

**商汤科技开发有限公司**

版权所有侵权必究

**使用kubeadm部署k8s集群文档**

|  |  |
| --- | --- |
| 系统名称 | 使用kubeamd部署k8s集群文档 |
| 项目负责人 |  |
| 作者 | Rock |
| 文档提交日期 | 2018-11-08 |

**V1.2**

**版本列表**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 时间 | 备注 | 负责人 |
| 0.1 | 2018/11/08 | 初版作成 | Rock |
|  |  | 优化、修改、补充部分文档内容 |  |
|  |  | 优化、修改文档 |  |
|  |  | 优化文档 |  |
|  |  |  |  |

目录

[1 环境准备 4](#_Toc529542942)

[1.1 设备准备 4](#_Toc529542943)

[1.2 预装各插件版本情况 4](#_Toc529542944)

[2 基础设置 4](#_Toc529542945)

[2.1 关闭防火墙 4](#_Toc529542946)

[2.2 关闭selinux 5](#_Toc529542947)

[2.3 添加内核参数文件 /etc/sysctl.d/k8s.conf 5](#_Toc529542948)

[2.4 关闭swap 5](#_Toc529542949)

[2.5 添加主机名与IP对应关系 5](#_Toc529542950)

[2.6 同步时间 5](#_Toc529542951)

[3 安装docker 6](#_Toc529542952)

[3.1 查看所有docker版本 6](#_Toc529542953)

[3.2 安装docker 指定版本 6](#_Toc529542954)

[3.3 验证版本 6](#_Toc529542955)

[3.4 卸载其他版本docker 6](#_Toc529542956)

[3.4.1 查看包 6](#_Toc529542957)

[3.4.2 卸载docker服务 7](#_Toc529542958)

[4 安装kubeadm，kubelet和kubectl 7](#_Toc529542959)

[4.1 添加阿里云YUM软件源 7](#_Toc529542960)

[4.2 安装kubeadm，kubelet和kubectl 7](#_Toc529542961)

[5 使用kubeadm创建单个Master集群 8](#_Toc529542962)

[5.1 插件镜像拉取 8](#_Toc529542963)

[5.2 初始化Master 10](#_Toc529542964)

[5.3 安装flannel网络插件 11](#_Toc529542965)

[5.4 安装Pod Network 12](#_Toc529542966)

[5.4.1 接下来安装flannel network add-on 13](#_Toc529542967)

[5.5 master参与工作负载 15](#_Toc529542968)

[5.6 测试DNS 16](#_Toc529542969)

[5.7 向Kubernetes集群中添加Node节点 16](#_Toc529542970)

[6 kubernetes dashboard 18](#_Toc529542971)

主要文档来源

http://www.cnblogs.com/benjamin77/p/9783797.html

3 master安装文档：

https://blog.csdn.net/zhangwenjiang001/article/details/85049281 #这个很好

https://kubernetes.io/docs/setup/independent/high-availability/ #这是官方文档

# 环境准备

## 设备准备

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 物理主机 | 系统 | 内网IP | 节点名称 | 角色 |
| CentOS7.4 | 10.5.6.19 | k8s-master | Master |

## 预装各插件版本情况

|  |  |
| --- | --- |
| 服务/插件名称 | 版本号 |
| docker | 18.06.1-ce |
| k8s.gcr.io/kube-proxy | v1.12.2 |
| k8s.gcr.io/kube-apiserver | v1.12.2 |
| k8s.gcr.io/kube-controller-manager | v1.12.2 |
| k8s.gcr.io/kube-scheduler | v1.12.2 |
| k8s.gcr.io/etcd | 3.2.24 |
| k8s.gcr.io/coredns | 1.2.2 |
| quay.io/coreos/flannel | v0.10.0-amd64 |
| k8s.gcr.io/pause | 3.1 |
| Helm | 2.11.0 |
| quay.io/kubernetes-ingress-controller/nginx-ingress-controller | 0.20.0 |
| k8s.gcr.io/defaultbackend | 1.4 |
| registry.cn-hangzhou.aliyuncs.com/google\_containers/kubernetes-dashboard-amd64 | v1.10.0 |

