第二阶段-----王凯

NETWORK(网络架构)

day01: 计算机网络,网络通信参考模型,数据链路层解析

day02: VLAN技术及应用,TRUNK,网络层解析

day03: OSPF,传输层,ACl,NAt

day04: STP,VRRP

day05: 企业级网络架构

SHELL(脚本)

day01: shell概述,shell变量,数值运算

day02: if判断, ||&&,for&whell循环

day03: case语句,函数及中断控制,字符串处理

day04: 正则表达式,sed基础

day05: sed高级应用,awk基础

day06: awk高级应用

第二阶段-----丁明一

OPERATION(运维)

day01: nginx基础,nginx虚拟主机,https加密网站

day02: 部署LNMP,nginx高级技术

day03: nginx调度,nginx优化

day04: Session与Cookie,部署memcached,Session共享

day05: Tomcat基础,Toncat应用,Varnish缓存服务器

day06: Git版本控制基础,进阶,RPM打包

day07: VPN隧道服务器,NTP服务器搭建

CLUSTER(集群)

day01: 集群及LVS简介,LVS-NAT集群,LVS-DR集群

day02: keepalived热备,keepalived+LVS、HAProxy

day03: Ceph概述,部署Ceph集群,Ceph块存储

day04: Ceph块存储应用,Ceph分布式文件系统,对象存储

PROJECT(项目搭建)

day01: 服务器硬件介绍,部署LNMP动态网站

day02: 网站架构演变,LNP+Mariadb数据库分离,Web集群

day03: Keepalived高可用,部署Ceph分布式存储

day04: 部署Git版本控制系统,优化Web服务器

网络架构

计算机网络的功能

数据通信、资源共享、增加数据可靠性、提高系统处理能力

计算机网络的发展

60年代:分组交换 提高数据传输效率,减少服务器的负担.

70-80年代:TCP/IP 统一网络环境.

90年后:Web技术

标准：一致同意的规则

ISO（国际标准化组织）在网络通信中创建了OSI（开放系统互联）模型。

ANSI（美国国家标准化局）

ITU-T（国际电信联盟-电信标准部）

IEEE（电气和电子工程师学会）

按照网络规模和使用范围分类为

WAN：广域网 LAN:局域网

网络设备

交换机、路由器

网络设备生产厂商：cisco思科，华为。

网络拓扑结构

1. 点对点

两台设备之间有一条单独的连接，通常用于广域网连接

1. 星型拓扑

1 ) 优点：易于实现、易于网络扩展、易于故障排查

2 ) 缺点：中心节点压力大、组网成本较高

3、网型拓扑结构

1. 各个节点至少与其他两个节点相连
2. 可靠性高、组网成本高

OSI

1. 国际标准化组织（ISO）

开放系统互连参考模型OSI

OSI是一个开放式体系结构，它规定将网络分为七层

2、协议分层

为了降低网络设计的复杂性，将协议进行了分层设计

应用层：网络服务与最终用户的一个接口

表示层：数据的表现形式，如加密、压缩。

会话层：建立、管理、中止会话，例如断点续传。

传输层：定义传输数据的协议端口号，以及流控和差错校验。

网络层：进行逻辑地址寻址，实现不同网络之间的通信。

数据链路层：建立逻辑连接、进行硬件地址寻址、差错校验等功能。

物理层：建立、维护、断开物理连接。

TCP/IP 常见协议

应用层：HTTP80、https443、FTP21、TFTP69、SMTP25 、SNMP161、DNS53

传输层：TCP、UDP

网络层：ICMP、IGMP、IP、ARP

VLAN TRUNK MSTP

PDU（协议数据单元）

传输层 段 segment

网络层 包 packet

数据链路层 帧 frame

物理层 比特 bit = 8 byte 字节

TCP/IP相应层次的设备

应用层 计算机

传输层 防火墙

网络层 路由器

数据链路层 交换机

物理层 网卡

交换机的视图操作：

<huawei>用户视图

<huawei>system-view 进入系统视图

<huawei>display version 显示版本

<huawei>display current-configuration 查看当前配置 #空格翻页

<huawei>undo terminal monitor 关闭信息 terminal monitor 打开信息

[huawei]系统视图

[huawei]interface Ethernet 0/0/1 百兆接口 interface GigabitEthernet 0/0/1 百兆接口

[huawei]sysname tedu01 命名

Ctrl+Z 返回用户视图

quit退出

[huawei]aaa 三a模式 : 对账户,认真,统计的管理

[huawei]local-user admin01 password cipher admin 设置账户名和密码(暗文)

quit

user-interface console 0 进入0号控制台 idle-timeout 10 设置控制台会话时间10分钟

authentication-mode aaa 启用三a模式登录 退出账户后生效

Ctrl+] 退出账户

<huawei>save 保存配置

<huawei>reboot 重启

<huawei>reset saved-configuration 恢复出厂设置(删除配置文件)

==============================================================================

接口

以太网接口：

RJ-45水晶头

光纤接口：

FC 、ST、SC

LC 窄体方形光纤接头（目前主流）

MT-RJ

双绞线

1. 双绞线分类：

屏蔽双绞线 （STP）

线外包裹一层金属网膜，用于电磁环境非常复杂的工业环境中

非屏蔽双绞线 （UTP）

1. 双绞线标准与分类 ：

Cat5五类双绞线，适用于100Mbps的网络

Cat 5e衰减更小，适用于100Mbps的网络，串扰更少，性能优于5类线 （超五类）

Cat 6适用于1000Mbps的网络

Cat 7适用于10000Mbps的网络

4、双绞线的连接规范

1）线序

T568A：

白绿、绿、白橙、蓝、白蓝、橙、白棕、棕

T568B：

白橙、橙、白绿、蓝、白蓝、绿、白棕、棕

2）线缆的连接：

标准网线（直连线或直通线）：用于连接不同设备（A-A，B-B）

交叉网线：用于连接相同设备 （A-B）

全反线 ：不用于以太网的连接，主要用于计算机的串口和路由器或交换机的console（控制口）相连

例外情况：版本较新设备可以随意使用标准与交叉网线而不受限制，设备本身具备自动识别功能。

5、物理层设备

网卡、中继器

======================

day02数据链路层

MAC地址

用来识别一个以太网上的某个单独的设备或一组设备

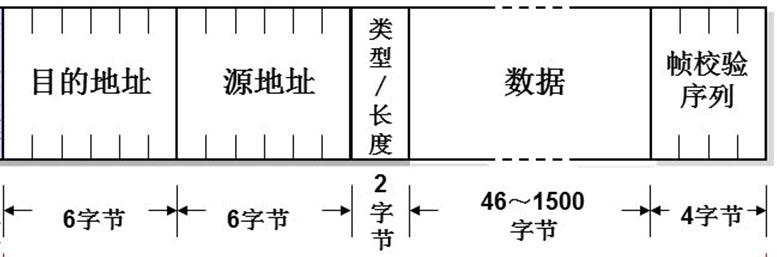
MAC地址长度48位bit(6个字节)，前24位代表厂商，后24位代表网卡编号，

MAC 地址的第8位为0时表示该MAC地址为单播地址，为1时表示组播地址。48位都为1表示广播地址。

windows系统中用Ipconfig /all 查看MAC地址

注：一块物理网卡的地址一定是一个单播地址，也就是第8位一定为0

数据链路层的帧格式 mtu: 最大传输单元1500字节



交换机

1、交换机是用来连接局域网的主要设备，可以实现数据帧的转发 , 交换机在数据链路层工作.

2、交换机的工作原理

学习-->广播-->转发-->更新

3、广播域

广播域指接收同样广播消息的节点的集合，交换机的所有端口默认属于同一个广播域

交换机的基本配置

1、常用命令

1）查看MAC地址表

特权：display mac-address

**MAC地址**

用来识别一个以太网上的某个单独的设备或一组设备

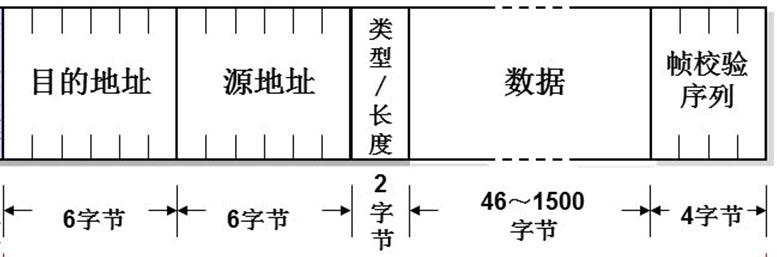
MAC地址长度48位bit(6个字节)，前24位代表厂商，后24位代表网卡编号，

MAC 地址的第8位为0时表示该MAC地址为单播地址，为1时表示组播地址。48位都为1表示广播地址。

windows系统中用Ipconfig /all 查看MAC地址

注：一块物理网卡的地址一定是一个单播地址，也就是第8位一定为0

数据链路层的帧格式 mtu: 最大传输单元1500字节



**交换机**

1、交换机是用来连接局域网的主要设备，可以实现数据帧的转发 , 交换机在数据链路层工作.

2、交换机的工作原理

学习-->广播-->转发-->更新

3、广播域

广播域指接收同样广播消息的节点的集合，交换机的所有端口默认属于同一个广播域

**交换机的基本配置**

1、常用命令

1）查看MAC地址表

特权：display mac-address

1. **什么是VLAN**

虚拟局域网

1. VLAN的优势

-广播控制

-增加安全性

-提高带宽利用

-降低延迟

**创建VLAN的方法**

在全局配置模式中：vlan 2（创建vlan2）

接口加入vlan

interface Ethernet 0/0/3 #进入三口视图

1）进入将要加入vlan的接口然后输入

port link-type access #将三口设置为access

port default vlan 2 #将三口加入vlan 2

1. 同时将多个接口加入vlan 3
2. 全局： port-group 1 #创建1号接口组

group-member ethernet 0/0/5 0/0/6 #将5口6口加入一号组 (前面加undo可删除指定接口)

port link-type access #将1号组里所有接口设置为access.表示将此接口定义为即将为某一个vlan服务

port default vlan 3 #定义此接口为vlan 3

1. 同时创建多个vlan

vlan batch 2 3 #创建vlan 2 vlan 3

**trunk中继链接**

1、作用：实现交换机之间的单一链路传递多个vlan的信息

2、链路类型：

接入链路(access): 可以承载1个 vlan

中继链路(trunk)：可以承载多个 vlan

0/0/8]port link-type trunk #修改链路类型为trunk(交换机设置)

0/0/8]port trunk allow-pass vlan all #允许所有vlan通行

clear configuration interface e0/0/8 #恢复接口默认配置

1. 链路聚合

.eth-trunk

-多条线路负载均衡,带宽提高

-容错,当一条线路失效时,不会造成网络失效

interface eth-trunk1 #开启链路聚合接口1

quit

in e0/0/1

0/0/1]eth-trunk1

0/0/1]in e0/0/2

0/0/2]eth-trunk1

quit

interface eth-trunk1

port link-type trunk

port trunk allow-pass vlan all

方法2

interface eth-trunk1 #开启链路聚合接口1

trunkport ethernet 0/0/1 0/0/2 #加入1号2号口

port link-type trunk

port trunk allow-pass vlan all

一个接口在配置链路聚合时必须是默认状态,如果有设置,需清空设置后再配置

day03

路由器原理及静态路由

1、路由

跨越从源主机到目标主机的一个互联网络来转发数据包的过程

能够连接不同广播域的设备,并且隔离所有广播

2、路由表

路由器根据路由表做路径选择

display ip routing-table

3、路由表的获得

1）、直连路由：配置IP地址，端口UP状态，形成直连路由。

2）、非直连网段：需要静态路由或动态路由，将网段添加到路由表中。

4、静态路由

1）、特点：

由管理员手工配置的，是单向的，因此需要在两个网络之间的边缘路由器上需要双方对指，否则就会造成流量有去无回，缺乏灵活性，适用于小型网络。

2)、配置

全局模式：

ip route-static 目标网络ID 子网掩码 下一跳IP

IP route-static 192.168.3.0 24 192.168.2.2

缺省路由（默认路由）

缺省路由是一种特殊的静态路由

简单地说,缺省路由就是在没有找到任何匹配的具体路由条目的情况下才使用的路由，适用于只有一个出口的末节网络（比如企业的网关出口）

全局:ip route 0.0.0.0 0.0.0.0 下一跳

查看路由表

特权：show ip route

C直连路由

S静态路由

动态路由

S\*默认路由 是特殊的动态路由.可以匹配任意网段,当路由表中找不到明确路由时使用,一般用于企业出口网关

============================================

# 三层交换技术

1、作用

使用三层交换技术实现VLAN间通信

三层交换=二层交换+三层转发

2、虚拟接口（SVI）

三层交换机上配置的VLAN接口为虚拟接口

3、三层交换机的配置

1）、在三层交换机启用路由功能

全局：ip routing

2）、配置虚拟接口的IP 地址

全局：

interface vlanif1 #

ip address  *192.168.1.254 255.255.255.0*

interface vlanif2

ip address  *192.168.2254 255.255.255.0*

interface vlanif3

ip address  *192.168.3.254 255.255.255.0*

ICMP Internet 控制报文协议,检测和犯规网络状态信息

ping -t #持续检测网络

ping 192.168.2.1 -l 100 #改变ping时发送的文件大小为100byte(默认为32byte)

三层交换机配置:

创建vlan g0/0/8配置trunk 配置SVI的ip地址

二层交换机配置:

创建vlan 2 3 e0/0/2加入vlan2 e0/0/3加入vlan3

e0/0/8配置trunk 配置SVI的ip地址

======================================================

动态路由

1、动态路由特点

根据网络拓扑或流量变化，由路由器通过路由协议自动设置, 减少了管理任务，但占用了网络带宽

适合ISP服务商、广域网、园区网等大型网络

OSPF协议

* Open Shortest Path First ( 开放式最短路径优先）
* OSPF区域
  + 为了适应大型的网络，OSPF在AS内划分多个区域
  + 每个OSPF路由器只维护所在区域的完整链路状态信息
* 区域ID
  + 区域ID可以表示成一个十进制的数字
  + 也可以表示成一个IP
* 骨干区域Area 0
  + 负责区域间路由信息传播

