k8s 部署和使用

Name ： 曲中岭
Email:zlingqu@126.com
Q Q :441869115

第一章 部署准备

1.1 目的

使用 k8s 搭建 Docker 集群，实现相关功能，比如自动扩容、缩容、滚动更新等。

1.2 规划

OS ： CentOS_7.5 x64
Host1 ： 172.16.6.37(docker07),master 节点
Host2 ： 172.16.6.38(docker08),node 节点
Host3 ： 172.16.6.39(docker09),node 节点
Host4 ： 172.16.6.40(docker10),node 节点
Docker-ce ： 18.09.0

序号	类目	master 节点	node 节点	版本	安装方式
1	IP	172.16.6.37	172.16.6.38/39/40		
2	主机名	docker07	docker08/09/10		
3	docker	√	√	18.09.0	系统服务
4	kubeadm	√	√	v1.13.3	rpm
5	kubectl	√	√	v1.13.3	rpm
6	kubelet	√	√	v1.13.3	rpm
7	kube-proxy	√	√	v1.13.3	container
8	flannel	√	√	v1.13.3	container
9	pause	√	√	3.1	container
10	apiserver	√		v1.13.3	container
11	controller-manager	√		v1.13.3	container
12	scheduler	√		v1.13.3	container
13	etcd	√		3.2.24	container

pod 网络：10.244.0.0/16
service 网络：10.92.0.0/12
节点网络：172.20.0.0/16

1.3 k8s 的两种部署方式

方式 1

kubeadm 方式部署，k8s 可以把 k8s 自身的大部分应用管控起来，即运行于 pod 上，但是 kubelet 和 docker 不能这样实现自托管，这两个主机运行为守护进程，因此，只需要在所有主机都安装 kubelet 和 docker,构建 k8s 集群。相当于是自举。etcd 也是托管于 pod 上运行，使用 kubeadm 进行部署，安装过程相对简单。这些主件的 pod 一般为静态 pod (不属于 k8s 管理)，也可以运行为自托管的 pod.每个主机都要运行 flannel 这个主件，可以运行为 pod。flannel 为动态 pod。

kubeadm 的介绍可以查看如下链接

https://github.com/kubernetes/kubeadm/blob/master/docs/design/design_v1.10.md

安装步骤如下三步

1.master 和 node 安装 kubelet,kubeadm,docker

2.mster:kubeadm init，集群初始化

3.nodes:kubeadm join，node 节点加入集群

方式 2

手动配置，主节点和 node 都主要组件运行为系统级的守护进程，每一步都需要手动处理，如证书和配置过程都是用手动配置的。另外，这种方式在 github 上有 playbook 自动化实现

a).master:安装 apiserver,scheduler,controller-manager,etcd,flanel

b).node:安装 kublet,kub-proxy,docker(container engine),flannel,需要多个节点

c).etcd:安装 etcd 存储服务器，建议配置为高可用

这种方式，可以到 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG-1.11.md#downloads-for-v1112 下载相关的安装包，注意，master 或者 node 都是要安装 server 端的包。client 是交互时使用，也需要安装，不建议使用这种方式安装，难度较高

本文仅介绍使用 kubeadm 实现 k8s 集群安装

第二章 docker 安装

操作对象:

172.16.6.31
172.16.6.32
172.16.6.33

安装方法有很多，这里选择其中一种，rpm 方式。

2.1 安装

添加 docker 源:

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

或者使用国内阿里/清华的源:

```
yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

yum-config-manager --add-repo https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/centos/docker-ce.repo
```

从指定源安装 docker-ce ：

```
yum install docker-ce --enablerepo=docker-ce-stable -y

systemctl start docker
systemctl enable docker
```

查看是否开机运行：

```
systemctl list-unit-files|grep docker
```

2.2 确认

```
docker version
```

```
[root@docker07 ~]#
[root@docker07 ~]# docker version
Client:
 Version:           18.09.0
 API version:       1.39
 Go version:        go1.10.4
 Git commit:        4d60db4
 Built:             Wed Nov  7 00:48:22 2018
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.0
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.4
  Git commit:       4d60db4
  Built:            Wed Nov  7 00:19:08 2018
  OS/Arch:          linux/amd64
  Experimental:     false
[root@docker07 ~]#
```

2.3 ubuntu 安装（补充）

方法有很多，这里只说一种。

```
curl -sSL https://get.docker.com/ | sh
service start docker
sysv-rc-conf --list|grep docker
update-rc.d  docker  start 90 3 4 5 . stop 20 0 1 2 6 .
sysv-rc-conf --list|grep docker
docker version
```

第三章 kubeadm 等安装

操作主机:所有

所有主机安装 kubeadm、kubectl、kubelet

3.1 添加源

这里使用阿里云，也可使用其他源。另外，需要提醒的是，这几个包有个特别的地方，就是在下载后重新组装成的 rpm，而不是直接下载 rpm，所以必须在线安装。

```
cat >> /etc/yum.repos.d/k8s.repo << EOF
[k8s]
name=aliyun_k8s
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
enabled=1
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF
```

3.2 安装

```
yum install kubeadm
```

```
[root@docker09 ~]# yum install kubeadm
已加载插件：fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.shu.edu.cn
 * extras: mirrors.aliyun.com
 * updates: mirrors.163.com
k8s
k8s/primary
k8s
正在解决依赖关系
--> 正在检查事务
---> 软件包 kubeadm.x86_64.0.1.13.3-0 将被 安装
--> 正在处理依赖关系 kubernetes-cni >= 0.6.0，它被软件包 kubeadm-1.13
--> 正在处理依赖关系 kubelet >= 1.6.0，它被软件包 kubeadm-1.13.3-0.x8
--> 正在处理依赖关系 kubectl >= 1.6.0，它被软件包 kubeadm-1.13.3-0.x8
--> 正在处理依赖关系 cri-tools >= 1.11.0，它被软件包 kubeadm-1.13.3-0
--> 正在检查事务
---> 软件包 cri-tools.x86_64.0.1.12.0-0 将被 安装
---> 软件包 kubectl.x86_64.0.1.13.3-0 将被 安装
---> 软件包 kubelet.x86_64.0.1.13.3-0 将被 安装
--> 正在处理依赖关系 socat，它被软件包 kubelet-1.13.3-0.x86_64 需要
```

自动安装依赖 kubectl 、kubelet、 kubernetes-cni

```
 Package                     架构          版本              源
===========================================================================
正在安装:
 kubeadm                     x86_64        1.13.3-0          k8s
为依赖而安装:
 conntrack-tools             x86_64        1.4.4-4.el7       base
 cri-tools                   x86_64        1.12.0-0          k8s
 kubectl                     x86_64        1.13.3-0          k8s
 kubelet                     x86_64        1.13.3-0          k8s
 kubernetes-cni              x86_64        0.6.0-0           k8s
 libnetfilter_cthelper       x86_64        1.0.0-9.el7       base
 libnetfilter_cttimeout      x86_64        1.0.0-6.el7       base
 libnetfilter_queue          x86_64        1.0.2-2.el7_2     base
 socat                       x86_64        1.7.3.2-2.el7     base

事务概要
===========================================================================
安装  1 软件包 (+9 依赖软件包)
```

