.dashboard.com

http:

paths:

- path: /

backend:

serviceName: kubernetes-dashboard

servicePort: 80

[root@k8s-master-240-16 kubernetes-dashboard]#

[root@k8s-master-240-16 kubernetes-dashboard]# kubectl create -f ingress-kubernetes-dashboard.yaml ##部署服务

[root@k8s-master-240-16 kubernetes-dashboard]# kubectl get ingress

NAME HOSTS ADDRESS PORTS AGE

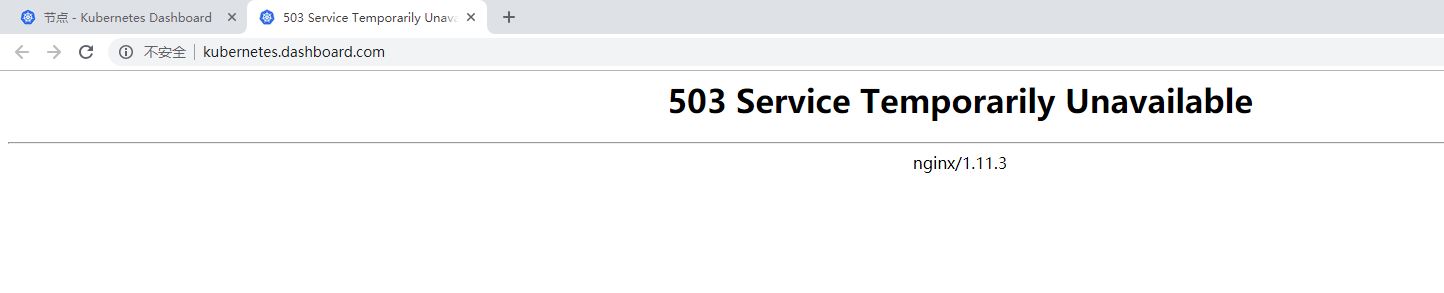
kubernetes-dashboard kubernetes.dashboard.com 80 5m26s

[root@k8s-master-240-16 kubernetes-dashboard]#

## 11.4、服务验证和排错

添加hosts记录访问测试：

192.168.240.20 kubernetes.dashboard.com



为什么访问报错，返回默认页面了。因为前面部署kubernetes-dashboard服务时namespace落在了kube-system。但是这里ingress-kubernetes-dashboard.yaml没有指定namesapce，默认到了default，解决方案是增加namespace: kube-system

先停掉服务，再增加配置，再重新部署。

[root@k8s-master-240-16 kubernetes-dashboard]# kubectl delete -f ingress-kubernetes-dashboard.yaml

[root@k8s-master-240-16 kubernetes-dashboard]# cat ingress-kubernetes-dashboard.yaml

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

namespace: kube-system

name: kubernetes-dashboard

labels:

app: kubernetes-dashboard

annotations: kubernetes-dashboard

spec:

rules:

- host: kubernetes.dashboard.com

http:

paths:

- path: /

backend:

serviceName: kubernetes-dashboard

servicePort: 80

[root@k8s-master-240-16 kubernetes-dashboard]#

[root@k8s-master-240-16 kubernetes-dashboard]# kubectl create -f ingress-kubernetes-dashboard.yaml

查看ingress

[root@k8s-master-240-16 kubernetes-dashboard]# kubectl get ingress

No resources found.