**三层交换机:**

启动OSPF路由进程

ospf 1 #开启进程ospf 1

指定OSPF协议运行的接口和所在的区域(宣告)

*area 0 #进入骨干区域area 0*

*network 192.168.1.0 0.0.0.255 #使用反掩码添加接口*

*network 192.168.2.0 0.0.0.255*

*network 192.168.3.0 0.0.0.255*

*network 192.168.4.0 0.0.0.255*

*ip route-static 0.0.0.0 0.0.0.0 192.168.4.2 #设置默认路由*

***路由器:***

ospf 1 #开启进程ospf 1

*area 0 #进入骨干区域area 0*

*network 192.168.4.0 0.0.0.255 #使用反掩码添加接口*

排错，分别检查三个方面：

vlan

ip

路由

# 传输层

. 网络层提供点到点的连接

. 传输层提供端与端的连接

传输层实现了端到端的数据传递

0～65535 端口总数

0保留 3306是数据库端口号

1～1023 常用端口

使用1024之后的端口号

**传输层协议**

TCP传输控制协议: 可靠 ，面向连接，效率低 (上网)

UDP用户数据报协议: 不可靠，无连接，效率高 (游戏)

**TCP的连接方式**



SYN 想与对方主机建立连接

ACK 确认对方信息

FIN 想与对方主机断开连接

三次握手

syn----->ack ,syn------>ack

四次断开

fin------>ack----->fin----->ack

**UDP的应用:**

69 TFTP 简单文件传输协议

53 DNS 域名服务

123 NTP 时间同步

**UDP的流控和差错控制**

.UDP缺乏可靠机制

.UDP只有校验和来提供差错控制

-需要上层协议来提供差错控制:例如TFTP协议

**访问控制列表（ACL）**

读取第三层、第四层 头部信息

根据预先定义好的规则对数据进行过滤

对已经建设好的网络，可以通过ACL管理，使网络环境更加健康

**标准ACL**

基于源IP地址过滤数据包

标准访问控制列表的访问控制列表号是2000-2999

**扩展ACL**

基于源IP地址、目的IP地址、指定协议、端口来过滤数据包

扩展访问控制列表的访问控制列表号是3000-3999

0 严格匹配

1 不做匹配

192.168.2.10

192.168.2.0 0.0.0.255 #只看前三位是否匹配

192.168.3.10

192.168.2.0 0.0.255.255 #只看前两位是否匹配

**要求:只拒绝192.168.2.1访问192.168.1.1,允许其他人访问**

acl 2000 #使用基本acl,号码是2000

rule deny/permit(拒绝或放行) source 192.168.2.1 0.0.0.0 #匹配4位 #拒绝源地址是192.168.2.1的主机

in g0/0/1

traffic-filter inbound acl 2000 #使用刚刚创建的acl规则过滤进入这个接口的数据

**要求:只让192.168.2.1通行,拒绝其他所有人通过**

acl 2001

rule permit source 192.168.2.1 0 #允许192.168.2.1通行

in g0/0/1

undo traffic-filter inbound #删除所有已设置的acl

traffic-filter inbound acl 2001 #使用acl2001过滤进入这个接口的数据

**要求:通过高级配置禁止192.168.2.1访问192.168.1.1的ftp服务,**

**禁止192.168.2.2访问192.168.1.1的http服务,所有其他主机不受限制**

acl 3000 #使用高级acl3000

rule deny tcp source 192.168.2.1 0 destination 192.168.1.1 0 destination-port eq 21

#禁止192.168.2.1访问192.168.1.1的ftp服务

rule deny tcp source 192.168.2.2 0 destination 192.168.1.1 0 destination-port eq 21

#禁止192.168.2.2访问192.168.1.1的http服务

in g0/0/1 #进入一号口

traffic-filter inbound acl 3000 #使用acl3000过滤进入这个接口的数据

day04

**NAT 网络地址转换:**

可以将内部私有ip地址转换为公网ip地址,从而实现全球互联

192.168.0.1 100.0.0.1 <-------------------> 200.0.0.1 192.168.0.1

内网ip 运营商转换的公网ip 运营商转换的公网ip 内网ip

NAT优点: NAT缺点:

-节省公有合法ip地址 -延迟增大

-处理地址重叠 -配置和维护的复杂性

-安全性

内部私有地址范围

A 10.0.0.0～10.255.255.255 /8

B 172.16.0.0～172.31.255.255 /16

C 192.168.0.0～192.168.255.255 /24

特殊地址:127.0.0.1(本机回环)

:169.254.\*.\* (代表本机暂时无ip可用,不能与其他ip通信) #内部保留地址

100M 500～800元 动态公网ip 故障率

50M 40000元 固定公网ip(公司用)

NAT实现方式

-静态转换 一般用于服务器发布服务到公网，一台服务器使用一个公网ip

-Easy ip 只发布某个服务到公网时使用(员工上网时使用)

**配置静态nat转换,一对一转换,通常配置在服务器(优先匹配)**

#interface g0/0/1

#nat static global 100.0.0.2 inside 192.168.2.1 #设置192.168.2.1的静态nat转换为100.0.0.2

#nat static global 100.0.0.3 inside 192.168.2.2 #设置192.168.2.2的静态nat转换为100.0.0.3

**配置easy ip ,多对一转换,通常在仅需访问公网的环境使用**

#interface g0/0/1

#undo nat static global 100.0.0.2 inside 192.168.2.1 #取消nat静态转换设置

#undo nat static global 100.0.0.3 inside 192.168.2.2 #取消nat静态转换设置

#acl 2000 #创建acl 2000

# rule permit source any #设置acl允许所有内部IP地址通过

#in g0/0/1 #进入路由外网接口0/0/1

#nat outbound(外出绑定) 2000 #设置easy ip 按照acl2000执行

**STP 生成树协议------------------->**智能交换机中默认为开启状态

可以在环形网络中临时断开一个接口,防止广播风暴产生，在网络不通时,又可以打开这个接口,

使阻塞接口起到备份链路的作用(广播风暴会让网络崩溃)

display stp brief #查看线路连接情况

forwarding 转发状态

discarding 阻塞状态(临时断开的接口) #在物理线断开的情况下,STP会自动打开临时关闭的接口

undo stp enble #关闭stp 1.1ping1.2 #产生广播风暴

1.1ping1.3 #风暴影响,无法ping通

**VRRP协议-------->高可用协议之一**

VRRP能够在不改变组网的情况下,将多台路由器虚拟成一个虚拟路由器,

通过配置虚拟路由器的ip地址为默认网关,实现网关的备份

进入三层交换机1

#interface vlanif1

#ip address 192.168.1.252 24

#vlan 2

#interface vlanif2

#ip address 192.168.2.2 24

#interface g0/0/2

#port link-type accress

#port default vlan 2

#ospf 1

#area 0

#network 192.168.1.0 0.0.0.255

#network 192.168.2.0 0.0.0.255

#interface vlanif1

#vrrp vrid 1 virtual-ip 192.168.1.254 #设置vrrp,创建vrid1 ip为192.168.1.254

进入三层交换机2

#interface vlanif1

#ip address 192.168.1.25324

#vlan 3

#interface vlanif 2

#ip address 192.168.3.2 24

#interface g0/0/2

#port link-type accress

#port default vlan 2

#ospf 1

#area 0

#network 192.168.1.0 0.0.0.255

#network 192.168.3.0 0.0.0.255

#interface vlanif1

#vrrp vrid 1 virtual-ip 192.168.1.254

进入路由器

#interface g0/0/0

#ip address 192.168.2.1 24

#interface g0/0/1

#ip address 192.168.3.1 24

#interface g0/0/2

#ip address 192.168.4.254 24

#ospf 1

#area 0

#network 192.168.2.0 0.0.0.255

#network 192.168.3.0 0.0.0.255

#network 192.168.4.0 0.0.0.255

**完善网络的负载均衡**

**通过之前配置的STP加上vrrp完善网络的负载均衡功能。**

**配置三层交换机1为vlan1的主路由器、vlan2的备份路由器，三层交换机2为vlan1的备份路由器、vlan2的主路由器，实现负载均衡的效果**

进入交换机1:

#vlan 2

#in e0/0/3

#port link-type access

#port default vlan 2

#port-group 1

#group-member e0/0/1 to e0/0/2

#port link-type trunk

#port trunk allow-pass vlan all

进入交换机2:

#port-group 1

#group-member e0/0/1 to e0/0/2

#port link-type trunk

#port trunk allow-pass vlan all

进入三层交换机1:

#vlan 2

#in vlanif1

#IP address 192.168.1.252 24

#in vlanif2

#ip address 192.168.2.252 24

#port-group 1

#group-member g0/0/1 to g0/0/3

#port link-type trunk

#port trunk allow-pass vlan all

#in vlanif1

#vrrp vrid 1 virtual-ip 192.168.1.254

#in vlanif2

#vrrp vrid 2 virtual-ip 192.168.2.254

进入三层交换机2:

#vlan 2

#in vlanif1

#IP address 192.168.1.253 24

#in vlanif2

#ip address 192.168.2.253 24

#port-group 1

#group-member g0/0/1 to g0/0/3

#port link-type trunk

#port trunk allow-pass vlan all

#in vlanif1

#vrrp vrid 1 virtual-ip 192.168.1.254

#in vlanif2

#vrrp vrid 2 virtual-ip 192.168.2.254

#vrrp vrid 2 pricrity 110

#dis vrrp brief //查看

day05

**综合项目拓扑:搭建企业网络**

**1、规划ip,vlan**

**2、链接终端设备线缆到接入层交换机(s3700)**

**3、在所有接入层交换机创建vlan10,vlan20,vlan30,vlan40**

#vlan batch 10 20 30 40

1. **将接入终端设备所在的接口加入相应的vlan**

#in e0/0/5

#port link-type access

#port default vlan 10

第二台加入vlan20,以此类推,最后使用display vlan检查

1. **在vlan40的接入层交换机配置链路聚合 (查询结果dis current-configuration)**

#interface eth-trunk 1 (2) #创建eth-trunk 1 和 2

#trunkport ethernet 0/0/1 0/0/2 (0/0/3 0/0/4) #将1 2 接口加入eth-trunk 1

1. **添加s5700两台,作为网关设备,并将链接vlan40的设备接口进行链路聚合**

第一台s5700

#interface Eth-trunk 1

#trunkport GigabitEthernet 0/0/4 0/0/5

第二台s5700

#interface Eth-trunk 2

#trunkport GigabitEthernet 0/0/4 0/0/5

**7、在两台s5700上分别创建vlan 10 20 30 40**

1. **将两台s5700用线缆链接**
2. **将所有交换机之间的链路都配置为trunk**

注意:链路聚合的接口要进入聚合口中配置

#interface eth-trunk 1

#port link-type trunk

#port trunk allow-pass vlan all

普通接口配置:

