概念及架构

Kubernetes：舵手，谷歌基于Borg使用golang开发的容器编排工具。

特性：自动装箱，自我修复，水平扩展，服务发现，负载均衡，自动发布和回滚，密匙和配置管理，存储编排（存储卷自动编排），批量处理执行。

集群，把多个主机当成一个主机使用，把多个工作机当作一个工作节点使用。

K8s的集群为master/node节点架构。

Master上的重要工具：API server，scheduler，controller-manager（都是守护进程daemon）

Node：kubelet，docker（容器引擎），。。。

客户端请求发送给masters，masters中的scheduler选择最合适处理该请求的node来完成任务。

API server：负责接受来自各种来源的请求（来自其他master，node或外部），所有服务的访问接口。

Scheduler（调度器机制）：两级调度。一级选择所有符合需求的节点，二级选择最优的节点来完成任务。

Kube-controller-manager（控制器管理器）：负责管理所有的controller，监控所有的controller是否健康。对于控制器管理器是否健康，由于即使有多个master，但是也只有一个控制管理器工作，所以使用冗余机制，使用>=3的master来维持控制管理器的使用。

Pod：k8s调度的原子单元。pod内可建立多个container，同一个pod内的container共享网络，存储卷等资源。但一般一个pod内只放一个container，就算有多个container，其他container也只是为了辅助主container来建立。当建立了pod后，会自动或手动添加一个label来标记这个pod用来干什么。

>两种pod：自主式pod，控制器管理的pod

> Controller（pod控制器）：持续监控node里的所有pod是否健康，loop循环监控node里的pod是否健康，发现一个pod死机，直接重启一个一样的pod来代替。分为以下几种：ReplicationController副本管理器，ReplicaSet，DeploymentSet管理无状态，statefulset管理有状态，daemonSet唯一pod管理，job工作管理）,deploymentSet支持下级管理器HPA：水平pod自动伸缩控制器。

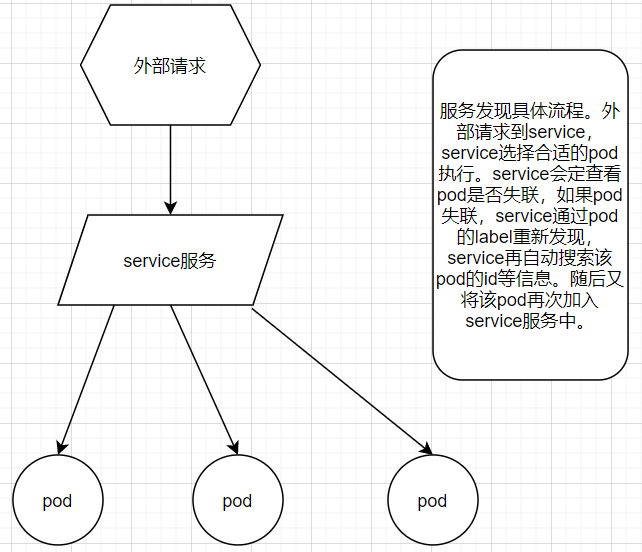
Label\_Selector（标签选择器）：根据标签来过滤符合条件的pod或者其他资源的机制，可以分类管理庞大的pod。即每个pod都有一个资源键值对（像key：application，value：nginx）来记录每个pod是干什么的。

Kubelet：在node上，可接受任务，试图启动pod，创建存储卷等。

Docker（容器引擎）：负责运行pod等任务。

Kubernetes可以托管自己（即在kubernetes上部署kubernetes）。

关于服务发现



Etcd：全局共享内存，需做成高可用（配置三个或以上）。

Flannel：网络配置

部署

环境：

Master，etcd：172.18.0.67

Node1：172.18.0.68

Node2：172.18.0.69

前提：

1. 基于主机名通信：/etc/hosts
2. 时间同步
3. 关闭firewalld和iptables.service

Os：centos 7.3.1611

安装配置步骤：

1. etcd cluster，仅master节点
2. flannel，集群所有节点都装
3. 配置k8s的master：仅master节点

Kubernetes-master

启动服务：

Kube-apiserver，kube-scheduler，kube-controller-manager

1. 配置k8s的各个node节点

Kubernetes-node

先设定启动docker服务

启动k8s服务：

Kube-proxy，kubelet

具体安装过程：

Kubeadm安装：

1. master，nodes：安装kubelet，kubeadm，docker
2. master：kubeadm init
3. nodes：kubeadm join

