Day04 kubernetes笔记

课程主要内容：

1、pod如何实现内部容器对外网提供服务？？

2、一组pod副本如何实现服务发现？？？

3、service VIP

4、service 如何和pod 进行关联呢？？？

5、service如何实现服务的发现呢？？？

6、service 和 业务服务的关系？？？

7、service 负载均衡的几种策略

课程内容回顾

1、常用的（已经学习过的）资源对象有哪些？？？

1）rc

2) rs 副本控制器

3) deployment 无状态服务部署对象

4) statefulset 有状态服务部署对象

5) daemonset

6) volume

7) labels

8) pod k8s管理最小单元

9) hpa

2、pod部署服务有几种模式？？？  
 pod内部只部署一个容器（通常情况下，一个pod内部就部署一个服务）

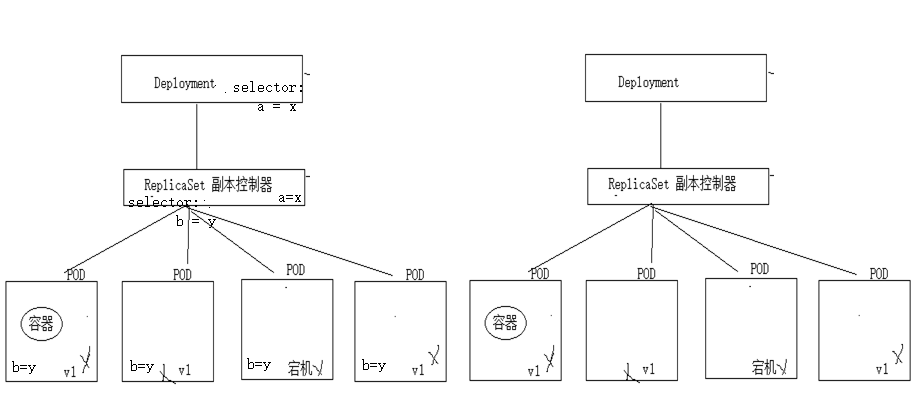
Pod内部部署多个容器： 多个容器服务必须紧密耦合的关系，才允许部署在一个pod内部，在一个请求链路上的服务，紧密的调用关系，这样上下游服务，叫做紧密耦合的服务。

3、无状态服务，有状态服务分别使用哪个资源对象进行部署管理？

无状态服务： deployment

有状态服务： statefulset

4、deployment 如何 replicaset 关联起来的？ ReplicaSet如何和pod关联起来的？？



# 1、外网访问pod

## 1.1、Pod结构

Pod的特点：容器

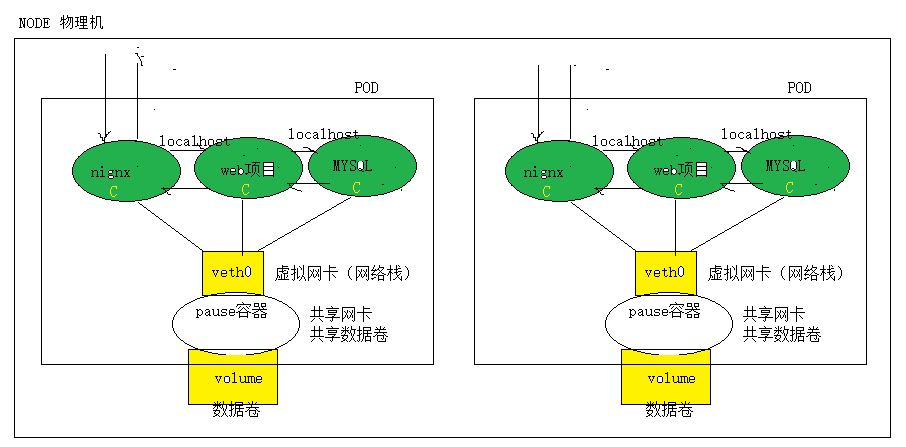
1、有自己的IP地址

2、有自己的hostname

3、有自己的端口

Pod实际上也相当于是一个独立的容器（虚拟机器），而这个pod容器内部封装的是由docker引擎所创建的容器，可以理解为pod就是一个虚拟化分组，pod内存可以存储一个或者多个容器。