#interface e0/0/1 //注意区分千兆口和百兆口

#port link-type trunk

#port trunk allow-pass vlan all

1. **给第一台s5700配置ip**

vlan 10 192.168.10.252

vlan 20 192.168.20.252

vlan 30 192.168.30.252

vlan 40 192.168.40.252

给第二台s5700配置ip

vlan 10 192.168.10.253

vlan 20 192.168.20.253

vlan 30 192.168.30.253

vlan 40 192.168.40.253

然后使用每个vlan的pc主机测试与本vlan网关的链接状况

1. **配置网关设备开启vrrp,实现虚拟网关功能,并达到负载均衡**

第一台s5700负责担任vlan10,vlan20的主

负责担任vlan30,vlan40的备份

#interface vlanif 10

#vrrp vrid 10 virtual-ip 192.168.10.254

#vrrp vrid 10 priority 110 //设置vlan10的主身份

#interface vlanif 20

#vrrp vrid 20 virtual-ip 192.168.20.254

#vrrp vrid 20 priority 110 //设置vlan20的主身份

#interface vlanif 30

#vrrp vrid 30 virtual-ip 192.168.30.254

#interface vlanif 40

#vrrp vrid 40 virtual-ip 192.168.40.254

#dis vrrp brief //查看

第二台s5700负责担任vlan10,vlan20的备份

负责担任vlan30,vlan40的主

#interface vlanif 10

#vrrp vrid 10 virtual-ip 192.168.10.254

#interface vlanif 20

#vrrp vrid 20 virtual-ip 192.168.20.254

#interface vlanif 30

#vrrp vrid 30 virtual-ip 192.168.30.254

#vrrp vrid 30 priority 110 //设置vlan30的主身份

#interface vlanif 40

#vrrp vrid 40 virtual-ip 192.168.40.254

#vrrp vrid 40 priority 110 //设置vlan40的主身份

#dis vrrp brief //查看

1. **添加两个AR2220路由器,并与三层交换机链接,链接接口分别设置ip地址**

第一台s5700 第二台s5700

# vlan batch 50 60 # vlan batch 70 80

#in vlanif 50 #in vlanif 70

#ip add 192.168.50.2 24 #ip add 192.168.70.2 24

#in vlanif 60 #in vlanif 80

#ip add 192.168.60.2 24 #ip add 192.168.80.2 24

#in g0/0/23 #in g0/0/23

#port link-type access #port link-type access

#port default vlan 50 #port default vlan 70

#in g0/0/24 #in g0/0/24

#port link-type access #port link-type access

#port default vlan 60 #port default vlan 80

第一台路由器 第二台路由器

#in g0/0/0 #in g0/0/0

#ip add 192.168.50.1 24 #ip add 192.168.60.1 24

#in g0/0/1 #in g0/0/1

#ip add 192.168.70.1 24 #ip add 192.168.80.1 24

1. **设置ospf自动路由**

第一台路由器

#ospf 1

#area 0

#network 192.168.50.0 0.0.0.255

#network 192.168.70.0 0.0.0.255

第二台路由器

#ospf 1

#area 0

#network 192.168.60.0 0.0.0.255

#network 192.168.80.0 0.0.0.255

第一台s5700

#ospf 1

#area 0

#network 192.168.10.0 0.0.0.255

#network 192.168.20.0 0.0.0.255

#network 192.168.30.0 0.0.0.255

#network 192.168.40.0 0.0.0.255

#network 192.168.50.0 0.0.0.255

#network 192.168.60.0 0.0.0.255

第二台s5700

#ospf 1

#area 0

#network 192.168.70.0 0.0.0.255

#network 192.168.80.0 0.0.0.255

#network 192.168.10.0 0.0.0.255

#network 192.168.20.0 0.0.0.255

#network 192.168.30.0 0.0.0.255

#network 192.168.40.0 0.0.0.255

1. **添加一个s3700,作为外网设备,并配置IP地址**

#in vlanif1

#ip add 100.0.0.10 8

第一个路由配置与外网链接的接口ip地址

#in g0/0/2

#ip add 100.0.0.1 8

第二个路由配置与外网链接的接口ip地址

#in g0/0/2

#ip add 100.0.0.2 8

1. **给路由器配置默认路由,在ospf中宣告默认路由**

第一台路由器

#ip route-static 0.0.0.0 0.0.0.0 100.0.0.10

#quit

#ospf 1

#default-route-advertise always

第二台路由器

#ip route-static 0.0.0.0 0.0.0.0 100.0.0.10

#quit

#ospf 1

#default-route-advertise always

1. **使用acl定义范围,再配置easy ip连接外网**

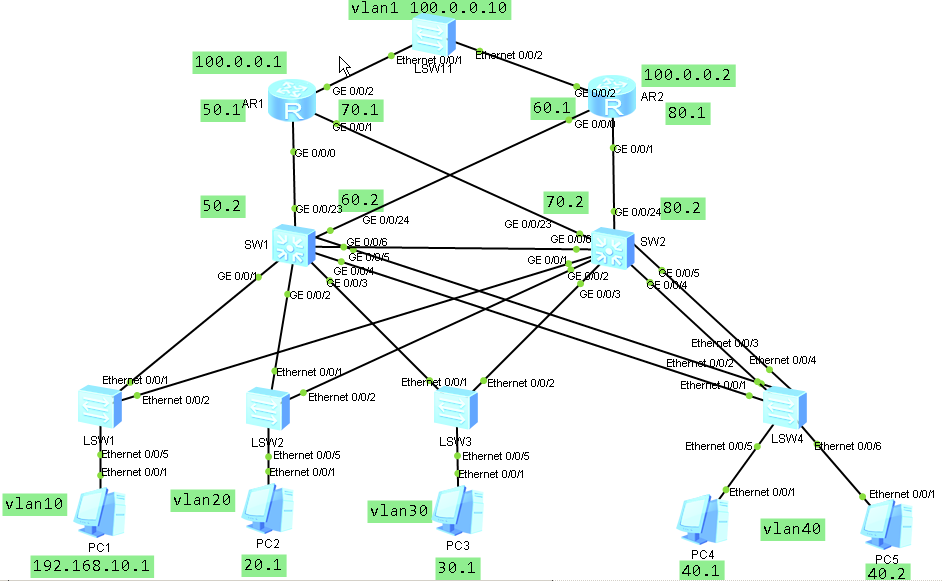
#acl 2000

#rule permit source any

#in g0/0/2

#nat outbound 2000

注:两台路由相同操作



SHELL

day01

**Shell是在Linux内核与用户之间的解释器程序，通常指的是bash，负责向内核翻译及传达用户/程序指令**

**Shell的使用方式：**

1.交互执行指令：人工干预，执行效率底。

2.非交互执行指令：安静地在后台执行，执行效率高，方便写脚本。

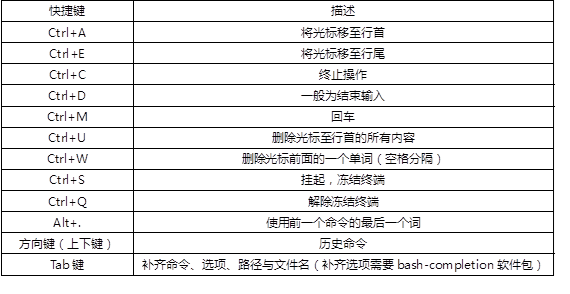
若需要临时使用另一种Shell环境，可以直接执行对应的Shell解释器程序，比如只要执行ksh可以切换到ksh命令行环境。

#yum -y install ksh //下载ksh解释器

#cat /etc/shells

#ksh //进入ksh环境

#exit //返回到切换前的bash环境



# shell脚本的三个步骤:

1. 声明所用解释器
2. 写脚本功能描述,变量含义,使用位置,作者等信息
3. 编写执行指令

# 脚本执行方法:

1. 添加x权限
2. 使用解释器直接执行脚本(新开启解释器)
3. 使用source(可以用.代表) 执行脚本(使用当前解释器,脚本中有exit的话,执行后会退出当前用户)

# 系统变量包括: set/env可查看全部变量

环境变量: UID HOME USER PWD SHELL HOSTNAME PS1 PS2

预定义变量:

$0 显示执行脚本的名称

$# 显示位置变量的个数

$\* 显示所有位置变量

$? 显示确认上一条命令执行结果(0为真,非0为非真)

$$ 显示当前脚本的进程号

位置变量:

$1 脚本名后面跟的第一个参数

$2 脚本名后面跟的第一个参数

依次类推

**bash -x 脚本名 //-x选项可以显示脚本中每条命令执行的细节，有助排错**

env : 查看所有环境变量

set : 查看所有变量

" " : 界定范围

' ' : 界定范围，屏蔽特殊的含义

` ` 或 $() : 可以把通过命令得到的结果当作参数使用

read -p : 产生交互 例:read -p " 请输入用户名" :变量名

stty -echo : 关闭回显

stty echo : 开启回显

export a=10 //定义全局变量，使子进程可以使用变量(新建bash环境也能使用)

export -n a //取消定义全局变量

# 整数运算工具:expr

方法一:expr命令

expr 值1 + 值2 //加法,值可以用常量,也可以用变量

expr 值1 - 值2 //减法

expr 值1 \\* 值2 //乘法,乘号需要用\转义

expr 值1 / 值2 //除法

expr 值1 % 值2 //求模,值1除以值2,然后取余数

方法二:$[]或$(())

echo $[值1+值2]

echo $[值1\*值2]

echo $[值1/值2]

echo $[值1%值2]

方法三:let命令 通常对变量进行计算,不显示结果

x=1234

#let y=x+22

#echo $y

显示为1256

#let a++ //a=a+1

#let a-- //a=a-1

#let a+=78 //a=a+78

#let a-=78 //a=a-78

#let a\*=78 //a=a\*78

#let a/=78 //a=a/78

#let a%=78 //a=a/78的余数

echo "scale=3;10/3" | bc //设置小数点后位数后进行运算

echo "1+1.1" | bc //bc可以运算非整数

命令的历史记录：

vim /etc/profile(环境变量全局文件) //修改历史记录

找到 HISTSIZE=1000 修改为合适的值

重新登陆帐号使用 history 检测修改结果

使用history -c 可以清空历史记录缓存

使用 > /root/.bash\_history 可以清空历史记录

# 编写搭建yum仓库脚本:

#!/bin/bash

echo "[development]

name=development

baseurl=ftp://172.25.254.250/centos-1804

enabled=1

gpgcheck=0" > /etc/yum.repos.d/dd.repo

退出后执行脚本

ping 172.25.254.250

yum clean all

yum repolist //检查结果

# 编写安装vsftpd的脚本

#!/bin/bash

yum -y install vsftpd &> /dev/null

systemctl start vsftpd

systemctl enable vsftpd

firefox ftp://172.25.0.11

**变量 使用固定的名称存储可能变化的量,增加脚本灵活性,可以在脚本中反复使用**

自定义变量,需要管理员定义变量的值,变量的名称使用大小写字母,不能以数字开头,不能使用特殊符号

定义变量 变量名称=变量的值

查看变量内容 echo $变量名称

取消变量 unset 变量名称

# 编写批量添加用户的脚本

#!/bin/bash

for i in {1..10}

do

useradd user$i 2>> /opt/e.txt //创建账户并收集错误信息

echo 123456 | passwd --stdin user$i > /dev/null //修改密码并不显示相关信息

done

chmod +x /opt/user //添加执行权限

/opt/user //执行脚本

id user1 //查看用户创建结果

cat /opt/e.txt //查看错误信息

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

day02

# 条件测试

方法 ： test 表达式 或者 [ 表达式 ]

字符串

例：[ $USER == root ] //判断当前用户是root

echo $? //查看结果，0是上一条指令判断成功，非0是失败

[ $USER ！= root ] //判断当前用户不是root

&& 前面命令成功后才执行后面的命令

|| 前面命令失败后才执行后面的命令

; 前面命令执行后再执行后面的命令,无逻辑关系

例:# [ abc != 123 ] || echo "ok" || echo "no"

[ abc != 123 ] && echo "ok" && echo "no"

[ -z ] //判断是空

[ ! -z ] //判断非空

**练习**

**创建添加用户的脚本，如果用户没有输入用户名，则退出脚本**

#!/bin/bash

read -p "请输入用户名:" u

[ -z $u ] && echo "您没有输入名字!" && exit

stty -echo

read -p "请输入密码:" p

stty echo

[ -z $p ] && echo "您没有输入密码!" && exit

useradd $u

echo $p | passwd --stdin $u

# 数字

-eq 等于（Equal）

-ne 不等于（Not Equal）

-ge 大于或等于（Greater or Equal）

-le 小于或等于（Lesser or Equal）

-gt 大于（Greater Than）

-lt 小于（Lesser Than）

# x=20

# [ $x -eq 20 ] && echo "相等" || echo "不相等"

相等

# [ $x -ne 30 ] && echo "不等于" || echo "等于"

不等于

# [ $x -gt 10 ] && echo "大于" || echo "否"

大于

# [ $x -ge 10 ] && echo "大于或等于" || echo "否"

大于或等于

# [ $x -lt 10 ] && echo "小于" || echo "否"

否

# [ $x -le 10 ] && echo "小于或等于" || echo "否"

否

# 每分钟检测系统中登陆账户的数量，如果大于3人登陆，发邮件给管理员报警

#!/bin/bash

s=`who | wc -l`

[ $s -gt 3 ] && echo "报警！超过3人登陆系统" | mail -s test root

脚本写完后赋于x权限，并放入计划任务中

chmod +x /opt/test2

crontab -e

\* \* \* \* \* /opt/test2

文件

[ -e abc ] //判断abc文件有没有，不考虑文件类型

echo $?

[ -d abc ] //判断abc文件有没有，且必须是个目录

echo $?

[ -f abc ] //判断abc文件有没有，且必须是个普通文件

echo $?

**检测有无$1目录，有则退出脚本，无则创建**

#!/bin/bash

[ -d $1 ] && exit

mkdir $1

测试 bash test3 abc

**if判断**

单分支

if 条件测试

then 命令序列

fi

if 条件测试;then

命令序列

fi

双分支

if 条件测试;then

命令序列

else

命令序列2

fi

**编写测试主机连接状况的脚本，成功显示"通了",失败显示"不通"**

#!/bin/bash

ping -c 3 -i 0.2 -W 1 172.25.0.10 &> /dev/null

if [ $? -eq 0 ];then

echo "通了"

else

echo "不通"

fi

**多分支**

if 条件测试;then

命令序列

elif 条件测试;then

命令序列2

else

命令序列3

fi

**使用if多分支编写随机数脚本，根据用户给的数字，分别显示“猜中了” “猜大了” “猜小了” 三种提示。**

#!/bin/bash

a=$[RANDOM%10]

read -p "请给一个数字(0-10)" n

[ -z $n ] && exit

if [ $a -eq $n ];then

echo "猜对了" && exit

elif [ $n -gt $a ];then

echo "猜大了"

else

echo "猜小了"

fi

**for循环**

for 变量名 in 变量值1 变量值2 。。。。

do

执行命令

done

for循环关心2个点:

1，循环次数

2，循环过程中使用的变量

for i in xx yy zz kk ee

do

echo "$i"

done

#!/bin/bash

for i in 1 2 3

do

echo "nb"

done

#!/bin/bash

a=3

for i in `seq $a`

do

echo "nb"

done

使用for循环检测某网段网络状况。

#!/bin/bash

a=0

b=0

for i in {1..15}

do

ping -c 3 -i 0.2 -W 1 172.25.0.$i &> /dev/null

if [ $? -eq 0 ];then

echo "172.25.0.$i通了"

let a++ //使变量a的值加1

else

echo "172.25.0.$i不通"

let b++

fi

done

echo "$a台通了，$b台不通"

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

day03

# case分支 ,作用与if类似，简化版本,但功能不如if

格式: case 变量 in

模式1)

命令序列1 ;;

模式2)

命令序列2 ;;

模式3)

命令序列3 ;;

\*)

命令序列4

esac

# 编写一键安装nginx的脚本

#!/bin/bash

yum -y install gcc openssl-devel pcre-devel

cd /opt

tar -xf /opt/nginx-1.12.2.tar.gz

cd nginx-1.12.2

./configure

make

make install

/usr/local/nginx/sbin/nginx (-s stop) 开启nginx/关闭nginx

netstat -ntulp | grep nginx 查看进程所连接的端口

# 使用case编写小工具

#!/bin/bash

case $1 in

-t|--T|tt)

touch $2 ;;

-v|V)

vim $2 ;;

-c|C|cc)

cat $2 ;;

\*)

echo "t|v|c"

esac

**用case编写nginx的快捷方式的脚本**

#! /bin/bash

case $1 in

start|sta)

/usr/local/nginx/sbin/nginx;;

stop|o)

/usr/local/nginx/sbin/nginx -s stop;;

restart|re)

/usr/local/nginx/sbin/nginx -s stop

/usr/local/nginx/sbin/nginx;;

status|u)

netstat -ntulp | grep -q nginx

[ $? -eq 0 ]&& echo "启动了" || echo "未启动";;

\*)

echo "start|stop|restart|status";;

esac

# 使用shell函数

将多组指令定义为一个函数名,在脚本中可以反复使用函数名,提高效率

格式1:

function 函数名 {

命令序列

.. ..

}

格式2:

函数名(){

命令序列

.. ..