安装k8s前准备：

关闭所有防火墙

使三台虚拟机能相互访问，master、node1、node2

设置本机名

hostname set-hostname 虚拟机名称

在/etc/hosts中添加其他虚拟机的ip地址和名称

vi /etc/hosts

添加如：

127.0.0.1 虚拟机名称

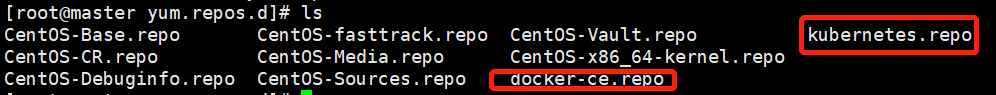
然后复制hosts文件到其他虚拟机，如：

scp /etc/hosts node1:/etc/

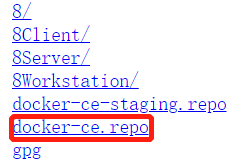
保证每台虚拟机中hosts文件都相同

开始安装>>>

需生成docker仓库的配置文件和k8s仓库的配置文件



去往阿里云的云仓库下载docker-ce.repo



复制下载链接，使用wget下载

如果没有wget，使用yum install –y wget 安装

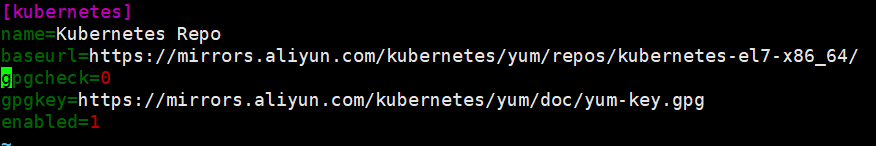


下载好后一般不需要修改

编辑kubernetes仓库文件



添加内容



确定仓库文件可用



复制仓库文件到node1、node2





