**LVS（Linux Virtual Server）即Linux虚拟服务器，是由章文嵩博士主导的开源负载均衡项目，目前LVS已经被集成到Linux内核模块中。该项目在Linux内核中实现了基于IP的数据请求负载均衡调度方案**

**三种工作模式：**

**1、基于NAT的LVS模式负载均衡**

**2、基于TUN的LVS负载均衡**

**3、基于DR的LVS负载均衡**

**LVS负载均衡调度算法**

**根据前面的介绍，我们了解了LVS的三种工作模式，但不管实际环境中采用的是哪种模式，调度算法进行调度的策略与算法都是LVS的核心技术，LVS在内核中主要实现了一下十种调度算法。**

**1.轮询调度**

**轮询调度（Round Robin 简称'RR'）算法就是按依次循环的方式将请求调度到不同的服务器上，该算法最大的特点就是实现简单。轮询算法假设所有的服务器处理请求的能力都一样的，调度器会将所有的请求平均分配给每个真实服务器。**

**2.加权轮询调度**

**加权轮询（Weight Round Robin 简称'WRR'）算法主要是对轮询算法的一种优化与补充，LVS会考虑每台服务器的性能，并给每台服务器添加一个权值，如果服务器A的权值为1，服务器B的权值为2，则调度器调度到服务器B的请求会是服务器A的两倍。权值越高的服务器，处理的请求越多。**

**3.最小连接调度**

**最小连接调度（Least Connections 简称'LC'）算法是把新的连接请求分配到当前连接数最小的服务器。最小连接调度是一种动态的调度算法，它通过服务器当前活跃的连接数来估计服务器的情况。调度器需要记录各个服务器已建立连接的数目，当一个请求被调度到某台服务器，其连接数加1；当连接中断或者超时，其连接数减1。**

**（集群系统的真实服务器具有相近的系统性能，采用最小连接调度算法可以比较好地均衡负载。)**

**4.加权最小连接调度**

**加权最少连接（Weight Least Connections 简称'WLC'）算法是最小连接调度的超集，各个服务器相应的权值表示其处理性能。服务器的缺省权值为1，系统管理员可以动态地设置服务器的权值。加权最小连接调度在调度新连接时尽可能使服务器的已建立连接数和其权值成比例。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。**

**5.基于局部的最少连接**

**基于局部的最少连接调度（Locality-Based Least Connections 简称'LBLC'）算法是针对请求报文的目标IP地址的 负载均衡调度，目前主要用于Cache集群系统，因为在Cache集群客户请求报文的目标IP地址是变化的。这里假设任何后端服务器都可以处理任一请求，算法的设计目标是在服务器的负载基本平衡情况下，将相同目标IP地址的请求调度到同一台服务器，来提高各台服务器的访问局部性和Cache命中率，从而提升整个集群系统的处理能力。LBLC调度算法先根据请求的目标IP地址找出该目标IP地址最近使用的服务器，若该服务器是可用的且没有超载，将请求发送到该服务器；若服务器不存在，或者该服务器超载且有服务器处于一半的工作负载，则使用'最少连接'的原则选出一个可用的服务器，将请求发送到服务器。**

**6.带复制的基于局部性的最少连接**

**带复制的基于局部性的最少连接（Locality-Based Least Connections with Replication 简称'LBLCR'）算法也是针对目标IP地址的负载均衡，目前主要用于Cache集群系统，它与LBLC算法不同之处是它要维护从一个目标IP地址到一组服务器的映射，而LBLC算法维护从一个目标IP地址到一台服务器的映射。按'最小连接'原则从该服务器组中选出一一台服务器，若服务器没有超载，将请求发送到该服务器；若服务器超载，则按'最小连接'原则从整个集群中选出一台服务器，将该服务器加入到这个服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服务器从服务器组中删除，以降低复制的程度。**

**7.目标地址散列调度**

**目标地址散列调度（Destination Hashing 简称'DH'）算法先根据请求的目标IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且并未超载，将请求发送到该服务器，否则返回空。**