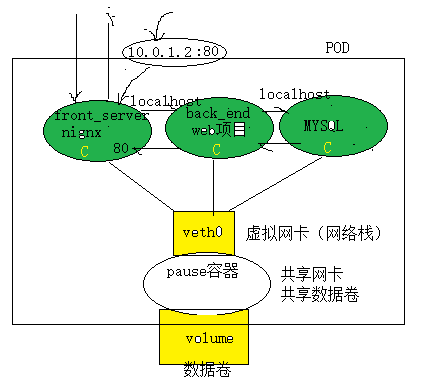
思考： pod运行在node节点（物理机）内部，也就是pod实际上是物理机内部的一个进程，pod如何对外网提供服务？？？



问题： 是否可以通过外网直接访问pod的ip：port ???

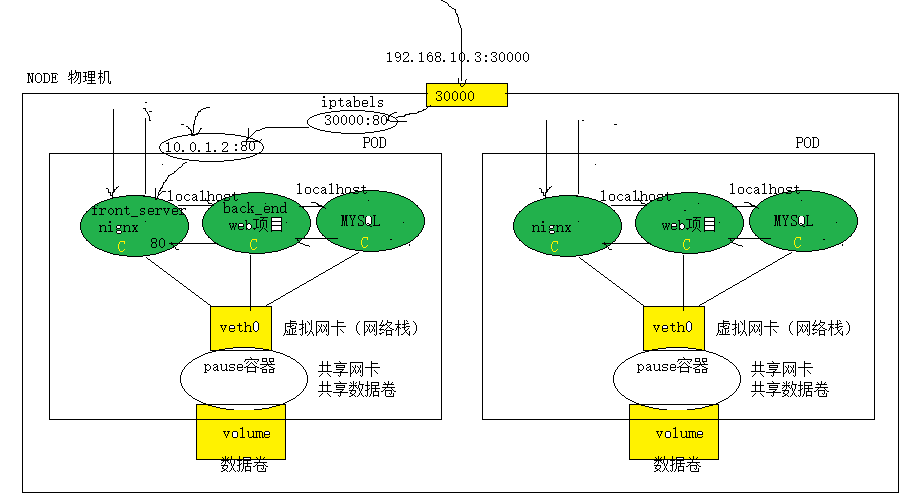
当然不可以的，pod只是一个进程，没有与之对应的物理实体。要在现实的网络中进行通信，必须要有与之对应的物理实体。也就是说pod要想和外部进行通信，必须借助于物理网卡。

## 1.2、内部访问



通过pod的ip地址，加上端口可以在node节点局域网内部实现pod的访问，pod的端口是front\_server的端口，访问pod实现请求转发，直接把请求转发给front\_server即可。

