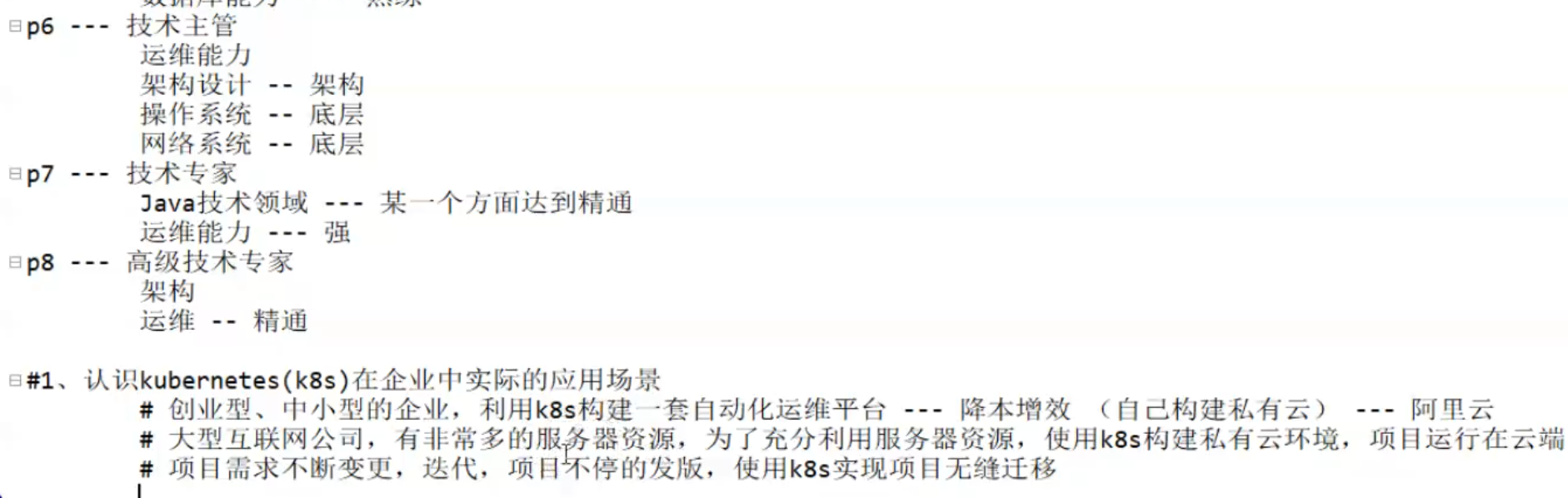
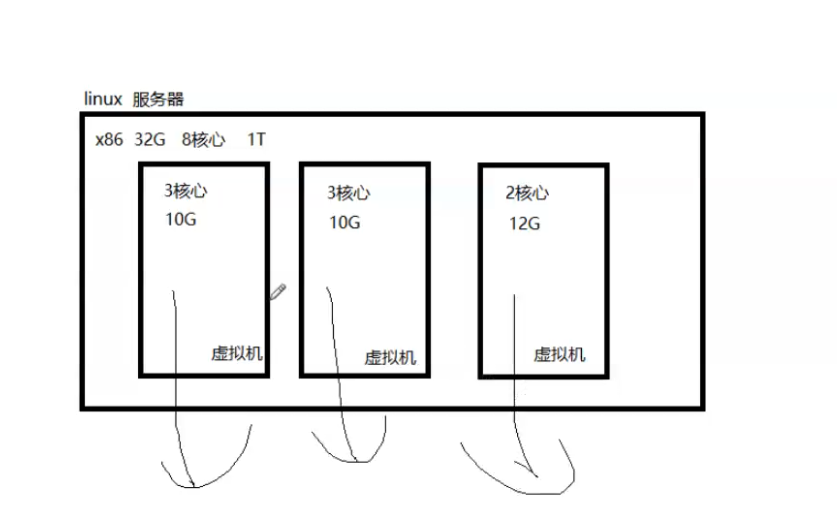
Openstack不仅仅是服务分布式，组件也是分布式，提供了很好的扩展作用。

自动化运维平台。

## K8s的应用场景：



## 云技术—物理资源隔离技术



虚拟化分类

# 全虚拟化架构 --- 在os系统内部 ，可以安装多个操作系统，达到物理资源分离。

# OS层虚拟化架构 --- 在os系统内部，可以复制多个操作系统，达到物理资源分离。

# 硬件层虚拟化 --- 直接在硬件上安装多个操作系统，达到物理资源分离。

注：Hypervisor 虚拟机监视器，之所以能实现虚拟化技术，主要是靠Hypervisor来进行物理资源调度，分配。

#总结

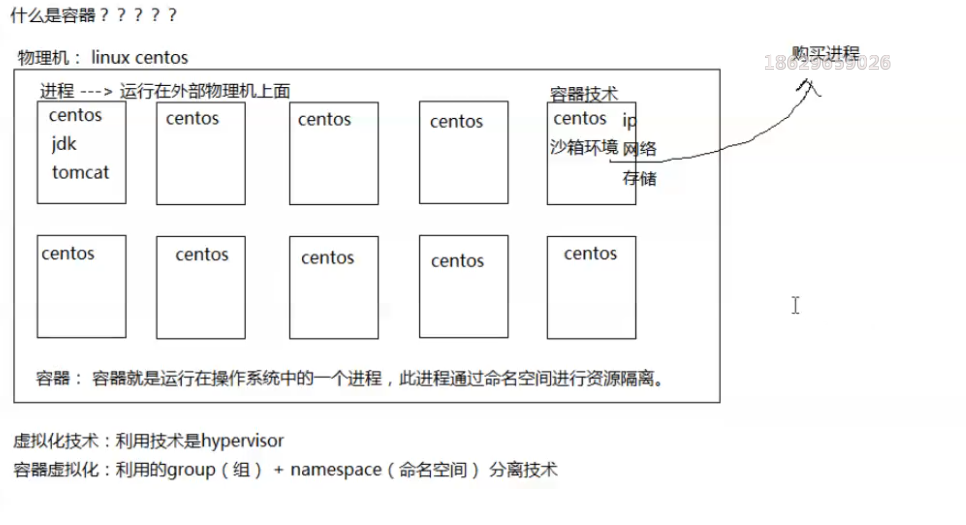
# 其实就是在物理硬件上面安装多个操作系统，通过操作系统的隔离性来达到资源的隔离。

# 什么是容器？？？

# 容器就是运行操作系统（物理机操作系统，虚拟机操作系统），是一个操作系统的进程。

进程级别的虚拟化技术。

虚拟化和容器：



## docker&OpenStack&kvm

什么是云平台？

云计算平台也称为云平台，是指基于硬件资源和软件资源的服务，提供计算、网络和存储能力。

<https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97%E5%B9%B3%E5%8F%B0/9471851?fr=aladdin>