第四章 部署集群

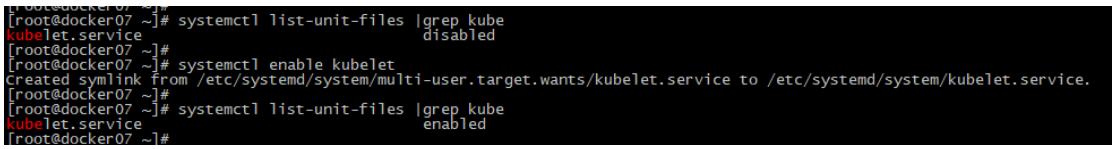
操作对象：

172.16.6.31
172.16.6.32
172.16.6.33

4.1 环境准备

4.1.1 kubelet 加入开机启动

```
systemctl enable kubelet
```



此时无法启动 kubelet，因为还未初始化完成，但需要将此服务加入开机启动

4.1.2 禁止 firewalld

```
systemctl stop firewalld
systemctl disable firewalld
```

4.1.3 调整内核参数

主要调整以下三个参数，并将其加入到/etc/rc.local 中。

```
echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables
echo 1 > /proc/sys/net/bridge/bridge-nf-call-ip6tables
echo 1 > /proc/sys/net/ipv4/ip_forward
```

4.1.4 host 配置

```
cat >> /etc/hosts << EOF
172.16.6.31 docker01
172.16.6.32 docker02
172.16.6.33 docker03
172.16.6.34 docker04
172.16.6.35 docker05
172.16.6.36 docker06
172.16.6.37 docker07
172.16.6.38 docker08
172.16.6.39 docker09
172.16.6.40 docker10
EOF
```

4.1.5 忽略 swap 错误

k8s 默认不支持 swap，如果开启了会 error 报错，处理方式有两种

方法 1：禁止 swap

```
swapoff -a && sed -i '/swap/d' /etc/fstab
```

方法 2：强制使用 swap

```
echo "KUBELET_EXTRA_ARGS=\"--fail-swap-on=false\"" > /etc/sysconfig/kubelet
```

并在初始化时添加如下参数

```
--ignore-preflight-errors=Swap
```

4.1.6 网络不通处理

初始化过程，默认会到 gcr.io/google_containers 站点拉取相关 k8s 的镜像信息，所需的镜像信息如 4.2.1 所列出。当前国内不能进行这些站点的访问，也就不能访问进行初始化安装。

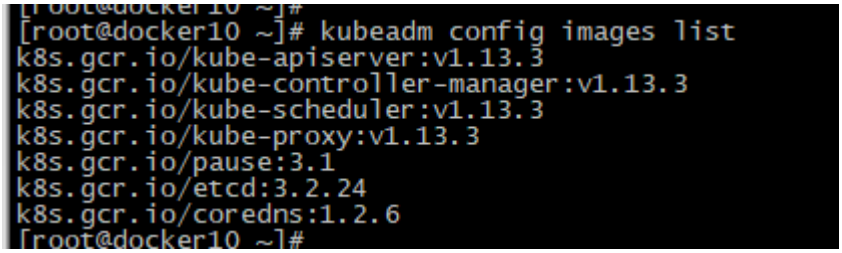
解决方法 1：使用国外的代理服务器或则其他方法，使能够从该站点下载对应镜像

解决方法 2：使用 docker 官方的克隆镜像，方法如 4.4.2 所示。

本文档使用方法 2，方法 1 不再演示。

4.2 启动 master 节点

4.2.1 所需的镜像



k8s.gcr.io/kube-apiserver:v1.13.3
k8s.gcr.io/kube-controller-manager:v1.13.3
k8s.gcr.io/kube-scheduler:v1.13.3
k8s.gcr.io/kube-proxy:v1.13.3
k8s.gcr.io/pause:3.1
k8s.gcr.io/etcd:3.2.24
k8s.gcr.io/coredns:1.2.6

注意 coredns、etcd 和 kube 模块的版本对应关系，可使用命令

```
kubeadm config images list
```

查到

4.2.2 拉取镜像

使用如下命令下载上述列出的镜像

```
docker pull mirrorgooglecontainers/kube-apiserver:v1.13.3
docker pull mirrorgooglecontainers/kube-controller-manager:v1.13.3
docker pull mirrorgooglecontainers/kube-scheduler:v1.13.3
docker pull mirrorgooglecontainers/kube-proxy:v1.13.3
docker pull mirrorgooglecontainers/pause:3.1
docker pull mirrorgooglecontainers/etcd:3.2.24
docker pull coredns/coredns:1.2.6
```

各模块还包含 64 位版本，比如 etcd 和 pause 可到如下页面查到。

https://hub.docker.com/r/mirrorgooglecontainers/etcd-amd64/tags

https://hub.docker.com/r/mirrorgooglecontainers/pause-amd64/tags

添加标签：

```
docker tag mirrorgooglecontainers/kube-apiserver:v1.13.3 k8s.gcr.io/kube-apiserver:v1.13.3
docker tag mirrorgooglecontainers/kube-controller-manager:v1.13.3 k8s.gcr.io/kube-controller-manager:v1.13.3
docker tag mirrorgooglecontainers/kube-scheduler:v1.13.3 k8s.gcr.io/kube-scheduler:v1.13.3
docker tag mirrorgooglecontainers/kube-proxy:v1.13.3 k8s.gcr.io/kube-proxy:v1.13.3
docker tag mirrorgooglecontainers/pause:3.1 k8s.gcr.io/pause:3.1
docker tag mirrorgooglecontainers/etcd:3.2.24 k8s.gcr.io/etcd:3.2.24
docker tag coredns/coredns:1.2.6 k8s.gcr.io/coredns:1.2.6
```

修改完成后，查看镜像

```
[root@docker07 ~]# docker images
REPOSITORY                                           TAG                 IMAGE ID            CREATED             SIZE
mirrorgooglecontainers/kube-apiserver                v1.13.3             fe242e556a99        2 weeks ago         181MB
k8s.gcr.io/kube-apiserver                            v1.13.3             fe242e556a99        2 weeks ago         181MB
mirrorgooglecontainers/kube-proxy                    v1.13.3             98db19758ad4        2 weeks ago         80.3MB
k8s.gcr.io/kube-proxy                                v1.13.3             98db19758ad4        2 weeks ago         80.3MB
mirrorgooglecontainers/kube-controller-manager       v1.13.3             0482f6400933        2 weeks ago         146MB
k8s.gcr.io/kube-controller-manager                   v1.13.3             0482f6400933        2 weeks ago         146MB
mirrorgooglecontainers/kube-scheduler                v1.13.3             3a6f709e97a0        2 weeks ago         79.6MB
k8s.gcr.io/kube-scheduler                            v1.13.3             3a6f709e97a0        2 weeks ago         79.6MB
coredns/coredns                                      1.3.1               eb516548c180        5 weeks ago         40.3MB
k8s.gcr.io/coredns                                   1.3.1               eb516548c180        5 weeks ago         40.3MB
mirrorgooglecontainers/etcd                          3.3.10              2c4adeb21b4f        2 months ago        258MB
k8s.gcr.io/etcd                                      3.3.10              2c4adeb21b4f        2 months ago        258MB
coredns/coredns                                      1.2.6               f59dcacceff4        3 months ago        40MB
k8s.gcr.io/coredns                                   1.2.6               f59dcacceff4        3 months ago        40MB
mirrorgooglecontainers/etcd                          3.2.24              3cab8e1b9802        5 months ago        220MB
k8s.gcr.io/etcd                                      3.2.24              3cab8e1b9802        5 months ago        220MB
k8s.gcr.io/pause                                     3.1                 da86e6ba6ca1        14 months ago       742kB
mirrorgooglecontainers/pause                         3.1                 da86e6ba6ca1        14 months ago       742kB
[root@docker07 ~]#
[root@docker07 ~]#
```