## 1.3、外网服务

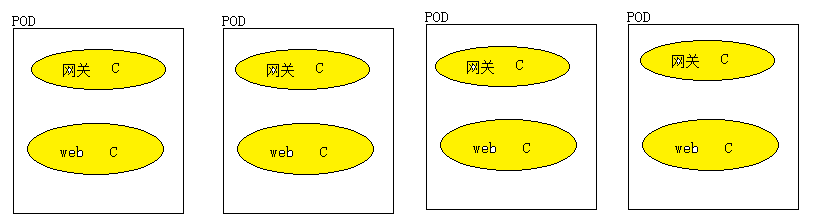


要和内部的pod进行通信，必须在物理机上开辟一个端口，通过物理机的端口对数据包（请求）进行转发，把请求转发给pod。实现pod内部服务外网访问。

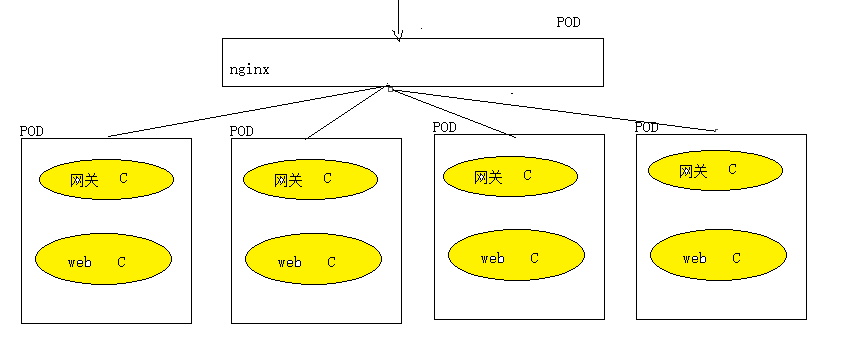
## 1.4、服务集群

使用k8s部署服务，所有的服务都部署在pod内部的容器中，服务集群就需要多个pod副本实现。

思考：那么多个pod副本实现负载均衡访问，该如何实现呢？？？



解决方案： 使用nginx实现负载均衡？？



使用nginx实时多个pod副本直接负载均衡访问。

问题：

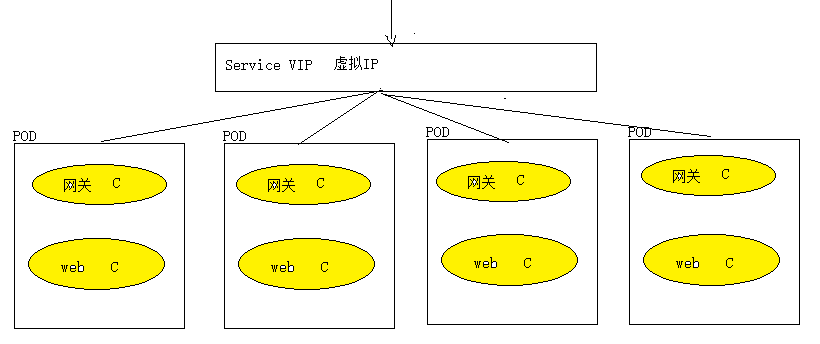
Pod是一个服务进程，有生命周期，或者pod随时可能宕机，k8s立马对pod进行重建，但是此时重建的pod的ip，hostname都发生的变化。此时nginx还能访问重建的pod吗？？

总结： nginx作为pod副本的负载均衡服务器，无法发现k8s重建，或者新建的pod,也就是说k8s重建的，新建的pod，使用nginx无法访问。因此使用nginx作为pod副本负载均衡访问服务，无法实现的。

# 2、Service VIP

## 2.1、Service VIP 是什么

Service VIP 是k8s提供一个虚拟IP, service 就是一个虚拟ip的资源对象。Service VIP就相当于是服务网关，所有的请求都要被service VIP进行拦截，然后进行转发，它屏蔽了底层 pod Ip,hostname变化所造成的影响，使得用户不需要关心pod在底层到底是如何变化的，或者不需要关心pod的ip,hostname是如何变化。