**8.源地址散列调度U**

**源地址散列调度（Source Hashing 简称'SH'）算法先根据请求的源IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且并未超载，将请求发送到该服务器，否则返回空。它采用的散列函数与目标地址散列调度算法的相同，它的算法流程与目标地址散列调度算法的基本相似。**

**9.最短的期望的延迟**

**最短的期望的延迟调度（Shortest Expected Delay 简称'SED'）算法基于WLC算法。举个例子吧，ABC三台服务器的权重分别为1、2、3 。那么如果使用WLC算法的话一个新请求进入时它可能会分给ABC中的任意一个。使用SED算法后会进行一个运算**

**A：（1+1）/1=2 B：（1+2）/2=3/2 C：（1+3）/3=4/3 就把请求交给得出运算结果最小的服务器。**

**10.最少队列调度**

**最少队列调度（Never Queue 简称'NQ'）算法，无需队列。如果有realserver的连接数等于0就直接分配过去，不需要在进行SED运算。**

# 实验：搭建Keepalived+LVS+NFS集群

**准备工作：**

**lb1: 192.168.7.10**

**lb2: 192.168.7.11**

**web1: 192.168.7.12 vmnet8**

**192.168.10.12 vmnet1**

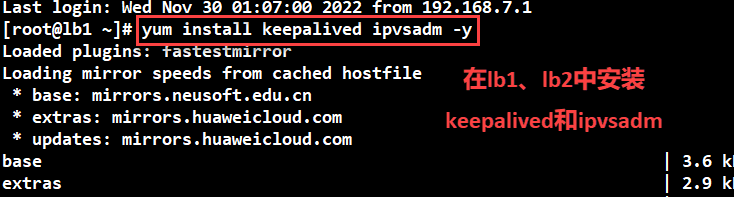
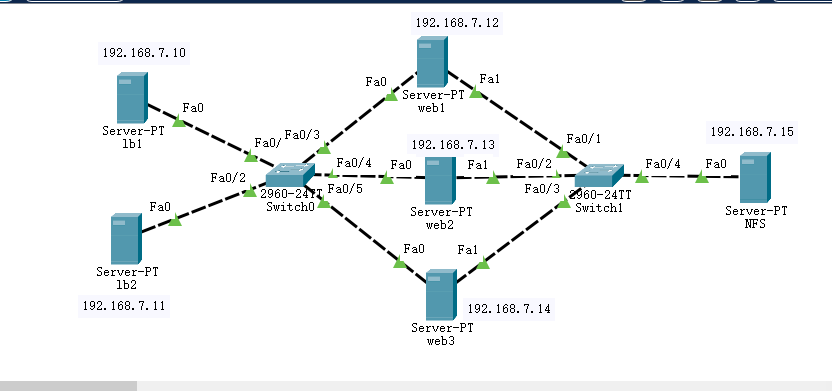
**web2: 192.168.7.13 vmnet8**

**192.168.10.13 vmnet1**

**web3: 192.168.7.14 vmnet8**

**192.168.10.14 vmnet1**

**nfs: 192.168.10.100 vmnet1**

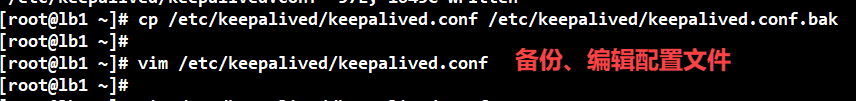


**keepalived可以实现高可用：**

keepalive：vrrp协议在Linux主机上以守护进程方式的实现； 能够根据配置文件生成ipvs规则，并对各RS的健康做检测；vrrp\_script, vrrp\_track；