# 云计算平台实现方案有几种？？

# 物理机构建云平台环境 ----- 无法充分利用资源 ，造成资源浪费

# 虚拟化技术构建云平台环境 ----- 隔离资源，可以更充分利用资源，虚拟机本身就耗费资源

# 容器化技术构建云平台环境 ----- 轻量级，非常好的云平台构建技术

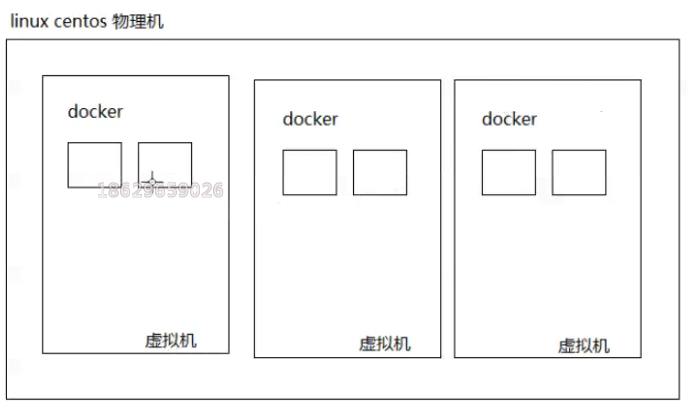
# docker 容器化技术

# 构建容器化云计算平台

# OpenStack管理容器化云平台，应该在虚拟化机器中运行容器

# 更多：更彻底的隔离，安全性考虑

# k8s 管理 容器 （安全，性能，效率）

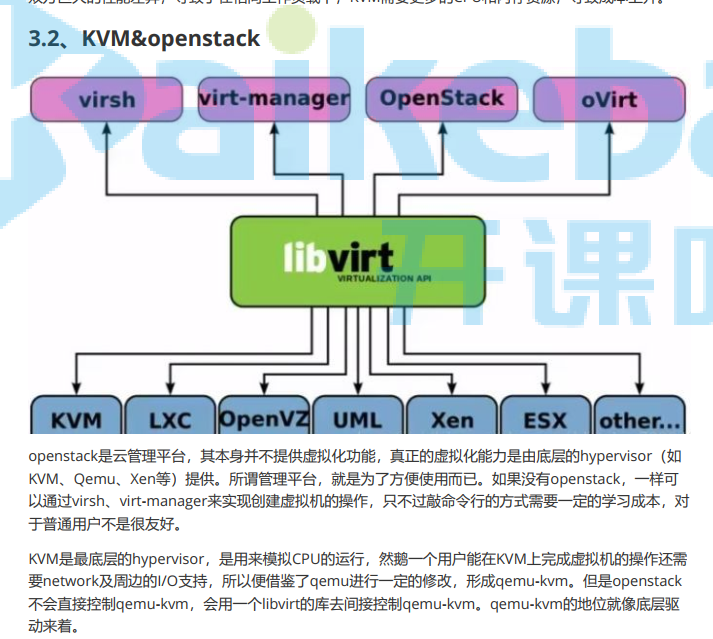


# kvm

# 相同的资源情况下，kvm会消耗更多CPU,内存资源。性能无法跟docker相比。

# OpenStack

# 提供一套可视化的方式管理虚拟机云平台。



# 问题：构建云计算平台，有那几中方案？

1、物理机构建云计算平台：物理机（硬件规格，建设机房，网络，基础服务，基础设置维护）--- 大批运维人员

中小型企业：

100 台 ----- excel，word记录下每一台机器部署什么服务，除了问题检查笔记本。

1000台 ----- excel，word记录下每一台机器部署什么服务，除了问题检查笔记本。

10000+ 台 ------ 人肉方式，成本非常高

2、KVM 时代： 使用虚拟机构建云计算平台

优点：充分利用物理硬件资源，用户使用服务器，只需要租用几个虚拟机即可，减少运维成本

缺点：虚拟化占用资源（CPU,IO,内存）太多，对性能是一个非常大的影响。

3、容器时代： ---- 构建公司私有云平台

轻量级

运行速度非常快

占用资源非常少

无缝迁移

## 云原生

# 一句话： 就是为了让应用程序（项目，基础服务：mysql,es...）运行在云上的解决方案，叫做云原生。

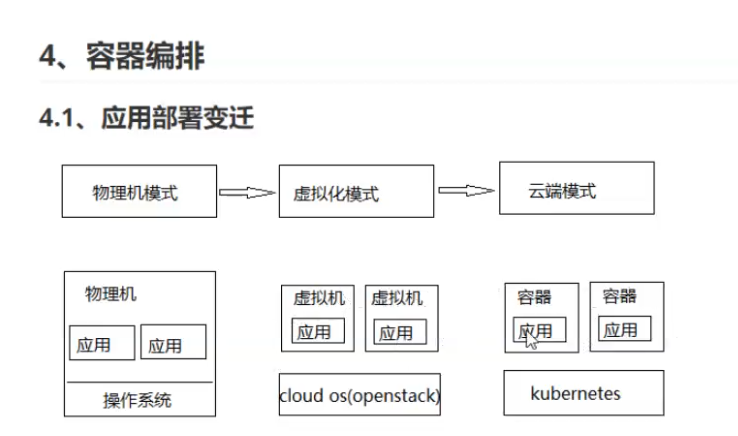
特点：

1、容器化 ： 应用程序都运行在容器中

2、微服务 ： 将单体架构拆分多个服务链式调用模式，进行流水线生产模式，实现CI/CD

3、可持续交付，可持续部署（需求：此项目实现快速迭代，快速上线）

4、DevOps : 开发，运维 密不可分。



## 容器编排技术

1、资源【容器】管理技术，容器编排技术

\* windows 资源管理器

\* linux 资源管理器

2、应用部署变迁

\* 物理机（硬件，机房，网络...）

\* 虚拟机（OpenStack管理）

\* 容器化（kubernetes k8s）

3、容器部署规模越来越大，如何管理？

\* 怎么扩容 --- 流量增大

\* 容器宕机怎么办？ 无法对外提供服务？

\* 版本迭代更新？

\* 服务监控？

\* 负载均衡调度？

\* 数据安全性？

4、容器云原生

\* 扩容 --- 一个指令搞定，一个按钮

\* 宕机 --- k8s自动创建新的服务，让服务永远不会消失，永远处于高可用。

\* 发版 --- 一个指令，按钮

5、容器编排技术

\* docker-compose

非常简单的轻量级的容器编排工具，只能做简单的容器编排，不能实现复杂容器编排

\* rancher

rancher 是可视化图形化的工具，管理编排容器。v2版本提供对k8s管理。中小型企业，可以使用rancher

\* swarm

docker 公司自己研发一套容器编排技术。docker自己用k8s

\* mesos

apache 提供的资源管理技术，提供比较粗颗粒度的容器管理方案。

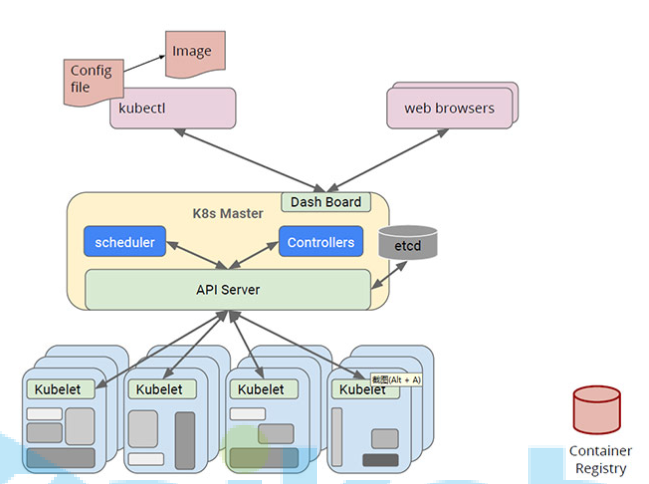
\* borg

google公司研发的一套容器编排技术，这套非常成熟，稳定，但是不开源。

\* kubernetes

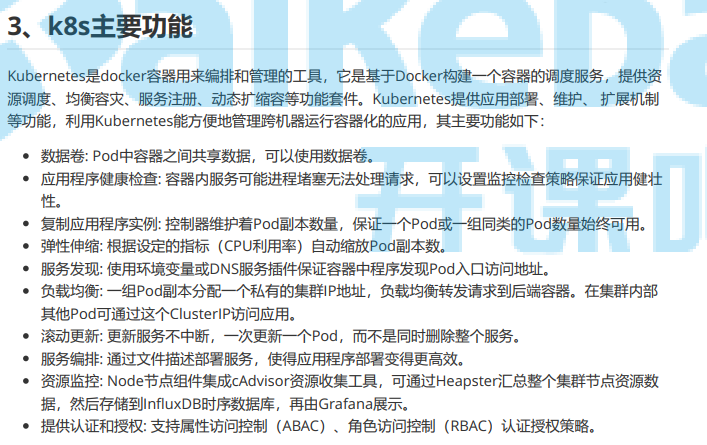
google公司研发的一套容器编排技术，开源，免费的。使用标签，yaml可编程的方式管理容器，非常灵活。