此时可以删除 mirrorgooglecontainers 相关的标签，我这里不再处理。

4.2.3 初始化集群

使用如下命令初始化集群

```
kubeadm init --kubernetes-version=v1.13.3 --pod-network-cidr=10.244.0.0/16 --service-cidr=10.96.0.0/12 --ignore-preflight-errors=Swap
```

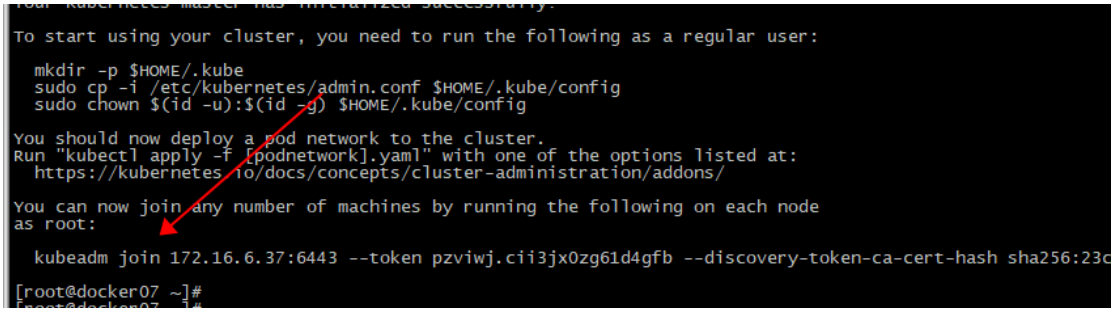
已经要进行 4.1 步骤，否则会有如下几个警告信息，

```
[root@docker07 ~]#
[root@docker07 ~]# kubeadm init --kubernetes-version=v1.13.3 --pod-network-cidr=10.244.0.0/16 --service-cidr=10.96.0.0/12
[init] Using Kubernetes version: v1.13.3
[preflight] Running pre-flight checks
        [WARNING Firewalld]: firewalld is active, please ensure ports [6443 10250] are open or your cluster may not function correctly
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 18.09.0. Latest validated version: 18.06
        [WARNING Hostname]: hostname "docker07" could not be reached
        [WARNING Hostname]: hostname "docker07": lookup docker07 on 1.2.4.8:53: no such host
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR Swap]: running with swap on is not supported. Please disable swap
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
[root@docker07 ~]#
[root@docker07 ~]#
```

其中第二个警告信息说，kubeadm 目前支持最高版本是 18.06，而我们安装的是 18.09，这个警告忽略即可。

```
[root@docker07 ~]#
[root@docker07 ~]# kubeadm init --kubernetes-version=v1.13.3 --pod-network-cidr=10.244.0.0/16 --service-cidr=10.96.0.0/12 --ignore-preflight-errors=Swap
[init] Using Kubernetes version: v1.13.3
[preflight] Running pre-flight checks
        [WARNING Swap]: running with swap on is not supported. Please disable swap
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 18.09.0. Latest validated version: 18.06
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [docker07 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.loc
.96.0.1 172.16.6.37]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [docker07 localhost] and IPs [172.16.6.37 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [docker07 localhost] and IPs [172.16.6.37 127.0.0.1 ::1]
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
```

初始化完成后，如下提示：



记录下上面这句话，用于 node 节点加入集群:

```
kubeadm join 172.16.6.37:6443 --token pzviwj.cii3jx0zg61d4gfb --discovery-token-ca-cert-hash sha256:23cbb3efbe8a2e2b73cc1442cf4b132b98511a451ffe14dacfe25b9594599c1a
```

4.3 启动 node 节点

4.3.1 安装必要包

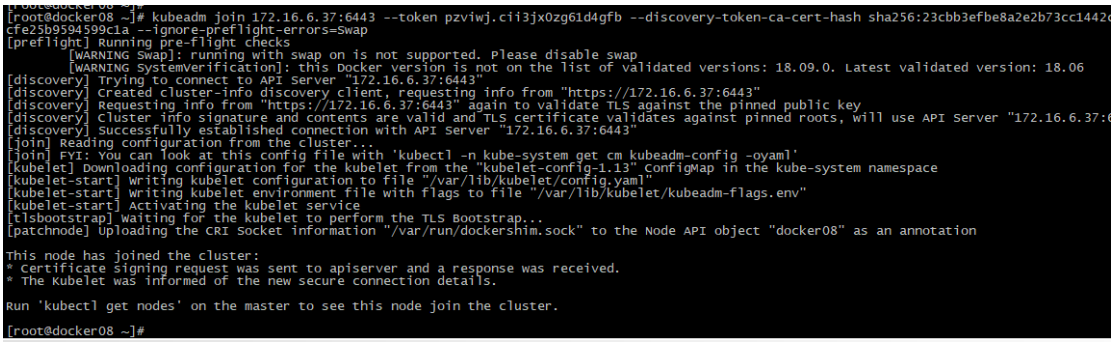
```
docker pull mirrorgooglecontainers/kube-proxy:v1.13.3
docker pull mirrorgooglecontainers/pause:3.1
docker tag mirrorgooglecontainers/kube-proxy:v1.13.3 k8s.gcr.io/kube-proxy:v1.13.3
docker tag mirrorgooglecontainers/pause:3.1 k8s.gcr.io/pause:3.1
```

4.3.2 加入集群

使用如下语句在 node 节点上执行即可加入集群，我这里所用了 swap

```
kubeadm join 172.16.6.37:6443 --token pzviwj.cii3jx0zg61d4gfb --discovery-token-ca-cert-hash sha256:23cbb3efbe8a2e2b73cc1442cf4b132b98511a451ffe14dacfe25b9594599c1a --ignore-preflight-errors=Swap
```

如下图：docker08 加入集群:



4.3.3 排错

如果出现如下错误

```
[root@docker09 ~]#
[root@docker09 ~]# kubeadm join 172.16.6.37:6443 --token pzviwj.cii3jx0zg61d4gfb --discovery-token-ca-cert-hash sh
cfe25b9594599c1a --ignore-preflight-errors=Swap
[preflight] Running pre-flight checks
        [WARNING Swap]: running with swap on is not supported. Please disable swap
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 18.09.0. Lates
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
[discovery] Trying to connect to API Server "172.16.6.37:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://172.16.6.37:6443"
[discovery] Requesting info from "https://172.16.6.37:6443" again to validate TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will
[discovery] Successfully established connection with API Server "172.16.6.37:6443"
[join] Reading configuration from the cluster...
[join] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
unable to fetch the kubeadm-config ConfigMap: failed to get config map: Unauthorized
[root@docker09 ~]#
[root@docker09 ~]#
```

unable to fetch the kubeadm-config ConfigMap: failed to get config map: Unauthorized

