docker编排工具

（一）

①docker compose：单机编排

②docker swarm：面向集群，将多个docker host整合为同一资源池，随后compoase只需要面对swarm整合出的资源池进行编排

③docker machine：初始化、预处理工具：将一个主机迅速初始化，使之能够加入swarm

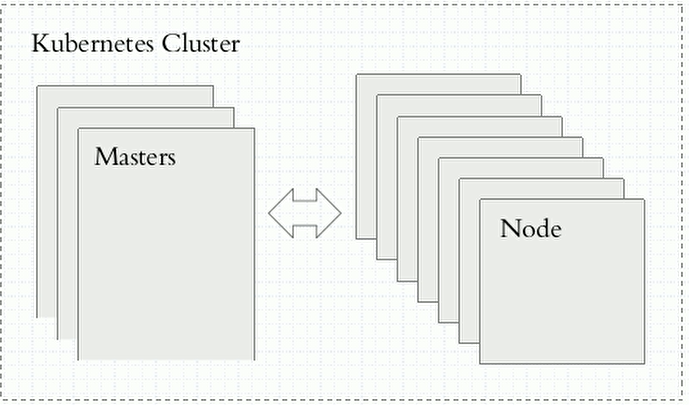
1. mesos，marathon
2. kubernates



Kubernetes

特点：自动装箱、自我修复、水平扩展、服务发现和负载均衡、自动发布和回滚、密钥和配置管理、存储编排、任务的批量处理执行

一、Kubernetes基础架构



三个master节点（高可用）； 多个Node节点：运行容器的工作节点

创建启动容器的请求都发给master 通过API Server接受、解析、处理请求；

调度器Scheduler分析各node现有的可用资源状态，找一个最佳适配的node，把容器调度到上面（两级调度，先筛选再优选），node上由kubelet接受请求然后由docker运行；

从镜像仓库拉取镜像（镜像仓库也可以以容器的形式部署在本身的这个k8s集群中，自托管）；

控制器负责监控pod的运行状态，pod挂了就向调度器发请求重建容器；控制器管理器controller-manager负责监控控制器的状态

一个pod内可以跑多个容器，一个pod内的容器共享网络名称空间、存储卷，

一个Pod中只有一个主容器，其他的称为边车容器（如filebeat）

Selector标签选择器，用标签（键值对）来标识Pod（元数据），用于pod的分类和管理

1. 控制器

无状态应用控制器：Deployment + HPA（对workload副本数进行自动水平扩缩容）

有状态应用控制器：StatefulSet

每个node上只运行一个副本：DaemonSet

作业：job 定时作业：Cronjob

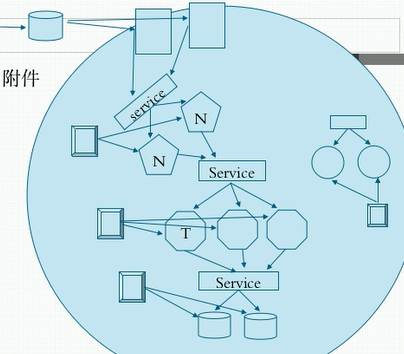
2、AddOns：附加组件

（1）service解析服务：由于pod对应的IP容易变动，因为pod的重启，其IP就会和原来不同，Service 提供了一个稳定访问的机制，提供稳定的IP。 Service 是一组逻辑上关联的 pod，利用selector，将label为app:MyApp的pod聚合在一起，形成个组，我们可以称为Endpoints，这也是一种k8s资源。