}

**输出字体带颜色**

echo -e "\033[31mabcd\033[0m"

-e 为 echo的扩展 \033[是固定格式 31m是色号 0m是恢复默认

**中断和退出**

wait 等待所有的后台进程都结束才结束脚本

break 退出循环，继续执行循环之后的任务

continue 退出当前循环，继续下一次循环

exit 退出脚本

# 编写脚本，将用户给的数字相加，用户给0时输出之前所有数字之和。

#!/bin/bash

x=0

while :

do

read -p "请输入一个整数" n

[ $n -eq 0 ] && break //当输入的整数等于0就退出,不为0就继续循环

x=$[x+n]

done

echo $x

**编写脚本文件,找出1~20以内6的倍数，并输出她的平方值**

#!/bin/bash

for i in {1..20}

do

x=$[i%6] //定义变量x等于余数

[ $x -ne 0 ] && continue //条件测试如果不是0则继续当前循环，否则执行平方任务

echo $[i\*i]

done

# **子串截取的用法：**

* ${变量名:起始位置:长度}

**子串替换的两种用法：**

* 只替换第一个匹配结果：${变量名/old/new}
* 替换全部匹配结果：${变量名//old/new}

**字符串掐头去尾：**

* 从左向右，最短匹配删除：${变量名#\*关键词}
* 从左向右，最长匹配删除：${变量名##\*关键词}
* 从右向左，最短匹配删除：${变量名%关键词\*}
* 从右向左，最长匹配删除：${变量名%%关键词\*}

方法一，截取时从0开始计算字符位

a=abcdef //首先定义变量

echo ${a:2:2} //从第3位字符开始往后截取2位，结果是只截取"cd"

echo ${a::3} //从第一位字符开始往后截取3位，从第1位开始时可以省略"0"不写

方法二，截取时从1开始计算字符位

expr substr $a 2 2 //从第2位字符开始往后截取2位，截取"bc"

方法三，截取时从1开始计算字符位

echo $a | cut -b 2-4 //截取从第2位到第4位字符

echo $a | cut -b -5 //截取从第1位到第5位字符

echo $a | cut -b 4- //截取从第4位到最后的字符

echo $a | cut -b 6 //只截取第6位字符

echo $a | cut -b 1,3,5 //截取第1位，第3位，第5位字符

**编写随机获取字符的脚本**

#!/bin/bash

a=abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 //定义变量a，一共62位字符

x=$[RANDOM%62] //创建随机数，范围是0～61之间

echo ${a:x:1} //使用字串截取，随机截1位

**编写8位随机字符串脚本**

#!/bin/bash

pass1=

a=abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

for i in {1..8} //循环8次

do

x=$[RANDOM%62] //创建随机数，范围是0～61之间

pass=${a:x:1} //使用字串截取，随机截1位

pass1=$pass1$pass //将每次循环的随机字符磊加赋值给pass1

done

echo $pass1 //循环8次结束后，得出结果

**字串替换**

a=aabbcc

echo ${a/a/8} //将第一个a替换为8

echo ${a//a/8} //将所有a替换为8

echo ${a//a/} //将所有a替换为空，等于删除所有a

**字串删除**

a=`head -1 /etc/passwd` //定义变量

echo $a //查看变量的值

从左往右删除（掐头）

echo ${a#root} //从左往右删除（掐头）root字串

echo ${a#\*:} //从左往右删除 到第一个:

echo ${a##\*:} //从左往右删除 到最后一个:

从右往左删除（去尾）

echo ${a%bash} //从右往左删除（去尾） bash字串

echo ${a%:\*} //从右往左删除 到第一个:

echo ${a%%:\*} //从右往左删除 到最后一个: 结果只显示root

root:x:0:0:root:/root:/bin/bash

思考：只取/root如何操作

例：将完成一半的结果定义为新变量

b=${a%:\*}

echo ${b##\*:}

**编写批量修改文件扩展名的脚本**

版本1：

#!/bin/bash

for i in `ls /opt/\*.txt` //找出所有扩展名是.txt的文件

do

n=${i%.\*} //将找到的文件用去尾功能得到纯粹的文件名

mv $i $n.doc //使用mv命令修改文件名，修改后以.doc结尾

done

版本2：

#!/bin/bash

for i in `ls /opt/\*.$1` //修改为位置变量，增加脚本灵活

do

n=${i%.\*}

mv $i $n.$2

done

**编写脚本,两个文件a和b,分别用a创建用户,用b创建密码**

#! /bin/bash

n=0

for i in `cat /opt/a`

do

useradd $i

let n++

p=`head -$n /opt/b | tail -1 `

echo $p | passwd --stdin $i

done

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

day04**定义变量的初值（备用值）**

echo ${x:-123} //变量x在有值时得到的是本身的值，为空时得到123

**使用定义初值编写创建账户并配置密码的脚本**

#!/bin/bash '

read -p "请输入用户名" u

useradd $u

stty -echo

read -p "请输入密码" p

echo ${p:-123} | passwd --stdin $u //用户没有输入密码直接回车后使用初值123

stty echo

**编写做数字相加运算的for循环脚本**

#!/bin/bash

read -p "请给一个正整数" n //用户输入的数字为n

x=0

for i in `seq $n` //使用seq命令将用户的数字从1开始依次赋值给变量i

do

x=$[x+i] //第一次循环后，x=0+1,第二次循环x=1+2

done

echo $x

${a:} 字串截取

${a/} 字串替换

${a#} 掐头

${a%} 去尾

${a:-} 定义初值

**使用预期交互软件，在脚本中使用交互式工具，首先用yum -y install expect安装，本机是172.25.0.11，即将连接172.25.0.10，并在对方主机中创建文件。**

#!/bin/bash

expect << EOF //使用expect工具

spawn ssh 172.25.0.10 //执行远程连接任务

expect "password:" { send "redhat\r" } //当看到password:时，输入密码redhat并回车，\r表示回车

expect "#" { send "touch /tmp.txt\r" } //当看到#号时，输入创建tmp.txt文本的命令，并回车

expect "#" { send "exit\r" }

EOF

# 正则表达式--------->为文本筛选提供帮助

**基本正则：**

先创建素材 vim abc.txt ，写入以下内容：

root123

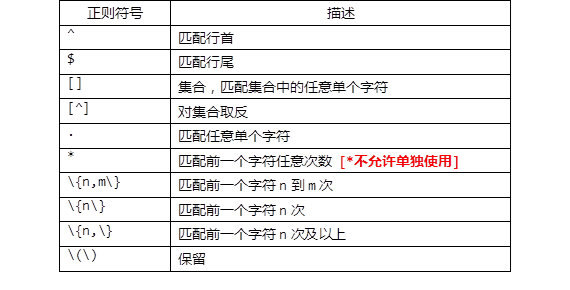
123root

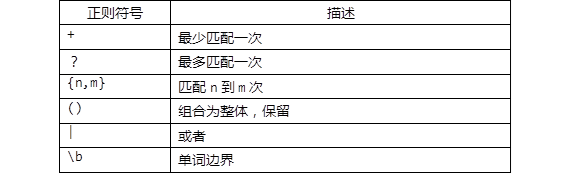
oabc

abctt

789

aaa





grep \w abc.txt 匹配字母数字下划线 等同于[a-Z 0-9 \_]

grep \s abc.txt 匹配空格,制表符(tab)

grep -P \d abc.txt 匹配数字[0-9]

grep "^root" abc.txt //在abc.txt文本中搜索以root单词开头的行

grep "root$" abc.txt //搜索以root单词结尾的行

grep "[root]" abc.txt //搜索文本中含有字母r或o或t的行

grep "[^root]" abc.txt //搜索文本中含有字母r或o或t之外的行

grep "..." abc.txt //搜索有任意三个字符的单词的行

grep "a\*" abc.txt //搜索a出现任意次数的行，0次也包括

grep ".\*" abc.txt //搜索任意字符的行

grep "ro\{1,2\}" abc.txt //搜索字母o出现1～2次的行，前面跟上字母r

grep "ro\{1\}" abc.txt //搜索字母o出现1次的行

grep "ro\{3,\}" abc.txt //搜索字母o出现3次以及3次以上的行

**扩展正则**

egrep "ro{1,2}" abc.txt //使用扩展正则搜索字母o出现1～2次的行，前面跟上字母r，效果与grep "ro\{1,2\}" abc.txt一样

# sed 流式编辑器，逐行处理，可以非交互式对文本进行修改查看等操作

用法1：

前置命令 | sed 选项 指令

用法2：

**sed 选项 指令 文件 //选项包括： -n屏蔽默认输出 -i 直接修改文档 -r 支持扩展正则**

sed 'p' abc.txt //使用sed显示abc.txt文档中所有行，默认会将每行多输出一次

sed -n 'p' abc.txt //使用sed显示abc.txt文档中所有行，-n是屏蔽默认输出，只显示一次所有行

使用行号做定址符

sed -n '1p' abc.txt //只打印第1行在屏幕上

sed -n '2,4p' abc.txt //打印第2行到第4行到屏幕

sed -n '2p;4p' abc.txt //打印第2行和第4行到屏幕

sed -n '2,+2p' abc.txt //打印第2行以及后面2行

sed -n '1~2p' abc.txt //打印第1行，第3行，第5行。。。。

sed -n '2~2p' abc.txt //打印第2行，第4行，第6行。。。。

使用正则做定址符

sed -n '/root/p' /etc/passwd //搜索包含root的行

sed -n '/^root/p' /etc/passwd //搜索以root开头的行

sed -n '/bash$/p' /etc/passwd //搜索以bash结尾的行

sed -n '/ro\{1,\}/p' /etc/passwd //搜索有ro开头的字串，并且o出现1次以及一次以上的行

sed -n -r '/ro{1,}/p' /etc/passwd //使用扩展正则搜索有ro开头的字串，并且o出现1次以及一次以上的行

sed -n '=' abc.txt //显示所有行的行号

sed -n '$=' abc.txt //显示最后一行的行号

sed -n '/root/=' abc.txt //显示有root字串的行的行号

sed 'd' abc.txt //删除文本中所有行

sed '1d' abc.txt //删除文本中第1行

sed '1,2d' abc.txt //删除文本中第1行到第2行

sed '1,+2d' abc.txt //删除文本中第1行以及后面2行

sed '1～2d' abc.txt //删除奇数行

sed '/root/d' abc.txt //删除有root的行

加-i选项后，会对文件做修改，注意不要用系统重要文件测试

sed -i 'p' abc.txt //将原本显示在屏幕的结果保存到文本中

sed -i '1d' abc.txt //删除第1行后，保存到文件

sed -i '$d' abc.txt //删除最后1行后，保存到文件

拷贝/etc/passwd文件到 /opt下，要求通过sed操作，使该文本中只保留解释器使用bash的用户的行

cp /etc/passwd /opt

sed -i '/bash$/!d' /opt/passwd //取反删除以bash结尾的行

使用sed的替换功能：

sed 's/2017/AAAA/' test.txt //使用s替换指令，将文本中每行的第一个2017更换为AAAA

sed 's/2017/AAAA/g' test.txt //使用s替换指令，将文本中每行的所有2017更换为AAAA

sed 's/2017/AAAA/2' test.txt //使用s替换指令，将文本中每行的第2个2017更换为AAAA

sed 's/2017//2' test.txt //使用s替换指令，将文本中每行的第2个2017更换为空，相当于删除2017

sed -n 's/2017/AAAA/p' test.txt //只查看修改成功的行

sed 's/\/bin\/bash/\/bin\/sh/' abc.txt //替换内容与/一样时，需要增加转义符号\

sed 's#/bin/bash#/bin/sh#' abc.txt //更换s分隔符号，将/bin/bash替换为/bin/sh

sed '1s/^/#/' abc.txt //为第一行增加注释符号

sed 's/^#an/an/' //为第一行取消注释

**删除文件中每行的第二个、最后一个字符。**

**思考：能否用下面的方法实现？**

方法一：

sed 's/.//2;s/.$//' nssw.txt

方法二：

sed -r 's/^(.)(.)(.\*)(.)$/\1\3/' nssw.txt

**将文件中每行的第一个、倒数第1个字符互换**

sed -r 's/^(.)(.\*)(.)$/\3\2\1/' nssw.txt //复制每行中第1个任意字符为\1 复制每行中间所有部分为\2 复制最后一个字符为\3 然后从新排序粘贴