是因为 token 过期了，默认有效期 24 小时。

解决方法：在 master 节点上，使用如下命令重新生产新的 token

```
kubeadm token create
```

```
[root@docker07 ~]#
[root@docker07 ~]# kubeadm token create
e4l7o1.expvenkjvafg93yt
[root@docker07 ~]#
[root@docker07 ~]#
```

使用新的 token 重新加入集群，如下图

```
[root@docker09 ~]#
[root@docker09 ~]# kubeadm join 172.16.6.37:6443 --token e4l7o1.expvenkjvafg93yt --discovery-token-ca-cert-hash sha256:23cbb3efbe8a2e2b7
cfe25b9594599c1a --ignore-preflight-errors=Swap
[preflight] Running pre-flight checks
        [WARNING Swap]: running with swap on is not supported. Please disable swap
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 18.09.0. Latest validated version: 1
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
[discovery] Trying to connect to API Server "172.16.6.37:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://172.16.6.37:6443"
[discovery] Requesting info from "https://172.16.6.37:6443" again to validate TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "172.1
[discovery] Successfully established connection with API Server "172.16.6.37:6443"
[join] Reading configuration from the cluster...
[join] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.13" ConfigMap in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "docker09" as an annotation

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the master to see this node join the cluster.
```

4. 3. 4　删除 node

```
kubectl drain docker08 --delete-local-data
kubectl delete node docker08
```

```
[root@docker07 ~]# kubectl delete node docker08
node "docker08" deleted
[root@docker07 ~]#
```

4. 4 配置网络

k8s 支持多种网络模型，比如

4. 4. 1　master 节点

使用如下语句安装：

```
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

4. 4. 2　node 节点

flannel 版本选择，查看如下：

https://quay.io/repository/coreos/flannel?tab=tags

使用如下命令下载镜像：

```
docker pull quay.io/coreos/flannel:v0.11.0-amd64
```

下载后，会被主节点调度，自动配置

4.5 部署 web-ui

到这里可以看到版本对应关系：
https://github.com/kubernetes/dashboard/releases
到这里选择镜像版本
https://hub.docker.com/r/mirrorgooglecontainers/kubernetes-dashboard-amd64/tags
比如我选择最新的 1.10.1 版本
docker pull mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.1

4.6 观察

集群启动后，在 master 节点上观察集群运行状态是否和规划相符

4.5.1 配置环境变量

输入以下语句

```
echo "export KUBECONFIG=/etc/kubernetes/admin.conf" >> ~/.bash_profile
source ~/.bash_profile
```

若不进行这一步，执行任何 kubectl 命令都将出现以下错误

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl get nodes
The connection to the server localhost:8080 was refused - did you specify the right host or port?
[root@docker07 ~]#
[root@docker07 ~]#
```

4.5.2 查看组件状态

```
kubectl get componentstatus
kubectl get cs
```

```
[root@docker07 ~]# kubectl get componentstatus
NAME                 STATUS    MESSAGE             ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-0               Healthy   {"health": "true"}
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl get cs
NAME                 STATUS    MESSAGE             ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-0               Healthy   {"health": "true"}
[root@docker07 ~]#
[root@docker07 ~]#
```

4.5.3 查看节点状态

```
[root@docker07 ~]# kubectl get nodes
NAME       STATUS     ROLES    AGE     VERSION
docker07   NotReady   master   2d23h   v1.13.3
docker08   NotReady   <none>   2d22h   v1.13.3
docker09   NotReady   <none>   80m     v1.13.3
docker10   NotReady   <none>   79m     v1.13.3
[root@docker07 ~]#
```

看到状态都是 NotReady 状态，因为未执行 4.4 步骤，执行后查看信息如下:

```
[root@docker07 ~]# kubectl get nodes
NAME       STATUS   ROLES    AGE     VERSION
docker07   Ready    master   3d1h    v1.13.3
docker08   Ready    <none>   3d1h    v1.13.3
docker09   Ready    <none>   3h23m   v1.13.3
docker10   Ready    <none>   3h22m   v1.13.3
[root@docker07 ~]#
```

4.5.4 查看名称空间

kubectl get namespace

kubectl get ns

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl get namespace
NAME          STATUS   AGE
default       Active   4d21h
kube-public   Active   4d21h
kube-system   Active   4d21h
[root@docker07 ~]#
[root@docker07 ~]# kubectl get ns
NAME          STATUS   AGE
default       Active   4d21h
kube-public   Active   4d21h
kube-system   Active   4d21h
[root@docker07 ~]#
```

可以看到一共有三个名称空间,

default : 默认的

kube-public: 公共的

kube-system: 系统级别的

4.5.5 查看资源

kubectl get deployments

```
[root@docker07 ~]# kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   1/1     1            1           2m52s
nginx   3/3     3            3           24h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-5d4d8c8458-wkskj   1/1     Running   0          2m56s
nginx-7cdbd8cdc9-6bxcj   1/1     Running   0          24h
nginx-7cdbd8cdc9-csb95   1/1     Running   0          24h
nginx-7cdbd8cdc9-qc5vh   1/1     Running   0          24h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl get deployments -n kube-system
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
coredns   2/2     2            2           4d22h
[root@docker07 ~]#
```

4.5.6 查看 service

kubectl get service

kubectl get svc

```
[root@docker07 ~]# kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        5d2h
myapp        NodePort    10.107.127.2    <none>        80:31274/TCP   4h
nginx        NodePort    10.103.94.20    <none>        80:30435/TCP   28h
[root@docker07 ~]#
[root@docker07 ~]# kubectl get services -n kube-system
NAME       TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)         AGE
kube-dns   ClusterIP   10.96.0.10    <none>        53/UDP,53/TCP   5d2h
[root@docker07 ~]#
```

从图中可以看到 cluster-ip，此 ip 即是 service 网络的 ip，在集群初始化时使用如下参数定义的网络

--service-cidr=10.96.0.0/12

图中 10.96、10.103、10.107 等都属于 10.96.0.0/12 网络。

需要说明的 kube-dns 的 IP：10.96.0.10 将作为所有 pod 中的默认 nameserver，以此来解析其他服务，服务之间的调用使用服务名。

使用如下命令

```
kubectl describe svc myapp
```

可以查看 service 的详情，包括很多信息，看下图

```
[root@docker07 ~]# kubectl describe svc myapp
Name:              myapp
Namespace:         default
Labels:            run=myapp
Annotations:       <none>
Selector:          run=myapp
Type:              ClusterIP
IP:                10.96.245.240
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         10.244.2.11:80,10.244.3.11:80,10.244.4.14:80
Session Affinity:  None
Events:            <none>
[root@docker07 ~]#
[root@docker07 ~]# kubectl describe svc nginx
Name:                     nginx
Namespace:                default
Labels:                   run=nginx
Annotations:              <none>
Selector:                 run=nginx
Type:                     NodePort
IP:                       10.103.94.20
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30435/TCP
Endpoints:                10.244.2.2:80,10.244.3.2:80,10.244.3.6:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
[root@docker07 ~]#
```

4.5.7 查看 endpoints

k8s 创建 service 的同时，会自动创建跟 service 同名的 endpoints。

使用如下语句可查看详细信息，包括 endpoint 后端对应的 pod 的 IP 地址