所以现在的关系是 Service（Cluster IP）–> Endpoints --> 一组Pod（Pod ip）

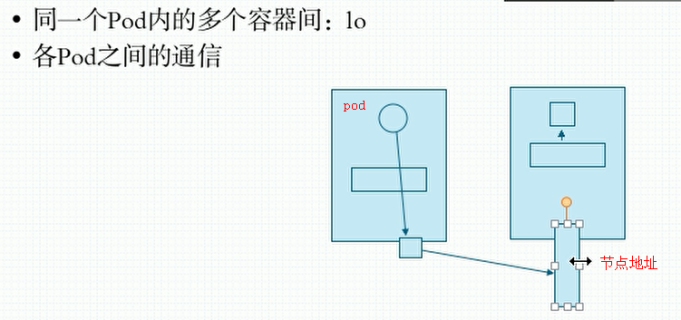
1. NMT

Nginx; M=数据库； Tomcat



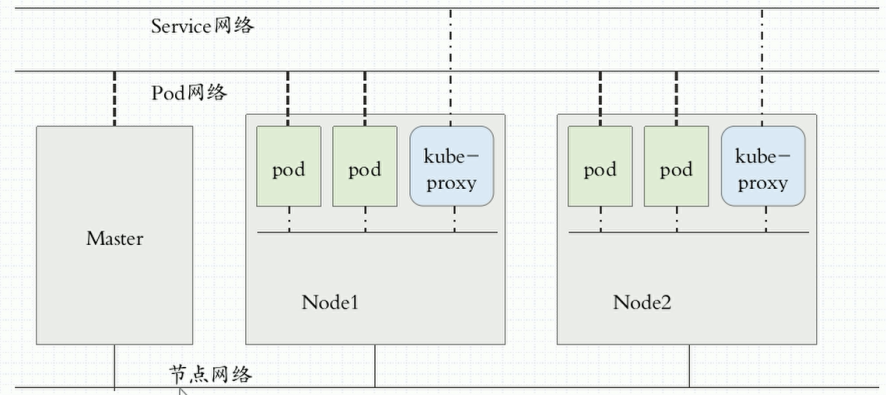
1. 网络

节点网络 -> 集群网络（service网络） -> pod网络



pod间直接通信： 叠加网络（Overlay Network）

service的创建和修改：kube-proxy



容器网络接口CNI：

flannel：网络配置（叠加网络）

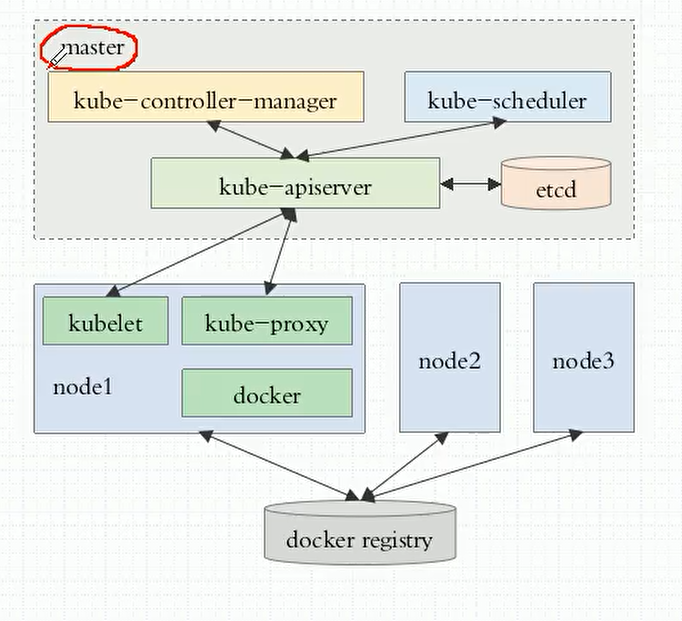
calico：网络配置+网络策略（BGP协议实现网络路由通信、三层隧道网络）

canel：网络配置（叠加网络）+ 网络策略（三层隧道）

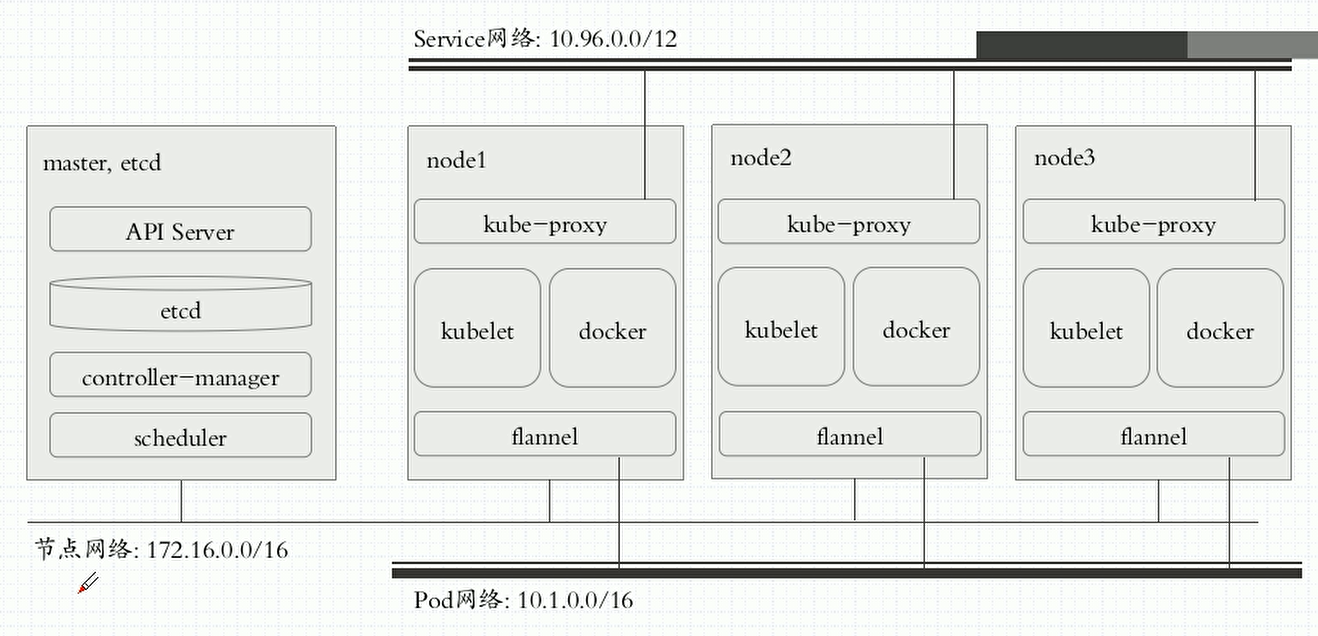