sed -r 's/^ +//;s/[0-9]//g' nssw.txt //使用替换，将行首的空格删除，再将文本中所有的数字删除

sed -r 's/([A-Z])/(\1)/g' nssw.txt //将文本中所有的大写字母加一个括号

**关闭防火墙和selinux**

systemctl stop firewalld

setenforce 0

**编写脚本,自动部署ftp,并开启上传功能**

#!/bin/bash

yum -y install vsftpd

cp /etc/vsftpd/vsftpd.conf /root/vsftpd.conf.bak

sed -i 's/^#anon/anon/' /etc/vsftpd/vsftpd.conf

systemctl restart vsftpd

systemctl enable vsftpd

chmod 777 /var/ftp/pub

++++++++++++++++++++++++++++++++++++++++++++++++++++

day05

tac(cat倒过来) /etc/passwd 上下倒过来显示

echo 123456 | rev 左右倒过来显示

sed -n '/root/{=;p}' /etc/passwd 打印包含root的行的内容及行号

sed -n '${=;p;=;p;=;p}' /etc/passwd 打印包含3遍最后一行的内容及行号

sed -r '2ijj' f1 //插入到2行前

sed -r '1ajj' f1 //插入到1行后

sed -r '2cjj' f1 //替换第二行

sed '2ixx\nyy' f1 //折行插入xx和yy

sed '2r m.txt' f1 //在第二行后插入显示m.txt的内容

sed '2,5w m.txt' f1 //把f1文件中的2到5行的内容放入到m.txt

**sed的高级运用**

使用sed复制粘贴，保持空间（剪切版）中默认有一个回车（空行）

h //覆盖的复制 H //追加的复制

g //覆盖的粘贴 G //追加的粘贴

sed '1h;5g' shu //覆盖的复制第1行（剪切版中的空行没有了），覆盖第5行

sed '1H;5G' shu //追加的复制第1行（剪切版中的空行还在），追加粘贴到第5行后

**习题**

#!/bin/bash

sed -n '/bash$/s/:.\*//p' /etc/passwd | while read var;do

echo -n "${var}-->" //用while循环提取用户名

sed -rn 's/root:([^:]+).\*/\1/p' /etc/shadow //提取用户的密码

done

grep 按行进行查找

vim 主要用来编辑文档，交互式

cat 显示文档所有内容

sed 主业是编辑文档，输出不是主要任务

awk 主业是过滤，搜索文本内容，并输出,可以显示不同的列

**格式1： awk [选项] '[条件]{指令}' 文件**

**格式2: 前置命令 | awk [选项] '[条件]{指令}'**

df -h / |tail -1 |awk '{print $4}' //使用awk查看根分区剩余空间

**指定awk分隔符，默认是空格或者tab键**

awk -F: '{print $7}' /etc/passwd //使用:作为分隔符，并输出第7列内容

**awk的内置变量**

FS 保存或设置字段分隔符,例如FS=":",与-F功能一样

$0 输出当前行所有内容

$n 指定分隔的第n个字段

NR 行号

NF 列号

awk -F: '{print NF}' user //输出文本每行有多少列

awk -F: '{print NR,NF}' user //输出文本有多少行，多少列

awk 'END{print NR}' //打印最后一行的行号

**使用awk输出常量**

awk -F: '{print "测试"}' user //输出常量

awk -F: '{print "用户",$1,"的shell是 ",$7}' user //输出内置变量与常量

df -h | awk '/\/$/{print $4}' //使用正则查找结尾是/的行，再输出第4列

**使用awk寻找登陆密码错误的连接者，先在连接时故意输错密码，在/var/log/secure中按G可以看到最后一行记录了连接信息，Failed 是连接失败。**

**使用server远程连接desktop**

awk '/Failed/{print $11}' /var/log/secure //在desktop中输入指令

awk '$6=="Failed"{print $11}' /var/log/secure

**awk处理的时机**

BEGIN{ } 行前处理，读取文件内容前执行，指令执行1次

{ } 逐行处理，读取文件过程中执行，指令执行n次

END{ } 行后处理，读取文件结束后执行，指令执行1次

awk 'BEGIN{print 10+10,2\*3}' //使用awk在逐行任务前做某些操作

awk 'BEGIN{x=0}/bash$/{x++}END{print x}' /etc/passwd //统计使用bash的用户个数

案例：

awk '/Failed/{print "尝试登陆主机的用户是"$11;x++}END{print "失败的次数" x}' secure //使用awk在逐行处理时一旦找到符合正则条件的行时就执行x++，逐行任务结束后输出x的值

案例:

方法1:

awk -F: 'BEGIN{print "User\tUID\tHome"}{print $1"\t"$3"\t"$6}END{print "Total\t" NR "\tline"}' /etc/passwd //使用awk输出带表头与结尾统计的内容 \t是tab效果

方法2:

awk -F: 'BEGIN{print "USER","UID","HOME"}{print $1,$3,$6}END{print "Total" ,"NR", "line"}' /etc/passwd | column -t

**认识awk处理条件的设置**

**1）使用正则、~ 匹配、！～不匹配 设置条件**

awk -F: '$4~/0/{print $4}' /etc/passwd //匹配第四列有0的,然后打印这一列

awk -F: '$4!~/0/{print $4}' /etc/passwd //匹配第四列没有0的,然后打印这一列

awk -F: '$NF~/bash$/{print $1}' /etc/passwd

awk -F: '$3==0&&$4==0{print $1}' /etc/passwd

awk -F: '$3==0||$4==0{print $1}' /etc/passwd

**2）使用数值/字符串比较设置条件**

比较符号：==(等于) !=（不等于） >（大于）

>=（大于等于） <（小于） <=（小于等于）

awk '$1~/^127/||$1~/^192/ {print $1} ' /etc/hosts //输出/etc/hosts文件内127或192开头的记录

seq 100 |awk '$1%7==0 || $1~/7/{print $1}' //输出100以内能被7整除或包含7的数字记录

**习题:找到使用bash的本地用户,列出这些用户的密码记录,按每行"用户名-->密码记录"保存**

#!/bin/bash

awk -F: '/bash$/{print $1}' /etc/passwd |while read var ;do

grep "${var}" /etc/shadow | awk -F: '{print $1, "---->",$2}'

done

3）逻辑测试条件

NR 行

NF 列

awk '{print NF}' nr //输出每行的总列数

awk '{x+=NF}END{print x}' nr //计算一个文档不同行的列数总和

文档内容如下：

abc xyz 123

abc

xyz 123

abc xyz 123 abc xyz 123

abc xyz 123 abc xyz

按 “名字 --> 加密密码” 的格式输出使用bash的用户

#!/bin/bash

n=`awk -F: '/bash$/{print $1}' /etc/passwd` //找到使用bash的用户，并输出名字到变量n中

for i in $n //使用for循环依次显示变量n中的名字

do

grep $i /etc/shadow |awk -F: '{print $1,"-->" ,$2}' //筛出在shadow文件中的

done

# awk的高级运用

**awk中使用if判断**

单分支

awk '{if(条件测试){执行指令}}' 文件

awk '{[条件测试]{执行指令}}' 文件

双分支

awk '{if(条件测试){执行指令} else{执行指令}}' 文件

多分支

awk '{if(条件测试){执行指令} elif(条件测试){执行指令} else{执行指令}}' 文件

**应用1:统计uid小于等于500的用户**

awk -F: '$3<=500{a++}END{print a}' /etc/passwd

awk -F: '{if($3<=500){a++}else{print a}' /etc/passwd

awk -F: '{if($3<=500){a++}else{b++}}END{print "uid<500",a"\nuid>500",b}' /etc/passwd

awk -F: '{if($3<10){i++;j++}elif($3<50){j++}else{l++}}END{print i,j,l }' /etc/passwd

**应用2:数组的定义及使用**

格式:数组名[下标]=元素值

用法:for(变量 in 数组名) (print 数组名[变量])

array=(a b c d e)

echo ${array[2]}

c

awk 'BEGIN{spa[0]="oo";spa[1]="xx";print spa[0]}'

oo

awk 'BEGIN{spa[0]="oo";spa[1]="xx";print spa[1]}'

xx

netstat -ant //连接检测

netstat -an | awk '$1~/tcp/{A[$NF]++}END{for (i in A)print i,A[i]}' //检测网页访问量

awk '{A[$1]++}END{for(i in A)print A[i],i}' /var/log/httpd/access\_log | sort -n //统计访问本机网页的IP及次数

ps -eo %mem,%cpu,ucmd

统计占用cpu最多的程序

ps -eo %mem,%cpu,ucmd | awk '{A[$3]+=$2}END{for(i in A)print A[i],i}'|sort -n

统计占用内存最多的程序

ps -eo %mem,%cpu,ucmd | awk '{A[$3]+=$1}END{for(i in A)print A[i],i}'|sort -n

echo -n //输出并不折行

echo -e //输出并支持固定间隔

echo -e "\n" //输出并折行

echo -e "\t" //固定tab间隔,可多个

echo -e "\r" //回到行首并且覆盖

for i in {00..99};do

> echo -ne "id: ${i}\r"

> sleep 0.5

> done

echo -ne "\033[60Gok" //在第60列打印ok

脚本:

astr=" |-->"

for i in {1..60};do

echo -ne "\033[65G${i}%"

echo -ne "\033[${i}G${astr}"

sleep 0.2

done

watch /实时监控

watch -n 1 'ifconfig eth0' /每秒钟查看网卡流量

-n \设置时间间隔

++++++++++++++++++++++++++++++++++++++++++++++++++++++

OPERATION

day01

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

web服务器对比:

--Apache Nginx Tengine Lighttpd

--Tomcat IBM WebSphere Jboss

\*Nginx (engine X)

更快,更高,更小. Nginx 实验室最大并发量 5W

Apache 实验室最大并发 3W

--是一个高性能的HTTP和反向代理服务器,同时也是一个IMAP/POP3/SMTP 代理服务器

--官网http://nginx.org/

网站: 客户端----服务器

发送请求

回应数据

\*nginx软件是模块化设计的软件,现代化的软件都是模块化的

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NGINX 安装

安装包名称:lnmp\_soft.tar.gz

#md5sum /home/student/桌面/lnmp\_soft.tar.gz

87f559d57dc20ae9f6cd4391804b09e1

#检查安装包是否可用

#scp /home/student/桌面/lnmp\_soft.tar.gz root@192.168.4.5"/

#将安装包发至四台虚拟机 :

\*192.168.4.5 \*192.168.4.10 \*192.168.2.100 \*192.168.2.200

源码安装NGINX

1.安装依赖包

1)官方网站[文档]中会有所要安装的依赖包

#yum -y install gcc pcre-devel openssl-devel #安装依赖包

2).安全:策略都是针对用户和组

人---启动QQ [QQ继承人的权限]

root ---->httpd----客户访问

root执行命令----->以nginx普通用户启动服务 #http安装后自动创建apache用户

#yum -y install gcc pcre-devel openssl-devel #安装依赖包

#useradd -s /sbin/nologin nginx #手动创建nginx用户

#tar -xf nginx-1.10.3.tar.gz #解包 nginx在lnmp中

#cd nginx-1.10.3 #cd到解包目录

#./configure \ #若不加参数则默认安装\ 换行

>--prefix=/usr/local/nginx \ #可以不写,默认安装在该路径

>--user=nginx \ #指定用户启动程序

>--group=nginx \ #指定组

>--with-http\_ssl\_module #开启ssl(加密)服务

[-with-模块][-without-模块]

#make && make install #make编译成二进制

#ls /usr/local/nginx/ #安装好后 该目录存在

conf 配置文件 html 网页 logs 日志 sbin 主程序

#/usr/local/nginx/sbin/nginx #启动程序

#/usr/local/nginx/sbin/nginx -s stop #关服务

#/usr/local/nginx/sbin/nginx -s reload #重新加载配置文件

#/usr/local/nginx/sbin/nginx -V #查看编译参数

#ln -s /usr/local/nginx/sbin/nginx /sbin/ #在sbin创建nginx的软连接

2.常用选项

1)指定配置文件,启动服务:

#/usr/local/nginx/sbin/nginx -c

2)查看相关进程

#ps aux | grep nginx

3)查看端口信息

#netstat -anutlp | grep nginx

u:udp t:tcp n:number p:PID/Program a:all l:listen

3.nginx 版本升级

以下情况需要升级: 1.版本老旧,需要升级 2.安装时缺少模块,升级相同版本

#tar -xf ngnix-1.12.2

#cd ngnix-1.12.2

#./configure

>--prefix=/usr/local/nginx \

>--user=nginx \

>--group=nginx \

>--with-http\_ssl\_module

#make

#mv /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginxold

#cp /root/nginx-1.12.2/objs/nginx /usr/local/nginx/sbin/

#用新的主程序替换老的nginx

#make upgrade #重启服务 命令不好用会报错

#killall nginx #杀死原nginx

#/usr/local/nginx/sbin/nginx #重启服务

#/usr/local/nginx/sbin/nginx -V #查看版本 1.12 即为更新完成

/src 是nginx的C语言源代码 /objs是./configure之后生成的源码包,

make编译之后生成nignx可执行文件.

目前两个nginx 执行程序:

/usr/local/nginx/sbin/nginx(1.10)

/root/lnmp\_soft/nginx-1.12.2/objs/nginx(.12)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NGINX 的具体应用

主配置文件: /usr/local/nginx/conf/nginx.conf

httpd配置

<virtualhost \*:80>

Documentroot /var/www/html

Servername www.example.com

</virtualhost>

nginx配置

server {

listen 80; #端口号

server\_name www.baidu.com; #域名设置

root html; #与documentroot 对应

root "/usr/local/nginx/html"; #绝对路径

location / {

root html;

index index.html index.htm; #默认首页

} #若找不到第一个找第二个......

}

注:若找不到index.html则访问index.htm

1)用户认证 1.访问web页面需要用户密码认证 用户名tom密码123456

vim /usr/local/nginx/conf/nginx.conf

.....

server{

....

server\_name localhost;

auth\_basic "Input Password:"; #信息提示

auth\_basic\_user\_file "/usr/local/nginx/pass"; #存放用户密码文件,需要管理员创建

.....

}

2)生成密码文件

#yum -y install httpd-tools

#htpasswd -c /usr/local/nginx/pass tom #生成密码文件,-c添加文件用户为tom

New password:

Re-type new password:

Adding password for user tom

#cat /usr/local/nginx/pass #查看用户及密码

#htpasswd /usr/local/nginx/pass shabi #追加创建用户

#nginx -s reload #重启服务

#firefox 192.168.4.5 #网页查看web. 提示输入用户名密码

#tailf logs/error.log #动态查看报错日志文件的最后几行,可以实时查看

3) Nginx 虚拟主机

http虚拟主机:基于域名.基于IP,基于端口.一台服务器,[多个网站]

nginx

server{

listen 80;

server\_name www.exampple.com #基于域名

}

server{

listen 80;

server\_name web.example.com

}

案例3: 实现两个基于域名的虚拟主机:域名分别为www.a.com和 www.b.com

对域名为a的站点进行用户密码验证,b可以直接访问

#vim /usr/local/nginx/conf/nginx.conf

.....

server {

listen 80;

server\_name www.a.com;

auth\_basic "Input Password:";

auth\_basir\_user\_file "/usr/local/nginx/pass";

location / {

root html;

index index.html index.htm;

}

}

server {

listen 80;

server\_name www.b.com;

location / {

root www;

index index.html index.htm

}

}

........

#nginx -s reload #刷新配置文件

#echo -e "192.168.4.5 www.a.com\n192.168.4.5 www.b.com" > /etc/hosts #添加域名

#firefox www.a.com // www.b.com #实现访问

++++++++++++++++++++++++++++++++++++++++

day02

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

昨日回顾

案例3:虚拟主机(基于域名,IP,端口)----基于域名的虚拟主机

vim /usr/lcoal/nginx/conf/nginxconf

.......

server {

listen 80;

server\_name www.a.com;

location / {

root html;

index index.html index.htm;

}

}

server {

listen 80;

server\_name www.b.com;

location / {

root html;

index index.html index.htm;

}

}

.......