```
kubectl get endpoints kube-dns -o yaml
```

```
[root@docker07 ~]# kubectl get endpoints kube-dns -o yaml -n kube-system
apiVersion: v1
kind: Endpoints
metadata:
  creationTimestamp: "2019-02-15T08:33:12Z"
  labels:
    k8s-app: kube-dns
    kubernetes.io/cluster-service: "true"
    kubernetes.io/name: KubeDNS
  name: kube-dns
  namespace: kube-system
  resourceVersion: "347640"
  selfLink: /api/v1/namespaces/kube-system/endpoints/kube-dns
  uid: 551e3a64-30fc-11e9-a2d0-000c29b30ea9
subsets:
- addresses:
  - ip: 10.244.0.2
    nodeName: docker07
    targetRef:
      kind: Pod
      name: coredns-86c58d9df4-2nwl7
      namespace: kube-system
```

使用 kubectl get pods 也可以看到对应的 pod 的 IP

4.5.8 查看 pods

kubectl get pods

默认查看 default 的 pod

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-7cdbd8cdc9-6bxcj   1/1     Running   0          24h
nginx-7cdbd8cdc9-csb95   1/1     Running   0          24h
nginx-7cdbd8cdc9-qc5vh   1/1     Running   0          24h
[root@docker07 ~]#
[root@docker07 ~]#
```

kubectl get pods -n kube-public -o wide

使用-n 查看指定名称空间的 pod

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pods -n kube-system -o wide
NAME                                 READY   STATUS    RESTARTS   AGE    IP              NODE       NOMINATED NODE   READINESS GATES
coredns-86c58d9df4-2nwl7             1/1     Running   0          4d21h  10.244.0.2      docker07   <none>           <none>
coredns-86c58d9df4-5969g             1/1     Running   0          4d21h  10.244.0.3      docker07   <none>           <none>
etcd-docker07                        1/1     Running   0          4d21h  172.16.6.37     docker07   <none>           <none>
kube-apiserver-docker07              1/1     Running   0          4d21h  172.16.6.37     docker07   <none>           <none>
kube-controller-manager-docker07     1/1     Running   0          4d21h  172.16.6.37     docker07   <none>           <none>
kube-flannel-ds-amd64-2ffm8          1/1     Running   0          46h    172.16.6.40     docker10   <none>           <none>
kube-flannel-ds-amd64-fwwrr          1/1     Running   0          46h    172.16.6.38     docker08   <none>           <none>
kube-flannel-ds-amd64-kbc55          1/1     Running   0          46h    172.16.6.37     docker07   <none>           <none>
kube-flannel-ds-amd64-qp6b5          1/1     Running   0          46h    172.16.6.39     docker09   <none>           <none>
kube-proxy-dfr7w                     1/1     Running   0          2d     172.16.6.40     docker10   <none>           <none>
kube-proxy-dgtsm                     1/1     Running   0          4d21h  172.16.6.37     docker07   <none>           <none>
kube-proxy-l9b7l                     1/1     Running   0          2d     172.16.6.39     docker09   <none>           <none>
kube-proxy-zz2z6                     1/1     Running   0          4d21h  172.16.6.38     docker08   <none>           <none>
kube-scheduler-docker07              1/1     Running   0          4d21h  172.16.6.37     docker07   <none>           <none>
[root@docker07 ~]#
```

从这里可以看到整个集群的架构，这种部署方式就是将系统组件也作为 pod 运行。

使用--show-labels 可以将标签也一并显示出来，如下图的 run=myapp 就是标签

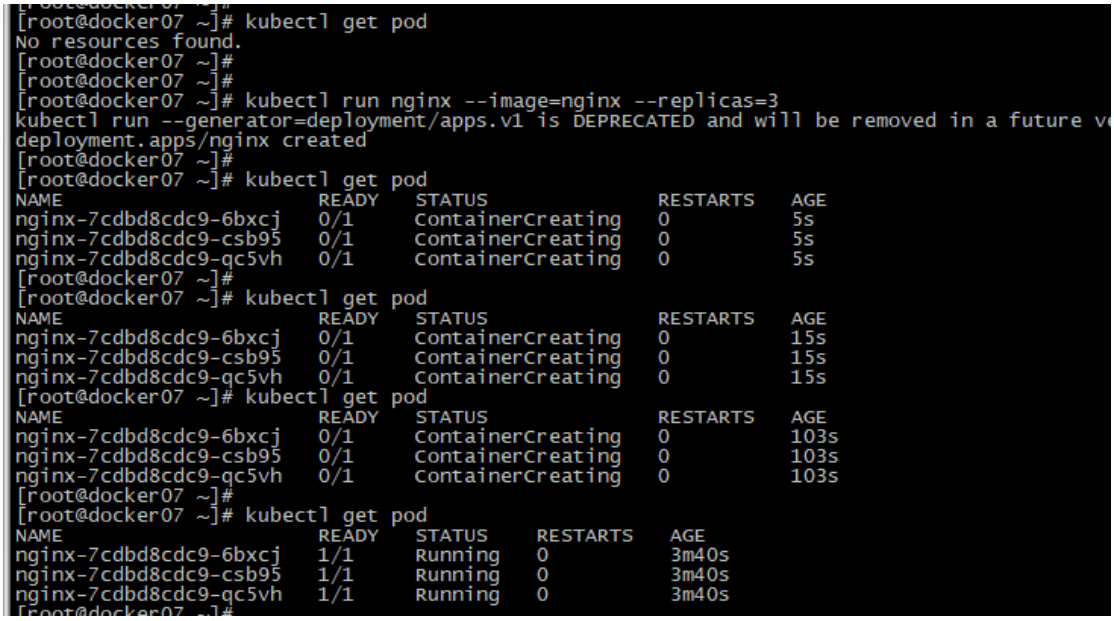
```
[root@docker07 ~]# kubectl get pods --show-labels
NAME                     READY   STATUS    RESTARTS   AGE   LABELS
busybox                  1/1     Running   15         15h   <none>
myapp-5d4d8c8458-5w4jc   1/1     Running   0          16m   pod-template-hash=5d4d8c8458,run=myapp
myapp-5d4d8c8458-kz5vw   1/1     Running   0          16m   pod-template-hash=5d4d8c8458,run=myapp
myapp-5d4d8c8458-qj72g   1/1     Running   0          16m   pod-template-hash=5d4d8c8458,run=myapp
nginx-7cdbd8cdc9-6bxcj   1/1     Running   0          44h   pod-template-hash=7cdbd8cdc9,run=nginx
nginx-7cdbd8cdc9-csb95   1/1     Running   0          44h   pod-template-hash=7cdbd8cdc9,run=nginx
nginx-7cdbd8cdc9-hhrl6   1/1     Running   0          18h   pod-template-hash=7cdbd8cdc9,run=nginx
[root@docker07 ~]#
[root@docker07 ~]#
```

第五章 测试

5.1 创建集群实例-nginx

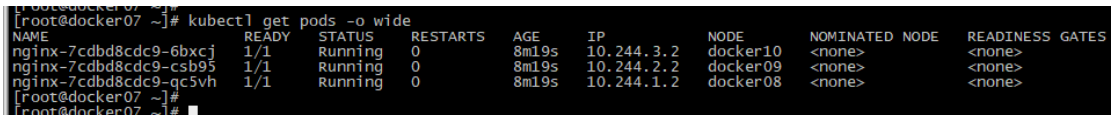
5.5.1 创建 nginx 的 pod