1. 名称空间 namespace :

为pod提供管理的边界，本身不限制pod间的相互访问；不过可以通过iptables来隔离名称空间之间pod的相互访问；

03-初始化Kubernetes集群



部署架构：

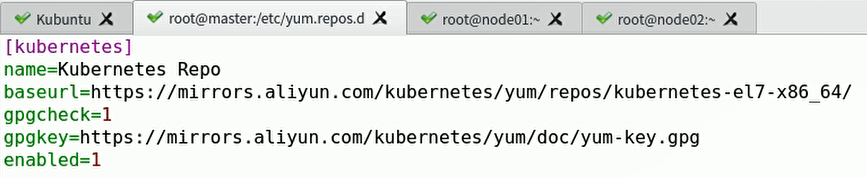


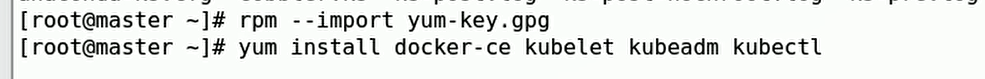
部署方式：

1. 手动部署到物理机节点上、手动创建各种CA认证
2. kubeadm：
   1. 所有节点上装好kubelet、kubeadm和docker并进行初始化
   2. 在作为master的节点上，kubeadm init ：用容器（pod，静态pod）运行api server、scheduler、controller-manager、etcd
   3. 余下的node节点上，kubeadm join： 用容器（pod，托管的动态pod）运行kube-proxy
   4. （所有节点都需要部署flannel）

