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 搭建集群，实现相关功能。

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：10.96.0.0/12
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 节点: kubeadm init，集群初始化

3.nodes 节点: 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 节点: 安装 apiserver,scheduler,controller-manager,etcd,flanel

b).node 节点: 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，有一定难度。

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：所有节点

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
```

如果找不到 yum-config-manager 命令
执行 yum install yum-utils

从指定源安装 docker-ce ：

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

安装指定版本

```
yum install docker-ce-18.06.3.ce
yum install docker-ce-18.03.1.ce

systemctl start docker
systemctl enable docker
```

查看是否开机运行：

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

有时候需要执行如下命令：

```
cat >> /usr/lib/systemd/system/docker.service << EOF
LimitNOFILE=1048576
LimitNPROC=1048576
EOF
```

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

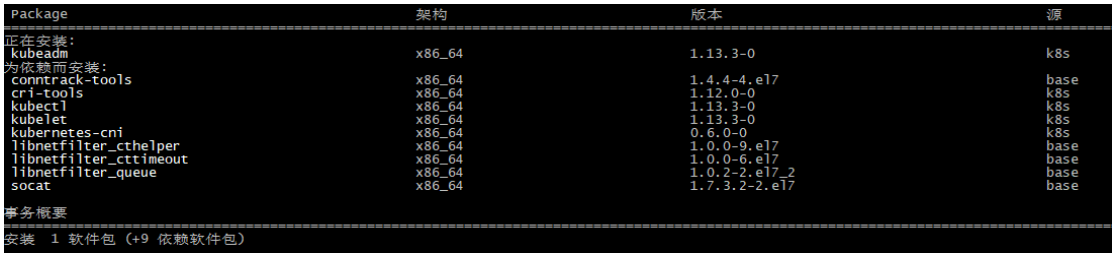
```
yum install kubeadm
```

指定版本安装

```
yum -y install kubectl-1.12.2  kubelet-1.12.2  kubeadm-1.12.2
```

```
[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



到这里可以查看都有哪些版本发布

https://github.com/kubernetes/kubernetes/releases

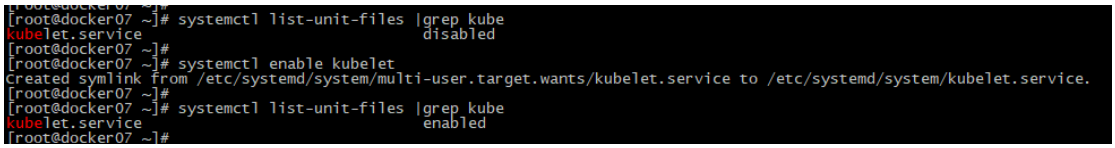
第四章 部署集群

操作对象：所有主机

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。

```
cat >> /etc/rc.local << EOF
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
EOF
```

或者

```
cat >> /usr/lib/sysctl.d/00-system.conf  << EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-arptables = 0
net.ipv4.ip_forward = 1
vm.swappiness=0
EOF
```

然后使其生效：

```
sysctl -p
systemctl restart network
```

如果要使用 ipvs，还要执行如下内容

```
cat > /etc/sysconfig/modules/ipvs.modules <<EOF
#!/bin/bash
```

```
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack_ipv4
EOF
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && lsmod | grep -e ip_vs -e nf_conntra
```

保证有如下内容，并生效

```
cat >> /etc/security/limits.conf << EOF
* soft noproc 65535
* hard noproc 65535
* soft nofile 65535
* hard nofile 65535
EOF
```

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/s&^&#&' /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。

4.1 启动 master 节点

4.2.1 所需的镜像

```
[root@docker10 ~]# kubeadm config images list
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
```

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
```

查到类似如下信息

```
[root@master ~]# kubeadm config images list
I0325 22:31:50.367151    1345 version.go:236] remote version is much newer: v1.13.4; falling back to: stable-1.12
k8s.gcr.io/kube-apiserver:v1.12.6
k8s.gcr.io/kube-controller-manager:v1.12.6
k8s.gcr.io/kube-scheduler:v1.12.6
k8s.gcr.io/kube-proxy:v1.12.6
k8s.gcr.io/pause:3.1
k8s.gcr.io/etcd:3.2.24
k8s.gcr.io/coredns:1.2.2
```

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

使用如下命令下载所需要的镜像，如果不下载，在 init 时自动下载

```
kubeadm config images pull --kubernetes-version=v1.12.2
```

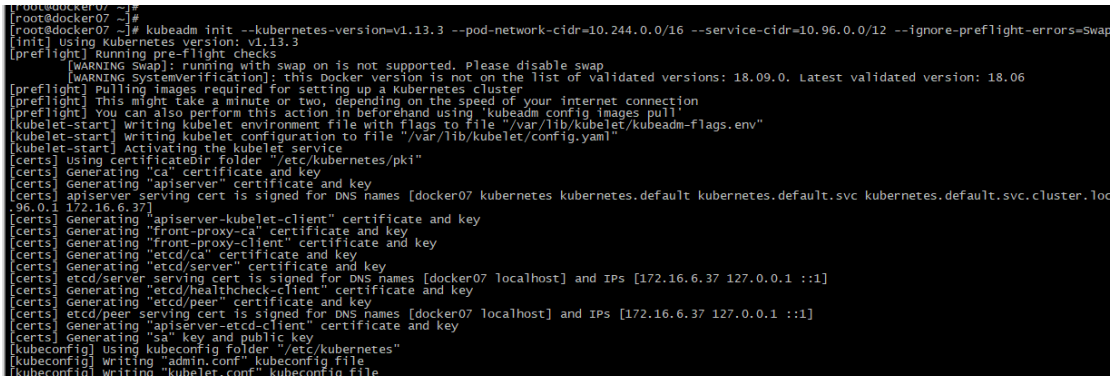
使用如下命令初始化集群

```
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
```

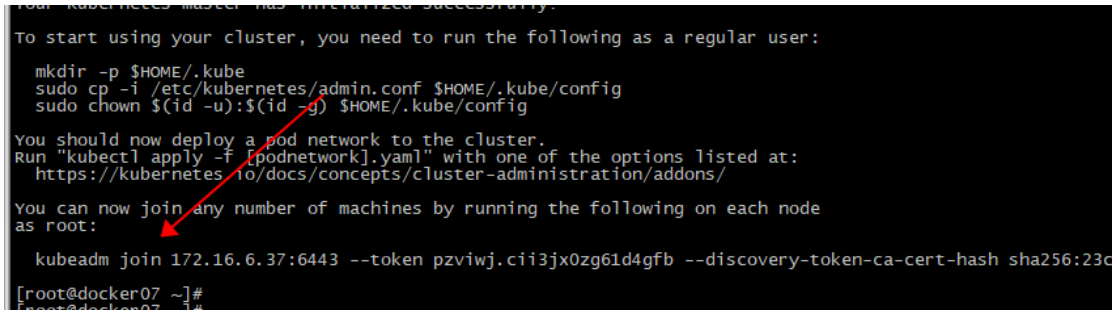
已经要进行 4.1 步骤，否则会有如下几个警告信息，