注意：service VIP 一旦创建就不会发生变化，service VIP高可用性由etcd来保证，service VIP作为一组pod业务的统一网关入口。

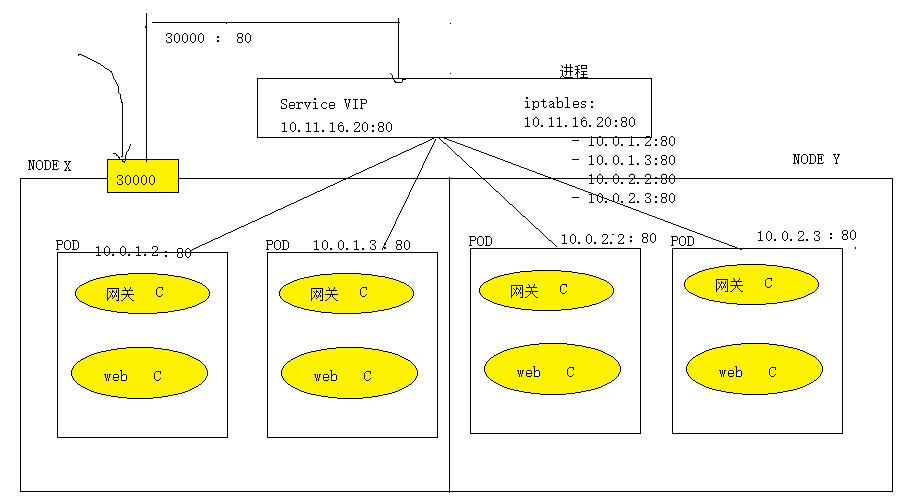
## 2.2、如何实现pod负载均衡

K8s 核心资源对象几种IP地址分类：

1、NODE IP ----- 物理机节点IP地址

2、POD IP ----- POD 进行IP地址

3、cluster IP ----- 虚拟机IP,由k8s抽象出的service资源对象具有的IP地址，此ip地址类型默认就是clusterIP, 此类型只能在局域网进行通信，无法对外网提供服务。