# 基础设置

## 关闭防火墙

# systemctl stop firewalld

# systemctl disable firewalld

## 关闭selinux

# sed -i 's/enforcing/disabled/' /etc/selinux/config

# setenforce 0

# getenforce

## 添加内核参数文件 /etc/sysctl.d/k8s.conf

# vim /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

vm.swappiness = 0

* 执行命令使修改生效

# modprobe br\_netfilter

# sysctl -p /etc/sysctl.d/k8s.conf

## 关闭swap

# swapoff -a # 临时关闭

# vim /etc/fstab #注释掉swap那一行， 永久关闭

## 添加主机名与IP对应关系

# cat /etc/hosts

10.5.6.19 k8s-master

# cat /etc/hostname

k8s-master

# hostname k8s-master

# logout

## 同步时间

# yum install ntpdate -y

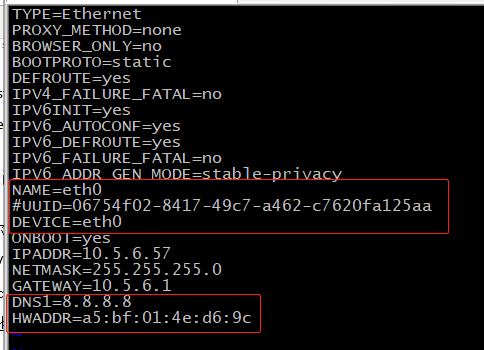
# ntpdate ntp.api.bz

## 修改网卡名称为eth0

修改名称

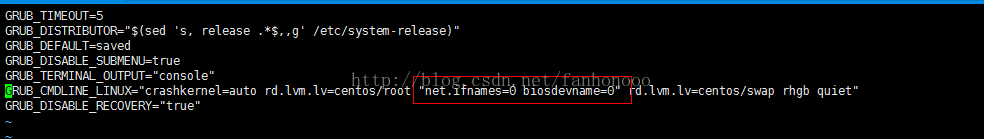
# cd /etc/sysconfig/network-scripts/

# vim /ifcfg-eno16777736  #网卡名称改为eth0，并注释UUID，添加HWADDR网卡mac地址



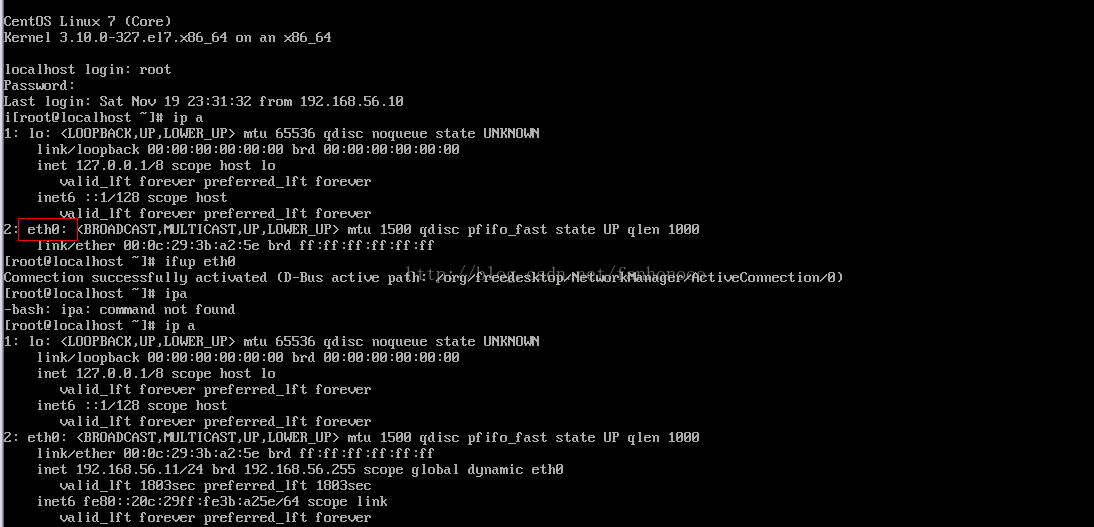
# mv ifcfg-eno16777736 ifcfg-eth0 #修改文件名

# vim /etc/default/grub #禁用该可预测命名规则



# grub2-mkconfig -o /boot/grub2/grub.cfg  #重新生成GRUB配置并更新内核参数

# init 6 #重启生效



# 安装docker

**所有节点操作**

Docker安装来源：

https://blog.csdn.net/nie312122330/article/details/81543847

#此处按公司要求指定安装 : 18.06.1版本docker

# yum install -y yum-utils device-mapper-persistent-data lvm2

# wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo -O /etc/yum.repos.d/docker.repo

## 查看所有docker版本

# yum list docker-ce.x86\_64 --showduplicates | sort -r

## 安装docker 指定版本

# yum makecache fast && yum -y install docker-ce-18.06.1.ce-3.el7

# systemctl start docker.service && systemctl enable docker.service

# systemctl status docker.service

## 验证版本

# docker version

# docker -v

Docker version 18.06.1-ce, build e68fc7a

## 卸载其他版本docker

### 查看包

# rpm -qa |grep docker

docker-client-1.13.1-75.git8633870.el7.centos.x86\_64

docker-1.13.1-75.git8633870.el7.centos.x86\_64

docker-common-1.13.1-75.git8633870.el7.centos.x86\_64

### 卸载docker服务

# yum -y remove docker-client-1.13.1-75.git8633870.el7.centos.x86\_64

# 安装kubeadm，kubelet和kubectl

kubeadm： 引导集群的命令

kubelet：集群中运行任务的代理程序

kubectl：命令行管理工具

## 添加阿里云YUM软件源

**所有节点操作**

# cat << EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64/

gpgcheck=0

gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

enabled=1

EOF

## 安装kubeadm，kubelet和kubectl

**所有节点操作**

# yum install -y kubelet kubeadm kubectl

*如果指定安装1.12.2版本的kubelet kubeadm kubectl 方法：*

*已经安装过1.12.2的服务器上执行：*

*# rpm -qa |grep -E 'kubelet|kubeadm|kubectl'*

*kubeadm-1.12.2-0.x86\_64*

*kubectl-1.12.2-0.x86\_64*

*kubelet-1.12.2-0.x86\_64*

*# yum install kubectl-1.12.2-0.x86\_64 kubelet-1.12.2-0.x86\_64 kubeadm-1.12.2-0.x86\_64*