```
[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。



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
```

kubeadmin init 主要做了以下工作：

- [init]：指定版本进行初始化操作
- [preflight] ： 初始化前的检查和下载所需要的 Docker 镜像文件
- [kubelet-start] ：生成 kubelet 的配置文件”/var/lib/kubelet/config.yaml”，没有这个文件 kubelet 无法启动，所以初始化之前的 kubelet 实际上启动失败。
- [certificates]：生成 Kubernetes 使用的证书，存放在/etc/kubernetes/pki 目录中。
- [kubeconfig] ：生成 KubeConfig 文件，存放在/etc/kubernetes 目录中，组件之间通信需要使用对应文件。
- [control-plane]：使用/etc/kubernetes/manifest 目录下的 YAML 文件，安装 Master 组件。
- [etcd]：使用/etc/kubernetes/manifest/etcd.yaml 安装 Etcd 服务。
- [wait-control-plane]：等待 control-plan 部署的 Master 组件启动。
- [apiclient]：检查 Master 组件服务状态。
- [uploadconfig]：更新配置
- [kubelet]：使用 configMap 配置 kubelet。
- [patchnode]：更新 CNI 信息到 Node 上，通过注释的方式记录。
- [mark-control-plane]：为当前节点打标签，打了角色 Master，和不可调度标签，这样默认就不会使用 Master 节点来运行 Pod。
- [bootstrap-token]：生成 token 记录下来，后边使用 kubeadm join 往集群中添加节点时

会用到

- [addons]：安装附加组件 CoreDNS 和 kube-proxy

4.2.4 销毁集群

删除所有 worker 节点
停止 master 上的 kubelet 服务
rm /etc/kubernetes/ -rf
rm /var/lib/kubelet/ -rf
rm /var/lib/etcd/ -rf
删除所有 pod
可以重新初始化

4.2 启动 worker 节点

4.3.1 安装必要包

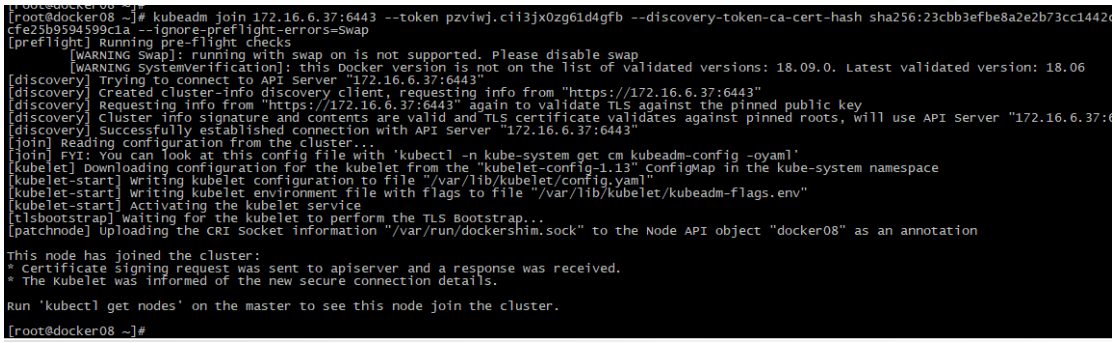
```
docker pull mirrorgooglecontainers/kube-proxy:v1.13.3
docker pull mirrorgooglecontainers/pause:3.1
docker tag mirrorgooglecontainers/kube-proxy:v1.13.3 k8s.gcr.io/kube-proxy:v1.13.3
docker tag mirrorgooglecontainers/pause:3.1 k8s.gcr.io/pause:3.1
```

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：



如果使用 ansible，可使用如下语句一次性加入

```
[root@jenkins ansible-playbook]#
[root@jenkins ansible-playbook]#
[root@jenkins ansible-playbook]#
[root@jenkins ansible-playbook]# ansible -i hosts k8s-node -m shell -a "kubeadm join 192.168.11.20:644
3 --token oh4fi8.1tnp7ow0p65hhnmv --discovery-token-ca-cert-hash sha256:c57eacc156efe97764dc546cbc8297
9185285c6eac32a2b3959b64d5d8348003"
192.168.11.21 | SUCCESS | rc=0 >>
[preflight] running pre-flight checks
[discovery] Trying to connect to API Server "192.168.11.20:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://192.168.11.20:6443"
```

ansible -i hosts k8s-node -m shell -a "kubeadm join 192.168.11.20:6443 --token oh4fi8.1tnp7ow0p65hhnmv --discovery-token-ca-cert-hash sha256:c57eacc156efe97764dc546cbc82979185285c6eac32a2b3959b64d5d8348003"

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.cii3jxOzg61d4gfb --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

另外，token 有效期 24 小时，如果过期使用如下命令重新生成

```
kubeadm token create --print-join-command
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.
```

也可使用以下命令查看已经颁发的 token

```
kubeadm token list
```

如果出现以下错误：
请检查网络是否同，node 和 master 时间是否一致
Failed to request cluster info, will try again: [Get https://192.168.11.20:6443/api/v1/namespaces/kube-public/configmaps/cluster-info: x509: certificate has expired or is not yet valid]

```
[root@node1321 ~]#
[root@node1321 ~]#
[root@node1321 ~]# kubeadm join 192.168.11.20:6443 --token i6qmpa.qxyfeotj3sboo491 --discovery-token-ca-cert-hash sha256:c57eacc156efe97764dc546cbc82979185285c6eac32a2b3959b64d5d8348003
[preflight] running pre-flight checks
        [WARNING RequiredIPVSKernelModulesAvailable]: the IPVS proxier will not be used, because the following required kernel modules are not loaded: [ip_vs ip_vs_rr ip_vs_wrr ip_vs_sh] or no builtin kernel ipvs support: map[ip_vs:{} ip_vs_rr:{} ip_vs_wrr:{} ip_vs_sh:{} nf_conntrack_ipv4:{}]
you can solve this problem with following methods:
 1. Run 'modprobe -- ' to load missing kernel modules;
2. Provide the missing builtin kernel ipvs support

[discovery] Trying to connect to API Server "192.168.11.20:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://192.168.11.20:6443"
[discovery] Requesting info from "https://192.168.11.20:6443" again to validate TLS against the pinned public key
[discovery] Failed to request cluster info, will try again: [Get https://192.168.11.20:6443/api/v1/namespaces/kube-public/configmaps/cluster-info: x509: certificate has expired or is not yet valid]
[discovery] Failed to request cluster info, will try again: [Get https://192.168.11.20:6443/api/v1/namespaces/kube-public/configmaps/cluster-info: x509: certificate has expired or is not yet valid]
[discovery] Failed to request cluster info, will try again: [Get https://192.168.11.20:6443/api/v1/namespaces/kube-public/configmaps/cluster-info: x509: certificate has expired or is not yet valid]
```

4.3.4 删除 worker 节点

```
kubectl delete node docker08
```

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

也可以在 worker 节点执行如下命令，直接离开集群

```
kubeadm reset
```

4.3.5 清除网络

当一个 node 从一个集群剔除，并开始加入另一个集群时，可能由于已有的网络，而报类似于如下的错误

```
t" network: failed to set bridge addr: "cni0" already has an IP address different from 10.244.2.1/24
```

此时需要重置 k8s、停止 docker/kubelet、删除配置、清除网络、重启 docker，重新加入集群

```
kubeadm reset

清空规则:
iptables -F && iptables -t nat -F && iptables -t mangle -F && iptables -X

systemctl stop kubelet
```

```
systemctl stop docker
rm -rf /etc/kubernetes/
rm -rf /var/lib/cni/
rm -rf /var/lib/kubelet/*
rm -rf /etc/cni/
ifconfig cni0 down
ifconfig flannel.1 down
ifconfig docker0 down
ip link delete cni0
ip link delete flannel.1
rm -rf /var/run/flannel
systemctl start docker
```

4.3 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
```

如果出现以下错误，需要开启代理端口：

```
[root@master ~]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
unable to recognize "https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml": Get http://localhost:8080/api?timeout=32s: dial tcp [::1]:8080: connect: connection refused
unable to recognize "https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml": Get http://localhost:8080/api?timeout=32s: dial tcp [::1]:8080: connect: connection refused
unable to recognize "https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml": Get http://localhost:8080/api?timeout=32s: dial tcp [::1]:8080: connect: connection refused
```

请先执行 5.1.1 的配置

4.4.2 node 节点（选做）

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

如果网络是通的，则不需要这一步，worker 节点会自动下载 flannel.