```
kubectl get pod
kubectl run nginx --image=nginx --replicas=3
kubectl get pod
```



使用-o wide 参数可看到更加详细的信息

```
kubectl get pods -o wide
```



从图中我们可以看到，每个 node 节点运行了一个 pod

5.5.2 创建 pod 的 service

pod 包含容器，着眼于多节点的服务，而服务访问的入口由 service 提供，service 提供类似于 lvs 类似的功能，做流量分发，使用如下命令创建 service

```
kubectl expose deployment nginx --port=80 --target-port=80 --type=NodePort
```

--type 的可选参数:

NodePort 节点 pod，节点会暴露端口到宿主机网卡，集群外部可以访问

ClusterIP 集群内部 pod

LoadBalancer

ExternalName

使用如下命令可查看端口监听情况:

```
kubectl get service
kubectl get svc
```

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl expose deployment nginx --port=80 --target-port=80 --type=NodePort
service/nginx exposed
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
nginx-7cdbd8cdc9-6bxcj   1/1     Running   0          47m   10.244.3.2   docker10   <none>           <none>
nginx-7cdbd8cdc9-csb95   1/1     Running   0          47m   10.244.2.2   docker09   <none>           <none>
nginx-7cdbd8cdc9-qc5vh   1/1     Running   0          47m   10.244.1.2   docker08   <none>           <none>
[root@docker07 ~]#
[root@docker07 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        3d22h
nginx        NodePort    10.103.94.20   <none>        80:30435/TCP   107s
```

其中 nginx 服务的类型（TYPE）是 NodePort，pod 所在宿主机将使用 30435 端口进行转发流量转发到 pod

5.5.3 访问 pod

使用以下命令

```
kubectl get service
```

可以查看对应的 cluster-ip 是：10.103.94.20，可通过此 IP 在安装了 flannel 的节点上进行访问：

```
[root@docker09 ~]# curl 10.103.94.20:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
```

我们看到还有一个 30435 端口， 该端口将监听在 node 节点上，如下图

```
[root@docker09 ~]#
[root@docker09 ~]# netstat -tnlp|grep 30435
tcp6       0      0 :::30435                :::*                    LISTEN      4151/kube-proxy
[root@docker09 ~]#
[root@docker09 ~]#
```

所以可以在集群外部通过该 node 节点的 30435 端口直接访问

5.2 创建另一个实例-myapp

5.2.1 创建 myapp

我这里使用 ikubernetes/myapp 镜像，可通过访问 http://**/hostname.html 返回主机名，通过 http://***返回版本，用于测试负载均衡和版本回退等各种功能。

```
kubectl run myapp --image=ikubernetes/myapp:v6 --replicas=3
kubectl expose deployment myapp --port=80 --target-port=80
kubectl get svc
```

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        5d18h
myapp        ClusterIP   10.96.245.240   <none>        80/TCP         2s
nginx        NodePort    10.103.94.20    <none>        80:30435/TCP   43h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl get service
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        5d18h
myapp        ClusterIP   10.96.245.240   <none>        80/TCP         13s
nginx        NodePort    10.103.94.20    <none>        80:30435/TCP   43h
[root@docker07 ~]#
[root@docker07 ~]# kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        5d18h
myapp        ClusterIP   10.96.245.240   <none>        80/TCP         18s
nginx        NodePort    10.103.94.20    <none>        80:30435/TCP   43h
[root@docker07 ~]#
```

5.2.2 删除控制器

```
kubectl delete deployment myapp
```

```
[root@docker07 ~]# kubectl delete deployment myapp
deployment.extensions "myapp" deleted
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pods
NAME                     READY   STATUS        RESTARTS   AGE
myapp-5d4d8c8458-6rz5g   1/1     Terminating   0          12m
myapp-5d4d8c8458-c4lbz   1/1     Terminating   0          12m
myapp-5d4d8c8458-k78rl   0/1     Terminating   0          12m
nginx-7cdbd8cdc9-6bxcj   1/1     Running       0          24h
nginx-7cdbd8cdc9-csb95   1/1     Running       0          24h
nginx-7cdbd8cdc9-qc5vh   1/1     Running       0          24h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-7cdbd8cdc9-6bxcj   1/1     Running   0          24h
nginx-7cdbd8cdc9-csb95   1/1     Running   0          24h
nginx-7cdbd8cdc9-qc5vh   1/1     Running   0          24h
[root@docker07 ~]#
```

需要说明的是，及时删除了调度器，对应的 service 还是存在的

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl delete deployment myapp
deployment.extensions "myapp" deleted
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        5d17h
myapp        NodePort    10.107.127.2   <none>        80:31274/TCP   19h
nginx        NodePort    10.103.94.20   <none>        80:30435/TCP   43h
[root@docker07 ~]#
[root@docker07 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        5d17h
myapp        NodePort    10.107.127.2   <none>        80:31274/TCP   19h
nginx        NodePort    10.103.94.20   <none>        80:30435/TCP   43h
[root@docker07 ~]#
[root@docker07 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        5d17h
myapp        NodePort    10.107.127.2   <none>        80:31274/TCP   19h
nginx        NodePort    10.103.94.20   <none>        80:30435/TCP   43h
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
busybox                  1/1     Running   15         15h
nginx-7cdbd8cdc9-6bxcj   1/1     Running   0          44h
nginx-7cdbd8cdc9-csb95   1/1     Running   0          44h
nginx-7cdbd8cdc9-hhrl6   1/1     Running   0          18h
[root@docker07 ~]#
[root@docker07 ~]#
```

5.2.3 删除 service

kubectl delete service myapp

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        5d17h
myapp        NodePort    10.107.127.2   <none>        80:31274/TCP   19h
nginx        NodePort    10.103.94.20   <none>        80:30435/TCP   43h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl delete service myapp
service "myapp" deleted
[root@docker07 ~]#
[root@docker07 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        5d17h
nginx        NodePort    10.103.94.20   <none>        80:30435/TCP   43h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]#
```

5.2.4 删除 pod

kubectl delete po myapp-5d4d8c8458-wkskj

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl  get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-5d4d8c8458-wkskj   1/1     Running   0          10m
nginx-7cdbd8cdc9-6bxcj   1/1     Running   0          24h
nginx-7cdbd8cdc9-csb95   1/1     Running   0          24h
nginx-7cdbd8cdc9-qc5vh   1/1     Running   0          24h
[root@docker07 ~]#
[root@docker07 ~]# kubectl delete myapp-5d4d8c8458-wkskj
error: resource(s) were provided, but no name, label selector, or --all flag specified
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl delete po myapp-5d4d8c8458-wkskj
pod "myapp-5d4d8c8458-wkskj" deleted
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl  get pods
NAME                     READY   STATUS              RESTARTS   AGE
myapp-5d4d8c8458-gn28c   0/1     ContainerCreating   0          11s
nginx-7cdbd8cdc9-6bxcj   1/1     Running             0          24h
nginx-7cdbd8cdc9-csb95   1/1     Running             0          24h
nginx-7cdbd8cdc9-qc5vh   1/1     Running             0          24h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl  get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-5d4d8c8458-gn28c   1/1     Running   0          29s
nginx-7cdbd8cdc9-6bxcj   1/1     Running   0          24h
nginx-7cdbd8cdc9-csb95   1/1     Running   0          24h
nginx-7cdbd8cdc9-qc5vh   1/1     Running   0          24h
[root@docker07 ~]#
```