Installing:

kubeadm x86\_64 1.12.2-0 kubernetes 7.2 M

kubectl x86\_64 1.12.2-0 kubernetes 7.7 M

kubelet x86\_64 1.12.2-0 kubernetes 19 M

Installing for dependencies:

cri-tools x86\_64 1.12.0-0 kubernetes 4.2 M

kubernetes-cni x86\_64 0.6.0-0 kubernetes 8.6 M

socat x86\_64 1.7.3.2-2.el7 base 290 k

# systemctl enable kubelet && systemctl start kubelet

# echo "KUBELET\_EXTRA\_ARGS=--fail-swap-on=false" > /etc/sysconfig/kubelet #忽略报错

# systemctl daemon-reload && systemctl start kubelet

# systemctl status kubelet **#正常这里都会起不来，只有等Kubeadm init初始化成功了，才可以正常**

# systemctl stop kubelet

* 注意：使用Docker时，kubeadm会自动检查kubelet的cgroup驱动程序，并/var/lib/kubelet/kubeadm-flags.env在运行时将其设置在文件中。如果使用的其他CRI，则必须在/etc/default/kubelet中cgroup-driver值修改为cgroupfs：

# mkdir -p /var/lib/kubelet/

# vim /var/lib/kubelet/kubeadm-flags.env

KUBELET\_KUBEADM\_ARGS=--cgroup-driver=cgroupfs --cni-bin-dir=/opt/cni/bin --cni-confdir=/etc/cni/net.d --network-plugin=cni

# systemctl daemon-reload && systemctl restart kubelet

# systemctl status kubelet

# 使用kubeadm创建单个Master集群

## 插件镜像拉取

**所有节点操作**

*注意：如果版本为1.13.0，则这里把1.12.2改为1.13.0，coredns改为 1.2.6。*

*请全部用* **:%s///g***方式修改，否则可能出现漏改问题*

# cat k8s.sh

docker pull mirrorgooglecontainers/kube-apiserver:v1.12.2

docker pull mirrorgooglecontainers/kube-controller-manager:v1.12.2

docker pull mirrorgooglecontainers/kube-scheduler:v1.12.2

docker pull mirrorgooglecontainers/kube-proxy:v1.12.2

docker pull mirrorgooglecontainers/pause:3.1

docker pull mirrorgooglecontainers/etcd:3.2.24

docker pull coredns/coredns:1.2.2

docker tag mirrorgooglecontainers/kube-proxy:v1.12.2 k8s.gcr.io/kube-proxy:v1.12.2

docker tag mirrorgooglecontainers/kube-scheduler:v1.12.2 k8s.gcr.io/kube-scheduler:v1.12.2

docker tag mirrorgooglecontainers/kube-apiserver:v1.12.2 k8s.gcr.io/kube-apiserver:v1.12.2

docker tag mirrorgooglecontainers/kube-controller-manager:v1.12.2 k8s.gcr.io/kube-controller-manager:v1.12.2

docker tag mirrorgooglecontainers/etcd: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

docker rmi mirrorgooglecontainers/kube-apiserver:v1.12.2

docker rmi mirrorgooglecontainers/kube-controller-manager:v1.12.2

docker rmi mirrorgooglecontainers/kube-scheduler:v1.12.2

docker rmi mirrorgooglecontainers/kube-proxy:v1.12.2

docker rmi mirrorgooglecontainers/pause:3.1

docker rmi mirrorgooglecontainers/etcd:3.2.24

docker rmi coredns/coredns:1.2.2

# /bin/bash k8s.sh

# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

k8s.gcr.io/kube-proxy v1.12.2 15e9da1ca195 2 weeks ago 96.5MB

k8s.gcr.io/kube-apiserver v1.12.2 51a9c329b7c5 2 weeks ago 194MB

k8s.gcr.io/kube-controller-manager v1.12.2 15548c720a70 2 weeks ago 164MB

k8s.gcr.io/kube-scheduler v1.12.2 d6d57c76136c 2 weeks ago 58.3MB

k8s.gcr.io/etcd 3.2.24 3cab8e1b9802 6 weeks ago 220MB

k8s.gcr.io/coredns 1.2.2 367cdc8433a4 2 months ago 39.2MB

k8s.gcr.io/pause 3.1 da86e6ba6ca1 10 months ago 742kB

# kubeadm version

kubeadm version: &version.Info{Major:"1", Minor:"12", GitVersion:"v1.12.2", GitCommit:"17c77c7898218073f14c8d573582e8d2313dc740", GitTreeState:"clean", BuildDate:"2018-10-24T06:51:33Z", GoVersion:"go1.10.4", Compiler:"gc", Platform:"linux/amd64"}

## 初始化Master

如果不先docker pull一下image的话，会报错的，所以先做了5.1中的操作。

*注意：cpu数量>=2，版本需要从1.12.2改到1.13.0，coredns改为1.2.6。*

# kubeadm init --kubernetes-version=1.12.2 --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=10.5.6.19

...