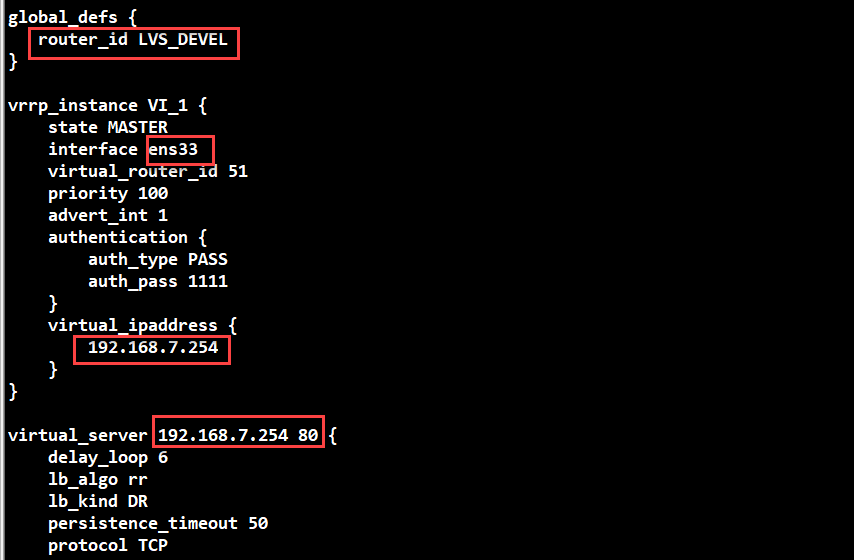
keepalive组件:

核心组件：(checkers vrrp stack ipvs wrapperwatch dog)、IO复用器、内存管理、配置文件分析器



**node1和node2作为调度器，调度器负责调动web1 web2 web3**

**keepalived可以实现调度器高可用**



**vrrp\_instance VI\_1 {**

**state MASTER //表示主**

**interface ens33**

**virtual\_router\_id 51**

**priority 100 //优先级为100**

**advert\_int 1**

**authentication {**

**auth\_type PASS**

**auth\_pass 1111**

**}**

**virtual\_ipaddress {**

**192.168.7.254 //虚拟IP地址**

**}**

**}**

**virtual\_server 192.168.7.254 80 {**

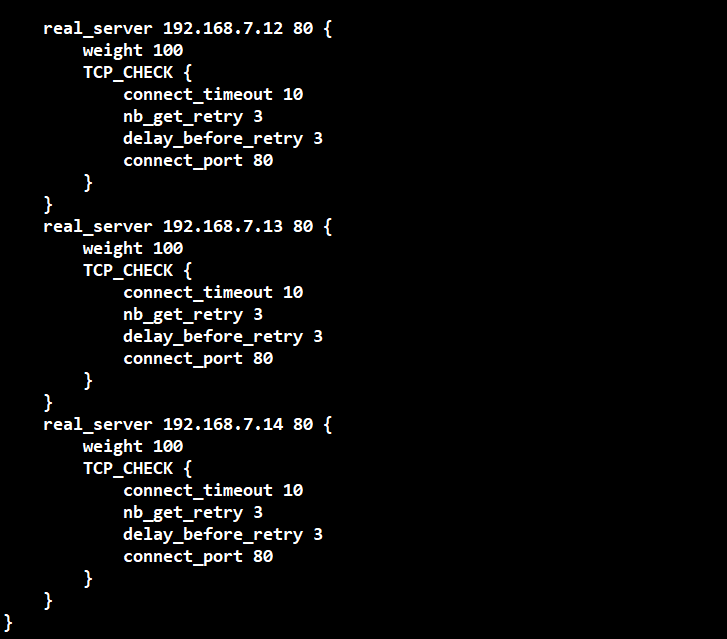
**delay\_loop 6**

**lb\_algo rr //算法是轮训**

**lb\_kind DR //lvs采用直接路由模式**

**persistence\_timeout 50**

**protocol TCP**

****





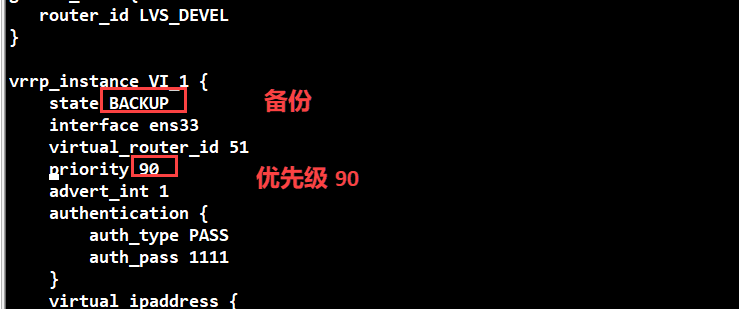
**vrrp\_instance VI\_1 {**

**state BACKUP //表示备份**

**interface ens33**

**virtual\_router\_id 51**

**priority 90 //优先级是90,一定要低于MASTER**

 **advert\_int 1**

**authentication {**

**auth\_type PASS**

**auth\_pass 1111**

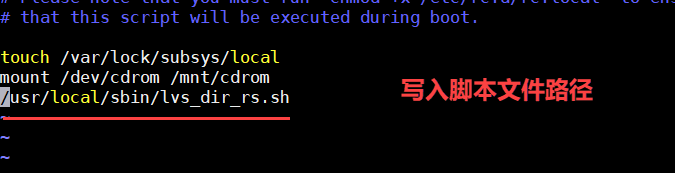
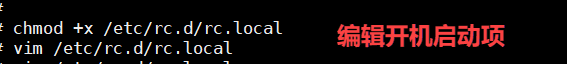
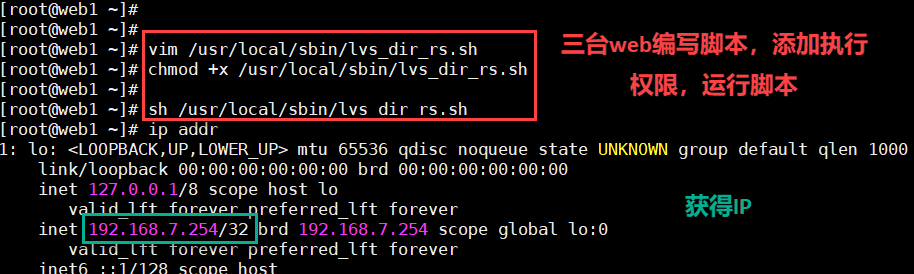
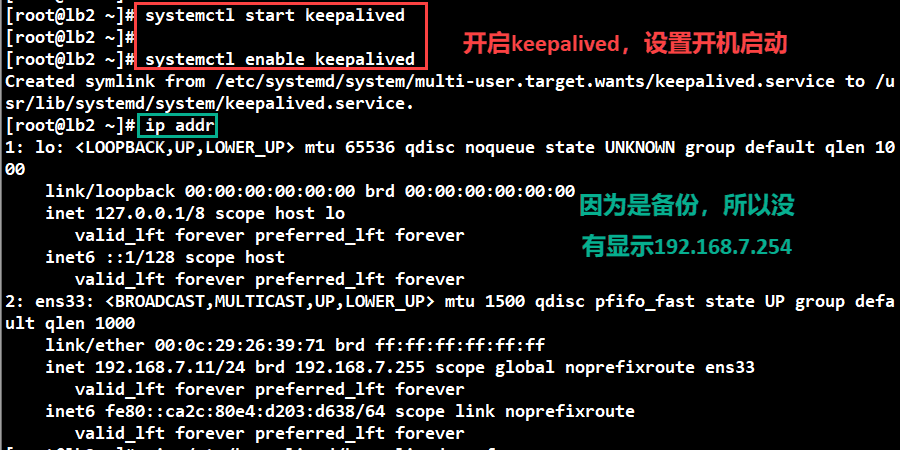
**}**