4.4 多 master 节点部署

4.5.1 规划

haproxy + keepalievd	192.168.11.19	192.168.12.19	192.168.13.19
k8s-master	192.168.11.20	192.168.12.20	192.168.13.20
k8s-worker	192.168.11.21	192.168.12.21	192.168.13.21
k8s-worker	192.168.11.22	192.168.12.22	192.168.13.22
k8s-worker	192.168.11.23	192.168.12.23	192.168.13.23

其中 VIP：192.168.11.3/24

4.5.2 keepalived 部署

在 192.168.11/12/13.19 上安装 keepalived

```
yum install keepalived
```

配置文件如下，三台中，不同的地方使用黄色背景标注
cat /etc/keepalived/keepalived.conf

```
! Configuration File for keepalived

global_defs {
    notification_email {
      acassen@firewall.loc
      failover@firewall.loc
      sysadmin@firewall.loc
    }
```

```
    notification_email_from Alexandre.Cassen@firewall.loc
    smtp_server 192.168.200.1
    smtp_connect_timeout 30
    router_id 1119
    vrrp_mcast_group4 224.0.100.100
}

vrrp_instance k8s_api_server {
    state MASTER
    interface eth0
    virtual_router_id 59
    priority 100
    advert_int 1

    authentication {
        auth_type PASS
        auth_pass 1111fdsfas
    }

    virtual_ipaddress {
        192.168.11.3/24 dev eth0 label eth0:0

    }
}
```

```
systemctl enable keepalived
systemctl start keepalived
```

4.5.3 haproxy 部署

在 192.168.11/12/13.19 上安装 haproxy

```
yum install haproxy
```

配置文件如下，三个节点配置完全相同

```
cat /etc/haproxy/haproxy.cfg
global
        chroot  /var/lib/haproxy
        daemon
        group haproxy
```

```
        user haproxy
        log 127.0.0.1 local2
        pidfile /var/lib/haproxy/haproxy.pid
        maxconn 20000
        spread-checks 3
        nbproc 8
        stats socket /var/lib/haproxy/stats

defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

frontend  k8s-api  *:6443
    mode                tcp
    default_backend     k8s-master
    option              tcplog

backend k8s-master
    mode        tcp
    balance     roundrobin
    option      tcplog
    server      master1120  192.168.11.20:6443 check maxconn 2000
    server      master1220  192.168.12.20:6443 check maxconn 2000
    server      master1320  192.168.13.20:6443 check maxconn 2000
```

```
systemctl enable haproxy
systemctl start haproxy
```

4.5.4 k8s 初始化

在 11.20 上，使用如下命令显示初始化配置并保存到文件中

```
kubeadm config print init-defaults > k8s-init.yaml
```

修改相关内容，其中红色部分是我修改过的。

```
apiVersion: kubeadm.k8s.io/v1beta1
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 192.168.11.20
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: master1120
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
---
apiServer:
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta1
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controlPlaneEndpoint: "192.168.11.3:6443"
controllerManager: {}
dns:
  type: CoreDNS
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: k8s.gcr.io
kind: ClusterConfiguration
kubernetesVersion: v1.13.5
networking:
```

```
  dnsDomain: cluster.local
  podSubnet: 10.244.0.0/16
  serviceSubnet: 10.96.0.0/12
scheduler: {}
```

如果要使用 ipvs 而不是 iptables，此文件尾部还应添加如下内容

```
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
mode: "ipvs"
```

使用如下命令初始化集群

```
kubeadm init --config k8s-init.yaml
```

4.5.5 添加 master

在 1120 上传递证书到其他 master

```
cat > k8s.sh << EOF
USER=root
IP="master1220 master1320"
for host in ${IP}; do
        ssh "${USER}"@$host "mkdir -p /etc/kubernetes/pki/etcd"
        scp /etc/kubernetes/pki/ca.* "${USER}"@$host:/etc/kubernetes/pki/
        scp /etc/kubernetes/pki/sa.* "${USER}"@$host:/etc/kubernetes/pki/
        scp /etc/kubernetes/pki/front-proxy-ca.* "${USER}"@$host:/etc/kubernetes/pki/
        scp /etc/kubernetes/pki/etcd/ca.* "${USER}"@$host:/etc/kubernetes/pki/etcd/
        scp /etc/kubernetes/admin.conf "${USER}"@$host:/etc/kubernetes/
done
EOF
```

sh k8s.sh

然后在 1220、1320 上节点上执行 join

```
kubeadm join 192.168.11.3:6443 --token abcdef.0123456789abcdef --discovery-token-ca-
cert-hash
sha256:ceef8a8b804a884fa16ab8e86c3eaa95aed01a95eae9285b68170bf56899f0d9 --
experimental-control-plane
```

注意最后的--experimental-control-plane 参数，普通 node 加入集群不需要这个参数。

参考: https://kubernetes.io/docs/setup/independent/high-availability/

另外，token 有效期 24 小时，如果过期使用如下命令重新生成

```
kubeadm token create --print-join-command
```

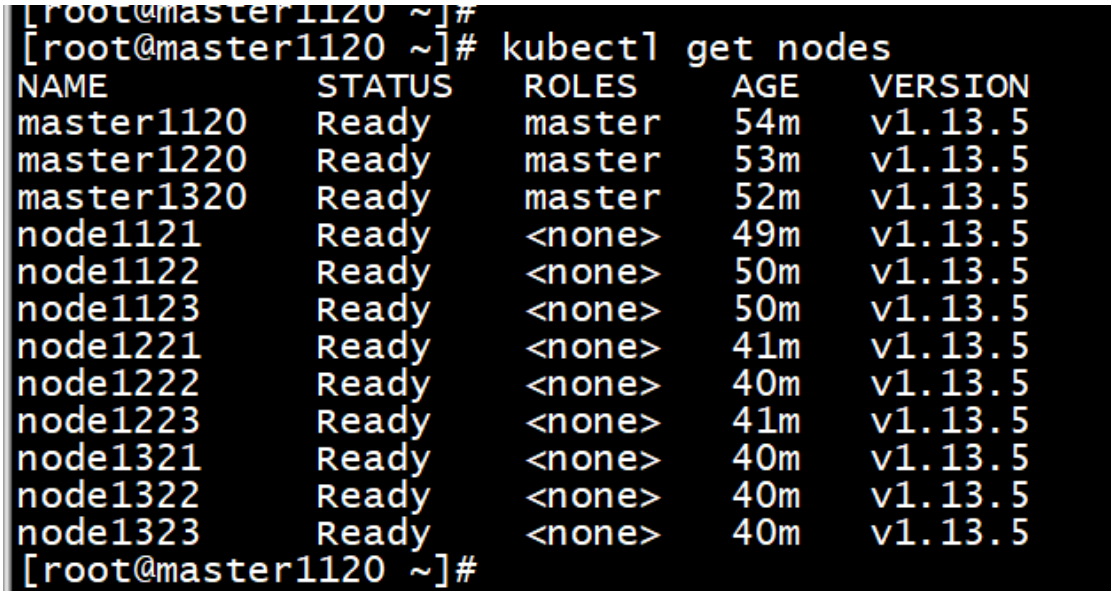
也可使用以下命令查看已经颁发的 token

```
kubeadm token list
```

4.5.6 添加 worker 节点

和单 master 一样，添加 worker 节点，并配置 flannel 网络。

4.5.7 观察



```
NAME                                READY   STATUS    RESTARTS   AGE
coredns-86c58d9df4-dwlgr            1/1     Running   0          54m
coredns-86c58d9df4-r8ggq            1/1     Running   0          54m
etcd-master1120                     1/1     Running   0          53m
etcd-master1220                     1/1     Running   0          53m
etcd-master1320                     1/1     Running   0          52m
kube-apiserver-master1120           1/1     Running   0          53m
kube-apiserver-master1220           1/1     Running   0          53m
kube-apiserver-master1320           1/1     Running   0          52m
kube-controller-manager-master1120  1/1     Running   1          53m
kube-controller-manager-master1220  1/1     Running   0          53m
kube-controller-manager-master1320  1/1     Running   0          52m
kube-proxy-x2jvq                    1/1     Running   0          40m
kube-scheduler-master1120           1/1     Running   1          53m
kube-scheduler-master1220           1/1     Running   0          53m
kube-scheduler-master1320           1/1     Running   0          52m
[root@master1120 ~]#
```

查看 etcd 健康状况：

```
docker run --rm -it \
--net host \
-v /etc/kubernetes:/etc/kubernetes quay.io/coreos/etcd etcdctl \
--cert-file /etc/kubernetes/pki/etcd/peer.crt \
--key-file /etc/kubernetes/pki/etcd/peer.key \
--ca-file /etc/kubernetes/pki/etcd/ca.crt \
--endpoints https://127.0.0.1:2379 cluster-health
```

```
[root@master1120 ~]#
[root@master1120 ~]# docker run --rm -it \
> --net host \
> -v /etc/kubernetes:/etc/kubernetes quay.io/coreos/etcd etcdctl \
> --cert-file /etc/kubernetes/pki/etcd/peer.crt \
> --key-file /etc/kubernetes/pki/etcd/peer.key \
> --ca-file /etc/kubernetes/pki/etcd/ca.crt \
> --endpoints https://127.0.0.1:2379 cluster-health
member 32225956760089 is healthy: got healthy result from https://192.168.12.20:2379
member 4ebbb444774b731c is healthy: got healthy result from https://192.168.11.20:2379
member aa54a44501678029 is healthy: got healthy result from https://192.168.13.20:2379
cluster is healthy
[root@master1120 ~]#
[root@master1120 ~]#
```