从上图可以看出我们删除一个 pod，还会再启动一个 pod，这就是自愈。

强制删除 pod，加上参数：--grace-period=0 --force

```
kubectl delete pod myapp-5d4d8c8458-g9wkt --grace-period=0 --force
```

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl delete pod myapp-5d4d8c8458-g9wkt --grace-period=0 --force
warning: Immediate deletion does not wait for confirmation that the running resource has been te
pod "myapp-5d4d8c8458-g9wkt" force deleted
[root@docker07 ~]#
```

5.2.5 扩/缩容

```
kubectl scale --replicas=3 deployment/myapp
```

```
[root@docker07 ~]# kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   1/1     1            1           15m
nginx   3/3     3            3           24h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl scale --replicas=3 deployment/myapp
deployment.extensions/myapp scaled
[root@docker07 ~]#
[root@docker07 ~]# kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   1/3     3            1           16m
nginx   3/3     3            3           24h
[root@docker07 ~]#
[root@docker07 ~]# kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   2/3     3            2           17m
nginx   3/3     3            3           24h
[root@docker07 ~]#
[root@docker07 ~]# kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   3/3     3            3           17m
nginx   3/3     3            3           24h
[root@docker07 ~]#
```

5.2.6 动态查看执行过程

使用如下命令可以动态查看过程，包括扩/缩容、版本更新、回退等

```
kubectl rollout status deployment myapp
```

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl scale --replicas=10 deployment/myapp
deployment.extensions/myapp scaled
[root@docker07 ~]# kubectl rollout status deployment myapp
Waiting for deployment "myapp" rollout to finish: 5 of 10 updated replicas are available...
Waiting for deployment "myapp" rollout to finish: 6 of 10 updated replicas are available...
Waiting for deployment "myapp" rollout to finish: 7 of 10 updated replicas are available...
Waiting for deployment "myapp" rollout to finish: 8 of 10 updated replicas are available...
Waiting for deployment "myapp" rollout to finish: 9 of 10 updated replicas are available...
deployment "myapp" successfully rolled out
[root@docker07 ~]#
```

如果所示是进行节点扩容时的动态查看

5.2.7 滚动更新

Kubectl rollout undo deployment myapp　#回滚，默认回滚到上一个版本

使用如下命令进行滚动更新：

```
kubectl set image deployment/myapp myapp=ikubernetes/myapp:v5
```

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl set image deployment/myapp myapp=ikubernetes/myapp:v5
deployment.extensions/myapp image updated
[root@docker07 ~]#
[root@docker07 ~]# kubectl rollout status deployment myapp
Waiting for deployment "myapp" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "myapp" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "myapp" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "myapp" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "myapp" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "myapp" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "myapp" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "myapp" rollout to finish: 1 old replicas are pending termination...
deployment "myapp" successfully rolled out
[root@docker07 ~]#
[root@docker07 ~]#
```

同时使用如下命令进行动态观察

```
kubectl rollout status deployment myapp
```

也可使用 curl 请求 service 的的方法进行观察：

```
[root@docker07 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        5d18h
myapp        ClusterIP   10.96.245.240   <none>        80/TCP         19m
nginx        NodePort    10.103.94.20    <none>        80:30435/TCP   43h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# while ture;do curl 10.96.245.240 ;sleep 1;done
-bash: ture: 未找到命令
[root@docker07 ~]#
[root@docker07 ~]# while true;do curl 10.96.245.240 ;sleep 1;done
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
```

```
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
```

5.2.8 回退

使用如下命令进行版本回退，默认只回退到上一个版本：

```
kubectl rollout undo deployment myapp
```

```
[root@docker07 ~]# kubectl rollout undo deployment myapp
deployment.extensions/myapp rolled back
[root@docker07 ~]#
[root@docker07 ~]# kubectl rollout status deployment myapp
Waiting for deployment "myapp" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "myapp" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "myapp" rollout to finish: 2 old replicas are pending termination...
Waiting for deployment "myapp" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "myapp" rollout to finish: 1 old replicas are pending termination...
deployment "myapp" successfully rolled out
[root@docker07 ~]#
```

```
[root@docker07 ~]# while true;do curl 10.96.245.240 ;sleep 1;done
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v5 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
Hello MyApp | Version: v6 | <a href="hostname.html">Pod Name</a>
```

使用如下命令查看版本历史：

```
kubectl rollout history deployment myapp
```

使用如下命令指定版本回退：

```
kubectl rollout undo deployment myapp --to-revision=1
```

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl rollout history deployment myapp
deployment.extensions/myapp
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
3         <none>
4         <none>

[root@docker07 ~]#
[root@docker07 ~]# kubectl rollout undo deployment myapp --to-revision=1
deployment.extensions/myapp rolled back
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl rollout history deployment myapp
deployment.extensions/myapp
REVISION  CHANGE-CAUSE
2         <none>
3         <none>
4         <none>
5         <none>
```

要理解下这里版本的关系，按照时间顺序排列，1 最早，比如我回退到 1 版本，那么 5 将取代 1，历史版本中不会保留 1，也就是说历史版本中不会有重复版本。

5. 2. 9　内部 dns 理解

验证一：

```
kubectl get svc -n kube-system
dig -t A myapp.default.svc.cluster.local +short @10.96.0.10
kubectl get svc
```

```
[root@docker07 ~]#
[root@docker07 ~]# kubectl get svc -n kube-system
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)         AGE
kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP   5d18h
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# dig -t A myapp.default.svc.cluster.local +short @10.96.0.10
10.96.245.240
[root@docker07 ~]#
[root@docker07 ~]# dig -t A nginx.default.svc.cluster.local +short @10.96.0.10
10.103.94.20
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        5d18h
myapp        ClusterIP   10.96.245.240   <none>        80/TCP         53m
nginx        NodePort    10.103.94.20    <none>        80:30435/TCP   44h
[root@docker07 ~]#
```

```
[root@docker07 ~]# kubectl get pod -n kube-system -o wide
NAME                           READY   STATUS    RESTARTS   AGE     IP
coredns-86c58d9df4-2nwl7       1/1     Running   0          5d18h   10.244.0.2
coredns-86c58d9df4-5969g       1/1     Running   0          5d18h   10.244.0.3
```

```
[root@docker07 ~]# dig -t A nginx.default.svc.cluster.local +short @10.244.0.2
10.103.94.20
[root@docker07 ~]# dig -t A nginx.default.svc.cluster.local +short @10.244.0.3
10.103.94.20
[root@docker07 ~]#
[root@docker07 ~]# dig -t A kubernetes.default.svc.cluster.local +short @10.244.0.3
10.96.0.1
[root@docker07 ~]#
```