#nginx-s reload

扩展: 制作基于端口的虚拟主机,基于IP的虚拟主机

vim /usr/lcoal/nginx/conf/nginxconf

.......

server {

listen 80;

server\_name www.a.com;

location / {

root html;

index index.html index.htm;

}

}

server {

listen 8000;

server\_name www.a.com;

location / {

root html;

index index.html index.htm;

}

}

扩展:制作基于IP的虚拟主机

listen 192.168.4.5:80

server\_name www.a.com;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

nginx网站加密

nginx http协议:为明文协议 http+ssl=https

加密类型

\*对称加密:AES,DES[很安全,目前为止没有被人破解,将密码转译为1024位密码](暴力破解,枚举) [单机加密很安全,,基于工作原理,网络加密不安全]

\*非对称加密:RSA,DSA 私钥,公钥

\*信息摘要:md5(已被破解),sha256,sha512.主要应用在数据完整性校验.

[root@zhw ~]# md5sum /mnt/1.sh

12cf2d15dfd66d4371cf3a0162c19f57 /mnt/1.sh

#上诉码只与文件内容有关

1) 生成公私钥

#cd /usr/local/nginx/conf #cd到conf下

conf]#openssl genrsa > cert.key #生成私钥

conf]#openssl req -new -x509 -key cert.key > cert.pem

-->国家名

-->所在省名

-->城市名

-->公司名

-->部门名

-->你名字

-->邮箱

conf]#ls

cert.key cert.pem #私钥及证书文件

2)更改配置文件

#vim nginx.conf

.....

\* server\_name www.c.com; #修改域名

\* ssl\_certificate cert.pem; #证书

\* ssl\_certificate\_key cert.key; #私钥

ssl\_session\_cache shared:SSL:1m; #缓存时间

ssl\_session\_timeout 5m; #超时时间

ssl\_ciphers HIGH:!aNULL:!MD5;

#网站一定要加密,密码不能为空,加密算法不能是md5

ssl\_prefer\_server\_ciphers on;

.....

conf]#echo "192.168.4.5 www.c.com" >> /etc/hosts

conf]#firefox https://www.c.com

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

什么是动态页面

静态: [jpg,doc,mp3,mp4,pdf,ppt,gif...]

动态: [java php python shell ........]

代码在服务器上执行一次,脚本执行结果给用户

\*LNMP环境的部署 linux nginx mysql php 解释器

1)#yum -y install mariadb mariadb-server mariadb-devel

#客户端软件 服务端软件 其他客户端软件依赖包

2)#yum -y install php php-mysql php-fpm

#解释器 php数据库扩展 进程管理器服务

#vim /usr/local/nginx/html/test.php #编写一个PHP脚本

<?php

$i=33; #测试不建立动态页面,用户访问结果之用

echo $i;

?>

#php /usr/local/nginx/html/test.php #试运行PHP脚本

33

3) 启动各个服务:

#systemctl start mariadb #启动mariadb服务

#systemctl start php-fpm #启动php服务

#ss -alnutp | grep 9000 #查看php服务(端口号:9000)

#ss -alnutp | grep 3306 #查看mariadb服务(端口号:3306)

#ss -alnutp | grep 80 #查看nginx服务(端口号:80)

程序(硬盘) ---->内存(进程)

进程 (30M)

线程(共享内存)

线程

线程(若进程死了,下面的线程均挂掉)

4)编辑php配置文件

vim /etc/php-fpm.d/www.conf

[www]

listen = 127.0.0.1:9000 #PHP端口号

listen.allowed\_clients = 127.0.0.1

user = apache

group = apache

pm = dynamic

pm.max\_children = 50 #最大进程数量

pm.start\_servers = 5 #最小进程数量

pm.min\_spare\_servers = 5 #最少需要几个空闲进程

pm.max\_spare\_servers = 35 #最多允许几个进程处于空闲状态

5) location匹配用户的地址栏(从域名往后的部分)

location / { # / 匹配优先级最低

deny 1.1.1.1; #location支持正则

allow all;}

location /abc {

deny all; }

location /dachui {

allow all;}

firefox http://www.a.com/

firefox http://www.a.com/abc #被拒绝

firefox http://www.a.com/dachui #允许

firefox http://www.a.com/qq #若匹配不到,则与根匹配

location ~ /abc #~正则匹配,地址栏包含abc即匹配 (模糊匹配)

\*部署php页面[动静分离]:

server {

listen 80;

server\_name www.a.com;

location / {

root html;

}

\* location ~ \.php$ { # \.取消.的正则

root html;

\* fastcgi\_pass 127.0.0.1:9000;

#pass 处理不了交给下家 php端口:9000

fastcgi\_index index.php;

\* include fastcgi.conf; #加载nginx的内置变量

}

}

#firefox http://www.a.com/test.php

#curl -u tom:123 http://www.a.com/test.php

输出结果 33

6)LNMP常见问题

Nginx的默认访问日志文件为/usr/local/nginx/logs/access.log

Nginx的默认错误日志文件为/usr/local/nginx/logs/error.log

PHP默认错误日志文件为/var/log/php-fpm/www-error.log

如果动态网站访问失败，可用参考错误日志，查找错误信息。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

流程:user----nginx-----PHP(代码)----mysql

mysql-e "select \* from mysql.usr" #-e非交互

#cp /lnmp\_soft/php\_scripts/mysql.php /usr/local/nginx/html/ #将已编辑好的mysql.php 放到nginx/html下

#firefox www.a.com/mysql.php #访问mysql.php

++++++++++++++++++++++++++++++++++++

day03

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

昨日回顾

nginx 静态网页 共享目录----/usr/local/nginx/html

nginx 脚本 需通过 php,mariadb 执行结果,最终将执行结果转发给用户

php的配置文件 /etc/php-fpm.d/www.conf

cat -A /etc/php-fpm.d/www.conf #-A 显示换行字符

client(客户端)----nginx(web)

判断(动态,静态)

location / {

root html; #/usr/local/nginx/

}

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000; #fastcgi 就是指fpm进程管理器

include fastcgi.conf;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

地址重写rewrite0(地址栏被重写)

eg: www.360buy.com--->www.jd.com

格式: rewrite 旧地址 新地址 [选项] #旧地址 支持正则表达式

\*案例1: 访问a.html 自动跳到b.html

......

server\_name www.a.com;

rewrite /a.html /b.html redirect; #www.a.com/a.html

......

#firefox www.a.com/a.html

#charset koi8-r; ---> charset utf-8;改后可以在b中输入中文

character set #字符集 选某一种编码

编码: 人--英语 计算机:01

ASCII 码只到127位 [后来的码均兼容ASCII码]

第一批中文编码,只编了常见的几千个字.

中文编码:GB2312

\*apple[万国编码] utf-8

\*案例2:访问192.168.4.5跳转到www.tmooc.cn

......

server\_name www.a.com;

rewrite ^/ http://www.tmooc.cn; # ^/ 匹配/开头即可.正则

......

真机: firefox http://www.a.com/hsad;laskjljaklsd;jl

\*案例3: 访问 www.360buy.com/book.html 跳转到

不同域名的相同子网站 www.jd.com/book.com

正则保留: 基本正则 \(\) 扩展正则() ----复制----->粘贴\1

eg: (a)(b)/\1\2 #\1为粘贴a的内容

......

server\_name www.a.com;

rewrite ^/(.\*) http://www.tmooc.cn/$1; #nginx中粘贴为$1

(.\*) 为复制匹配输入的所有 $1粘贴(.\*)复制的内容

\*案例4:不同浏览器访问相同连接返回不同内容

eg: 电脑ie http://www.sina.com.cn

手机uc http://www.sina.com.cn

/usr/local/nginx/html/(test.html)宽 电脑

/usr/local.nginx/html/xx/test.html(窄) 手机

准备两个页面:

#echo "I am Normal page" > html/test.html

#echo "firefox page" > html/firefox/test.html

......

if ($http\_user\_agent ~\* firefox) {#~代表模糊比较\*不区分大小写

rewrite ^/(.\*) /firefox/$1; #访问火狐下的$1

}

#否则访问根下的test.html

firefox http://192.168.4.5/test.html

google-chrome http://192.168.4.5/test.html

rewrite 选项参数:

--last 不再读其他的rewrite

--break 不再读其他reewrite语句,结束请求

--redirect 临时重定向

--permament 永久重定向

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*日志access.log的格式:

第一列:$remote\_addr 客户端IP地址

第二列: -

第三列:$remote\_user 用户名(tom)

第四列:[$time\_local] 时间及时区(东八区)

第五列:"$request" 用户的请求 (get /test.php)

第六列:$status 结果状态 [302/303重定向.200代表正常.404页面找不到] /favicon.ico--->标签最左边的图标文件.eg: 百度的

第七列:$body\_bytes\_sent 网页大小 字节

第八列:"$http\_referer" 空 eg:refer=www.baidu.com 百度上访问的

\*第九列:$http\_user\_agent 存用户信息 操作系统 浏览器/版本号

谷歌浏览器chrome

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NGINX 反向代理

nginx(web服务器,代理服务器)

代理:有2功能:调度(负载均衡), 健康检查(当服务器坏了又好了时,自动连接使用)

案例:

使用4台RHEL7虚拟机，其中一台作为Nginx代理服务器，该服务器需要配置两块网卡，IP地址分别为192.168.4.5和192.168.2.5，两台Web服务器IP地址分别为192.168.2.100和192.168.2.200。客户端测试主机IP地址为192.168.4.10

步骤一:部署实施后端web服务器(两台)

#yum -y install httpd

echo "192.168.2.100" > /var/www/html/index.html

#systemctl restart httpd

#firewall-cmd -set-default-zone=trusted #clone虚拟机不用

#setenforce 0

#yum -y install httpd

echo "192.168.2.200" > /var/www/html/index.html

#systemctl restart httpd

#firewall-cmd -set-default-zone=trusted

#setenforce 0

步骤二:在192.168.4.5 proxy上 配置nignx服务器,添加服务池

#在对应的server上增加

upstream webserver { #upstream 定义集群 webserver 集群名字

ip\_hash; #算法改成了ip\_hash.相同客户端访问相同的web服务器

server 192.168.2.100:80 weight=1 max\_fails=1 fail\_timeout=30;

server 192.168.2.200:80 weight=2 max\_fails=1 fail\_timeout=30;

server 192.168.2.101 down;

}

\*weight 权重

\*max\_fails 连接web并认为web坏了的最大失败次数

\*fail\_timeout=30 隔30秒去连接web服务器,

\*down 标记服务器 彻底坏了,不参与集群

server {

listen 80;

server\_name www.a.com;

charset utf-8;

location / {

\* proxy\_pass http://webserver; #调用集群 代理转发

root html; #root优先级低

index index.html index.htm;

}

}

\*调度算法:

轮询算法(roundrobin,RR)

ip\_hash; 相同客户端访问相同的web服务器

\*ip\_hash 原理:

192.168.4.10 | md5sum 0-9A-F(16进制)

数字%2===1 nginx ---->web2

#ip不变 md5sum值不变,数字不变.有n台主机,就对n取余

案例2: 上述均建立在http协议上进行,其他服务(dns,ftp...)无法进行

http协议===>7层 http ftp dhcp

tcp,udp===>4层

nginx支持tcp,udp代理(4层代理).

\*要求1) nginx-1.9 以上 2) 增加模块 --with-stream

+++++++++++++++++++++++++++++++++++++

day04

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

昨日回顾

nginx(web服务器,代理服务器:实现调度)

调度算法: RR IP\_hash;

http代理服务器(不是所有软件都使用http协议)

nginx-1.9 (TCP,UDP)

加密 --with-http\_ssl\_module

TCP,UDP --with-stream

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*安装--with-stream 模块,提供四层代理功能.

#yum -y install gcc pcre-devel openssl-devel

#tar -xf nginx-1.12.2.tar.gz

#cd nginx-1.12.2

#./configure

--with-http\_ssl\_module #开启ssl加密功能

--with-stream #开启4层反向代理功能

#make&&make install #编译安装

案例2:ssh代理服务器 (配置文件加到http外)

....

stream{

upstream backend {

server 192.168.2.100:22;#可只写一server

server 192.168.2.200:22;

}

server {

listen 12345;#代理也正使用22port,所以需改成其他端口>1024

proxy\_connect\_timeout 1s; #可选参数

proxy\_timeout 3s; #可选参数

proxy\_pass backend; #转发

}

}

http{}

....

#nginx -s reload

#ssh 192.168.4.5 -p 12345

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例3:nginx常见问题处理

1).如何自定义返回给客户端的404错误页面

优化前:

#firefox http://192.168.4.5/xxxxx

修改Nginx配置文件，自定义报错页面:

#vim conf/nginx.conf

......

charset utf-8;

error\_page 404 /404.html; #404报错页面

......

# echo "<h1><color=32>您要的页面不存在" > html/404.html

#firefox www.a.com/xxxasd --->404.html

\*404转到其他页面

......

error\_page 404 /404.html;

location = /404.html {

rewrite ^/404.html http://www.tmooc.cn;

}

......

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

常见错误返回码描述:

状态码 功能描述

200 一切正常

301 永久重定向

302 临时重定向

401 用户名或密码错误

403 禁止访问(客户端IP地址被拒绝)

404 文件不存在

414 请求URI头部过长

500 服务器内部错误

502 Bad Gateway

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2)如何查看看服务器状态信息

1.实时并发量是多少?

2.等待的数量?

3.总连接数量是多少?

4.PV量和UV量是多少?

page view 页面被人刷新看了的次数.

user view 用户访问的人数.

查看需要的module

./configure --help | grep "with-"

1.重新安装nginx, nginx提供一个专门的模块:

./configure

--with-http\_stub\_status\_module #查看状态信息

--with-http\_ssl\_module

--with-stream

make && make install #清空原有的配置

2.修改配置文件

vim conf.nginx.conf

....

server{

location /status { #/status 可替换成任意

stub\_status on;

#allow IP地址; eg: allow myip

#deny IP地址; eg: deny all

}

}

.....