安装组件，在组件后加版本号则安装对应的版本组件



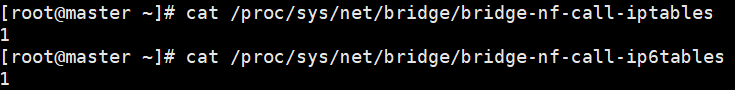
编辑Docker的service文件



重新加载daemon，重启docker



确保以下两个值都为1

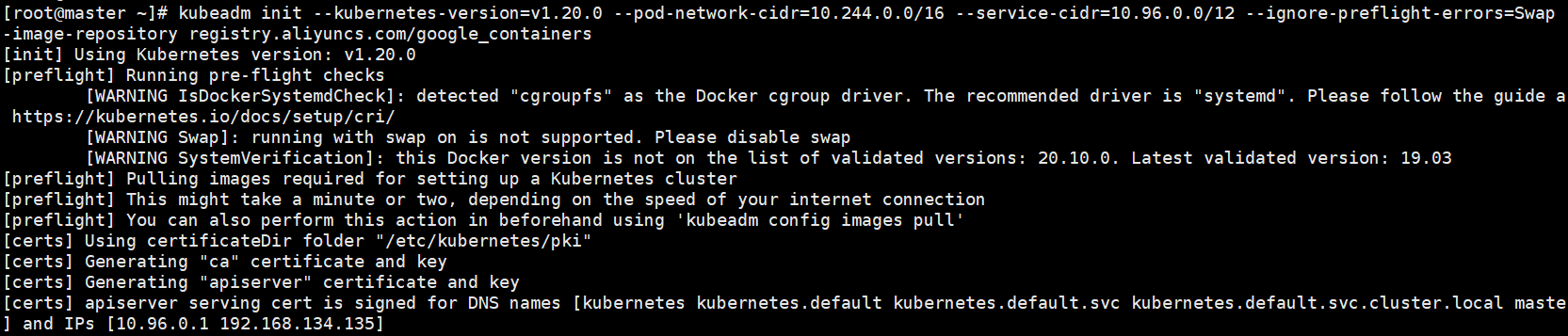


编辑kubelet文件

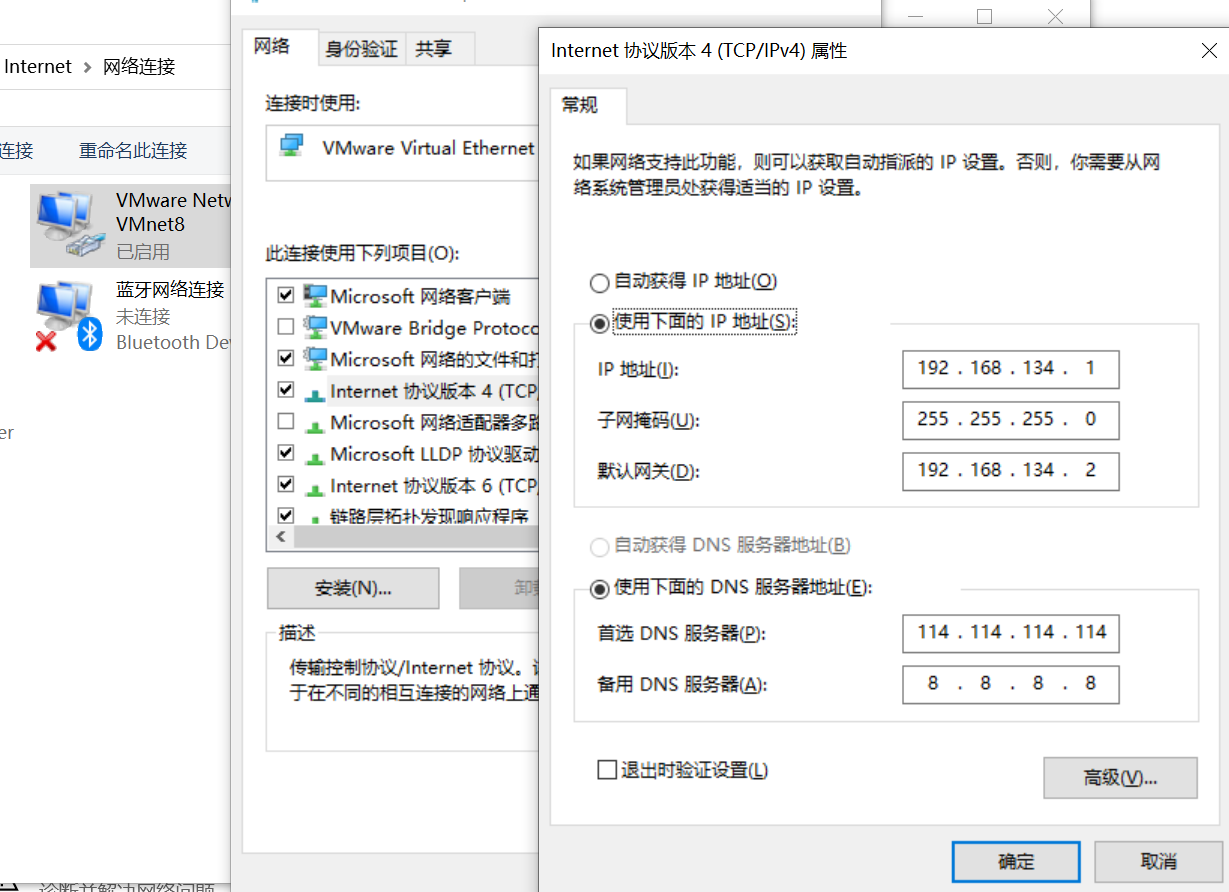




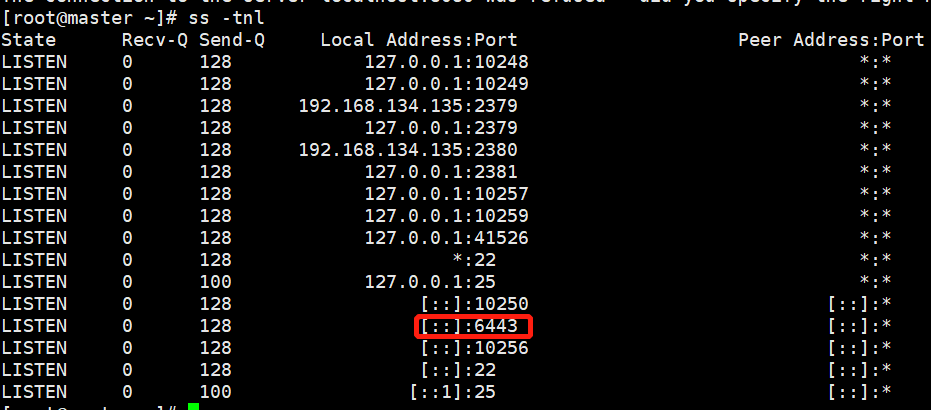
下载相关镜像



如果registry.aliyuncs.com无法ping同则需要去虚拟网块那里配置（大概）



执行完上一段后查看6443端口是否被监听



保存hash码，其他节点通过这个hash码加入集群

kubeadm join 192.168.134.135:6443 --token ytcor4.3fbo7eplhk5jzohl \

--discovery-token-ca-cert-hash sha256:21c50e955a980fcd61ff2f183ef3007c8d911931e52cd974c5c69d5448270762