**virtual\_ipaddress {**

**192.168.7.254**

**}**

**}**



**web1 web2 web3**

**#!/bin/bash**

**vip=192.168.7.254**

**ifconfig lo:0 $vip broadcast $vip netmask 255.255.255.255 up**

**route add -host $vip lo:0**

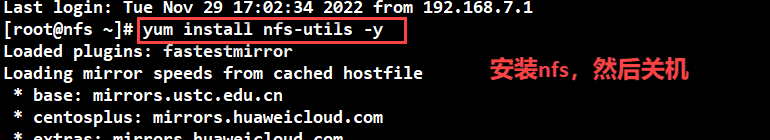
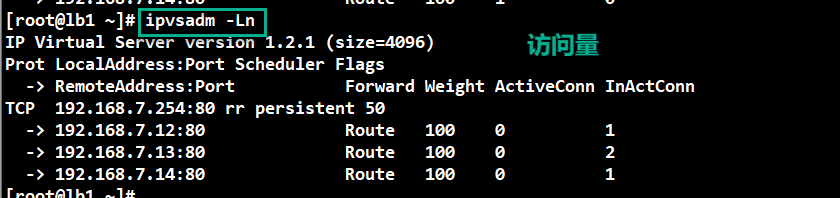
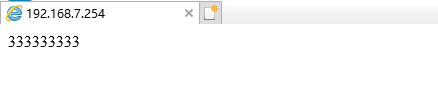
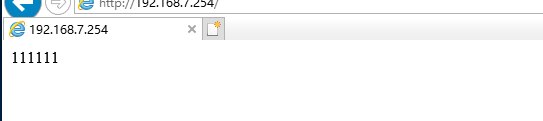
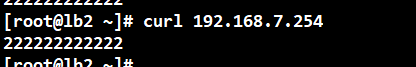
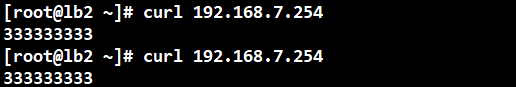
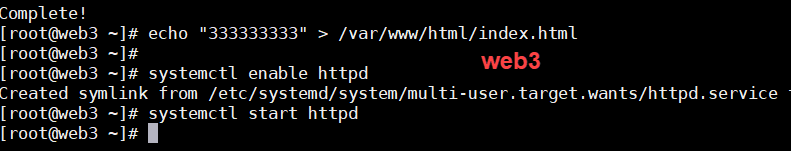
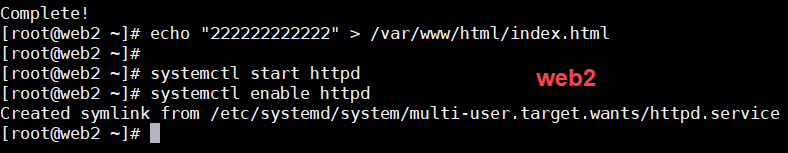
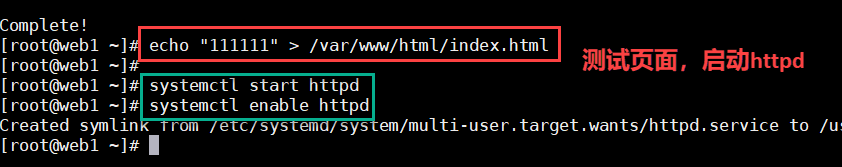
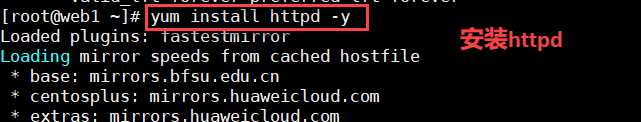
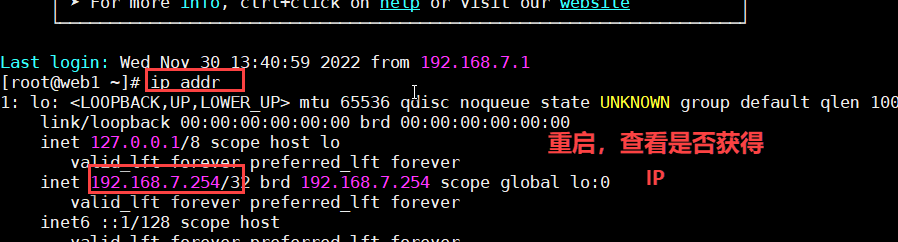
**echo "1" > /proc/sys/net/ipv4/conf/lo/arp\_ignore**