部署：

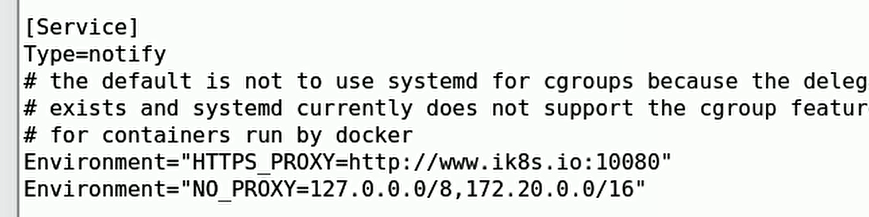
1、准备镜像

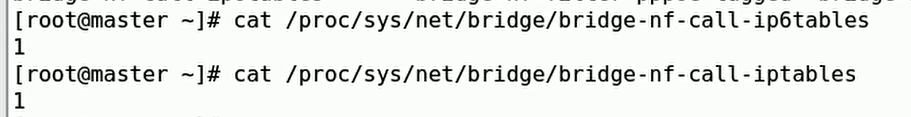






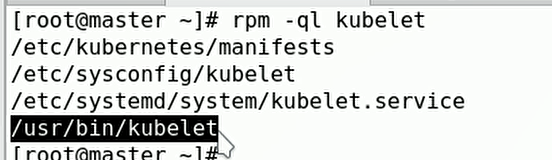
2、启动docker，做好环境设定





1. 设定kubelet

启动项：



设定kubelet配置忽略SWAP报错信息

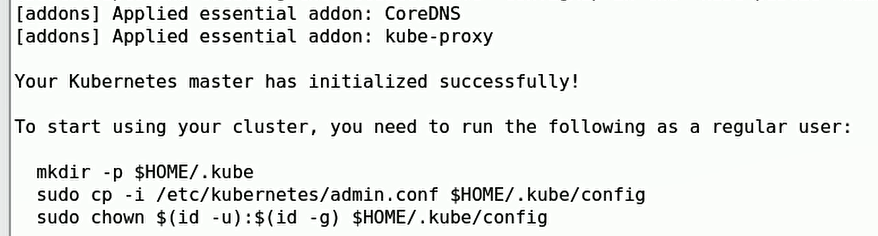
#vim /etc/sysconfig/kubelet

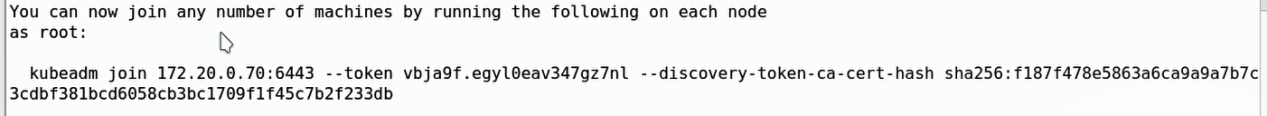
KUBELET\_EXTRA\_ARGS=”--fail-swap-on=false”

启动： systemctl start kubelet

1. kubeadm init

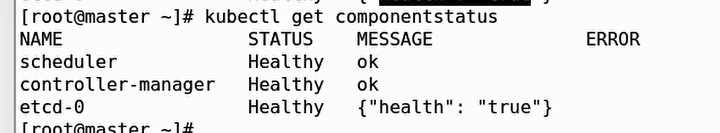






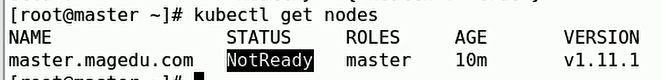
保存好tocken 和ca-cert-hash

查看组件状态： kubectl get componentstatus (kubectl get cs)