#nginx -s reload

#firefox 192.168.4.5/status

Active connections: 1

server accepts handled requests

1 1 1

#客户端发的请求次数,服务端处理的请求次数 请求数量

Reading: 0 Writing: 1 Waiting: 0

\*Active connections 当前活动的连接数

\*Accept:已经接受客户端的连接总数量

\*Handled:已经处理客户端的连接总数量

一般与accepts一致，除非服务器限制了连接数量

\*Requests:客户端发送的请求数量

\*Reading：当前服务器正在读取客户端请求头的数量

\*Writing：当前服务器正在写响应信息的数量。

\*Waiting：当前多少客户端在等待服务器的响应

#wc -l /user/local/nginx/logs/access.log #查看PV量

#awk '{IP[$1]++}END{for(i in IP){print i,IP[i]}}' access.log

#查看UV量

模拟100个人访问 网页

#ab -c 100 -n 100 http://192.168.4.5/ #-c client -n number

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3)优化nginx并发量

1.ab高并发测试 2000以上访问出错

ab -c 2000 -n 2000 http://192.168.4.5/

#-c client访问客户 -n 访问次数 number

2.修改配置文件(增加nginx的并发连接)

vim conf/nginx.conf

......

worker\_processes 2; #与CPU核心数量一致,启动进程的数量

events{

worker\_connection 65535; #每个worker最大并发连接数

}

......

3)优化linux内容参数(最大文件数量)

任何操作系统:资源限制

打开文件 最大个数 1024

#ulmit -a 查看所有限制值

#ulimit -Hn 100000 #设置硬限制[不可超过的限制] 临时规则

#ulimit -Sn 100000 #设置软限制 [可以突破的限制---警告值] 临时规则

#ulimit -a

#vim /etc/security/limits.conf

\* soft nofile 1000000 #用户或组 硬限制/软限制限制的项目 限制值

\* hard nofile 1000000

#ab -c 5000 -n 5000 http://192.168.4.5/ #能突破1024

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4) 优化nginx数据包头缓存

#bash /root/lnmp\_soft/buffer.sh

414 URI头部过长

client --------------->server

请求 [数据包]网卡--内存[1k存地址栏] nginx

修改配置文件

#vim conf/nginx.conf

......

http{

client\_header\_buffer\_size 1k; #默认请求包头信息的存 large\_client\_header\_buffers 4 4k; ##大请求包头部信息的缓存个数与容量

.....

}

nginx -s reload

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5)浏览器本地缓存静态数据

真机火狐中输入: about:cache 查看浏览器缓存信息

该实验适合 长期不变的数据.(jpg,logo,MP3,MP4)

浏览器带缓存功能,缓存什么.缓存多久,取决于服务器

vim conf/nginx.conf

....

server {

....

location ~ \.(jpg|png|gif|mp3|mp4|flv)$ {

expires 30d;

}

...

}

#cp /usr/share/backgroud/day.jpg html

#nginx -s reload

#firefox 192.168.4.5/day.jpg #优化后火狐访问,查看本地缓存

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6) 日志切割

#mv logs/access.log logs/access.log.bak #老日志改名

#kill -USR1 $(cat logs/nginx.pid) #生成新日志

#ls logs/

access.log access.log.bak

kill -l #查看kill的选项参数. kill 传递信号

-18 继续 -19 停止 -15温柔的杀死 -9强制杀死

脚本:每周5的3点3分自动执行日志分割脚本:

#vim /usr/local/nginx/logbak.sh

#!/bin/bash

date=`date +%Y%m%d`

logpath=/usr/local/nginx/logs

mv $logpath/access.log $logpath/access-$date.log

mv $logpath/error.log $logpath/error-$date.log

kill -USR1 $(cat $logpath/nginx.pid)

#crontab -e

03 03 \* \* 5 /usr/local/nginx/logbak.sh

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7)对页面开启压缩功能

所有浏览器都支持gzip解压

[root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf

http {

.. ..

gzip on; //开启压缩

gzip\_min\_length 1000; //小文件不压缩1000字节

gzip\_comp\_level 4; //压缩比率级别1-9压缩效果最好

gzip\_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;

//对特定文件压缩，类型参考mime.types

.. ..

}

#cat conf/mime.types

类型 扩展名

application/pdf pdf

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

8)服务器内存缓存

如果需要处理大量静态文件，可以将文件缓存在内存，下次访问会更快。

http {

open\_file\_cache max=2000 inactive=20s;

open\_file\_cache\_valid 60s;

open\_file\_cache\_min\_uses 5;

open\_file\_cache\_errors off;

//设置服务器最大缓存2000个文件句柄，关闭20秒内无请求的文件句柄

//文件句柄的有效时间是60秒，60秒后过期

//只有访问次数超过5次会被缓存 热点数据放缓存

//缓存如果错误了,不报错

}

lnmp\_soft/nginx.conf

+++++++++++++++++++++++++++++++++++++++

day05

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

远程给100,200 安装nginx

for i in 100 200

do

ssh 192.168.2.$i "yum -y install gcc pcre-devel openssl-devel"

done

uuidgen 自动生成uuid.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

验证 Session和Cookies

Session:存储在服务器,保存用户名,登录状态等信息.(eg:uuid.txt)

Cookies:由服务器下发给客户端,保存在客户端的一个文件里.

服务器传给client的数据包,包头中cookie:uuid.txt-->浏览器保存

F5刷新,浏览器拿出cookie:uuid.txt 给服务器

环境部署好后:

步骤五:

1).部署测试页面web1 部署测试页面web2 同理

[root@web1 ~]# cd lnmp\_soft/php\_scripts/

[root@web1 php\_scripts]# tar -xf php-memcached-demo.tar.gz

[root@web1 php\_scripts]# cd php-memcached-demo

[root@web1 php-memcached-demo]# cp -r \* /usr/local/nginx/html/

#firefox 192.168.2.100

#cd /var/lib/php/session/ #服务器创建的session位置

#ls

sess\_1l8gn1c42gbti6rujc95rgj360

#cat sess\_1l8gn1c42gbti6rujc95rgj360

login\_user|s:11:"zhouhuowang";logged\_in|s:4:"true";id|s:26:"1l8gn1c42gbti6rujc95

#firefox 192.168.2.100/index.php

1.浏览器中清除cookie缓存,则需要用户重新登入

2.服务器中清除session,则用户也需要重新登入

2)修改web1的php网页颜色

mem.php中修改 <body bgcolor=pink>

memcachel(内存):利用session共享解决 集群中的用户登入问题.

案例2:构建memcached服务

memcached :可以指定数据在内存的存放的时间.

web1

client-----proxy <--memcached(内存)

web2

1.在代理服务器安装memcached软件. (KV数据库) 192.168.4.5 装memcache数据库

#yum -y install memcached telnet #服务端,客户端

#systemctl start memcached.service #启动服务

#vim /usr/lib/systemd/system/memcached.service #第一个配置文件

#vim /etc/sysconfig/memcached #第二个配置文件

#telnet 192.168.4.5 11211 #连接至memcached

key=value

set key 0 180 3 #数据key,0 不压缩 180存180秒,3存3个字节

abc

STORED

get key #查询key

VALUE key 0 4

abcd

END

quit #退出数据库

memcached数据库其他指令:

add myname 0 180 10 //新建，myname不存在则添加，存在则报错

set myname 0 180 10 //添加或替换变量

replace myname 0 180 10 //替换，如果myname不存在则报错

get myname //读取变量

append myname 0 180 10 //向变量中追加数据

delete myname //删除变量

flush\_all //清空所有

quit

案例3:LNNMP+memcached 测试web1,2连接memcached.此处只测试了web1

安装php的memcache扩展

#yum -y install php-pecl-memcache

\*写一个php的脚本测试web1能否连接上memcache

#cp /root/lnmp\_soft/php\_scripts/mem.php html/ #脚本拷至html

注:脚本内IP需更改,若是访问本机memcache,则改为本机ip,且启动

memcache服务;若访问代理memcache则写代理服务器的ip.

#firefox 192.168.2.100/mem.php

访问结果失败 因为未php重启 生效memcache模块

#systemctl restart php-fpm

#firefox 192.168.2.100/mem.php

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例4:php实现session共享

1)后端两台服务器均需要修改配置文件

#vim /etc/php-fpm.d/www.conf

php\_value[session.save\_handler] = files

php\_value[session.save\_path] = /var/lib/php/session

改为------->

php\_value[session.save\_handler] = memcache

#主机参数中为memcache

php\_value[session.save\_path] = "tcp://192.168.2.5:11211"

#定义公共的memcached服务器在哪（服务器的IP和端口）

#systemctl restart php-fpm

#goodle-chrome 192.168.4.5/index.php

#现在就只需登录一次,就能进网页,并且刷新都不会跳出,web服务器轮换

在192.168.4.5上访问memcache

#telnet 192.168.4.5 11211

get hkahla2of6v2d91agb0qu5ge80 (session ID)

VALUE hkahla2of6v2d91agb0qu5ge80 0 80

login\_user|s:4:"zhou";logged\_in|s:4:"true";id|s:26:"hkahla2of6v2d91agb0qu5ge80";

END

#可以在4.5上查到,表示目前web1,web2上的session以共同存在4.5的memcache数据库上.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

++++++++++++++++++++++++++++++++++++++++

day06

高级运维

nginx+PHP

tomcat+java

---Java是一种跨平台的,面向对象的程序设计语言,Java技术具有卓越的通用性,高效性,平台移植性和安全性.

---Java[通用语言] 游戏 web网站 财务软件 office

---Php [网页]

软件包名称:JDK--分为openjdk,jdk(oracle),其他

包括java运行环境,java工具,java基础的类库

jre阉割版

包括java虚拟机(jvm),java核心类库和支持文件

步骤一:部署Tomcat服务器软件

1)安装jdk1.8

#yum -y install java-1.8.0-openjdk #安装jdk

#yum -y install java-1.8.0-openjdk-headless

#java -version #查看JAVA版本

\*常见servlet容器

--IBM websphere

--Oracle weblogic

--Apache tomcat

--RedHat JBOSS

2)安装tomcat

#tar -xf /root/lnmp\_soft/apache-tomcat-8.0.30.tar.gz

#mv apache-tomcat-8.0.30 /usr/local/tomcat

#移动至/usr/local/ 并改名为tomcat

#ls

bin /主程序目录 lib /库文件目录

logs /日志目录 temp /临时目录

work /自动编译目录jsp代码转换servlet

conf /配置文件目录 webapps /页面目录

#/usr/local/tomcat/bin/startup.sh

#/usr/local/tomcat/bin/shutup.sh #关闭程序

#netstat -auntlp | grep java

#启动的话能看到三个端口 8005,8080,8009

tomcat每次启动都要从内存读取足够多的随机数据

#strings /dev/random #tomcat从/dev/random读取随机数

#strings /dev/urandom #

如果8005端口启动慢需进行如下操作:

#mv /dev/random /dev/random.bak

#ln -s /dev/urandom /dev/random #做urandom软连接成random

访问tomcat:

#firefox 192.168.2.100:8080

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

步骤二:修改tomcat配置文件

1)创建测试JSP页面

#vim /usr/local/tomcat/webspps/ROOT/test.jsp #网页放置位置

<html>

<body>

<center>

Now time is: <%=new java.util.Date()%> //显示服务器当前时间

</center>

</body>

</html>

#/usr/local/tomcat/bin/shutdown.sh

#/usr/local/tomcat/bin/start.sh #新建页面不用重启

#firefox 192.168.2.100:8080/test.jsp #访问测试页面

格式:

tomcat

#vim /usr/local/tomcat/conf/server.xml #配置文件

<connector port=8080> #端口8080

<Host name="www.a.com" appBase="webapps"#域名 代码目录

unpackWARs="true" autoDeploy="true">

</Host>

<host name=www.b.com appBase=b> #代码放在b/ROOT/

</host>

#jar -cvf desk.war Desktop/ #打包文件 -c创建

---->desk.war

#jar -xf 解包

unpackWARs="true" 自动解包war

autoDeploy="true" 更新后自动部署

案例:创建两个虚拟主机

#vim conf/server.xml

<Host name="www.a.com" appBase="a"

unpackWARs="true" autoDeploy="true">

</Host>

<Host name="www.b.com" appBase="b"

unpackWARs="true" autoDeploy="true">

</Host>

#mkdir -p {a,b}/ROOT #创建a/ROOT b/ROOT目录

#echo "aaa" > a/ROOT/index.html #创建页面

#echo "bbb" > b/ROOT/index.html

#echo "192.168.2.100 www.a.com www.b.com" >> /etc/hosts

重启tomcat服务:

#/usr/local/tomcat/bin/shutdown.sh

#/usr/local/tomcat/bin/startup.sh

#firefox www.a.com:8080

#firefox www.b.com:8080

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

步骤二:修改www.b.com网站的首页目录为base

vim conf/server.xml

......

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

<Context path="" docBase="base"/>

.....

# mkdir b/base #创建b的根目录

# echo "BASE" > b/base/index.html

# bin/shutdown.sh

# bin/startup.sh

#firefox www.b.com:8080

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

步骤三:页面跳转

当用户访问www.a.com/test 打开/var/www/html目录下的页面

vim conf/server.xml

.......

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

<Context path="/test" docBase="/var/www/html/" />

#访问的子目录 跳转目录

# echo "Test" > /var/www/html/index.html

#bin/shutdown.sh

#bin/startup.sh #重启服务

#firefox www.a.com:8080/test

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

步骤四:配置Tomcat支持ssl加密网站

1)创建加密用的私钥和证书文件

#keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat/keystore

//-genkeypair 生成密钥对

//-alias tomcat 密钥别名

//-keyalg RSA 定义密钥算法为RSA算法

//-keystore 定义密钥文件存储在:/usr/local/tomcat/mi

密码: 按Y确定

2)修改配置文件

vim conf/server.xml

...

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

keystoreFile="/usr/local/tomcat/keystore" keystorePass="123456" #添加秘钥文件路径及密码

clientAuth="false" sslProtocol="TLS" />

....

#bin/shutdown.sh

#bin/startup.sh #重启服务

#firefox https://www.a.com:8443/ #8443为Tomcat的https的端口

#firefox https://www.b.com:8443/

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

步骤五:

/logs/catalina.log #为程序日志 ,排错日志

/logs/localhost\_access\_log #为cat的日志

#vim conf/server.xml

.... #在a,b页面下添加

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs" #日志存放目录

prefix=" a\_access" suffix=".txt"

#日志存放名 时间自动添加 日志扩展名

pattern="%h %l %u %t &quot;%r&quot; %s %b" /> #日志格式

......