## K8s架构

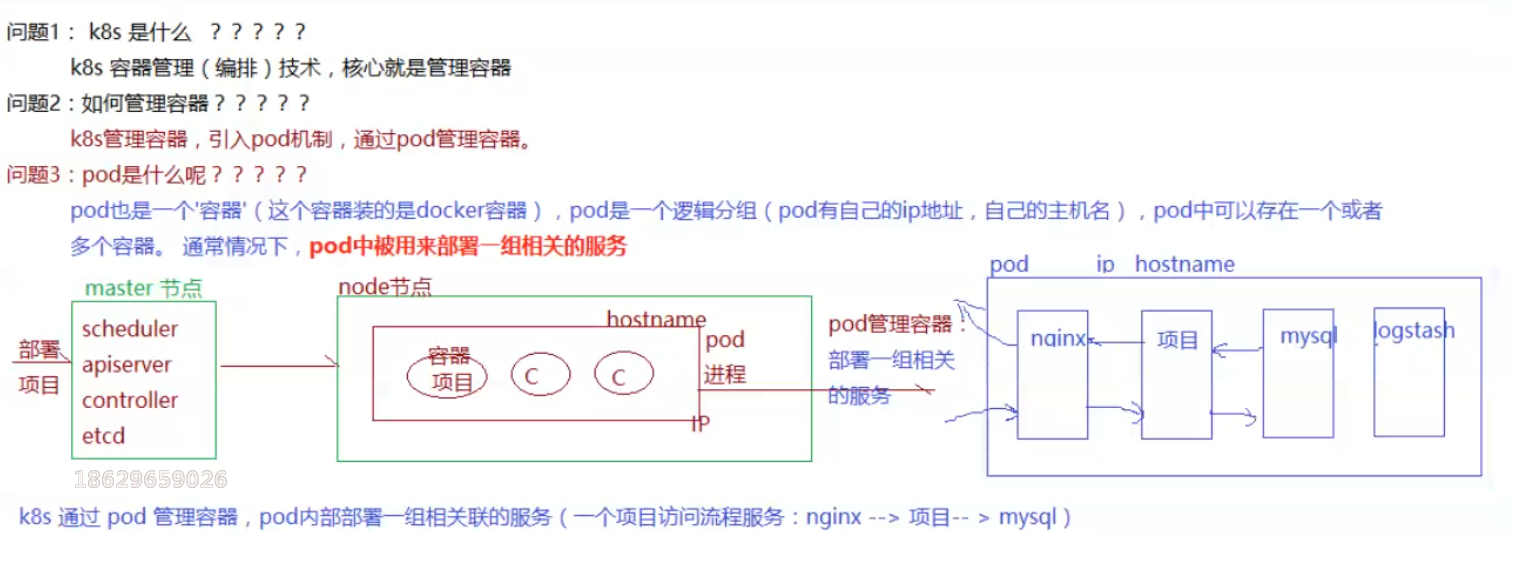




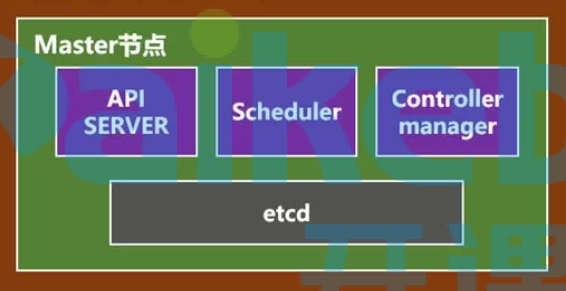
## 主要功能



K8s总结：



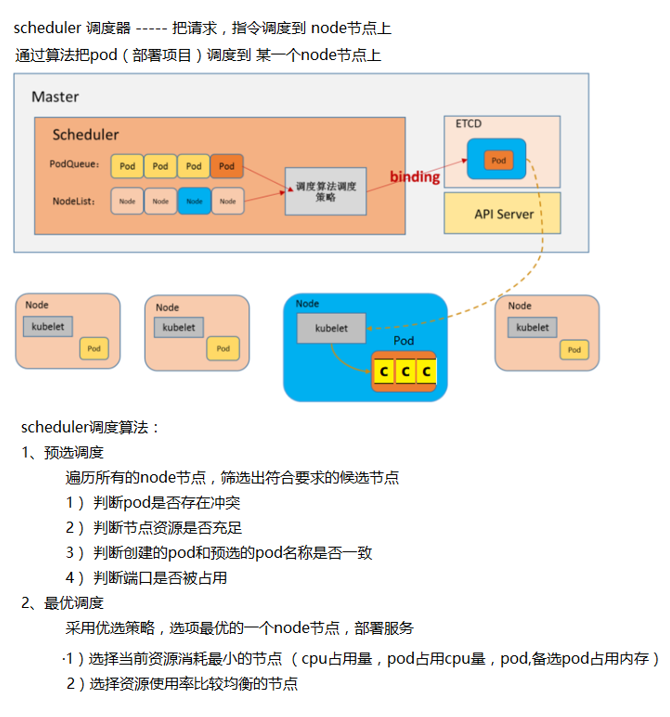
## Master节点组件



## Controllers



## Scheduler



## node节点组件



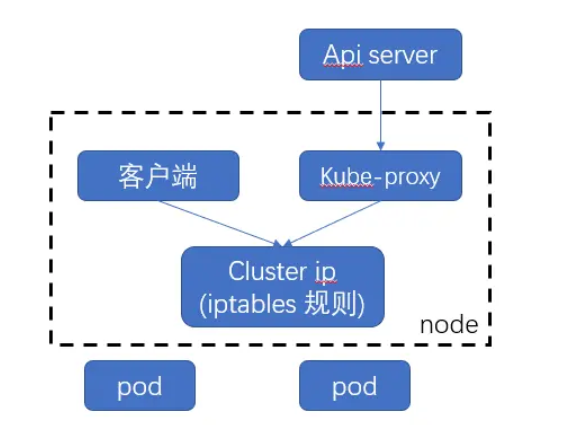
## Kubelet

【kubelet负责管理pods和它们上面的容器，images镜像、volumes、etc】，kubelet是Master在Node节点上的Agent，每个节点都会启动 kubelet进程，用来处理 Master 节点下发到本节点的任务，管理本机运行容器的生命周期，比如创建容器、Pod挂载数据卷、下载secret、获取容器和节点状态等工作。kubelet将每个Pod转换成一组容器。