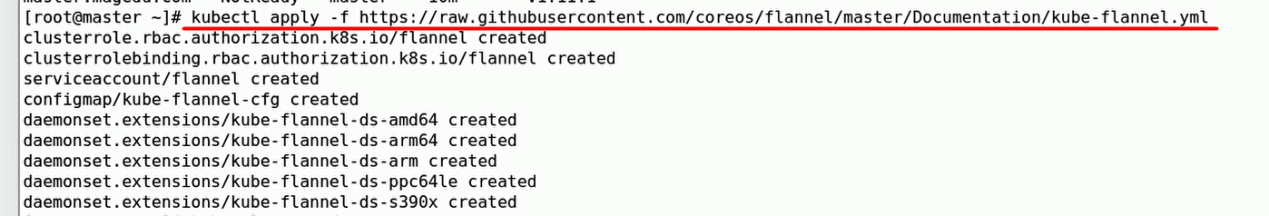
查看集群节点信息 kubectl get nodes

如果没装好flannel或其他网络组件的话 节点会显示未就绪状态

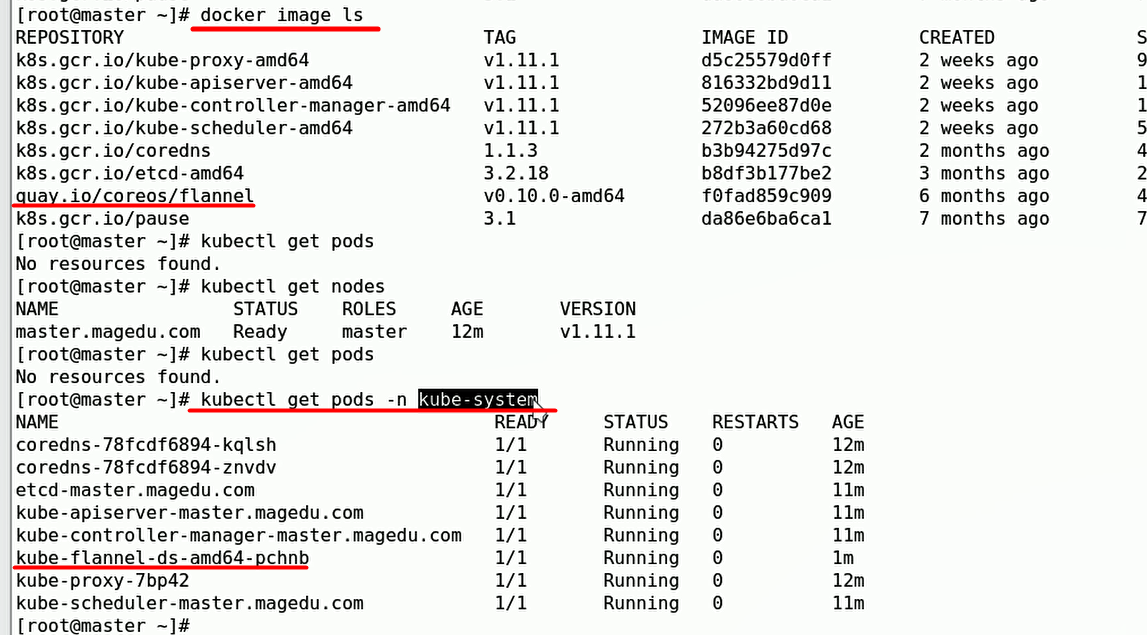


1. flannel

自动获取清单并下载镜像：



docker image ls 能看到flannel镜像后

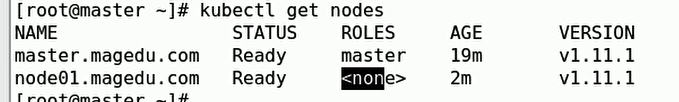


1. node节点加入集群

先用和master相同的配置启动docker kubelet

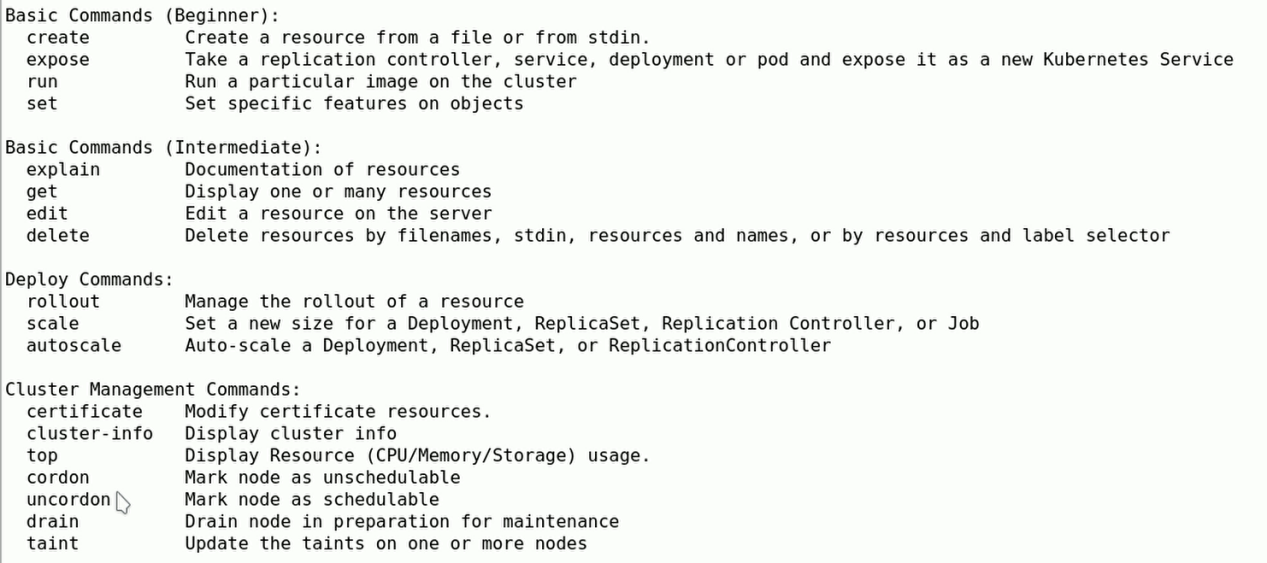
kubeadm join:

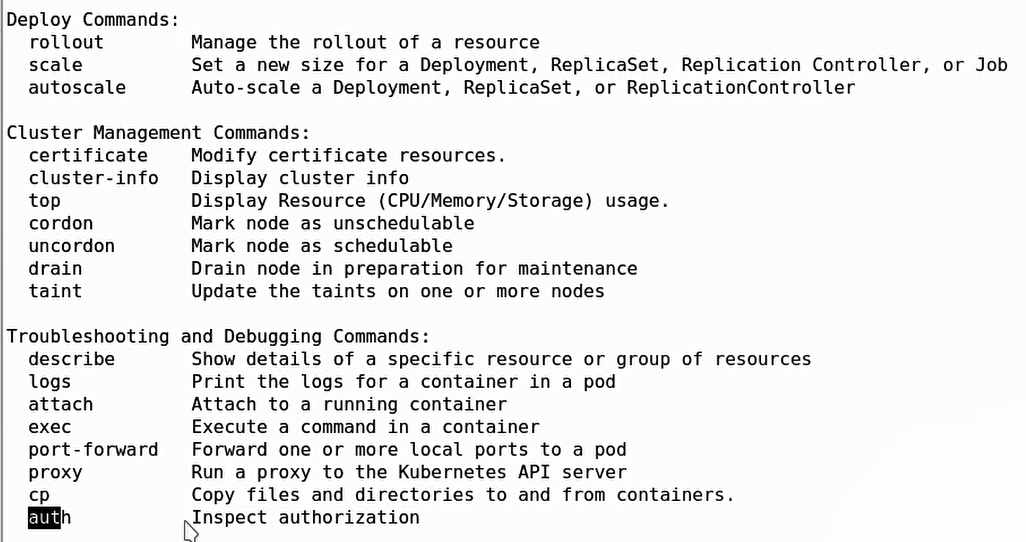


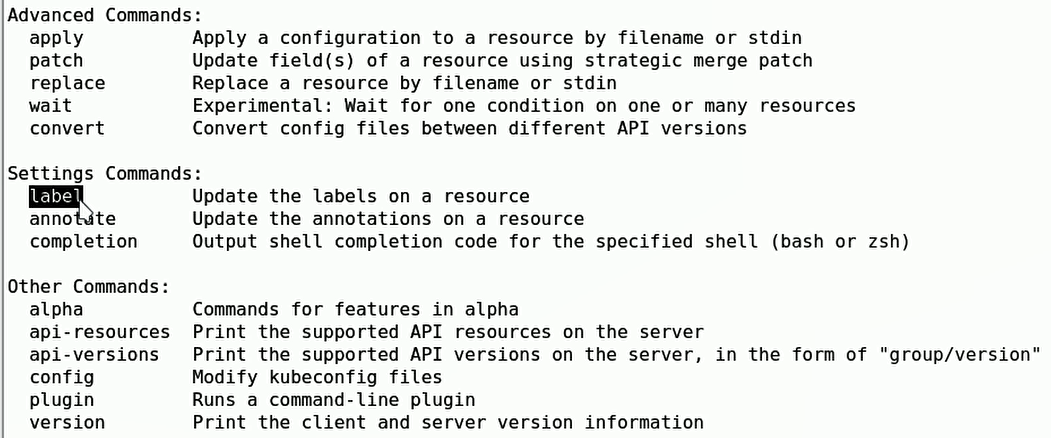


04、应用入门

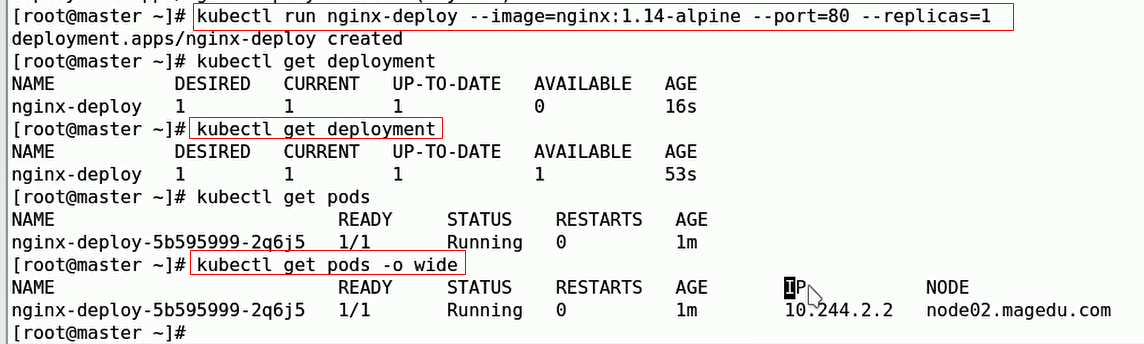
kubelet后面可带的参数：







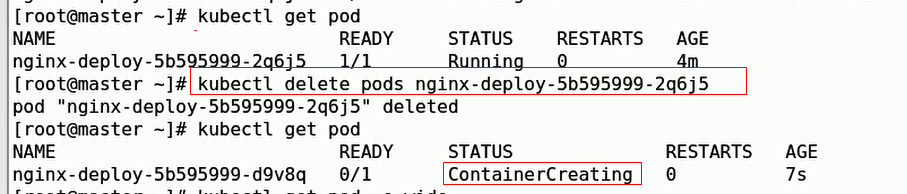
1. 创建pod



生成对应桥接网络地址：

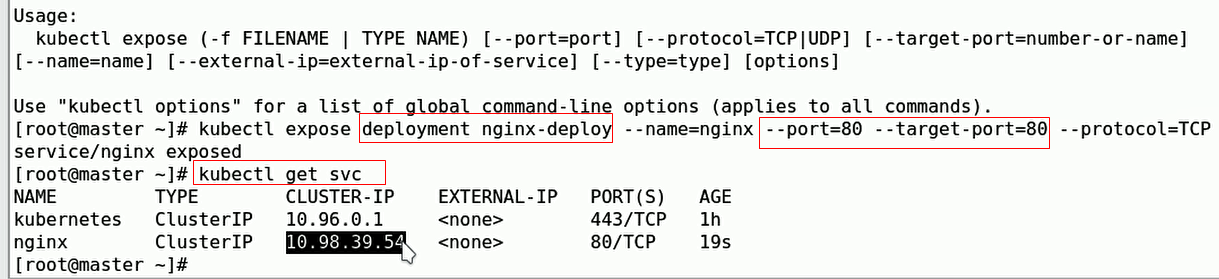


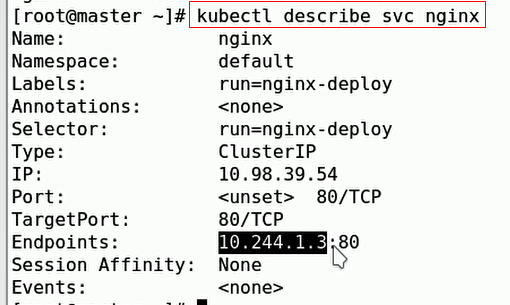
1. 删除pod（会自动创建新的）