Your Kubernetes master 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/

You can now join any number of machines by running the following on each node

as root:

kubeadm join 192.168.2.151:6443 --token xpim7a.3v0eualufhlt7v99 --discovery-token-ca-cert-hash sha256:9689bb23e5804d0679430de5cc2acc12aaebefe2b0676e68717d7b3c51ea69c9

kubeadm join --token <token> <master-ip>:<master-port> --discovery-token-ca-cert-hashsha256:<hash>

家里的：

kubeadm join 192.168.2.151:6443 --token xpim7a.3v0eualufhlt7v99 --discovery-token-ca-cert-hash sha256:9689bb23e5804d0679430de5cc2acc12aaebefe2b0676e68717d7b3c51ea69c9

* 注意：

如果报错：[ERROR Port-10250]: Port 10250 is in use

可以加：--ignore-preflight-errors=all

# mkdir -p $HOME/.kube

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

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

## 安装flannel网络插件

**所有节点操作**

# mkdir -p ~/k8s/ && cd ~/k8s

# wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

* 这里改成自己的网卡

# vim kube-flannel.yml

args:

- --ip-masq

- --kube-subnet-mgr

- --iface=eth0

仅仅修改此处即可：

containers:

- name: kube-flannel

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

command:

- /opt/bin/flanneld

args:

- --ip-masq

- --kube-subnet-mgr

**- --iface=eth0**

# kubectl apply -f kube-flannel.yml

* 修改镜像地址

registry.cn-shenzhen.aliyuncs.com/cp\_m/flannel:v0.10.0-amd64

这里也可以用docker tag打一下标签：

# docker pull registry.cn-shenzhen.aliyuncs.com/cp\_m/flannel:v0.10.0-amd64

# docker tag registry.cn-shenzhen.aliyuncs.com/cp\_m/flannel:v0.10.0-amd64 k8s.gcr.io/ flannel:v0.10.0-amd64:v0.10.0-amd64

那么镜像里也要换一下

但是可能下载不来，需要更换yum源：

# mv /etc/yum.repos.d/kubernetes.repo /etc/yum.repos.d/kubernetes.repo-bak

# 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

此处必须解决，否则 master节点一直显示notReady 。

* 查看结果

# kubectl get ds -l app=flannel -n kube-system

NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE

kube-flannel-ds-amd64 1 1 1 1 1 beta.kubernetes.io/arch=amd64 67m

kube-flannel-ds-arm 0 0 0 0 0 beta.kubernetes.io/arch=arm 67m

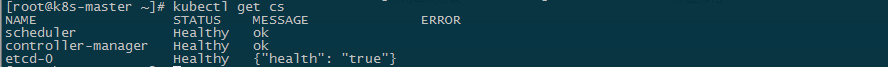
kube-flannel-ds-arm64 0 0 0 0 0 beta.kubernetes.io/arch=arm64 67m

kube-flannel-ds-ppc64le 0 0 0 0 0 beta.kubernetes.io/arch=ppc64le 67m

kube-flannel-ds-s390x 0 0 0 0 0 beta.kubernetes.io/arch=s390x 67m

*注意：*kube-flannel-ds-amd64 值为1才对。

* 查看集群状态



* 集群初始化如果遇到问题，可以使用下面的命令进行清理：

# kubeadm reset

ifconfig cni0 down

ip link delete cni0

ifconfig flannel.1 down

ip link delete flannel.1

# rm -rf /var/lib/cni/

**ROCK遇到问题：**

发现使用上面的kube-flannel.yml后，过一段时间后flannel容器会挂掉，最后**史顺义**帮解决，然后我对比了一下配置文件，发现他把其他部分全删掉了。本来有471行，删除后只有159了。下面是我拿过来，确认史顺义改好后可用的kube-flannel.yml文件：

---

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": "10.244.0.0/16",

"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

- --iface=eth0

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

## 安装Pod Network

#这时master状态为notready 是因为没有网络插件

[root@k8s-master k8s]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-master NotReady master 27m v1.12.2

### 接下来安装flannel network add-on

* 获取组件健康状态

# kubectl get cs

NAME STATUS MESSAGE ERROR

scheduler Healthy ok

controller-manager Healthy ok

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

# kubectl describe node k8s-master

Name: k8s-master

Roles: master

Labels: beta.kubernetes.io/arch=amd64

beta.kubernetes.io/os=linux

kubernetes.io/hostname=k8s-master

node-role.kubernetes.io/master=

Annotations: kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock

node.alpha.kubernetes.io/ttl: 0

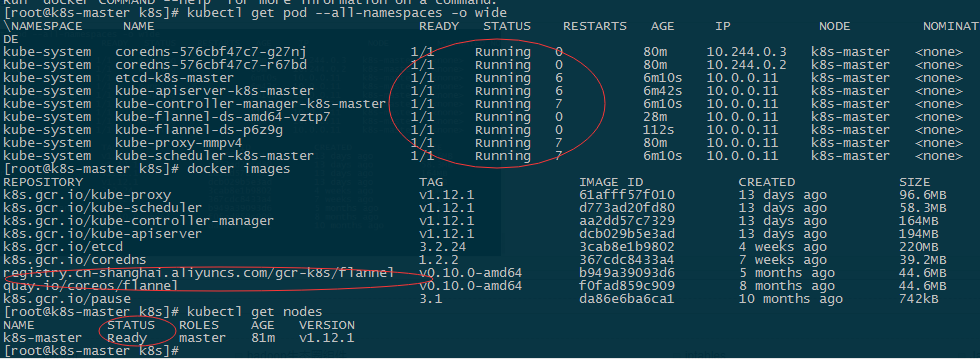
volumes.kubernetes.io/controller-managed-attach-detach: true