同时返回三个 etcd 集群的结果。

4.5.8 etcd 集群操作

查看集群成员：

docker run --rm -it --net host -v /etc/kubernetes:/etc/kubernetes quay.io/coreos/etcd etcdctl --cert-file /etc/kubernetes/pki/etcd/peer.crt --key-file /etc/kubernetes/pki/etcd/peer.key --ca-file /etc/kubernetes/pki/etcd/ca.crt --endpoints https://127.0.0.1:2379 member list

```
[root@master1220 ~]#
[root@master1220 ~]# docker run --rm -it --net host -v /etc/kubernetes:/etc/kubernetes quay.io/coreos/etcd etcdctl --cert-file
cd/peer.key --ca-file /etc/kubernetes/pki/etcd/ca.crt --endpoints https://127.0.0.1:2379 member list
32225956760089: name=master1220 peerURLs=https://192.168.12.20:2380 clientURLs=https://192.168.12.20:2379 isLeader=true
4ebbb444774b731c: name=master1120 peerURLs=https://192.168.11.20:2380 clientURLs=https://192.168.11.20:2379 isLeader=false
aa54a44501678029: name=master1320 peerURLs=https://192.168.13.20:2380 clientURLs=https://192.168.13.20:2379 isLeader=false
[root@master1220 ~]#
```

删除集群成员：

member remove {ID}

```
[root@master1220 ~]#
[root@master1220 ~]# docker run --rm -it --net host -v /etc/kubernetes:/etc/kubernetes quay.io/coreos/etcd etcdctl --cert-file /etc/kubernet
cd/peer.key --ca-file /etc/kubernetes/pki/etcd/ca.crt --endpoints https://127.0.0.1:2379 member remove 4ebbb444774b731c
Removed member 4ebbb444774b731c from cluster
[root@master1220 ~]#
[root@master1220 ~]# docker run --rm -it --net host -v /etc/kubernetes:/etc/kubernetes quay.io/coreos/etcd etcdctl --cert-file /etc/kubernet
cd/peer.key --ca-file /etc/kubernetes/pki/etcd/ca.crt --endpoints https://127.0.0.1:2379 member list
322259567600089: name=master1220 peerURLs=https://192.168.12.20:2380 clientURLs=https://192.168.12.20:2379 isLeader=true
aa54a44501678029: name=master1320 peerURLs=https://192.168.13.20:2380 clientURLs=https://192.168.13.20:2379 isLeader=false
[root@master1220 ~]#
[root@master1220 ~]#
```

添加成员

member add master1120 https://192.168.11.20:2380

```
[root@master1220 manifests]#
[root@master1220 manifests]# docker run --rm -it --net host -v /etc/kubernetes:/etc/kubernetes quay.io/coreos/etcd etcdctl --cert-file /etc/kubernete
s/pki/etcd/peer.key --ca-file /etc/kubernetes/pki/etcd/ca.crt --endpoints https://127.0.0.1:2379 member add master1120 https://192.168.11.20:2380
Added member named master1120 with ID ba63f35307d0914b to cluster

ETCD_NAME="master1120"
ETCD_INITIAL_CLUSTER="master1220=https://192.168.12.20:2380,master1320=https://192.168.13.20:2380,master1120=https://192.168.11.20:2380"
ETCD_INITIAL_CLUSTER_STATE="existing"
[root@master1220 manifests]#
```

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

我这里 k8s 使用 1.13.3 版本，太新了 ，dashboard 不支持。这一部分先不操作

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

使用如下内容的 yaml 部署（访问方式：http://***:32001）

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

apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system


---
# ------------------- ClusterRole and ClusterRoleBinding ------------------- #
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

```
metadata:
  name: system:kubernetes-dashboard
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/cluster-service: "true"
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kubernetes-dashboard-cluster-role
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/cluster-service: "true"
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:kubernetes-dashboard
subjects:
- kind: ServiceAccount
  name: kubernetes-dashboard
  namespace: kube-system

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

kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
```

```
        k8s-app: kubernetes-dashboard
    template:
      metadata:
        labels:
          k8s-app: kubernetes-dashboard
      spec:
        containers:
        - name: kubernetes-dashboard
          image: k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1
          ports:
          - containerPort: 9090
            protocol: TCP
          volumeMounts:
            # Create on-disk volume to store exec logs
          - mountPath: /tmp
            name: tmp-volume
          livenessProbe:
            httpGet:
              path: /
              port: 9090
            initialDelaySeconds: 30
            timeoutSeconds: 30
        volumes:
        - 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: 80
    targetPort: 9090
    nodePort: 32001
  selector:
    k8s-app: kubernetes-dashboard
```

第五章 常用命令

5.1 get

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

5.1.1 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 ~]#
```

5.1.2 get cs

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 ~]#
```

5.1.3 get node

查看节点

```
[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 ~]#
```

显示标签

```
kubectl get nodes --show-labels
```

```
[root@node3 ~]#
[root@node3 ~]# kubectl get nodes --show-labels
NAME      STATUS   ROLES    AGE    VERSION   LABELS
node1     Ready    <none>   129d   v1.12.2   beta.kubernetes.
node100   Ready    <none>   101d   v1.12.2   beta.kubernetes.
node2     Ready    <none>   129d   v1.12.2   beta.kubernetes.
node3     Ready    master   129d   v1.12.2   beta.kubernetes.
node4     Ready    <none>   129d   v1.12.2   beta.kubernetes.
[root@node3 ~]#
[root@node3 ~]#
[root@node3 ~]#
```

查看各个节点的 pod 网络

kubectl get node "-o=custom-columns=NAME:.metadata.name,podCIDR:spec.podCIDR"

```
[root@master1120 xmc-model-serving]# kubectl get node "-o=custom-columns=NAME:.metadata.name,podCIDR:spec.podCIDR"
NAME        podCIDR
gpu0312     10.244.11.0/24
gpu6802     10.244.4.0/24
master1120  10.244.3.0/24
master1220  10.244.1.0/24
master1320  10.244.2.0/24
node1121    10.244.20.0/24
node1122    10.244.5.0/24
node1123    10.244.16.0/24
node1221    10.244.15.0/24
node1222    10.244.17.0/24
node1223    10.244.13.0/24
node1321    10.244.19.0/24
node1322    10.244.6.0/24
node1323    10.244.18.0/24
```

5.1.4 get ns

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

5.1.5 get deploy

查看资源

```
kubectl get deployments
kubectl get deploy
```

```
[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 ~]#
```

5.1.6 get svc

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 ~]#
```

5.1.7 get endpoints

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

5.1.8 get pod

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。

5.1.9 get label

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 ~]#
```

-L app　显示 app 的标签值

-l app,abc

-l app=myapp

-l app=myapp，abc=bcd

-l app!=xmc-backend-service

-l “app in (xmc-backend-service,bcd)”