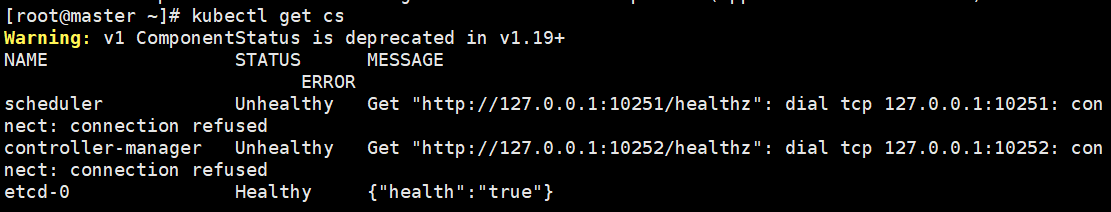
创建隐藏目录



Cp文件到隐藏目录



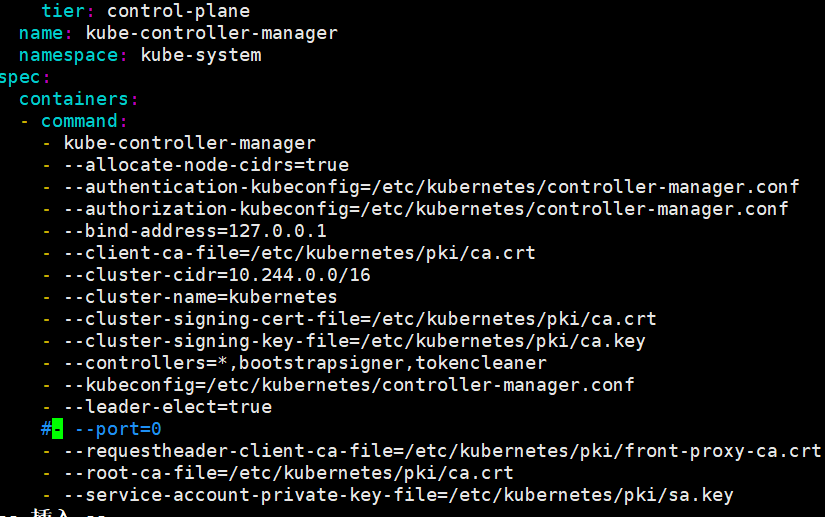
查看发现有两个组件不健康

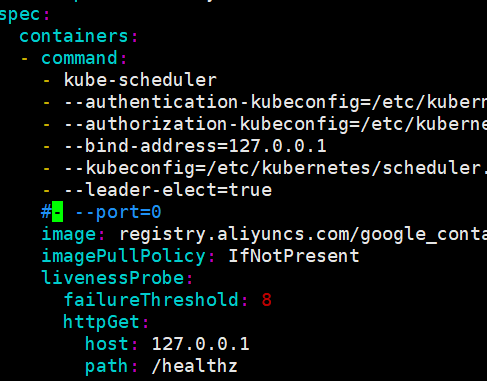


修改两个组件对应的配置文件



注释掉- --port=0





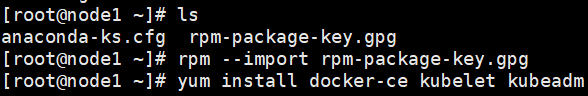
复制rpm钥匙到node1、node2





切换到node1

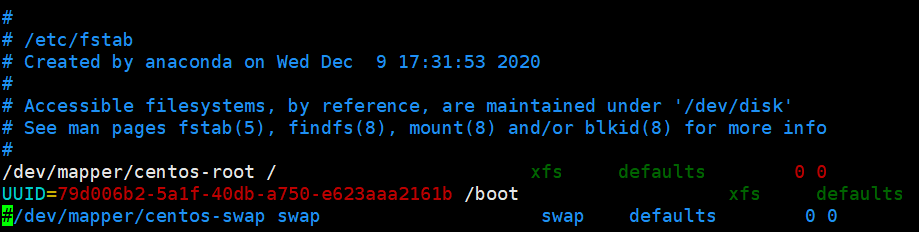
启用密钥，安装docker-ce、kubelet、kubeadm



Systemctl enable docker kubelet

启动docker

修改/etc/fstab

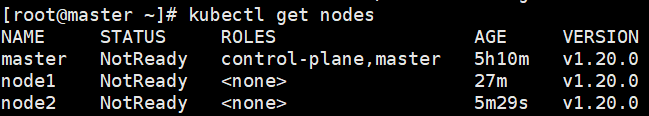


加入节点

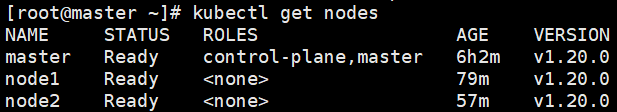


Node2重复node1的操作

返回master查看节点是否加入成功



Flannel部署。。。



K8s应用入门

Master上的应用pod，service，replicaset，deployment，statefulet，daemonset，job，cronjob，node

Kubectl：

Taint（污点）

关于无法连接创建好的pod问题

首先kubectl get po –owide 查看curl的服务的ip地址是否正确，查看flannd等状态是否正常，然后再去要访问的pod上的节点上查看cni网卡是否正常，如不正常则需要重启节点或重启network。

使用nodePort连接集群服务。

创建资源清单

方法：

Apiserver仅接收json格式的资源定义

Yaml格式提供配置清单，apiserver可自动将其转换为json格式，而后再提交

大部分资源的配置清单：

ApiVersion（v1、v1beta等）

Kind：资源类别(pod、deployment、statefulset、daemonset、job等）

Metadata：元数据

Name

Namespace

Labels：标签

Annotations：注解

资源的引用path：/api/group/version/namespaces/namespace/type/name

Spec:用户期望状态，disired state

Status：当前状态，current state，本字段由k8s维护

Pod控制器应用进阶

查看资源具体的属性说明：kubectl explain deployment.metadata

标签：

Key=value

Key:字母、数字、\_、-、.

Value:可以为空，只能字母或数字开头及结尾，中间可使用

标签选择器：

等值关系：=、!=、==

集合关系：

Key in (value1,value2,…)

Key not in (value1,value2,…)