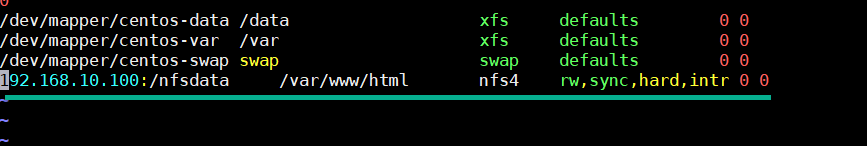
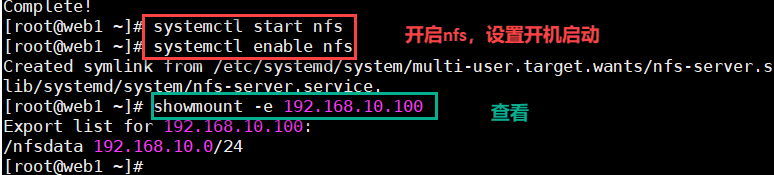
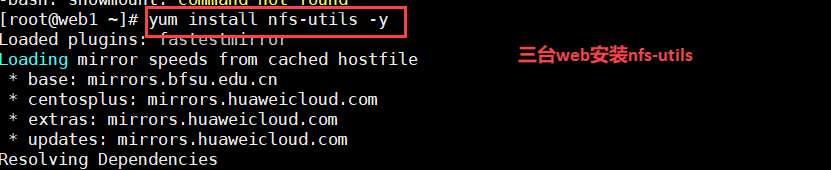
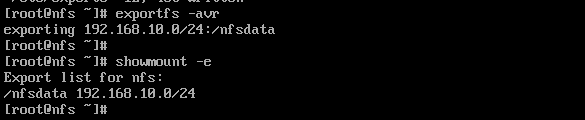
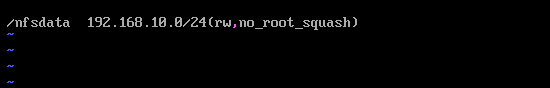
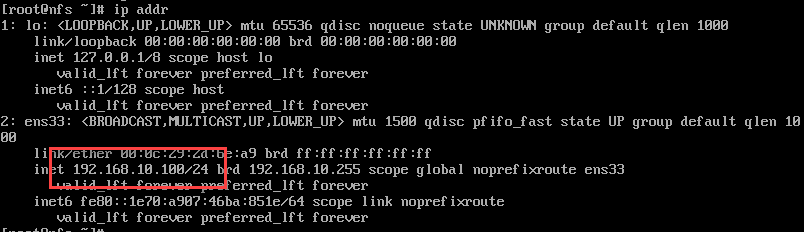
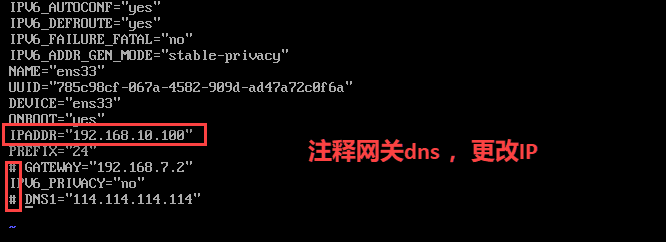
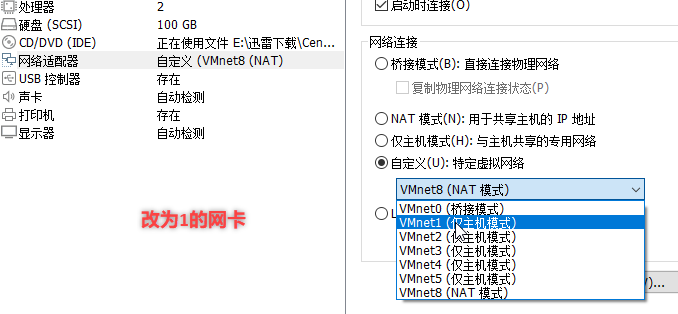
**echo "2" > /proc/sys/net/ipv4/conf/lo/arp\_announce**

**echo "1" > /proc/sys/net/ipv4/conf/all/arp\_ignore**

**echo "2" > /proc/sys/net/ipv4/conf/all/arp\_announce**

**sysctl -p &>/dev/null**





**/etc/fstab挂载NFS说明**

**intr：允许NFS中断文件操作和向调用它的程序返回值，默认不允许文件操作被中断**

**hard or soft -- 指定当程序无法通过NFS连接使用服务器输出的文件系统上的文件时，是否需要等待服务器重新在线。如果指定hard，用户无法不能终止程序等待NFS通讯恢复，除非同时指定了intr选项。如果指定了soft，用户可以设置附加选项timeo=<value>，<value>指定了错误报告之前的秒数。**

**intr -- 当服务器宕机或者无法到达时，允许中断NFS请求**