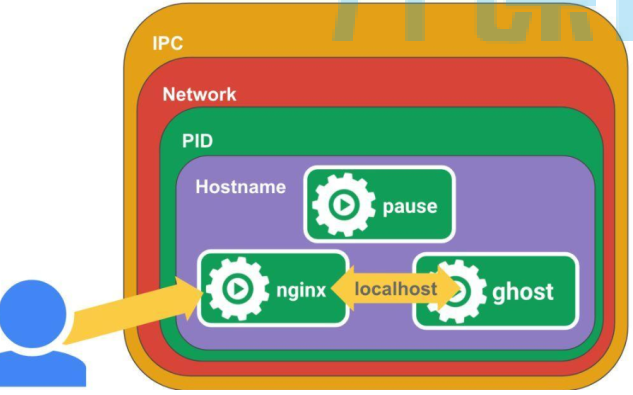
## kube-proxy

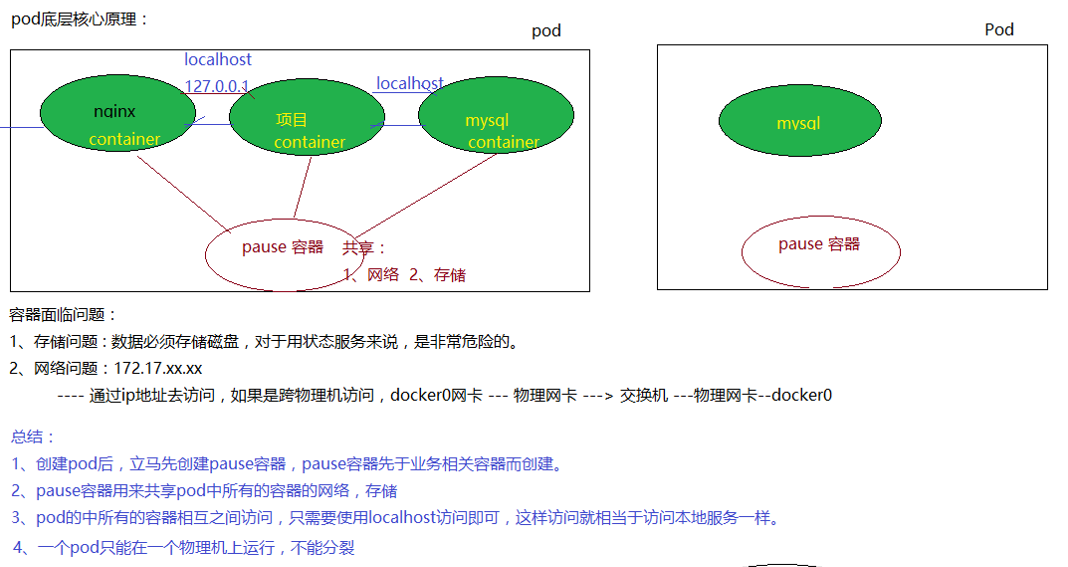
在Node节点上实现Pod网络代理，维护网络规则和四层负载均衡工作，kube-proxy 本质上，类似一个反向代理. 我们可以把每个节点上运行的 kube-proxy 看作 service 的透明代理兼LB.

kube-proxy 监听 apiserver 中service 与Endpoint 的信息, 配置iptables 规则,请求通过iptables 直接转发给 pod.

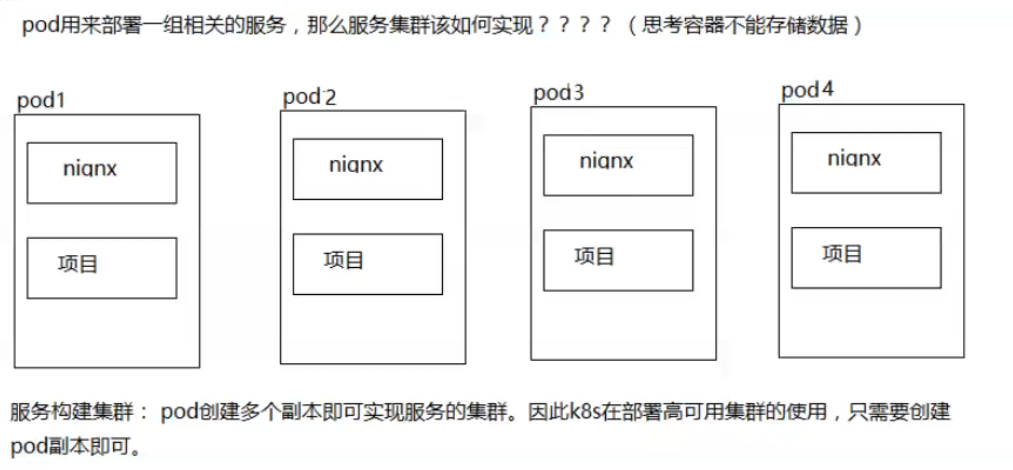


Pod是最小部署单元，一个Pod有一个或多个容器组成，Pod中容器共享存储和网络，在同一台Docker  
主机上运行  
1）pod基本结构

  
**Pause的作用：**  
我们看下在node节点上都会起很多pause容器，和pod是一一对应的。  
每个Pod里运行着一个特殊的被称之为Pause的容器，其他容器则为业务容器，这些业务容器共享Pause容器的网络栈和Volume挂载卷，因此他们之间通信和数据交换更为高效，在设计时我们可以充分利用这一特性将一组密切相关的服务进程放入同一个Pod中。同一个Pod里的容器之间仅需通过  
localhost就能互相通信。