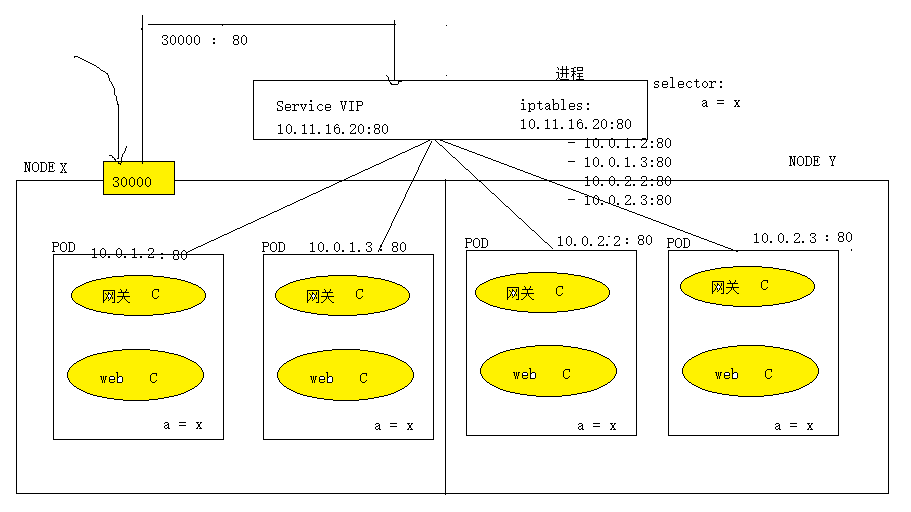
1、在物理机开辟一个端口，实现数据包（请求）转发

2、创建service VIP 资源对象，实现物理端口和service VIP 端口关系映射，从而从物理机端口转发的请求自动转发给service VIP

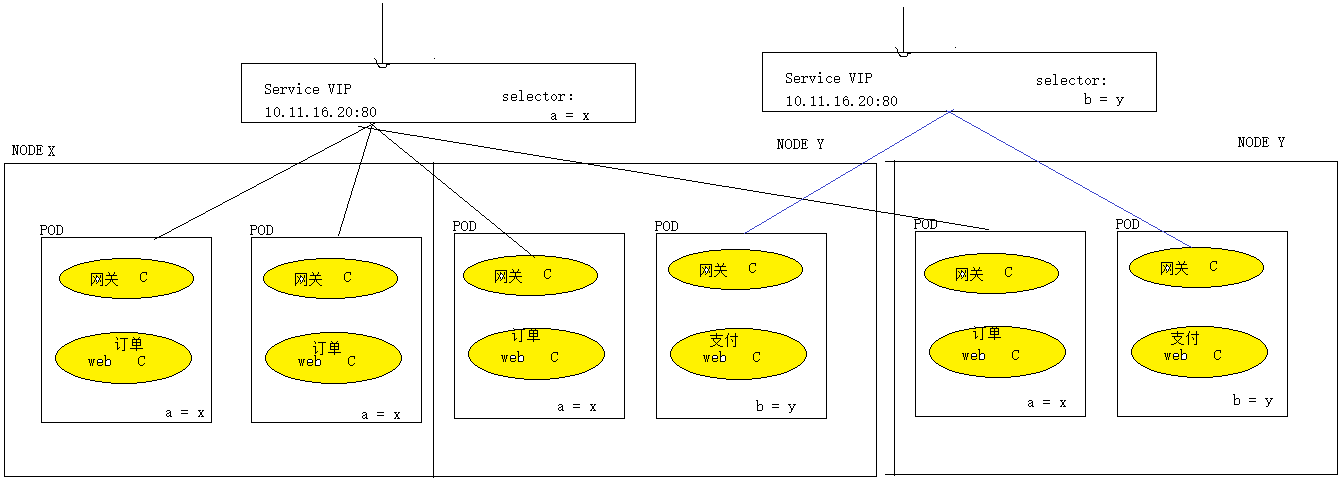
3、service 资源对象通过转发策略，把请求转发给相应的Pod.

## 2.3、Service关联pod

Service和pod如何关联起来的，service是通过标签选择器选择一组相关的pod,创建service，也就是说service只对一组pod的相关的pod副本提供服务。多个业务pod,就应该有多个service。