许多资源支持内嵌字段，定义其使用的标签选择器：

Matchlabels:直接给定键值

MatchExpressions：基于给定的表达式来定义使用标签选择器

操作符：in、notin、Exists、notExists

Annotation：注解，用于标注资源的元数据。

pod生命周期

状态：pending，running，failed，successed，unknown

创建·pod：

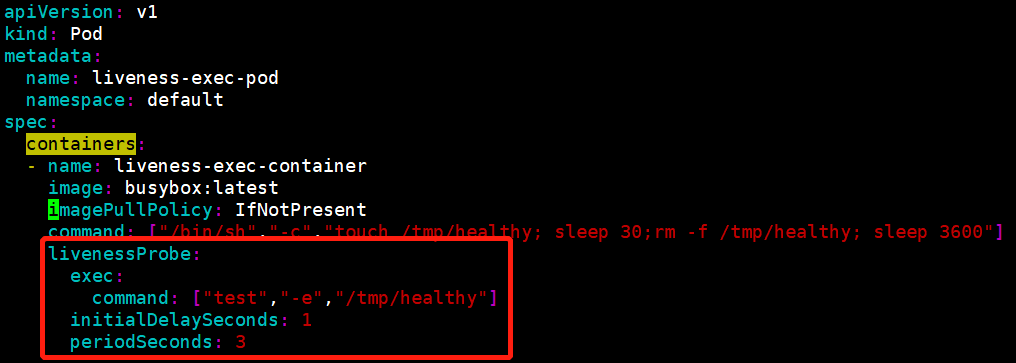
重要行为-初始化容器，容器探测：liveness、readiness

RestartPolicy：

Always、onFailure、Never、default to always

探针类型有三种：

ExecAction（存活性探测）



TCPSocketAction

HTTPGetAction

Kubectl patch –help 给资源打补丁

Service资源

Service服务的域名解析强依赖于CoreDNS或kube-dns，所以想要使用service服务，必须安装以上两种DNS服务

其中有三种网络：

Node network、pod network、cluster network

以上都是virtual ip（虚拟ip），

工作模式：userspace、iptables、ipvs

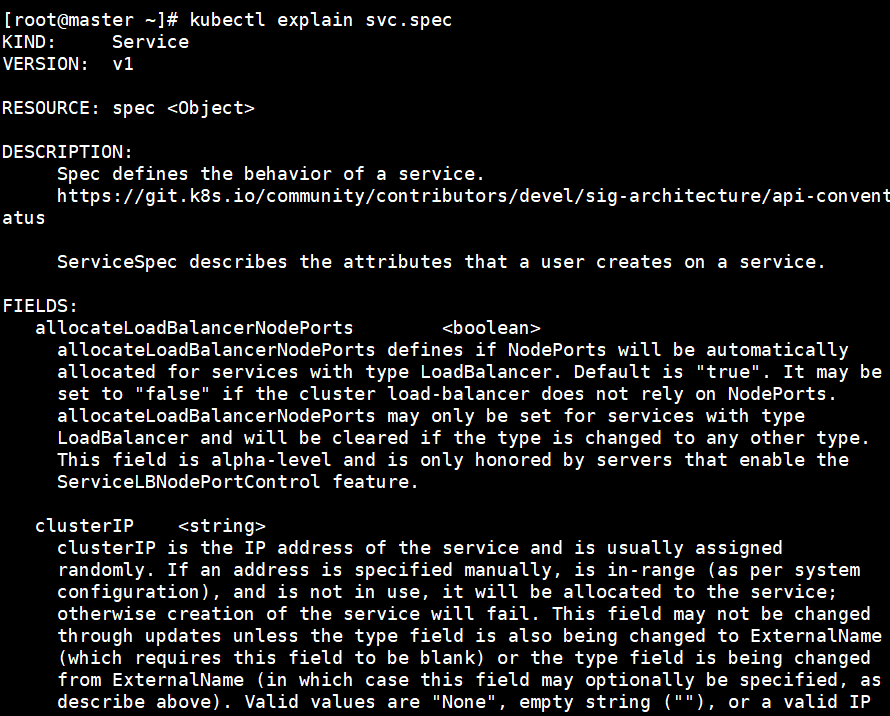
iptables规则就是service 规则，更方便的ipvs规则更加方便，在k8s1.1版本以后使用ipvs，如果没有配置ipvs则自动降级为iptables