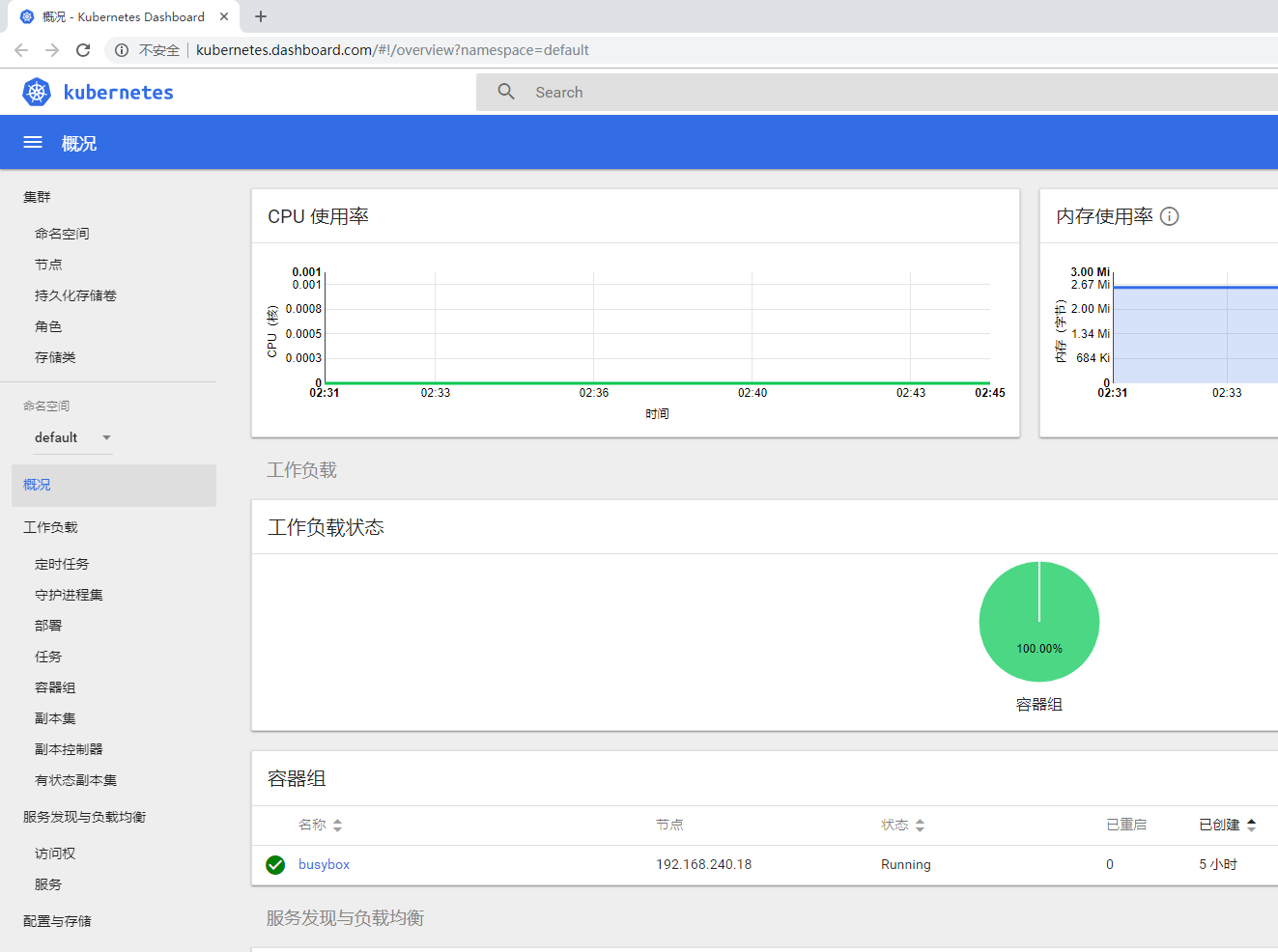
[root@k8s-master-240-16 kubernetes-dashboard]# kubectl get ingress --namespace=kube-system

NAME HOSTS ADDRESS PORTS AGE

kubernetes-dashboard kubernetes.dashboard.com 80 93s

[root@k8s-master-240-16 kubernetes-dashboard]#

验证访问



# 十二、参考文档

<https://blog.csdn.net/a735131232/article/details/83352517>

<https://www.kubernetes.org.cn/kubernetes%E8%AE%BE%E8%AE%A1%E7%90%86%E5%BF%B5>

<https://yq.aliyun.com/articles/11035>

<https://github.com/opsnull/follow-me-install-kubernetes-cluster/>

<https://www.cnblogs.com/zhaojiankai/p/7853525.html>