1. 创建service：expose

给特定应用的Pod暴露特定的端点，访问时通过service-ip:service-port而不是pod-ip:pod-port

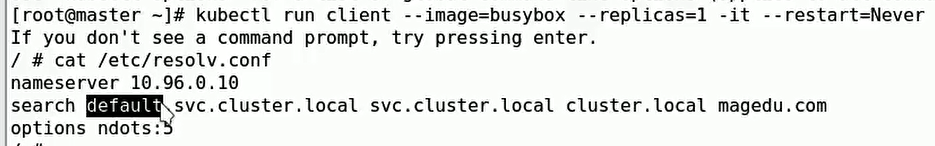




pod被删除重启之后endpoint便变化，Selector会重新选择满足标签run=nginx-deploy的节点



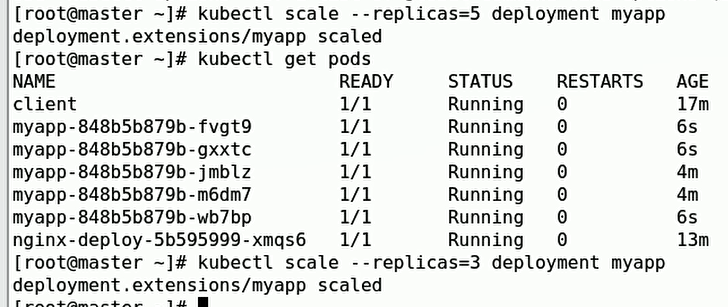
建个客户端去访问



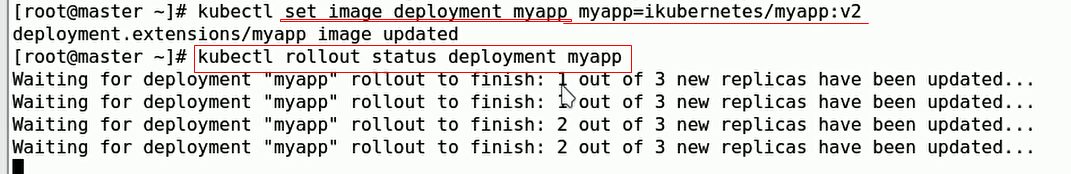
wget nginx 或者wget -O - -q http://nginx:80/