使用-l 筛选

```
[root@master xmc]#
[root@master xmc]# kubectl get pod -n xmc
NAME                                      READY   STATUS    RESTARTS   AGE
dm-model-serving-student-6775fcc894-sxfkb 0/1     Pending   0          4h38m
redis-8474575d46-hb9wr                    1/1     Running   0          4d3h
xmc-backend-service-5bb6dd5f7f-mzsk5      1/1     Running   0          4d4h
xmc-data-collector-76788d79c7-ln5tl       1/1     Running   0          4d4h
xmc-data-stream-6576659869-jbnsh          1/1     Running   0          22m
xmc-frontend-7f4bf7d87-f2xr9              1/1     Running   0          4d3h
xmc-metric-generator-5768d98655-cp87z     1/1     Running   0          4h44m
xmc-storage-service-74fdff799f-nmtjk      1/1     Running   0          4d4h
[root@master xmc]#
[root@master xmc]# kubectl get pod -n xmc -L app
NAME                                      READY   STATUS    RESTARTS   AGE     APP
dm-model-serving-student-6775fcc894-sxfkb 0/1     Pending   0          4h38m   dm-model-serving-student
redis-8474575d46-hb9wr                    1/1     Running   0          4d3h    redis
xmc-backend-service-5bb6dd5f7f-mzsk5      1/1     Running   0          4d4h    xmc-backend-service
xmc-data-collector-76788d79c7-ln5tl       1/1     Running   0          4d4h    xmc-data-collector
xmc-data-stream-6576659869-jbnsh          1/1     Running   0          22m     xmc-data-stream
xmc-frontend-7f4bf7d87-f2xr9              1/1     Running   0          4d3h    xmc-frontend
xmc-metric-generator-5768d98655-cp87z     1/1     Running   0          4h44m   xmc-metric-generator
xmc-storage-service-74fdff799f-nmtjk      1/1     Running   0          4d4h    xmc-storage-service
[root@master xmc]#
[root@master xmc]#
[root@master xmc]#
[root@master xmc]# kubectl get pod -n xmc -l app=xmc-backend-service
NAME                                 READY   STATUS    RESTARTS   AGE
xmc-backend-service-5bb6dd5f7f-mzsk5 1/1     Running   0          4d4h
[root@master xmc]#
[root@master xmc]# kubectl get pod -n xmc -l app=xmc-backend-service --show-labels
NAME                                 READY   STATUS    RESTARTS   AGE     LABELS
xmc-backend-service-5bb6dd5f7f-mzsk5 1/1     Running   0          4d4h    app=xmc-backend-service,pod-template-hash=5bb6dd5f7f
[root@master xmc]#
[root@master xmc]#
```

```
[root@master xmc]#
[root@master xmc]# kubectl get pod -n xmc -l "app in (redis,bdd)"
NAME                     READY   STATUS    RESTARTS   AGE
redis-8474575d46-hb9wr   1/1     Running   0          4d3h
[root@master xmc]#
[root@master xmc]#
[root@master xmc]# kubectl get pod -n xmc -l "app notin (redis,bdd)"
NAME                                        READY   STATUS    RESTARTS   AGE
dm-model-serving-student-6775fcc894-sxfkb   0/1     Pending   0          4h48m
xmc-backend-service-5bb6dd5f7f-mzsk5        1/1     Running   0          4d4h
xmc-data-collector-76788d79c7-ln5tl         1/1     Running   0          4d4h
xmc-data-stream-6576659869-jbnsh            1/1     Running   0          32m
xmc-frontend-7f4bf7d87-f2xr9                1/1     Running   0          4d3h
xmc-metric-generator-5768d98655-cp87z       1/1     Running   0          4h54m
xmc-storage-service-74fdff799f-nmtjk        1/1     Running   0          4d4h
[root@master xmc]#
[root@master xmc]#
```

5.2 describe

查看相信信息

5.2.1 describe deploy

```
kubectl describe deployments couchbase-server
```

5.2.2 describe node

```
kubectl describe node node2
```

5.2.3 describe ns

```
kubectl describe ns x2
```

5.3 logs

```
kubectl logs myapp -n quzl
```

如果一个 pod 中有多个容器，需要指明需要查看哪个容器，如下查看 busybox 容器日志

```
kubectl logs myapp –c myapp -n quzl
```

```
[root@master k8s-test]#
[root@master k8s-test]# kubectl logs myapp -c busybox -n quzl
[root@master k8s-test]#
[root@master k8s-test]# kubectl logs myapp  -n quzl
Error from server (BadRequest): a container name must be specified for pod myapp, choose one of: [myapp busybox]
[root@master k8s-test]#
[root@master k8s-test]#
[root@master k8s-test]# kubectl logs myapp -c myapp -n quzl
10.244.0.0 - - [31/Mar/2019:20:58:43 +0000] "GET / HTTP/1.1" 200 87 "-" "curl/7.29.0" "-"
10.244.0.0 - - [31/Mar/2019:20:58:50 +0000] "GET / HTTP/1.1" 200 116 "-" "curl/7.29.0" "-"
10.244.0.0 - - [31/Mar/2019:20:58:52 +0000] "GET / HTTP/1.1" 200 116 "-" "curl/7.29.0" "-"
10.244.0.0 - - [31/Mar/2019:20:58:53 +0000] "GET / HTTP/1.1" 200 116 "-" "curl/7.29.0" "-"
10.244.0.0 - - [31/Mar/2019:20:58:54 +0000] "GET / HTTP/1.1" 200 116 "-" "curl/7.29.0" "-"
[root@master k8s-test]#
```

5.4 rollout

回滚

查看版本历史，序号越大，越新

```
kubectl rollout history deployment xmc-backend-service -n xmc
```

回滚，默认回滚到上一个版本

```
kubectl rollout undo deployment xmc-backend-service -n xmc
```

指定版本回滚

```
kubectl rollout undo deployment xmc-backend-service --to-revision=1 -n xmc
```

第六章 测试

6.1 创建集群实例-nginx

6.1.1 创建 nginx 的 pod

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

```
[root@docker07 ~]# kubectl get pod
No resources found.
[root@docker07 ~]#
[root@docker07 ~]#
[root@docker07 ~]# kubectl run nginx --image=nginx --replicas=3
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future v
deployment.apps/nginx created
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pod
NAME                     READY   STATUS              RESTARTS   AGE
nginx-7cdbd8cdc9-6bxcj   0/1     ContainerCreating   0          5s
nginx-7cdbd8cdc9-csb95   0/1     ContainerCreating   0          5s
nginx-7cdbd8cdc9-qc5vh   0/1     ContainerCreating   0          5s
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pod
NAME                     READY   STATUS              RESTARTS   AGE
nginx-7cdbd8cdc9-6bxcj   0/1     ContainerCreating   0          15s
nginx-7cdbd8cdc9-csb95   0/1     ContainerCreating   0          15s
nginx-7cdbd8cdc9-qc5vh   0/1     ContainerCreating   0          15s
[root@docker07 ~]# kubectl get pod
NAME                     READY   STATUS              RESTARTS   AGE
nginx-7cdbd8cdc9-6bxcj   0/1     ContainerCreating   0          103s
nginx-7cdbd8cdc9-csb95   0/1     ContainerCreating   0          103s
nginx-7cdbd8cdc9-qc5vh   0/1     ContainerCreating   0          103s
[root@docker07 ~]#
[root@docker07 ~]# kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
nginx-7cdbd8cdc9-6bxcj   1/1     Running   0          3m40s
nginx-7cdbd8cdc9-csb95   1/1     Running   0          3m40s
nginx-7cdbd8cdc9-qc5vh   1/1     Running   0          3m40s
```

使用-o wide 参数可看到更加详细的信息

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

```
[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          8m19s   10.244.3.2   docker10   <none>           <none>
nginx-7cdbd8cdc9-csb95   1/1     Running   0          8m19s   10.244.2.2   docker09   <none>           <none>
nginx-7cdbd8cdc9-qc5vh   1/1     Running   0          8m19s   10.244.1.2   docker08   <none>           <none>
[root@docker07 ~]#
```

从图中我们可以看到，每个 node 节点运行了一个 pod

6.1.2 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

6.1.3 访问 pod

使用以下命令

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

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

6.2 创建另一个实例-myapp

6.2.1 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 ~]#
```

6.2.2 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 ~]#
```

6.2.3 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 ~]#
```

6.2.4 删除 pod

kubectl delete pod 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 ~]#
```

6.2.5 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 ~]#
```

6.2.6 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

6.2.7 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，更新版本：

```
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>
```

6.2.8 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 ~]#
[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 ~]#
```

```
[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>
^C
```

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。

6.2.9 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
```

6.2.10 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
```

第七章 资源清单

7.1 资源

7.1.1 资源对象

k8s 中所有的内容都抽象为资源， 资源实例化之后，叫做对象。

类别	名称
工作负载型资源对象	Pod、 Replicaset 、ReplicationController 、Deployments 、StatefulSets 、Daemonset、 Job、CronJob
服务发现及负载均衡	Service、 Ingress
配置与存储	Volume、Persistent Volume、CSI 、 configmap、 secret
集群资源	Namespace 、Node 、Role ClusterRole 、RoleBinding 、ClusterRoleBinding
元数据资源	HPA、PodTemplate、LimitRang