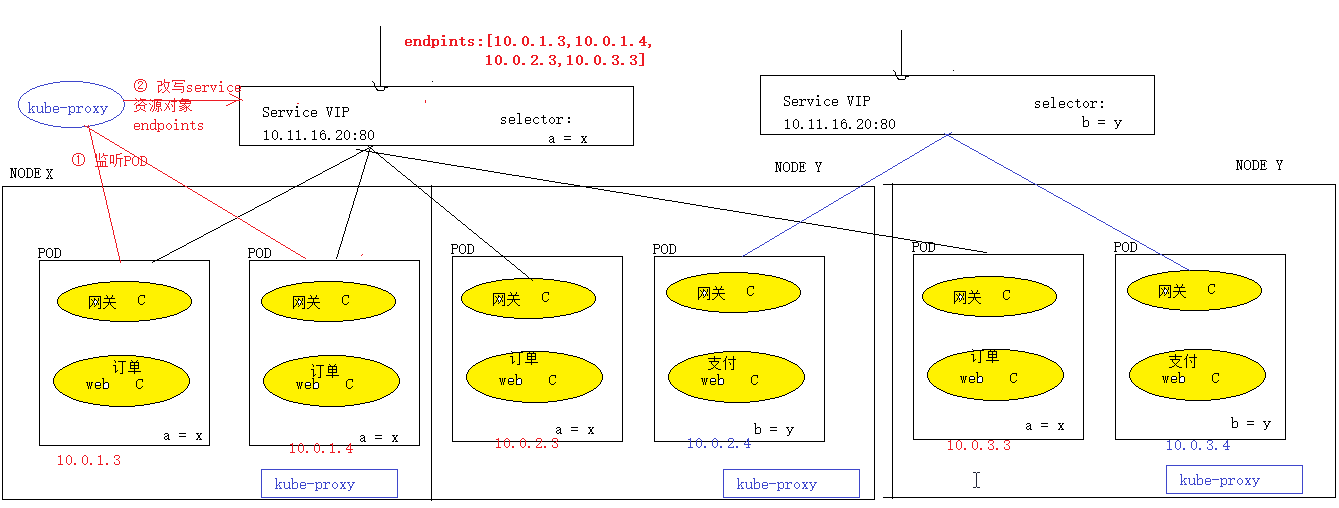
一个业务组pod副本，对应一个service。



## 2.4、Pod服务发现

Pod宕机，当k8s重建pod后，pod的ip地址，hostname发现了变化，service VIP 可以感知到pod ip，hostname已经发生了变化。

K8s node节点中提供一个核心组件：kube-proxy,这个代理组件主要用来实现服务发现，路由规则改写。



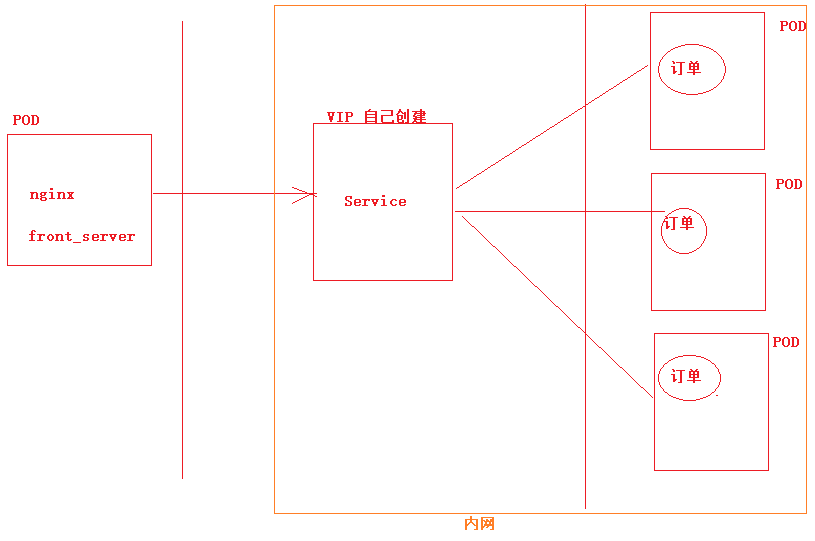
注意： kube-proxy在每一个node节点都存在一个进程，每一个node节点中kube-proxy都会监控pod的ip地址变化，发现pod的ip地址发生了变化，把变化的ip的更新到service资源对象的endpoints. 使得service vip可以及时感知到pod ip的变化，从而可以实现更好的，及时的，稳定的访问。

总结： k8s所有的资源对象都存储在etcd, 所有的资源对象都可以用yaml资源配置文件方式进行表示，因此这些资源以yaml方式存储。

# 3、负载均衡策略

## 3.1、Service VIP 是如何产生的？？

Service 资源对象是我们自己创建的，service VIP是k8s的核心资源对象，资源对象名称：Service,服务部署时候就必须创建service，可以通过k8s指令，yaml文件方式进行创建（可视化界面）