CreationTimestamp: Wed, 17 Oct 2018 21:24:01 +0800

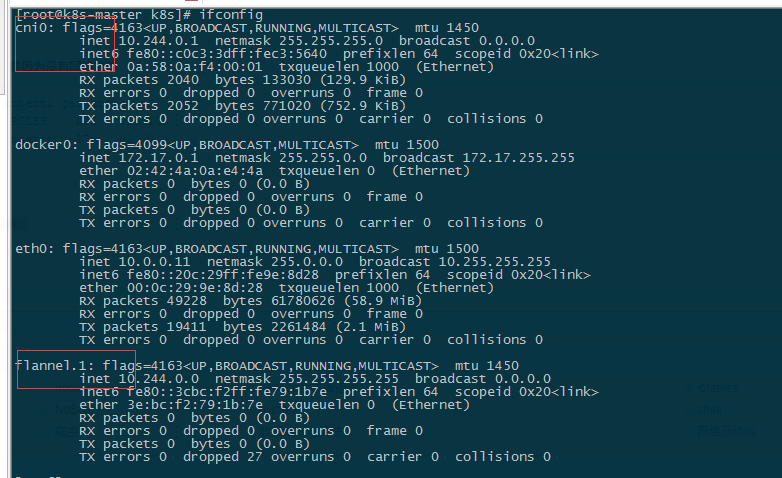
Taints: node-role.kubernetes.io/master:NoSchedule

node.kubernetes.io/not-ready:NoSchedule

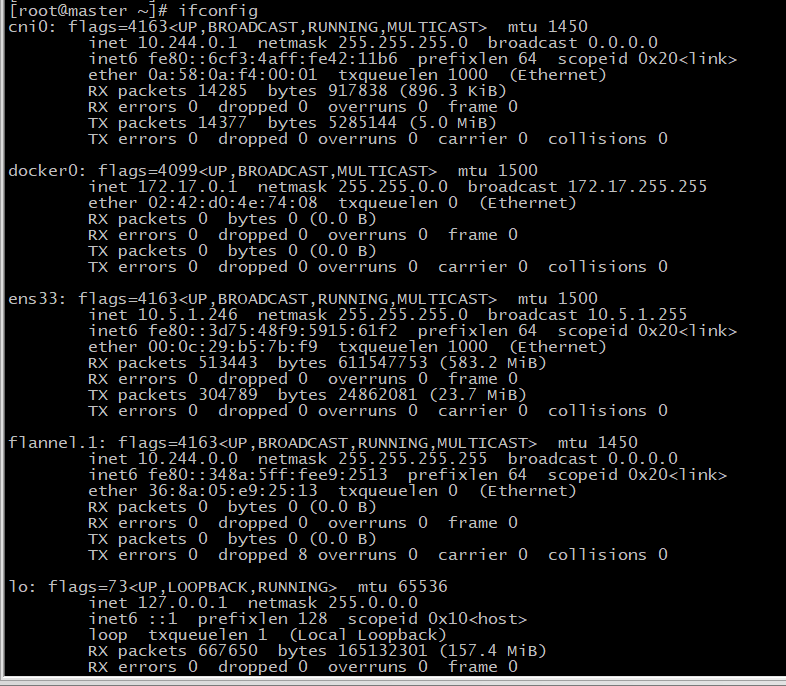
Unschedulable: false



上图输出显示首先会下载一个flannel镜像，namespace全部为running状态， master为ready。



下图为ROCK的图：



## master参与工作负载

出于安全考虑Pod不会被调度到Master Node上，也就是说Master Node不参与工作负载。这是因为当前的master节点node1被打上了node-role.kubernetes.io/master:NoSchedule的污点：

* 查看污点情况

# kubectl describe node k8s-master | grep Taint

Taints: node-role.kubernetes.io/k8s-master:NoSchedule

* 去除污点使k8s-master参与负载

# kubectl taint nodes k8s-master node-role.kubernetes.io/master-

node/k8s-master untainted

* 查看

# kubectl describe node k8s-master | grep Taint

Taints: <none>

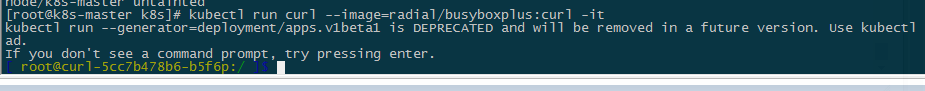
* 如果需要改回来不想让master节点参与到工作负载

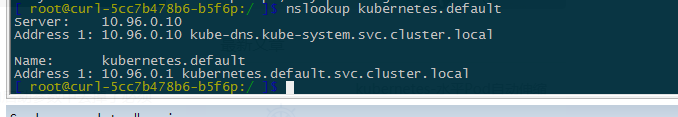
# kubectl taint node k8s-master node-role.kubernetes.io/master="":NoSchedule

## 测试DNS

# kubectl run curl --image=radial/busyboxplus:curl -it

注意：这里要用这个busyboxplus镜像，普通的busybox是无法解析到Name和Address的。



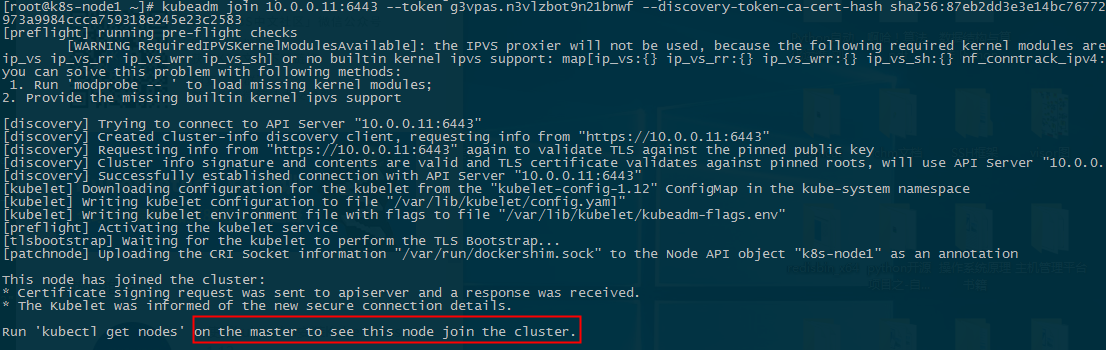


## 向Kubernetes集群中添加Node节点

***注意：节点先做这一步，再安装flannel，不然flannel安装不上****。*

下面我们将node1 node2这个主机添加到Kubernetes集群中， 在node1和node2上执行:

[root@k8s-node1 ~]# kubeadm join 10.5.6.19:6443 --token i4us8x.pw2f3botcnipng8e --discovery-token-ca-cert-hash sha256:d16ac747c2312ae829aa29a3596f733f920ca3d372d9f1b34d33c938be067e51



* 拷贝master文件到本地

# scp root@masterIP:/etc/kubernetes/admin.conf /etc/kubernetes/

# export KUBECONFIG=/etc/kubernetes/admin.conf"

# echo "export KUBECONFIG=/etc/kubernetes/admin.conf" >> ~/.bash\_profile

* 查看节点

https://img2018.cnblogs.com/blog/1215197/201810/1215197-20181028203625831-229481386.png

原因是节点k8-node1也要获取镜像，执行以上的获取镜像的脚本即可，两个节点分别重置集群，kubeadm reset，然后重新初始化。

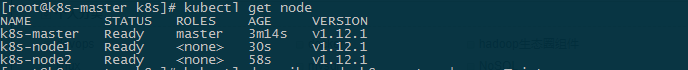
https://img2018.cnblogs.com/blog/1215197/201810/1215197-20181029083527539-306943815.png

* 从master节点如果需要移出这个node1节点

在master节点上执行：

# kubectl drain k8s-node1 --delete-local-data --force --ignore-daemonsets

# kubectl delete node k8s-node1



# 安装helm

Helm由客户端命helm令行工具和服务端tiller组成，Helm的安装十分简单。 下载helm命令行工具到master节点node1的/usr/local/bin下，这里下载的2.9.1版本：

## 下载helm

# wget https://storage.googleapis.com/kubernetes-helm/helm-v2.11.0-linux-amd64.tar.gz

如果无法下载，可以放网页上下载。

# tar -zxvf helm-v2.11.0-linux-amd64.tar.gz

# cd linux-amd64/

# cp helm /usr/local/bin/

## 创建tiller yaml文件及服务

为了安装服务端tiller，还需要在这台机器上配置好kubectl工具和kubeconfig文件，确保kubectl工具可以在这台机器上访问apiserver且正常使用。 这里的node1节点以及配置好了kubectl。

因为Kubernetes APIServer开启了RBAC访问控制，所以需要创建tiller使用的service account: tiller并分配合适的角色给它。 详细内容可以查看helm文档中的Role-based Access Control。 这里简单起见直接分配cluster-admin这个集群内置的ClusterRole给它。创建rbac-config.yaml文件：

# vim rbac-config.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: tiller

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: tiller

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: tiller

namespace: kube-system

# kubectl create -f rbac-config.yaml

serviceaccount/tiller created

clusterrolebinding.rbac.authorization.k8s.io/tiller created

# kubectl get ServiceAccount -n kube-system -o wide #查看

## 接下来使用helm部署tiller

# helm init --service-account tiller --skip-refresh

Creating /root/.helm

Creating /root/.helm/repository

Creating /root/.helm/repository/cache

Creating /root/.helm/repository/local

Creating /root/.helm/plugins

Creating /root/.helm/starters

Creating /root/.helm/cache/archive

Creating /root/.helm/repository/repositories.yaml

Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com

Adding local repo with URL: http://127.0.0.1:8879/charts

$HELM\_HOME has been configured at /root/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users' policy.

To prevent this, run `helm init` with the --tiller-tls-verify flag.

For more information on securing your installation see: https://docs.helm.sh/using\_helm/#securing-your-helm-installation

Happy Helming!

## 验证

tiller默认被部署在k8s集群中的kube-system这个namespace下：

# kubectl get pod -n kube-system -l app=helm

NAME READY STATUS RESTARTS AGE

tiller-deploy-6f6fd74b68-kk2z9 1/1 Running 0 3m17s

# helm version

Client: &version.Version{SemVer:"v2.11.0", GitCommit:"2e55dbe1fdb5fdb96b75ff144a339489417b146b", GitTreeState:"clean"}

Server: &version.Version{SemVer:"v2.11.0", GitCommit:"2e55dbe1fdb5fdb96b75ff144a339489417b146b", GitTreeState:"clean"}

# 使用Helm部署Nginx Ingress

为了便于将集群中的服务暴露到集群外部，从集群外部访问，接下来使用Helm将Nginx Ingress部署到Kubernetes上。 Nginx Ingress Controller被部署在Kubernetes的边缘节点上，关于Kubernetes边缘节点的高可用相关的内容可以查看我前面整理的Bare metal环境下Kubernetes Ingress边缘节点的高可用。 这里简单起见，只有一个edge节点。

## 将k8s-master(10.5.6.19)同时做为边缘节点，打上Label

# kubectl label node k8s-master node-role.kubernetes.io/edge=

node/node1 labeled

# kubectl get node

NAME STATUS ROLES AGE VERSION

k8s-master Ready edge,master 147m v1.12.2

## stable/nginx-ingress chart的值文件ingress-nginx.yaml

# vim ingress-nginx.yaml

controller:

service:

externalIPs:

- **10.5.1.19**

nodeSelector:

node-role.kubernetes.io/edge: ''

tolerations:

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

operator: Exists

effect: NoSchedule

defaultBackend:

nodeSelector:

node-role.kubernetes.io/edge: ''

tolerations:

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

operator: Exists

effect: NoSchedule

# helm repo update

Hang tight while we grab the latest from your chart repositories...