查看有哪些资源

```
kubectl api-resources
```

7.1.2 资源版本

一共有 5 个一级字段。

1) apiVersion （group/version）

查看哪些

```
kubectl api-versions
```

```
[root@master ~]#
[root@master ~]# kubectl api-versions
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
apps/v1
apps/v1beta1
apps/v1beta2
authentication.k8s.io/v1
authentication.k8s.io/v1beta1
authorization.k8s.io/v1
authorization.k8s.io/v1beta1
autoscaling/v1
autoscaling/v2beta1
autoscaling/v2beta2
batch/v1
batch/v1beta1
certificates.k8s.io/v1beta1
coordination.k8s.io/v1beta1
events.k8s.io/v1beta1
extensions/v1beta1
networking.k8s.io/v1
policy/v1beta1
rbac.authorization.k8s.io/v1
rbac.authorization.k8s.io/v1beta1
scheduling.k8s.io/v1beta1
storage.k8s.io/v1
storage.k8s.io/v1beta1
v1
[root@master ~]#
```

2) kind
3) Metadata

   name

   namespace

   labels

   annotations

   资源引用：/api/GROUP/VERSION/namespace/NAMESPACE/TYPE

4) Spec

   期望的状态，disired state

5) status

7.1.3 资源限制

如下，使用 resource 字段

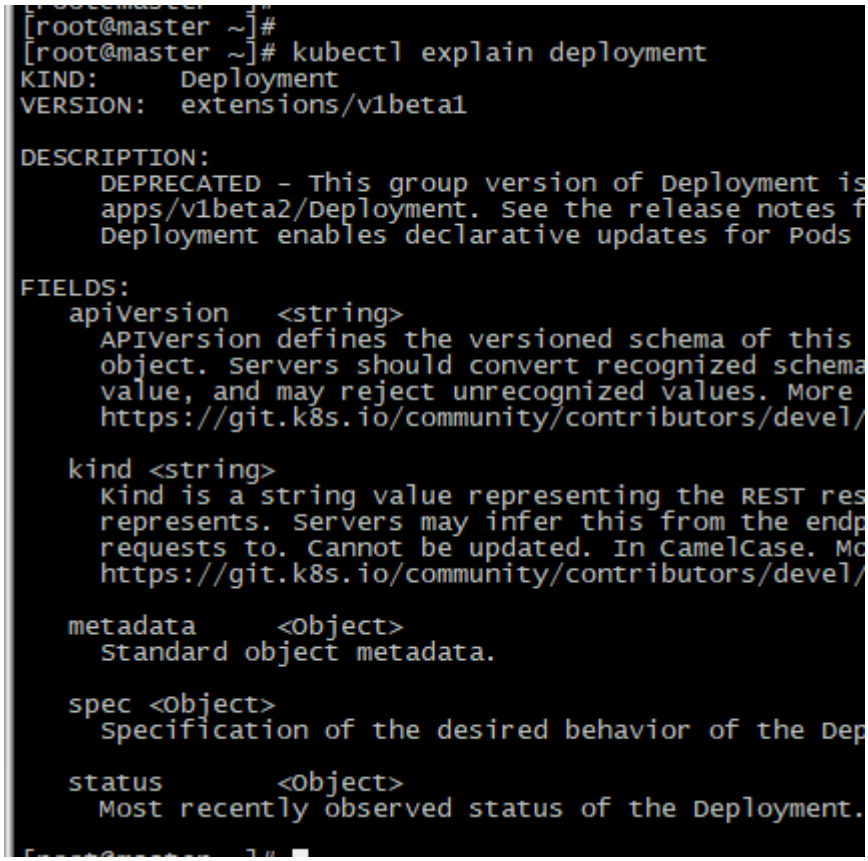
```
apiVersion: v1
kind: ReplicationController
metadata:
   name: redis
spec:
    replicas: 1
    selector:
```

```
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: redis
        resources:
          requests:
            cpu: 100m                #限制 cpu 的数量为 0.1 个，1000m=1 个
            memory: 100Mi            #限制内存为 100M
        ports:
        - containerPort: 6379
```

7.1.4 查看定义的方式

```
kubectl explain svc
kubectl explain pod.metadate.namespace
kubectl explain svc.spec.type
```

如下，查看 deployment



会有多种字段类型:

1）<string>，表示字符串

2）<Object>，表示对象，比如

deployment.metadata

name: mongodb

namespace: x2

3）<[]Object>，表示对象列表，比如

pod.spec.containers

- name: myapp

image: ikuberneters/myaqpp:v1

- name: busybox

image: busybox

4）<[]string>，表示字符串列表，比如

pod.spec.containers.command

command:

- "/bin/sh"

- "-c"

- " ***"

5）<boolean>，表示布尔类型，比如

6）<integer>，整型，比如

deployment.spec.replicas：4

7）<map[string]string>，字符类型的 map，比如

pod.metadata.labels

labels:
  app: redis
  abc: bcd

7.1.5 查看是否属于 namespace

```
# In a namespace
kubectl api-resources --namespaced=true

# Not in a namespace
kubectl api-resources --namespaced=false
```

7.2 coredns

资源记录格式:
    SVC_NAME.NS_NAME.DOMAIN.LTD.
而 k8s 默认的资源后缀是: svc.cluster.local.
例如一个名字叫做 redis 的 service 的资源记录是: Redis.default.svc.cluster.local.

7.3 NameSpace 资源

7.4.1 创建 ns

方法一: 命令行创建

```
kubectl create ns foo
```

方法二: 使用 yaml 文件创建

```
apiVersion: v1
kind: Namespace
metadata:
    name: x2
    labels:
        name: x2
```