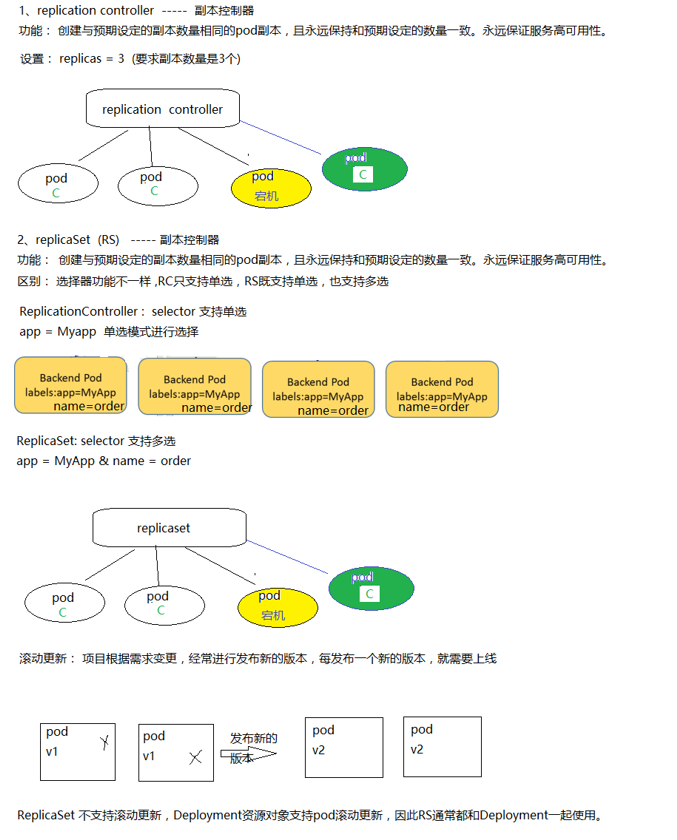


K8s部署集群

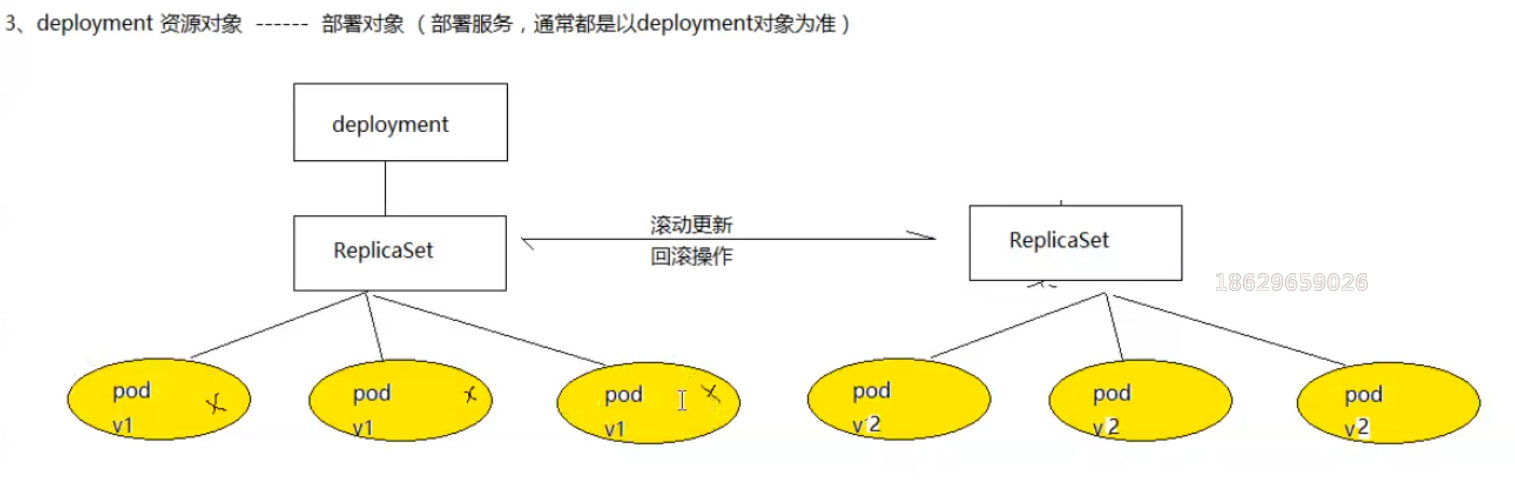


## 核心资源对象：

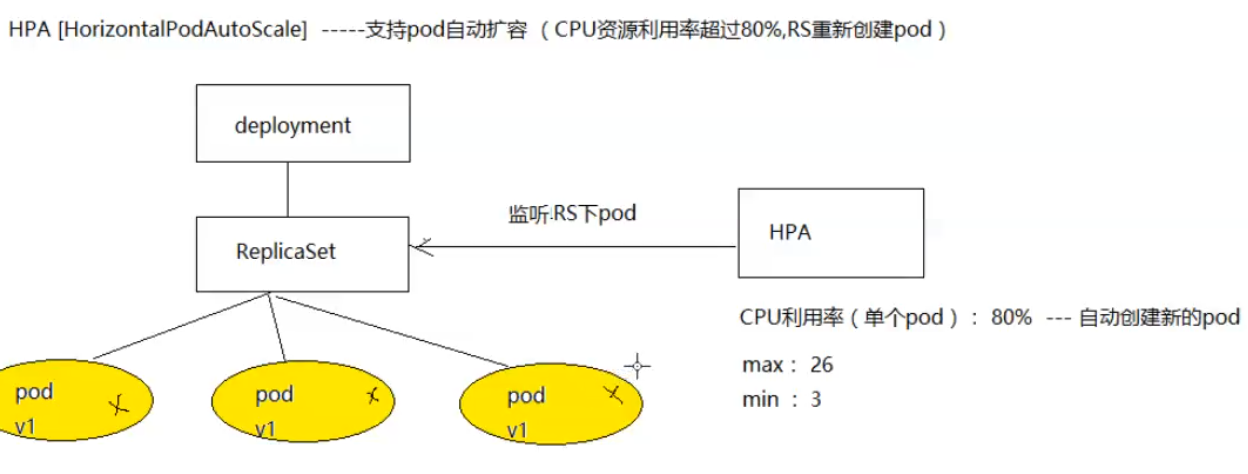
RC/RS

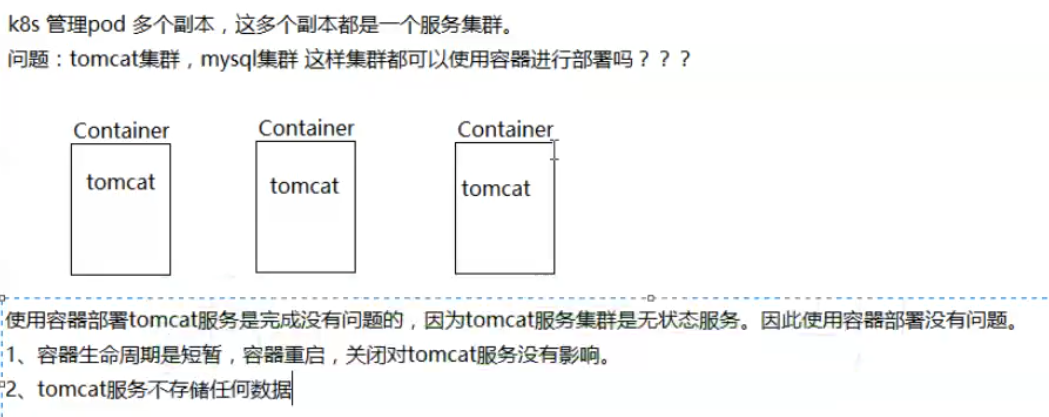


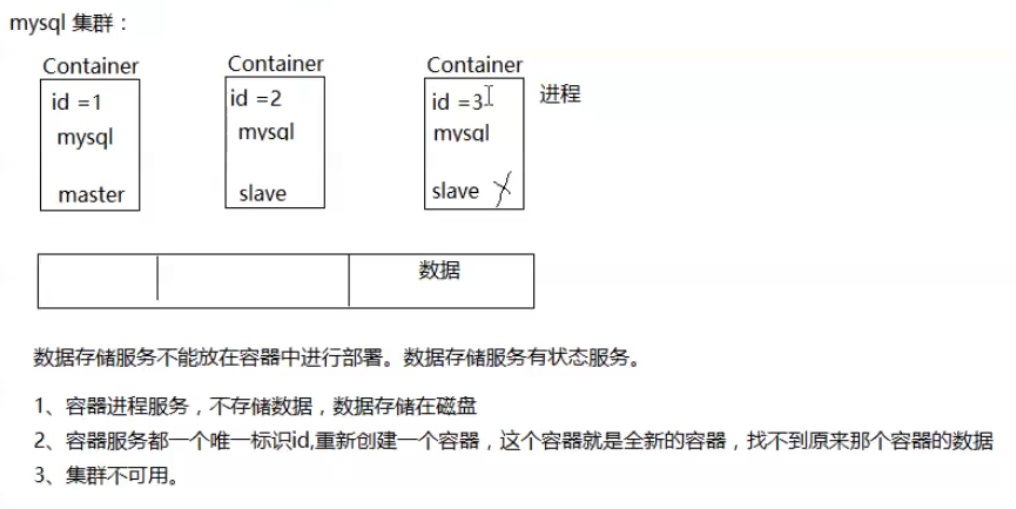
Deployment为Pod和ReplicaSet 提供了一个 声明式定义方法，用来替代以前的 ReplicationController  
来方便的管理应用。  
典型的应用场景：  
（1）、定义Deployment 来创建 Pod 和 ReplicaSet  
（2）、滚动升级和回滚应用  
（3）、扩容和索容  
（4）、暂停和继续 Deployment  
Deployment不仅仅可以滚动更新，而且可以进行回滚，如果发现升级到V2版本后，发现服务不可用，  
可以回滚到V1版本。



HPA(HorizontalPodAutoScale)  
Horizontal Pod Autoscaling 仅适用于 Deployment 和 ReplicaSet,在V1版本中仅支持根据Pod的CPU利用率扩容，在vlalpha版本中，支持根据内存和用户自定义的metric扩缩容