...Skip local chart repository

...Successfully got an update from the "stable" chart repository

Update Complete. ⎈ Happy Helming!⎈

# helm install stable/nginx-ingress \

-n nginx-ingress \

--namespace ingress-nginx \

-f ingress-nginx.yaml

# 此处需要等待几分钟，因为镜像还需要拉取和安装，因为镜像有513MB.

# docker images | grep ingress

quay.io/kubernetes-ingress-controller/nginx-ingress-controller 0.20.0 513MB

# kubectl get pod -n ingress-nginx -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE

nginx-ingress-controller-8658f85fd-5jnxt 1/1 Running 0 12m 10.244.0.10 k8s-master <none>

nginx-ingress-default-backend-684f76869d-dbhcz 1/1 Running 0 12m 10.244.0.9 k8s-master <none>

* 访问

如果访问http://10.5.6.19返回default backend，则部署完成:

# curl http://10.5.6.19/

default backend – 404

## 将TLS证书配置到Kubernetes中 – 此步应该后面会删掉

当使用Ingress将HTTPS的服务暴露到集群外部时，需要HTTPS证书，这里将\*.frognew.com的证书和秘钥配置到Kubernetes中。

后边部署在kube-system命名空间中的dashboard要使用这个证书，因此这里先在kube-system中创建证书的secret

# kubectl create secret tls frognew-com-tls-secret --cert=fullchain.pem --key=privkey.pem -n kube-system

secret/frognew-com-tls-secret created

**ROCK这里测试有问题，提示没有对应的目录。后来没做这一步的，下面dashboard用的火狐浏览器就行，其他浏览器都不行，原因可能就是缺少这一步的TLS证书的配置问题。**

# kubernetes dashboard

## 配置dashboard文件及生成—源文档配置，有问题，这里不要用了

# wget https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml

* 修改镜像地址：

打开刚下载的 kubernetes-dashboard.yaml 文件，修改其中的v1.10.0对应的镜像拉取地址：

# registry.cn-hangzhou.aliyuncs.com/google\_containers/kubernetes-dashboard-amd64:v1.10.0

* 修改Service：（150-162行，Dashboard Service）

kind: Service

apiVersion: v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

type: NodePort

ports:

- port: 443

targetPort: 8443

nodePort: 30000

selector:

k8s-app: kubernetes-dashboard

# docker pull registry.cn-hangzhou.aliyuncs.com/google\_containers/kubernetes-dashboard-amd64:v1.10.0

# kubectl apply -f kubernetes-dashboard.yaml

创建一个管理员角色：

# vim k8s-admin.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: dashboard-admin

namespace: kube-system

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1beta1

metadata:

name: dashboard-admin

subjects:

- kind: ServiceAccount

name: dashboard-admin

namespace: kube-system

roleRef:

kind: ClusterRole

name: cluster-admin

apiGroup: rbac.authorization.k8s.io

# kubectl apply -f k8s-admin.yaml

## ROCK找的方法

注意：上面8.1中最后还是有问题，最后rock重新找了个配置方法，这里测试成功了。

### 获取yaml文件及修改yaml文件

获取 kubernetes-dashboard.yaml 文件，打开下面github中的地址，进行复制：

https://github.com/yuezhongxin/Kubernetes.Sample/blob/master/kubernetes-dashboard.yaml

这里我把 kubernetes-dashboard.yaml 文件贴出来了：

# vim kubernetes-dashboard.yaml

# 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.

# Configuration to deploy release version of the Dashboard UI compatible with

# Kubernetes 1.8.

#

# Example usage: kubectl create -f <this\_file>

# ------------------- 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:v1.8.2 #此处我用的v1.10.0版本，所以改了一下Image的地址。

image: registry.cn-hangzhou.aliyuncs.com/google\_containers/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:

type: NodePort

ports:

- port: 443

targetPort: 8443

nodePort: 30000

selector:

k8s-app: kubernetes-dashboard

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: kubernetes-dashboard

subjects:

- kind: ServiceAccount

name: kubernetes-dashboard

namespace: kube-system

roleRef:

kind: ClusterRole

name: cluster-admin

apiGroup: rbac.authorization.k8s.io

### 拉取镜像及执行yaml文件

# docker pull registry.cn-hangzhou.aliyuncs.com/google\_containers/kubernetes-dashboard-amd64:v1.10.0

# kubectl apply -f kubernetes-dashboard.yaml

## 找到token，用于登录Kubernetes Dashboard

* 方法一： 正常dashboard secret token

# kubectl get secret -n kube-system | grep -i kubernetes-dashboard-token

# kubectl describe secret kubernetes-dashboard-token-gwnj7 -n kube-system

Name: kubernetes-dashboard-token-gwnj7

Namespace: kube-system

Labels: <none>

Annotations: kubernetes.io/service-account.name: kubernetes-dashboard

kubernetes.io/service-account.uid: 2e675bc9-e642-11e8-aed3-000c29b57bf9

Type: kubernetes.io/service-account-token

Data

====

ca.crt: 1025 bytes

namespace: 11 bytes

token: eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9..CiteRRskocxiVKmlVR2jqFh0\_4mpsV2YnMcsG98aOaSeE\_vhzrobLK0cKGOS55KjTWyztzC66ZC7ZVLLNTNRd4v0fdCJx1iKxxoqhViXYg7yWcvF\_2kAp4KBW8HherLhoIaa8-r-X7zI9Ekc2vZwaCw\_36PMY6utwa2jCX4boXvFYmU31qtrfOL718bYRBBDTnWfUZemdeI7-ZqlGTv5fEDK9a99FcJgigJEViyChblXB1A5I6jb6nfT7ccW-bL5vQn91MES3BwCctaqhp5xxzrHYg\_M9PgNJwTrVHyuT4xHeNzMWmvIi-oqhh9ujqwCSxNXIEodmLlP2Bw4mblBCg

* **方法二：** admin secret token

如果上面的token 有问题（无法登陆或者没有权限），可以用admin生成的token 去登陆

# kubectl get secrets -n kube-system | grep admin

admin-token-bp54r kubernetes.io/service-account-token 3 28d

## URL访问验证dashboard

1. 方法一：

通过映射端口30001进行访问

* 使用**火狐浏览器**访问： https://10.5.1.246:30001

1. 方法二(待验证)：

# kubectl proxy --address='localhost' --port=8086 --accept-hosts='^\*$'

Starting to serve on 127.0.0.1:8086

在master上：# curl http://127.0.0.1:8086

{

"paths": [

"/api",

"/api/v1",

"/apis",

"/apis/",

"/apis/admissionregistration.k8s.io",

"/apis/admissionregistration.k8s.io/v1beta1",

"/apis/apiextensions.k8s.io",

"/apis/apiextensions.k8s.io/v1beta1",

"/apis/apiregistration.k8s.io",

"/apis/apiregistration.k8s.io/v1",

"/apis/apiregistration.k8s.io/v1beta1",

"/apis/apps",

"/apis/apps/v1",

"/apis/apps/v1beta1",

"/apis/apps/v1beta2",

"/apis/authentication.k8s.io",

"/apis/authentication.k8s.io/v1",

"/apis/authentication.k8s.io/v1beta1",

"/apis/authorization.k8s.io",

"/apis/authorization.k8s.io/v1",

"/apis/authorization.k8s.io/v1beta1",

"/apis/autoscaling",

"/apis/autoscaling/v1",

"/apis/autoscaling/v2beta1",

"/apis/autoscaling/v2beta2",

"/apis/batch",

"/apis/batch/v1",

"/apis/batch/v1beta1",

"/apis/certificates.k8s.io",

"/apis/certificates.k8s.io/v1beta1",

"/apis/coordination.k8s.io",

"/apis/coordination.k8s.io/v1beta1",

"/apis/events.k8s.io",

"/apis/events.k8s.io/v1beta1",

"/apis/extensions",

"/apis/extensions/v1beta1",

"/apis/networking.k8s.io",

"/apis/networking.k8s.io/v1",

"/apis/policy",

"/apis/policy/v1beta1",

"/apis/rbac.authorization.k8s.io",

"/apis/rbac.authorization.k8s.io/v1",

"/apis/rbac.authorization.k8s.io/v1beta1",

"/apis/scheduling.k8s.io",

"/apis/scheduling.k8s.io/v1beta1",

"/apis/storage.k8s.io",

"/apis/storage.k8s.io/v1",

"/apis/storage.k8s.io/v1beta1",

"/healthz",

"/healthz/autoregister-completion",

"/healthz/etcd",

"/healthz/log",

"/healthz/ping",

"/healthz/poststarthook/apiservice-openapi-controller",

"/healthz/poststarthook/apiservice-registration-controller",

"/healthz/poststarthook/apiservice-status-available-controller",

"/healthz/poststarthook/bootstrap-controller",

"/healthz/poststarthook/ca-registration",

"/healthz/poststarthook/generic-apiserver-start-informers",

"/healthz/poststarthook/kube-apiserver-autoregistration",

"/healthz/poststarthook/rbac/bootstrap-roles",

"/healthz/poststarthook/scheduling/bootstrap-system-priority-classes",

"/healthz/poststarthook/start-apiextensions-controllers",

"/healthz/poststarthook/start-apiextensions-informers",

"/healthz/poststarthook/start-kube-aggregator-informers",

"/healthz/poststarthook/start-kube-apiserver-admission-initializer",

"/healthz/poststarthook/start-kube-apiserver-informers",

"/logs",

"/metrics",

"/openapi/v2",

"/swagger-2.0.0.json",

"/swagger-2.0.0.pb-v1",

"/swagger-2.0.0.pb-v1.gz",

"/swagger.json",

"/swaggerapi",

"/version"

]

命令总结：

# kubectl get pod -n kube-system -l app=helm #看单一的服务情况

# kubectl get pods -n kube-system -l app=helm --show-labels #查看label

# tab命令补全

# yum install -y bash-completion

# source /usr/share/bash-completion/bash\_completion

# source <(kubectl completion bash)

# echo "source <(kubectl completion bash)" >> ~/.bashrc