7.4.2 删除 ns

命令行删除

```
kubectl create ns foo
```

7.4 pod 资源

7.4.1 资源定义的完整格式

```
apiVersion: v1                          #必选，版本号，例如 v1,版本号必须可以用
kubectl api-versions 查询到 .
kind: Pod                               #必选，Pod
metadata:                               #必选，元数据
  name: string                          #必选，Pod 名称
  namespace: string                     #必选，Pod 所属的命名空间,默认为"default"
  labels:                               #自定义标签
    - name: string                      #自定义标签名字
  annotations:                          #为对象提供“元数据”，不能用于挑选资源对象，在某些场景下，必须添加
    - name: string
spec:                                   #必选，Pod 中容器的详细定义
  containers:                           #必选，Pod 中容器列表
  - name: string                        #必选，容器名称,需符合 RFC 1035 规范
    image: string                       #必选，容器的镜像名称
    imagePullPolicy: [ Always|Never|IfNotPresent ]  #获取镜像的策略 Alawys 表示下载镜像 IfnotPresent 表示优先使用本地镜像,否则下载镜像，Nerver 表示仅使用本地镜像
拉取最新 latest 镜像：默认是 always
拉取的不是 latest 镜像：默认是 IfnotPresent
    command: [string]                   #容器的启动命令列表，如不指定，使用打包时使用的启动命令
    args: [string]                      #容器的启动命令参数列表
#和 dockerfile 里面的 cmd、entpoints 有关，具体如何生效，见:
https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/
    workingDir: string                  #容器的工作目录
    volumeMounts:                       #挂载到容器内部的存储卷配置
    - name: string                      #引用 pod 定义的共享存储卷的名称，需用 volumes[]部分定义的的卷名
      mountPath: string                 #存储卷在容器内 mount 的绝对路径，应少于 512 字符
      readOnly: boolean                 #是否为只读模式
    ports:                              #需要暴露的端口库号列表
    - name: string                      #端口的名称
      containerPort: int                #容器需要监听的端口号
      hostPort: int                     #容器所在主机需要监听的端口号，默认与 Container 相同
      protocol: string                  #端口协议，支持 TCP 和 UDP，默认 TCP
    env:                                #容器运行前需设置的环境变量列表
    - name: string                      #环境变量名称
      value: string                     #环境变量的值
    resources:                          #资源限制和请求的设置
```

```
        limits:                                  #资源限制的设置
          cpu: string                            #Cpu 的限制，单位为 core 数，将用于 docker
run --cpu-shares 参数
          memory: string                          #内存限制，单位可以为 Mib/Gib，将用于
docker run --memory 参数
        requests:                                  #资源请求的设置
          cpu: string                            #Cpu 请求，容器启动的初始可用数量
          memory: string                           #内存请求,容器启动的初始可用数量
      livenessProbe:                             #对 Pod 内各容器健康检查的设置，当探测无
响应几次后将自动重启该容器，检查方法有 exec、httpGet 和 tcpSocket，对一个容器只需
设置其中一种方法即可
        exec:                                    #对 Pod 容器内检查方式设置为 exec 方式
          command: [string]                       #exec 方式需要制定的命令或脚本
        httpGet:                                 #对 Pod 内个容器健康检查方法设置为
HttpGet，需要制定 Path、port
          path: string
          port: number
          host: string
          scheme: string
          HttpHeaders:
          - name: string
            value: string
        tcpSocket:                           #对 Pod 内个容器健康检查方式设置为 tcpSocket 方
式
          port: number
        initialDelaySeconds: 0          #容器启动完成后首次探测的时间，单位为秒
        timeoutSeconds: 0                  #对容器健康检查探测等待响应的超时时间，单位
秒，默认 1 秒
        periodSeconds: 0                   #对容器监控检查的定期探测时间设置，单位秒，
默认 10 秒一次
        successThreshold: 0
        failureThreshold: 0
        securityContext:
          privileged: false
    restartPolicy: [Always | Never | OnFailure] #Pod 的重启策略，Always 表示一旦不管以何
种方式终止运行，kubelet 都将重启，OnFailure 表示只有 Pod 以非 0 退出码退出才重启，
Nerver 表示不再重启该 Pod
    nodeSelector: obeject                  #设置 NodeSelector 表示将该 Pod 调度到包含这个
label 的 node 上，以 key：value 的格式指定
    imagePullSecrets:                    #Pull 镜像时使用的 secret 名称，以 key：secretkey 格
式指定
    - name: string
    hostNetwork: false                     #是否使用主机网络模式，默认为 false，如果设置
为 true，表示使用宿主机网络
```

```
    volumes:                          #在该 pod 上定义共享存储卷列表
    - name: string                    #共享存储卷名称 （volumes 类型有很多种）
      emptyDir: {}                    #类型为 emtyDir 的存储卷，与 Pod 同生命周期的
一个临时目录。为空值
      hostPath: string                #类型为 hostPath 的存储卷，表示挂载 Pod 所在宿
主机的目录
        path: string                  #Pod 所在宿主机的目录，将被用于同期中 mount 的目录
      secret:                         #类型为 secret 的存储卷，挂载集群与定义的 secre
对象到容器内部
        scretname: string
        items:
        - key: string
          path: string
      configMap:                      #类型为 configMap 的存储卷，挂载预定义
的 configMap 对象到容器内部
        name: string
        items:
        - key: string
          path: string
```

7.4.2 一个 pod 包含一个容器

一般很少直接创建 pod，而是通过 pod 控制器创建的，所以这里举例使用 deployment 创建的 pod

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: quzl
spec:
  type: ClusterIP
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: nginx
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx
  namespace: quzl
spec:
   replicas: 2
   template:
```

```
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
        volumeMounts:
        - name: web-base
          mountPath: /usr/share/nginx/html
      volumes:
      - name: web-base
        hostPath:
          path: /data/nginx-html
```

7.4.3 一个 pod 包含多个容器

同时，这里还举例了共用存储卷。需要说明的是，虽然多个容器都位于同一个 pod，但是其文件系统并不是相同的。

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  namespace: quzl
spec:
  containers:
  - name: myapp
    image: ikubernetes/myapp:v1
    volumeMounts:
    - name: www-data
      mountPath: /usr/share/nginx/html
  - name: busybox
    image: busybox
    volumeMounts:
    - name: www-data
      mountPath: /data/www
    command:
    - "/bin/sh"
    - "-c"
    - "while true;do echo $(date) >> /data/www/index.html;sleep 10 ;done"
  volumes:
  - name: www-data
```

```
    hostPath:
      path: /data/nginx-html
```

另外，对于单个 pod 包含多个容器的情况，当使用 log 命令时，需要使用-c 指明要查看的是哪个容器

```
root@master k8s-test]#
root@master k8s-test]# kubectl logs myapp -c busybox -n quzl
root@master k8s-test]#
root@master k8s-test]# kubectl logs myapp  -n quzl
rror from server (BadRequest): a container name must be specified for pod myapp, choose one of: [myapp busybox]
root@master k8s-test]#
root@master k8s-test]#
root@master k8s-test]# kubectl logs myapp -c myapp -n quzl
0.244.0.0 - - [31/Mar/2019:20:58:43 +0000] "GET / HTTP/1.1" 200 87 "-" "curl/7.29.0" "-"
0.244.0.0 - - [31/Mar/2019:20:58:50 +0000] "GET / HTTP/1.1" 200 116 "-" "curl/7.29.0" "-"
0.244.0.0 - - [31/Mar/2019:20:58:52 +0000] "GET / HTTP/1.1" 200 116 "-" "curl/7.29.0" "-"
0.244.0.0 - - [31/Mar/2019:20:58:53 +0000] "GET / HTTP/1.1" 200 116 "-" "curl/7.29.0" "-"
0.244.0.0 - - [31/Mar/2019:20:58:54 +0000] "GET / HTTP/1.1" 200 116 "-" "curl/7.29.0" "-"
root@master k8s-test]#
```

7.5 Service 资源

7.5.1 Service 的实现技术

userspace: 1.1 之前使用，效率很低

iptables: 1.10 之后加入

ipvs: 1.11 之后加入

启用 ipvs 方法:

1） 安装 kubelet 后，配置文件中指定使用 ipvs

```
echo "KUBE_PROXY_MODE=ipvs" >> /etc/sysconfig/kubelet
```

2） 保证开机时自动加载如下模块

ip_vs,ip_vs_rr,ip_vs_wrr,ip_vs_sh,nf_conntrack_ipv4

如果未使用 ipvs，使用 join 加入集群时，会有以下警告。

```
[root@node1323 ~]# kubeadm join 192.168.11.20:6443 --token 6dmsrk.s6vijhgnfh120j2h --discovery-token-c
a-cert-hash sha256:cbff9300b8b76710d35d00954cd0ee36ec73fc4f26afa492cce38feb7fe29664
[preflight] running pre-flight checks
        [WARNING RequiredIPVSKernelModulesAvailable]: the IPVS proxier will not be used, because the f
ollowing required kernel modules are not loaded: [ip_vs_wrr ip_vs_sh ip_vs ip_vs_rr] or no builtin ker
nel ipvs support: map[ip_vs_sh:{} nf_conntrack_ipv4:{} ip_vs:{} ip_vs_rr:{} ip_vs_wrr:{}]
you can solve this problem with following methods:
 1. Run 'modprobe -- ' to load missing kernel modules;
2. Provide the missing builtin kernel ipvs support

[discovery] Trying to connect to API Server "192.168.11.20:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://192.168.11.20:6443"
[discovery] Requesting info from "https://192.168.11.20:6443" again to validate TLS against the pinned
 public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned
```

[WARNING RequiredIPVSKernelModulesAvailable]: the IPVS proxier will not be used, because the following required kernel modules are not loaded: [ip_vs_wrr ip_vs_sh ip_vs ip_vs_rr] or no builtin kernel ipvs support: map[ip_vs_sh:{} nf_conntrack_ipv4:{} ip_vs:{} ip_vs_rr:{} ip_vs_wrr:{}]

you can solve this problem with following methods:

1. Run 'modprobe -- ' to load missing kernel modules;
2. Provide the missing builtin kernel ipvs support

7.5.2 service 类型

1) ExternalName
用于 k8s 集群内部的 pod 访问集群外部的资源这种情况时。这里不再举例。

可以立即为一种增强的 ClunterIP 类型。

2) ClusterIP
举例：

```
apiVersion: v1
kind: Service
metadata:
    name: redis
    namespace: default
spec:
    type: ClusterIP
    ports:
    - port: 6379
      targetPort: 6379
    selector:
      app: redis
      role: logstor
     #sessionAffinity: None #默认
     #sessionAffinity: ClientIP #启用会话保持，类似于 nginx 的 ip hash
```