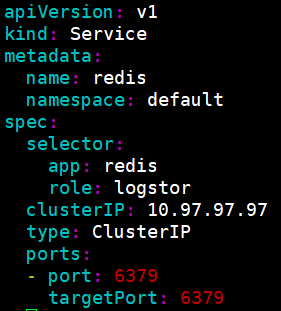
类型：

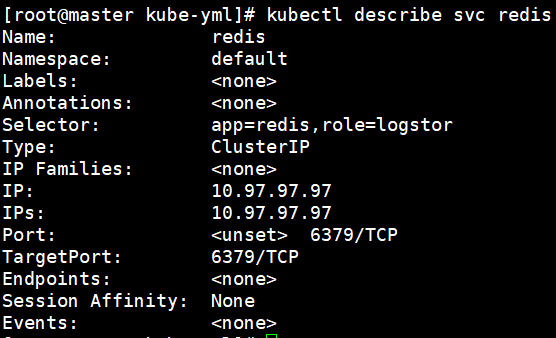
ExternalName、clusterIP、nodePort、loadBalancer

资源记录：

Svc\_name.ns\_name.domain.ltd.

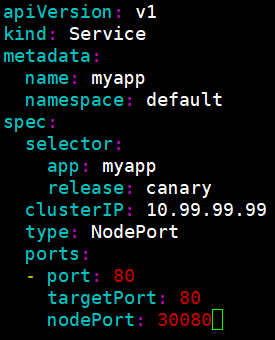






其中service到pod中间会存在一个实际存在的对象endpoints，先经过endpoints再到pod

Myapp-svc

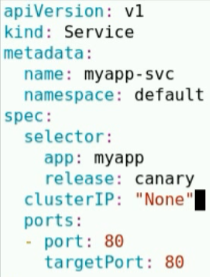


spec 中的参数

App: nginx-deployment，nignx-deployment代表一个具体的提供服务的pod

release: canary，暂不知道

clusterIP：为service在集群内部的ip，当请求到该service时，会通过该service的名字解析为该clusterIP



当clusterIP设为””时，则代表这是一个无头的服务

port：为service的端口

targetPort: 为暴露指定的pod的端口到service端口，不设置则默认为80

nodePort：30080，暴露service的80端口到集群的30080端口，使集群外部也能够访问service

Session Affinity：使来自同一个clientIP请求的始终调度到同一个pod中处理，默认为None

Ingress

ingress提供7层服务，