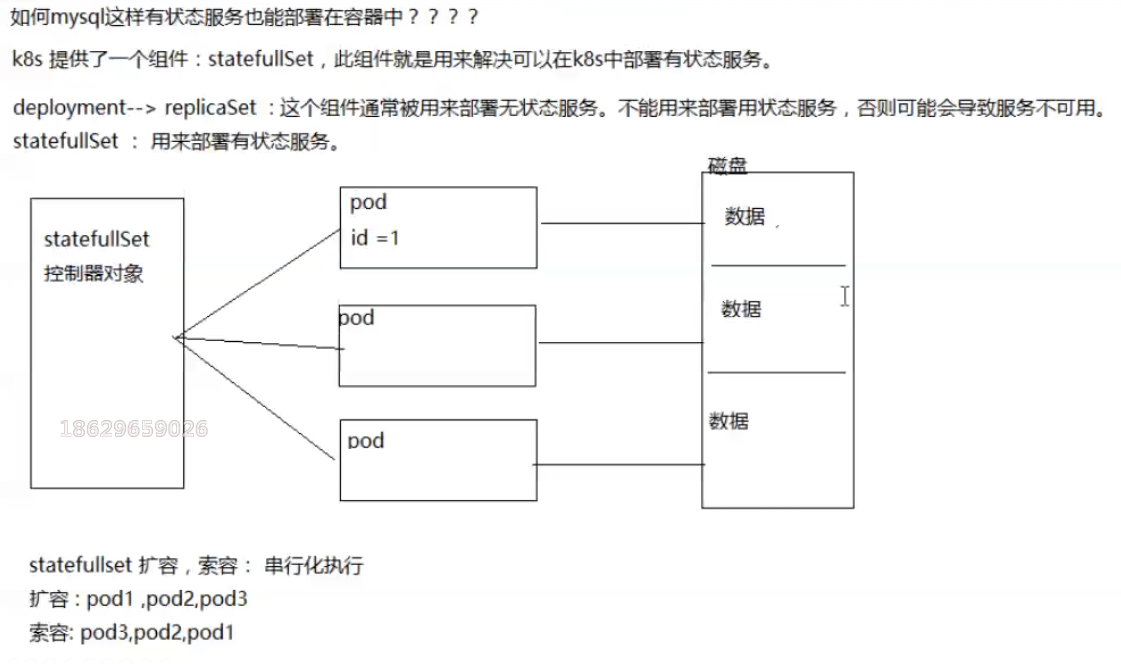




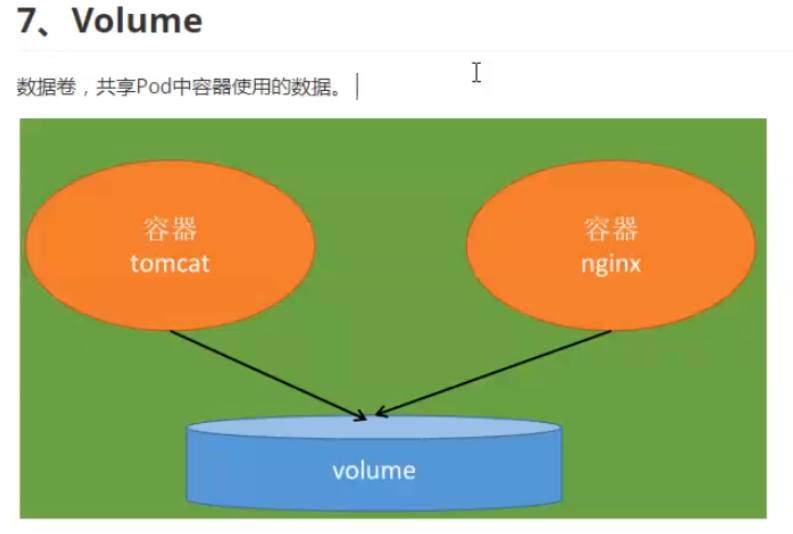
有无状态服务？



StatefullSet 是为了解决有状态服务的问题（对应Deployments 和 ReplicaSets 是为无状态服务而设计），其应用场景包括：  
（1）稳定的持久化存储，即Pod重新调度后还是能访问的相同持久化数据，基于PVC来实现  
（2）稳定的网络标志，及Pod重新调度后其 PodName 和 HostName 不变，基于Headlesss  
Service（即没有 Cluster IP 的 Service）来实现。  
（3）有序部署，有序扩展，即Pod是有顺序的，在部署或者扩展的时候要依据定义的顺序依次进行（即从 0 到 N-1,在下一个Pod运行之前所有之前的Pod必须都是Running 和 Ready 状态），基于 init  
containers 来实现。  
（4）有序收缩，有序删除（即从N-1 到 0）



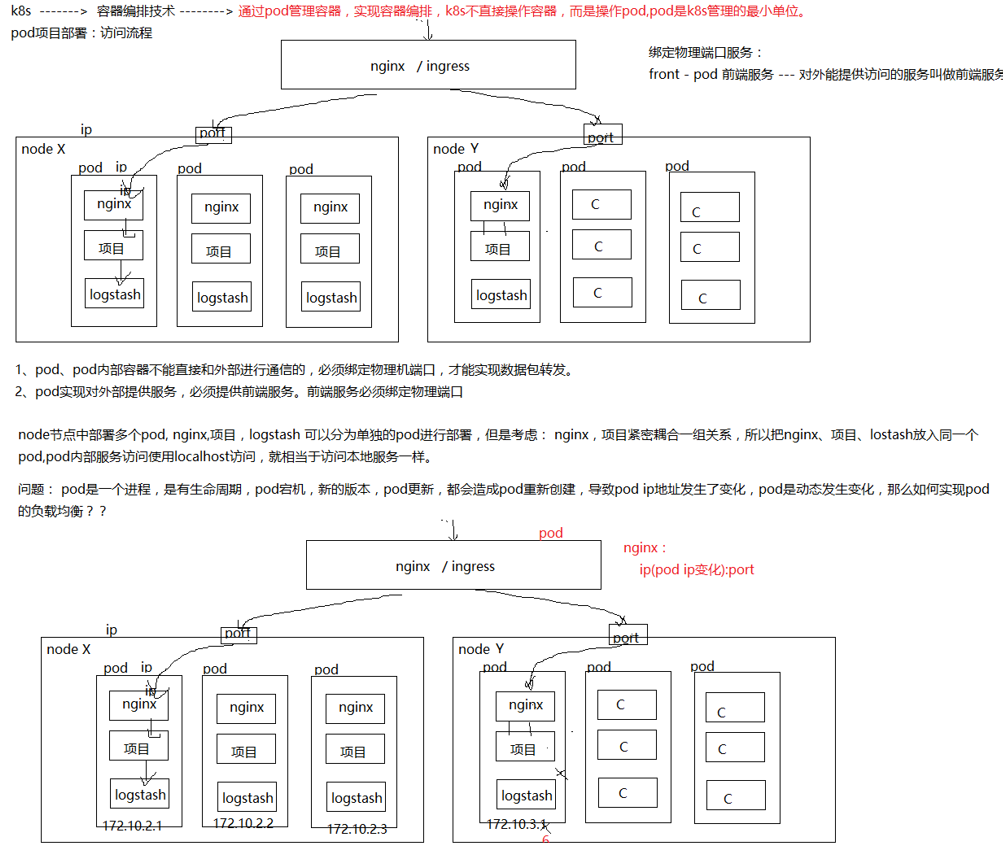
DaemonSet确保全部（或者一些 [ node打上污点（可以想象成一个标签）,pod如果不定义容忍这个污点，那么pod就不会被调度器分配到这个node ]）



Kubernetes中任意API对象都是通过Label进行标识，Label的实质是一系列的Key/Value键值对，其中  
key于value由用户自己指定。  
Label可以附加在各种资源对象上，如Node、Pod、Service、RC等，一个资源对象可以定义任意数量  
的Label，同一个Label也可以被添加到任意数量的资源对象上去。  
Label是Replication Controller和Service运行的基础，二者通过Label来进行关联Node上运行的Pod。