3) NodePort
举例：

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort //指定 service 类型
  selector:
    app: forme
  ports:
    - port: 80 // 供集群中其它 container 访问端口
      targetPort: 8020 //转向后端 pod 中 container 暴露的端口
      nodePort: 9090 //节点暴露的端口
```

默认的，nodePort 的范围是 30000-32767, k8s 会从中随机选择一个端口，可以通过修改 apiserver 的--service-node-port-range 的参数来修改默认范围，如：--service-node-port-range 8000-9000。

访问路径:
clinent --> NodeIP:NodePort --> ClusterIP:ClusterPort --> PodIP:PodPort

如果单独访问一个 node,该 node 可能压力过大，所以可以使用 LocadBalance 方式

4) LoadBalancer
用于创建 ipvs 规则，供外部的负载均衡器再次调取，比如阿里云的 slb，自动修改 slb 规则。

可以理解为一种增强的 NodePort 类型。

7.5.3 一个特殊的 service 类型
无头 service,无 IP 的 cluserip 类型。此时流程是 pod-->service name --> pod name

定义时：ClusterIP: "None "

7.6 ingress

ingress 也是一种标准的 k8s 资源，用于解决普通 service 不能七层调度的问题。

7.7 ConfigMap

7.7.1 创建

1）使用字符串创建

```
kubectl create configmap nginx-config --from-literal=nginx_port=80 --from-literal=srver_name=www.quzl.com -n quzl
```

查看:

```
[root@master k8s-test]#
[root@master k8s-test]# kubectl get configmap nginx-config -n quzl -o yaml
apiVersion: v1
data:
  nginx_port: "80"
  srver_name: www.quzl.com
kind: ConfigMap
metadata:
  creationTimestamp: 2019-03-31T23:57:00Z
  name: nginx-config
  namespace: quzl
  resourceVersion: "1010045"
  selfLink: /api/v1/namespaces/quzl/configmaps/nginx-config
  uid: aca0b768-5410-11e9-b7e8-5254009b4e6b
```

创建 pod 时引用:

```
        env:
        - name: PORT
          valueFrom:
```

```
            configMapKeyRef:
              name: nginx-config
              key: nginx_port
        - name: SERVER_NAME
          valueFrom:
            configMapKeyRef:
              name: nginx-config
              key: srver_name
```

```
      app: nginx
  spec:
    containers:
    - name: nginx
      image: nginx
      ports:
      - containerPort: 80
      volumeMounts:
      - name: web-base
        mountPath: /usr/share/nginx/html
      env:
      - name: PORT
        valueFrom:
          configMapKeyRef:
            name: nginx-config
            key: nginx_port
      - name: SERVER_NAME
        valueFrom:
          configMapKeyRef:
            name: nginx-config
            key: srver_name
```

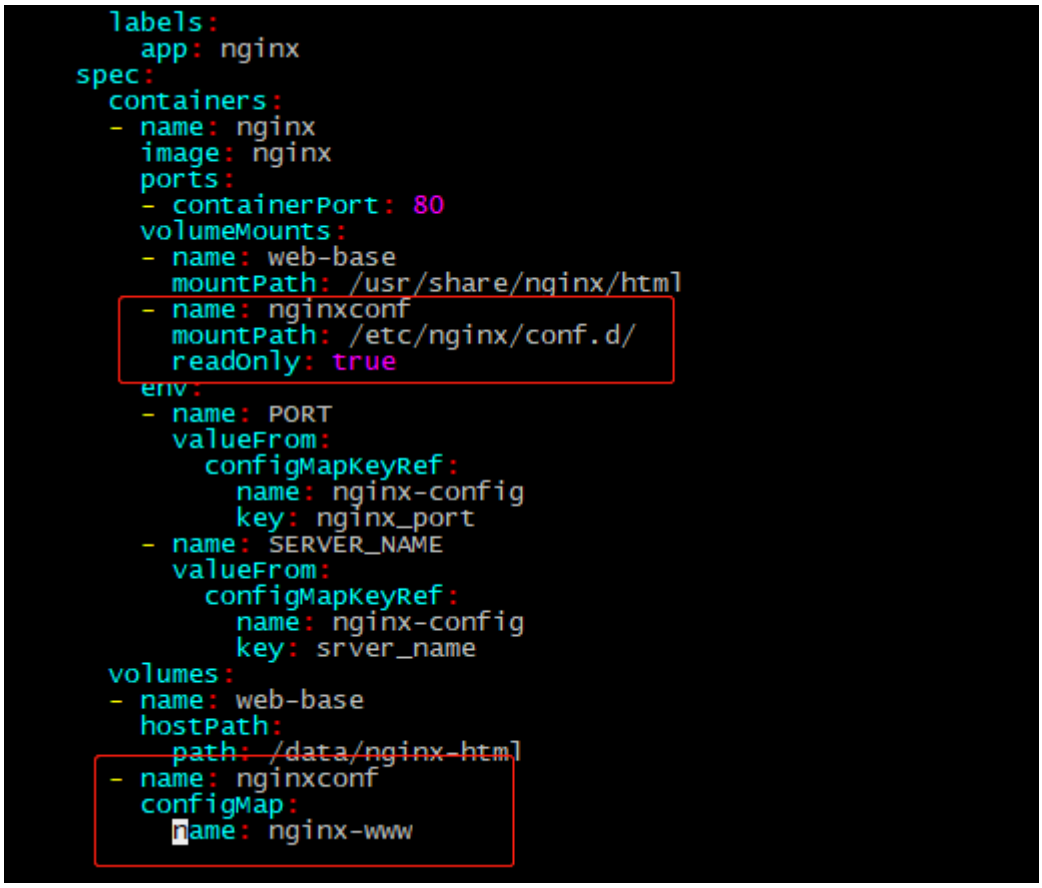
创建 pod 后查看是否生效:

```
[root@master k8s-test]#
[root@master k8s-test]# kubectl exec -it nginx-5b8f5f47cb-72prr -n quzl -- printenv
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx-5b8f5f47cb-72prr
TERM=xterm
PORT=80
SERVER_NAME=www.quzl.com
KUBERNETES_PORT_443_TCP_PROTO=tcp
NGINX_SERVICE_HOST=10.102.170.97
NGINX_PORT_80_TCP=tcp://10.102.170.97:80
```

2）使用文件创建，文件名就是 key，文件内容就是 value

```
kubectl create configmap nginx-www --from-file=www.conf
kubectl create configmap nginx-www --from-file=www.conf -n quzl
```

引用:



更多参考:
https://blog.csdn.net/liukuan73/article/details/79492374

7.8 控制器

7.8.1 ReplicationController

7.8.2 ReplicaSet

7.8.3 DeployMent

举例：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

7.8.4 DaemontSet

可以保证每个 node 上只有一个 pod。下面是一个例子：

```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: nvidia-device-plugin-daemonset-1.12
  namespace: kube-system
spec:
  updateStrategy:
    type: RollingUpdate
  template:
    metadata:
      # Mark this pod as a critical add-on; when enabled, the critical add-on scheduler
      # reserves resources for critical add-on pods so that they can be rescheduled after
      # a failure.  This annotation works in tandem with the toleration below.
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ""
      labels:
        name: nvidia-device-plugin-ds
    spec:
```

```
      tolerations:
      # Allow this pod to be rescheduled while the node is in "critical add-ons only" mode.
      # This, along with the annotation above marks this pod as a critical add-on.
      - key: CriticalAddonsOnly
        operator: Exists
      - key: nvidia.com/gpu
        operator: Exists
        effect: NoSchedule
      containers:
      - image: nvidia/k8s-device-plugin:1.11
        name: nvidia-device-plugin-ctr
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop: ["ALL"]
        volumeMounts:
          - name: device-plugin
            mountPath: /var/lib/kubelet/device-plugins
      volumes:
        - name: device-plugin
          hostPath:
            path: /var/lib/kubelet/device-plugins
```

7.8.5 Job

用的比较少

7.8.6 Cronjob

用的比较少

7.8.7 Statefulset