如上图，内部 dns 也是通过一个 service 进行服务分发，我们使用 dig 进行测试，可通过服务名 myapp、nginx 等进行解析。

dns 的默认搜索域是.default.svc.cluster.local

验证二：

我这里新建一个 busybox 的 pod，进入 busybox，查看 dns 的配置，可以看到 dns 的配置以及默认的搜索域等

```
/ # hostname
busybox
/ #
/ # cat /etc/resolv.conf
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
/ #
/ #
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
3: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1450 qdisc noqueue
    link/ether 0a:58:0a:f4:04:0b brd ff:ff:ff:ff:ff:ff
    inet 10.244.4.11/24 scope global eth0
       valid_lft forever preferred_lft forever
/ #
```

使用如下命令测试，得到和测试一相同的结果:

```
nslookup -type=A kubernetes.default.svc.cluster.local 10.96.0.10
```

```
/ # nslookup -type=A myapp.default.svc.cluster.local 10.96.0.10
Server:         10.96.0.10
Address:        10.96.0.10:53

Name:   myapp.default.svc.cluster.local
Address: 10.96.245.240

/ #
/ # nslookup -type=A nginx.default.svc.cluster.local 10.96.0.10
Server:         10.96.0.10
Address:        10.96.0.10:53

Name:   nginx.default.svc.cluster.local
Address: 10.103.94.20

/ #
/ # nslookup -type=A kubernetes.default.svc.cluster.local 10.96.0.10
Server:         10.96.0.10
Address:        10.96.0.10:53

Name:   kubernetes.default.svc.cluster.local
Address: 10.96.0.1
```

5.2.10 flannel 网络理解

flannel 作为一个 pod 运行，管理对应节点的 iptables 规则或者 ipvs 规则，使用如下命令

```
Iptable –vnL –t nat
```

查看当前的规则

每一个 node 节点中的 pod 属于同一 10.244.0.0/16 网络的子网，比如我这台是 10.244.4.0/24，所以整个集群内部所有 pod 的 ip 不会重复。

```
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
 1199 94359 KUBE-POSTROUTING  all  --  *      *       0.0.0.0/0            0.0.0.0/0            /* kubernetes postrouting rules *
    0     0 MASQUERADE  all  --  *      !docker0  172.17.0.0/16       0.0.0.0/0
  171 10588 RETURN     all  --  *      *       10.244.0.0/16        10.244.0.0/16
    3   252 MASQUERADE  all  --  *      *       10.244.0.0/16        !224.0.0.0/4
    0     0 RETURN     all  --  *      *      !10.244.0.0/16        10.244.4.0/24
    0     0 MASQUERADE  all  --  *      *      !10.244.0.0/16        10.244.0.0/16
```

如下图：为集群中每一个 pod（不管是否在此 node 上）创建一个 DNAT 规则，保证集群中任何一个 node 可以和集群中任何一个 pod 通信。

```
Chain KUBE-SEP-463KM6RTTDV6MG7S (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.4.23          0.0.0.0/0
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp to:10.244.4.23:80

Chain KUBE-SEP-4MQWJMQB3MP6HHHN (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.2.2           0.0.0.0/0
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp to:10.244.2.2:80

Chain KUBE-SEP-6E7XQMQ4RAYOWTTM (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.0.3           0.0.0.0/0
    0     0 DNAT       udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp to:10.244.0.3:53

Chain KUBE-SEP-6YBLMPJKOUHDFBST (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.3.2           0.0.0.0/0
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp to:10.244.3.2:80

Chain KUBE-SEP-7QJK43IC5KUCNJMY (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.3.18          0.0.0.0/0
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp to:10.244.3.18:80
```

为集群内的公共服务，比如 dns 的 53 端口、master 的 6443 端口也是作为 pod 运行的，所以也创建有对应的 DNAT 规则

```
Chain KUBE-SEP-HJ5LOISWSGGQY4OL (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       172.16.6.37          0.0.0.0/0
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp to:172.16.6.37:6443

Chain KUBE-SEP-IT2ZTR26TO4XFPTO (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.0.2           0.0.0.0/0
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp to:10.244.0.2:53

Chain KUBE-SEP-U6YIJEITHRETXKES (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.3.6           0.0.0.0/0
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp to:10.244.3.6:80

Chain KUBE-SEP-VJKWIWD27NYXSES2 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.3.19          0.0.0.0/0
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp to:10.244.3.19:80

Chain KUBE-SEP-YIL6JZP7A3QYXJU2 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.0.2           0.0.0.0/0
    0     0 DNAT       udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp to:10.244.0.2:53

Chain KUBE-SEP-ZXMNUKOKXUTL2MK2 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  all  --  *      *       10.244.0.3           0.0.0.0/0
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp to:10.244.0.3:53
```

为集群中的 service 服务创建规则，如下图中的 10.103.94.20、10.96.245.240、10.96.0.10 都是属于 service 网络的

```
Chain KUBE-SERVICES (2 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 KUBE-MARK-MASQ  tcp  --  *      *      !10.244.0.0/16        10.103.94.20         /* default/nginx: cluster IP */ tcp dpt:80
    0     0 KUBE-SVC-4N57TFCL4MD7ZTDA  tcp  --  *      *       0.0.0.0/0            10.103.94.20         /* default/nginx: cluster IP */ tcp dpt:80
    0     0 KUBE-MARK-MASQ  tcp  --  *      *      !10.244.0.0/16        10.96.245.240        /* default/myapp: cluster IP */ tcp dpt:80
    0     0 KUBE-SVC-NPJI2GAOYBRMPXVD  tcp  --  *      *       0.0.0.0/0            10.96.245.240        /* default/myapp: cluster IP */ tcp dpt:80
    0     0 KUBE-MARK-MASQ  tcp  --  *      *      !10.244.0.0/16        10.96.0.1            /* default/kubernetes:https cluster IP */ tcp dpt:443
    0     0 KUBE-SVC-NPX46M4PTMTKRN6Y  tcp  --  *      *       0.0.0.0/0            10.96.0.1            /* default/kubernetes:https cluster IP */ tcp dpt:443
    0     0 KUBE-MARK-MASQ  tcp  --  *      *      !10.244.0.0/16        10.96.0.10           /* kube-system/kube-dns:dns-tcp cluster IP */ tcp dpt:53
    0     0 KUBE-SVC-ERIFXISQEP7F7OF4  tcp  --  *      *       0.0.0.0/0            10.96.0.10           /* kube-system/kube-dns:dns-tcp cluster IP */ tcp dpt:53
    0     0 KUBE-MARK-MASQ  udp  --  *      *      !10.244.0.0/16        10.96.0.10           /* kube-system/kube-dns:dns cluster IP */ udp dpt:53
    0     0 KUBE-SVC-TCOU7JCQXEZGVUNU  udp  --  *      *       0.0.0.0/0            10.96.0.10           /* kube-system/kube-dns:dns cluster IP */ udp dpt:53
    0     0 KUBE-NODEPORTS  all  --  *      *       0.0.0.0/0            0.0.0.0/0            /* kubernetes service nodeports; NOTE: this must be the last rule in this
chain */ ADDRTYPE match dst-type LOCAL
```