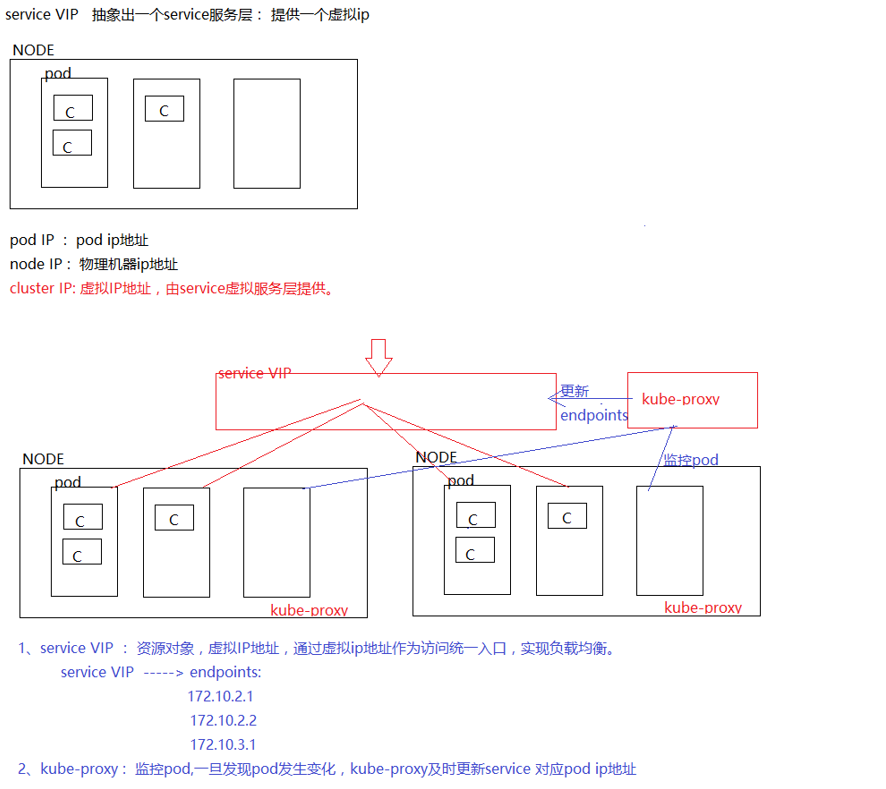
pod访问：



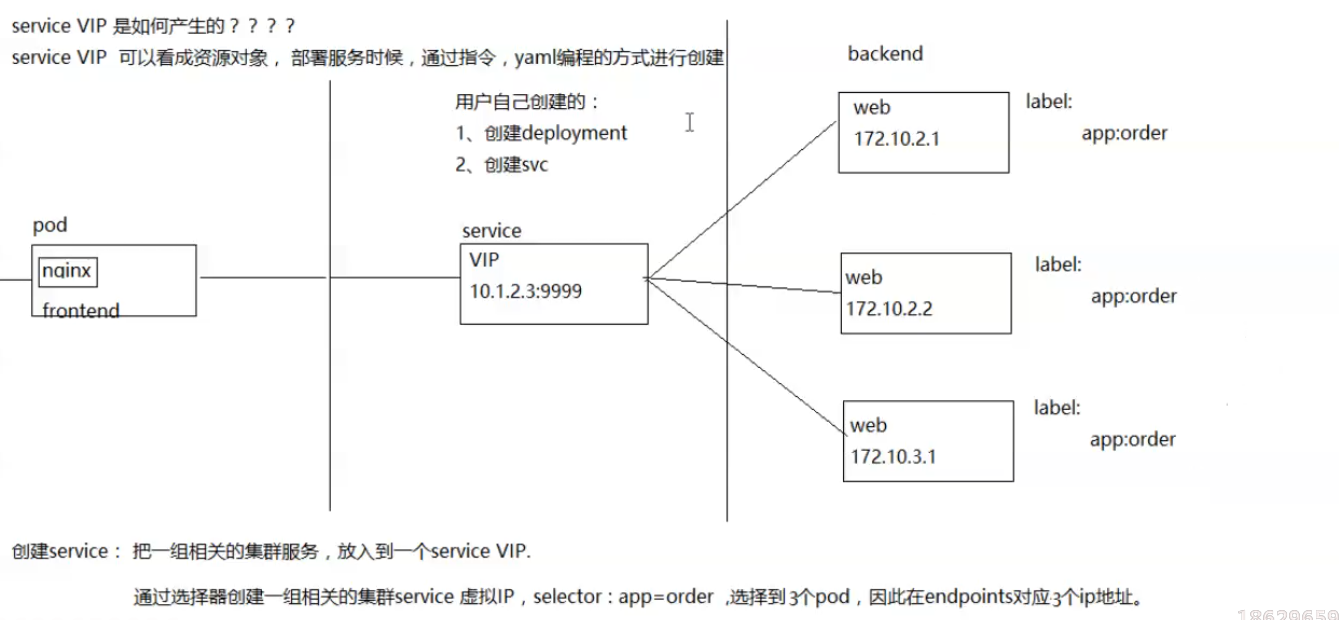
什么是service

Service是一个抽象的概念。它通过一个虚拟的IP的形式(VIPs)，映射出来指定的端口，通过代理客户端发来的请求转发到后端一组Pods中的一台（也就是endpoint）  
Service定义了Pod逻辑集合和访问该集合的策略，是真实服务的抽象。Service提供了统一的服务访问入口以及服务代理和发现机制，关联多个相同Label的Pod，用户不需要了解后台Pod是如何运行。 外部系统访问Service的问题: **->** 首先需要弄明白Kubernetes的三种IP这个问题 **-** Node IP：Node节点的IP地址 **-** Pod IP： Pod的IP地址 **-** Cluster IP：Service的IP地址

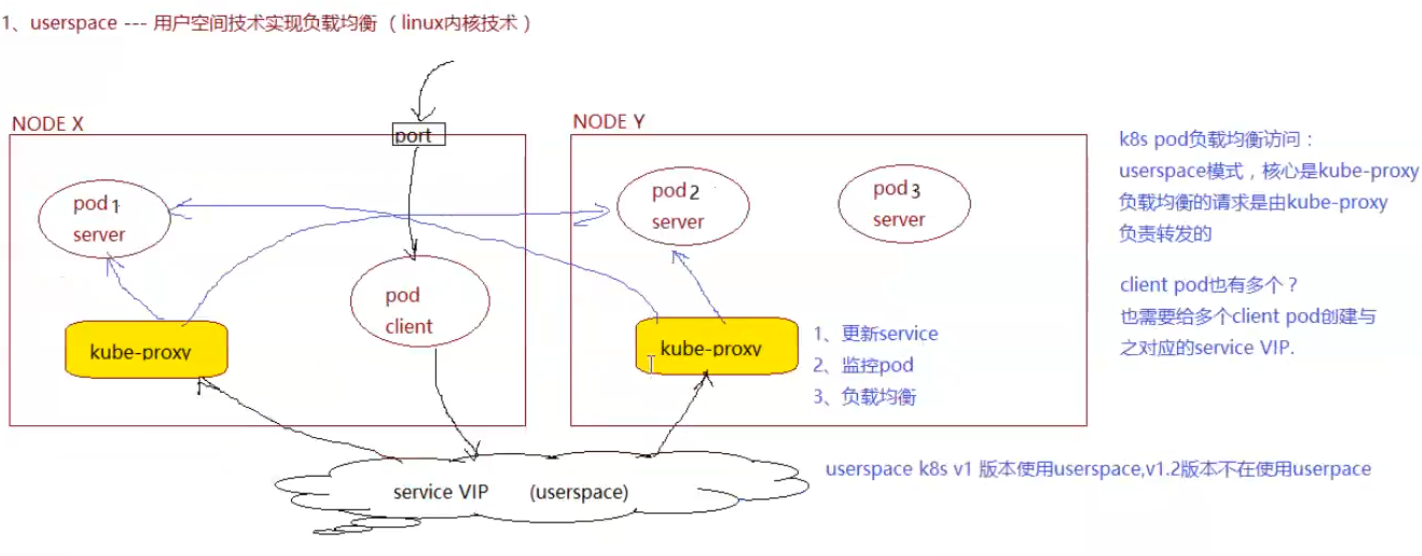
Service服务发现和VIP:

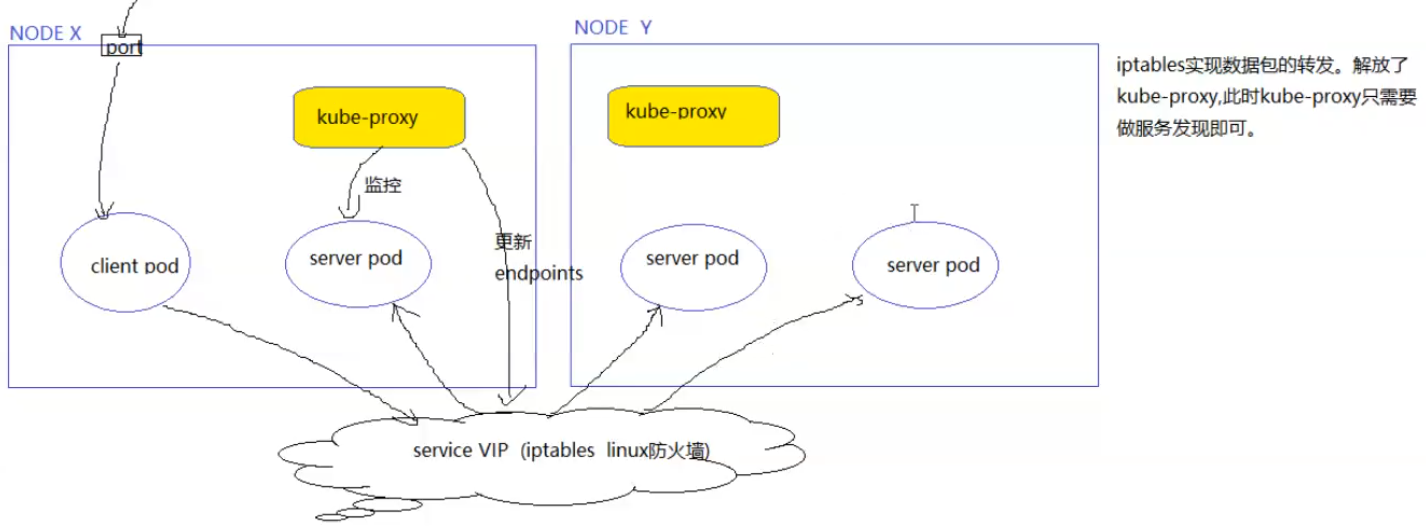


service如何产生：



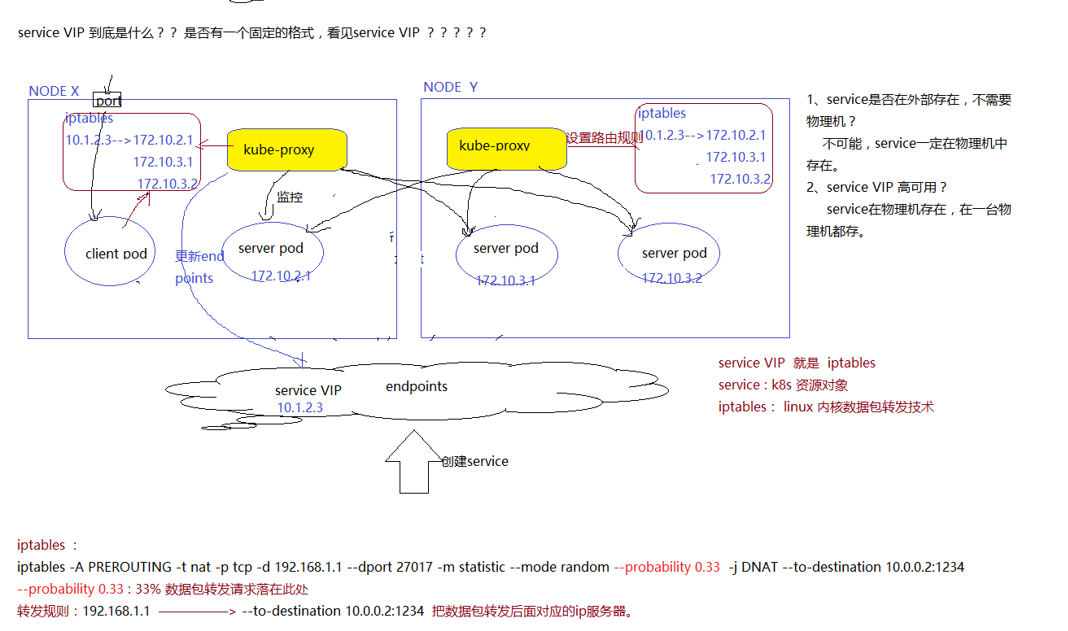
负载均衡：3种技术





ipvs和iptables转发方式一样，只不过负载均衡算法更丰富一些。

service ip



## 8.K8s环境搭建：

### 8.1.基础环境：

yum install -y conntrack ntpdate ntp ipvsadm ipset jq iptables curl sysstat libseccomp wget vim net-tools git iproute lrzsz base-completion tree bridge-utils unzip build-utils gcc

升级内核：k8s在内核4.4以上比较稳定

rpm -Uvh <http://www.elrepo.org/elrepo-release-7.0-4.el7.elrepo.noarch.rpm>

yum --enablerepo=elrepo-kernel install -y kernel-lt

sudo awk -F\' '$1=="menuentry " {print i++ " : " $2}' /etc/grub2.cfg

grub2-set-default 0

reboot

uname -a

### 8.2.安装docker：

（1）yum 包更新到最新

sudo yum update

（2）安装需要的软件包， yum-util 提供yum-config-manager功能，另外两个是devicemapper驱动依赖的

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

（3）设置yum源为阿里云

sudo yum-config-manager --add-repo http://mirrors.aliyun.com/dockerce/linux/centos/docker-ce.repo

（4）安装docker

sudo yum install docker-ce

（5）安装后查看docker版本

docker -v

### 8.3设置ustc的镜像

ustc是老牌的linux镜像服务提供者了，还在遥远的ubuntu 5.04版本的时候就在用。ustc的docker镜像加速器速度

很快。ustc docker mirror的优势之一就是不需要注册，是真正的公共服务。

https://lug.ustc.edu.cn/wiki/mirrors/help/docker

编辑该文件：

vi /etc/docker/daemon.json

在该文件中输入如下内容：

{

"registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]

}

### 8.4安装k8s

设置k8s yum源

cat <<EOF > kubernetes.repo  
[kubernetes]  
name=Kubernetes  
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64  
enabled=1  
gpgcheck=1  
repo\_gpgcheck=1  
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg  
EOF  
sudo mv kubernetes.repo /etc/yum.repos.d/

<https://www.cnblogs.com/jason-zhao/p/14663420.html>

安装k8s组件

yum install -y kubeadm-1.15.1 kubelet-1.15.1 kubectl-1.15.1

kubectl是Kubernetes集群的命令行工具，通过kubectl能够对集群本身进行管理，并能够在集群上进行容器化应用的安装部署。运行kubectl命令的语法如下所示：

kubectl [command] [TYPE] [NAME] [flags]

<https://blog.csdn.net/bbwangj/article/details/80814568>

Kubelet组件运行在Node节点上，维持运行中的Pods以及提供kuberntes运行时环境

<https://blog.csdn.net/jettery/article/details/78891733>

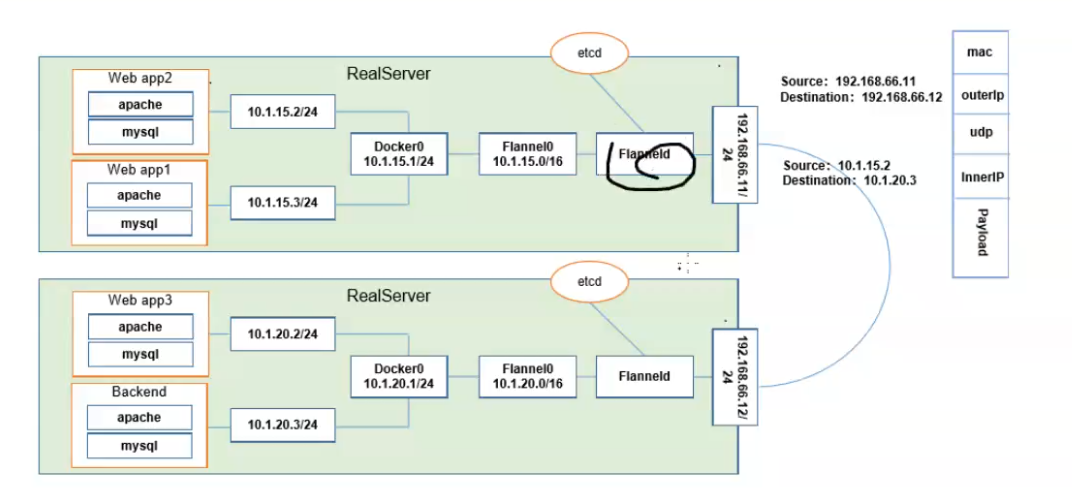
设置开机启动：

systemctl enable kubelet && systemctl start kubelet

mkdir -p $HOME/.kube

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

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



如果说某次执行kubeadm init初始化k8s集群失败了，在下一次执行kubeadm init初始化语句之前，先执行kubeadm reset命令。这个命令的作用是重置节点，大家可以把这个命令理解为：上一次kubeadm init初始化集群操作失败了，该命令清理了之前的失败环境。

<https://blog.csdn.net/curry10086/article/details/107579113>