部署服务：

1、先创建pod

2、再创建service

## 3.2、负载策略

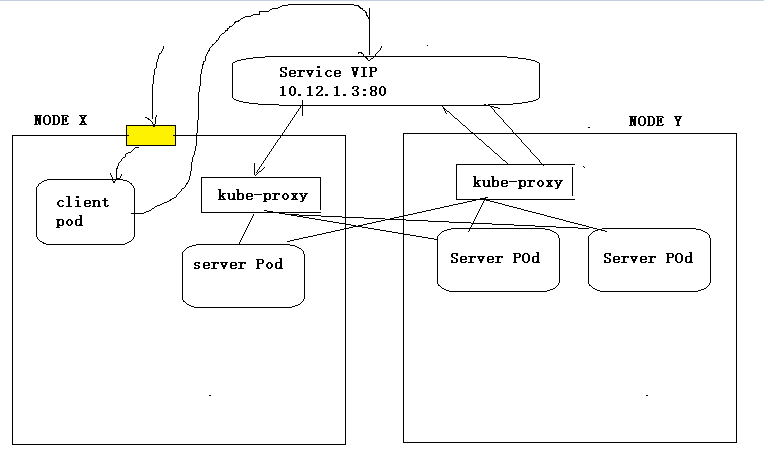
在k8s提供给pod副本负载均衡访问的负载策略中，有3种策略

1）、userspace

2)、iptables

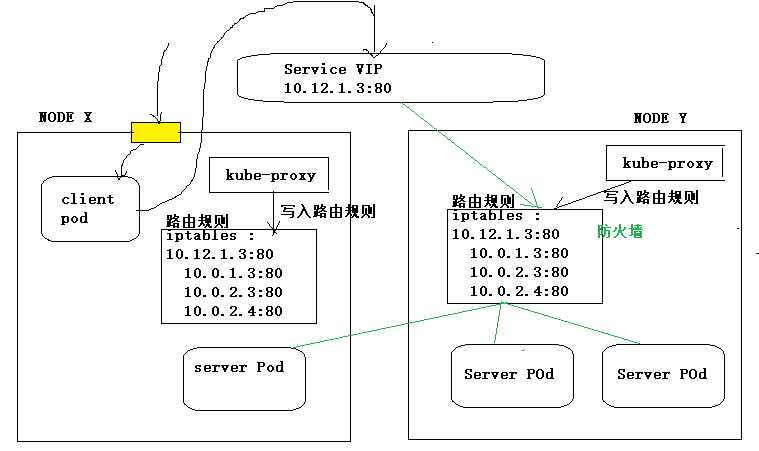
3)、ipvs

1、第一种策略： userspace策略



访问pod集群副本的时候，负载均衡是由kube-proxy来实现的，而不是service VIP来实现的。但是kube-proxy不仅仅需要实现负载均衡，还需要实现服务发现及注册，实现路由规则的改写。因此kube-proxy负载压力较大，性能不会太好。

## 3.3、Iptables实现负载均衡

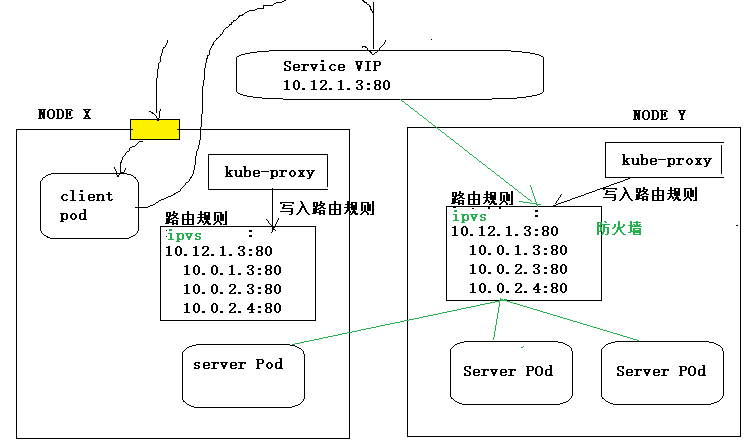


通过物理端口发送请求，由客户端pod接受后，把请求转发给后端服务集群的serviceVIP,有serviceVIP把请求转发给相应的pod.当请求发送到相应的node节点的时候，请求就会被iptables路由规则拦截，根据iptables路由规则进行请求分发。

问题： iptables 请求分发有几种策略？？？

在Linux中使用iptables完成tcp的负载均衡有两种模式：随机、轮询

## 3.4、Ipvs



IPVS支持的负载均衡算法有这么几种：

- rr: 轮询

- lc: 最小连接数

- dh: 目的地址hash

- sh: 源地址hash

- sed: 最短期望延迟

- nq: 无须队列等待

下节课开始： 实战 （结合前面4节课理论）