1. 改变服务的pod副本数 scale



1. 滚动升级 rollout，更新镜像 set image

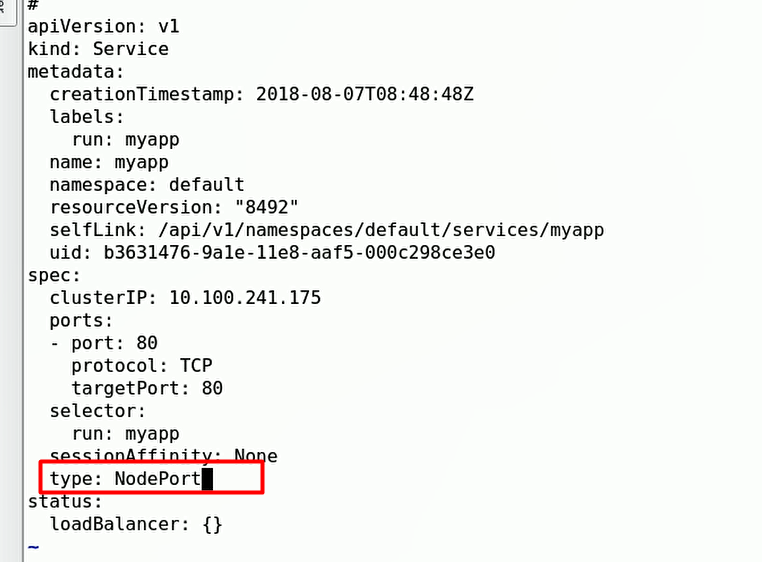


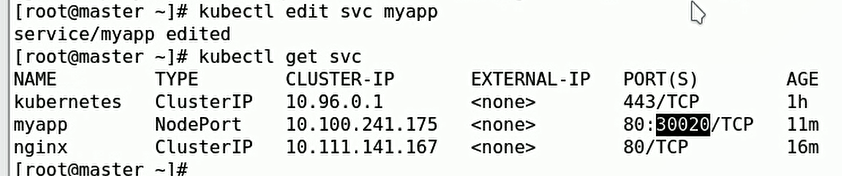
查看滚动更新进度：kubelet rollout status deployment

回滚更新：kubelet rollout undo deployment myapp

1. 使应用能被集群外客户端访问

kubectl edit myapp 改类型type: ClusterIP 为NodePort







05-定义资源清单