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例三:使用Varnish加速Web //nginx也可以做缓存!!!!

client----varnish-----web1

代理+缓存页面

缓存软件主流: squid,varnish,nginx,Apache,cache...

分离解析(view),DNS--->IP1

DNS---->IP2

CDN(Contect delivey network) 内容分发网络

1) 创建web1页面

2)源码安装varnish

#yum -y install gcc readline-devel

#yum -y install ncurses-devel

#yum -y install pcre-devel

#yum -y install python-docutils

#useradd -s /sbin/nologin varnish

#tar -xf /root/lnmp\_soft/varnish-5.2.1.tar.gz

#cd varnish-5.2.1

#./configure

#make && make install

#cp etc/example.vcl /usr/local/etc/default.vcl

#拷贝配置文件至 /usr/local/etc/default.vcl

3)修改代理配置文件

#vim /usr/local/etc/default.val

backend default { #后台服务器

.host = "192.168.2.100"; #ip

.port = "80"; #端口号

}

4)启动服务

#varnishd -f /usr/local/etc/default.vcl #启动varnish

默认缓存为内存

#varnishd -s malloc,128M

定义varnish使用内存作为缓存，空间为128M. 空间小 快

#varnishd -s file,/var/lib/varnish\_storage.bin,1G

定义varnish使用文件作为缓存 空间大,速度慢

5)查看varnish日志

[root@proxy ~]# varnishlog /varnish实时日志

[root@proxy ~]# varnishncsa //访问实时日志

6)更新缓存数据: (默认也会自动更新，但非实时更新)

# varnishadm

varnish> ban req.url ~ .\* #清空缓存数据,立即更新

++++++++++++++++++++++++++++++++++++

day07

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

版本控制软件服务器共享文件夹(文件)

服务器(版本控制):common/nginx.conf

常见免费软件:cvs,subversion(svn),git

开发----git--->/usr/local/nginx/html

rsync+inotifywait 可以事先实时更新

Git仓库:保存所有数据的地方

工作区:从仓库中提取除文件,放在磁盘上供你使用,修改.

暂存区:索引文件

步骤一:部署Git服务器

web1上:

#yum -y install git

#git --version #查看版本号

初始化一个空仓库

#mkdir /var/git

#git init /var/git/project --bare #init初始化一个空仓库

#cd /var/git/

projetct #创建好的仓库

步骤二:客户端测试:web2

clone 将远程服务器的仓库克隆到本地

config 修改git配置

add 添加修改到暂存区

commit 提交修改到本地仓库

push 提交修改到远程服务器

1)克隆git

#yum -y install git

#git clone root@192.168.2.100:/var/git/project

//下载project到当前目录

project/工作目录

project/.git/本地仓库

project/.git/文件(索引)

2)修改git配置

#git config --global user.email "you@example.com"

#git config --global user.name "Your Name"

[user] //填写用户名邮箱

email = you@example.com

name = Your Name

3)本地工作区对数据进行增删改查(必须要先进入仓库再操作数据

#cd /root/project

# echo "init date" > init.txt #创建文件

# mkdir demo #创建目录

# cp /etc/hosts demo

4)查看仓库中数据的状态

#git status

5)将工作区的修改提交到暂存区

#git add . # . 表示当前目录

6)将暂存区修改提交到本地仓库(当前目录)

#git commit -m "注释,可以为任何字符"

[master（根提交） cdf66bc] wnejian

//系统版本

7)将本地仓库中的数据推送到远程服务器(web2将数据推送到web1)

#git config --global push.default simple #

# git push //设置 push.default

可以在web1上clone project验证文件以上传到服务器

web1#git clone root@192.168.2.100:/var/git/project

#ls project

8)在web1的project上添加文件 在clone的project文件夹内

#echo "AAA" > A.txt

#echo "BBBB" > B.txt

#git add . //提交到暂存区

#git config --global user.email "you@example.com"

#git config --global user.name "Your Name"

#git commit -m "A B" //提交到本地仓库

#git config --global push.default simple

#git push //提交到源服务器

9)将服务器上的数据更新到本地（web1的数据更新到web2）

web2]#git pull

10)查看日志

#git log #查看全日志

#git log --pretty=oneline #简化版日志

#git log --oneline #更简化版

#git reflog

11)Windows需要安装git和tortoiseGit[将命令行转化为图形菜单].

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例二:head指针

HEAD指针是一个可以在任何分支和版本移动的指针，通过移动指针我们可以将数据还原至任何版本.每做一次提交操作都会导致git更新一个版本，HEAD指针也跟着自动移动。

步骤一:HEAD指针基本操作:

1)准备工作 (多对数据仓库进行修改、提交操作，以产生多个版本)

分3次追加信息[first second third]至new.txt

分3次从定向信息 [123 456 789] 至num.txt

eg:echo "first" >> new.txt

git add .

git commit -m "first" //三次设置完push到源库

2)#git reflog #查看版本信息

f18aae0 HEAD@{5}: commit: first

e45e187 HEAD@{6}: commit: A.txt

e82dcb9 HEAD@{7}: pull: Fast-forward

#git reset --hard f18aae0 #head移动到first版本

f18aae0 HEAD@{0}: reset: moving to f18aae0

#git reset --hard f18aae0 #回到最新版本

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例三:git分支

Git支持按功能模块、时间、版本等标准创建分支，分支可以让开发分多条主线同时进行，每条主线互不影响.

MASTER分支（MASTER是主分支，是代码的核心）。

DEVELOP分支（DEVELOP最新开发成果的分支）。

RELEASE分支（为发布新产品设置的分支）。

HOTFIX分支（为了修复软件BUG缺陷的分支）。

FEATURE分支（为开发新功能设置的分支）。

步骤一:查看并创建分支

1)查看当前分支

#git status #查看当前分支

#git branch -v #查看所有分支

2)创建分支

#git branch hotfix

#git branch feature #创建分支

#git branch -v #查看所有分支

步骤二:切换与合并分支

1)切换分支

#git checkout hotfix #切换分支

#git branch -v

2)在新分支上继续进行数据操作

#echo "fix a bug" >> new.txt

#git add .

#git commit -m "fix bug"

3)将hotfix修改的数据合并到master分支

注意，合并前必须要先切换到master分支，然后再执行merge命令

#git checkout master

#cat new.txt

#git merge hotfix #合并hotfix分支

#cat new.txt

4)将所有本地修改提交远程服务器

#git push

步骤三:解决版本分支的冲突问题

1)在不同分支中修改相同文件的相同行数据，模拟数据冲突。

#git checkout hotfix

#echo "AAA" > a.txt

#git add .

#git commit -m "add a.txt by hotfix"

#git checkout master

#echo "BBB" > a.txt

#git add .

#git commit -m "add a.txt by master"

#git merge hotfix

2)查看有冲突的文件内容，修改文件为最终版本的数据，解决冲突。

#cat a.txt

<<<<<<< HEAD #世界线的碰撞,迸溅出盛夏的花火

BBB #若版本一个新 一个旧就已新的为准

=======

AAA

>>>>>>> hotfix

#vim a.txt #修改文件,为最终需要的数据,解决冲突

BBB

#git add .

#git commit -m "resolved"

总结:分支指针与HEAD指针的关系。

创建分支的本质是在当前提交上创建一个可以移动的指针

如何判断当前分支呢？答案是根据HEAD这个特殊指针

案例四:

\*创建SSH协议服务器

\*创建Git协议服务器 #额外起服务,起端口

\*创建HTTP协议服务器 #浏览器上访问,只能下载,不能上传,修改

\*前面192.168.2.100上的操作 走的是SSH服务

#yum -y install git

#git init /var/git/project --bare #创建共享

步骤一:SSH协议服务器

1)创建基于密码验证的SSH协议服务器（web1主机操作）。

git init --bare /var/git/project

2)客户端访问的方式（web2主机操作）。

#git clone root@192.168.2.100:/var/git/project

# rm -rf project

3)客户端生成SSH密钥，实现免密码登陆git服务器（web2主机操作）

#ssh-keygen -f /root/.ssh/id\_rsa -N ''

//生成文件位置 秘钥密码为空

#ssh-copy-id 192.168.2.100

//将公钥传给web1

#git clone root@192.168.2.100:/var/git/project

//无需密码验证

步骤二:git协议服务器(只读操作的服务器)

1)安装git-daemon软件包(web1主机操作)

#yum -y install git-daemon

2)创建版本库(web1)

#git init --bare /var/git/project

3)修改配置文件,启动git服务

#vim /usr/lib/systemd/system/git@.service

修改前内容如下：

ExecStart=-/usr/libexec/git-core/git-daemon --base-path=/var/lib/git

--export-all --user-path=public\_git --syslog --inetd –verbose

修改后内容如下：

ExecStart=-/usr/libexec/git-core/git-daemon --base-path=/var/git #修改创建目录

--export-all --user-path=public\_git --syslog --inetd –verbose

#systemctl start git.socket #起服务

web2:

#git clone git://192.168.2.100/project

#下载project仓库

步骤三:HTTP协议服务器(只读操作服务器)

1)安装gitweb httpd软件包(web1)

#yum -y install httpd gitweb

2)修改gitweb配置文件,设置仓库根目录(web1)

#vim +11 /etc/gitweb.conf //打开文件光标移至11行

添加 $projectroot = "/var/git" //上面有注释

3)创建版本仓库(web1)

# git init --bare /var/git/project

4)启动httpd服务器

# systemctl start httpd

5)客户端访问(web2)

firefox 192.168.2.100/git/

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

昨日回顾

文件需要经常修改(版本)

版本控制软件(cvs,svn,git)

yum -y install git

mkdir /var/git

git init /var/git/pro1 --bare

git init /var/git/pro2 --bare

客户端 web1

git clone root@192.168.2.100:/var/git/pro1

cd project

vim a.txt //修改文件

git add . //git rm 文件/目录

git commit -m "注释" //更新到自己电脑上的库

git push //推到源仓库

git pull //下载源仓库内容

web2

git clone root@192.168.2.100:/var/git/pro1

可以针对不同用户设置仓库(目录)权限

HEAD 指针

git reset --hard 版本号

分支

master------------------------>

测试test------>

git branch 分支名 //创建分支

git branch -v //查看分支

git checkout 分支 //切换分支

git checkout master

git merge 分支 //合并分支

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例五:制作NGINX的RPM包

RPM是一个压缩包

先在自己的电脑上源码安装一遍nginx

命令把/usr/local/nginx/打包压缩

nginx-1.12.rpm

步骤一:安装rpm-build软件

1)安装rpm-build软件包

#yum -y install rpm-build

2)生成rpmbuild目录结构

# rpmbuild -ba nginx.spec //会报错，没有文件或目录

# ls /root/rpmbuild //自动生成的目录结构

BUILD BUILDROOT RPMS SOURCES SPECS SRPMS

RPM包 源码包 配置文件

3)准备工作,将nginx源码复制到SOURCES目录

# cp /root/lnmp\_soft/nginx-1.12.2.tar.gz /root/rpmbuild/SOURCES/

将源码拷贝到SOURCES下

4)创建并修改SPEC配置文件

GPL协议: 软件可以随便复制 可以随便修改 可以随便重新发布 强制继承性

copy modify release

#cd /root/rpmbuild/SPECS/

#vim nginx.spec

Name:nginx //源码包软件名称 必须正确

Version:1.12.2 //源码包软件的版本号 必须正确

Release:1 //制作的RPM包版本号

Summary:Nginx is a web server software. //RPM软件概述

#Group: //软件包属于什么yum组, 注释掉

License:GPL //软件遵循协议

URL:www.test.com //网址

Source0:nginx-1.12.2.tar.gz //源码包文件的全称 必须正确

#BuildRequires: //依赖关系

#Requires: //依赖关系

%description //软件详细信息

nginx [engine x] is an HTTP and reverse proxy server.

%post #安装后脚本,RPM安装后执行

useradd nginx

%prep

%setup -q //自动解压源码包,并cd进入目录

%build

./configure --user=nginx --group=nginx --with-http\_ssl\_module --with-stream --with-http\_stub\_status\_module

# ./configure 设置

make %{?\_smp\_mflags}

%install

make install DESTDIR=%{buildroot}

%files

%doc //对那些文件打包

/usr/local/nginx/\*

3)

#rpmbuild -ba nginx.spec

#ls RPMS/x86 64/

nginx-1.12.2-1.x86\_64.rpm nginx-debuginfo-1.12.2-1.x86\_64.rpm

#rpm -ivh /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-1.x86\_64.rpm

//安装配好的rpm包

#cd /SRPMS //该目录存放了源码包和spec文件的RPM包

#rpm -qpl nginx-1.12.2-10.src.rpm

nginx-1.12.2.tar.gz //源码包

nginx.spec //spec文件

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例1:配置GRE VPN

VPN Virtual Private Network (虚拟专用网络)

--在共用网络上建立专用私有网络,进行加密通讯

--多用于为集团公司的各个子公司建立连接

--连接完成后,各个地区的子公司可以想局域网一样进行连接

--在企业网络中有广泛应用

--偶尔可以用于翻墙

--目前主流的vpn技术(GRE,PPTP,L2TP+IPSec,SSL)

难度:GRE < PPTP < L2TP+IPSec

安全:GRE < PPTP < L2TP+IPSec

不加密 <部分加密<全加密

GRE Window不支持, 路由器等支持

PPTP L2TP+IPSec Window支持

准备: client eth3(201.1.2.10/24)

proxy eth0(192.168.4.5/24)

eth3(201.1.2.5/24)

步骤一:启用GRE模块(client和proxy都需要操作)

1)查看计算机当前是否加载gre模块

#lsmod | grep ip\_gre #确定是否加载了gre模块

2)加载模块ip\_gre

#modprobe ip\_gre #加载模块

#modinfo gre #查看模块信息

#rmmod 模块名 #禁用模块

步骤二:clien主机创建vpn隧道

1)创建隧道

# ip tunnel add tun0 mode gre remote 201.1.2.5 local 201.1.2.10

//ip tunnel add创建隧道（隧道名称为tun0），ip tunnel help可以查看帮助

//mode设置隧道使用gre模式

//local后面跟本机的IP地址，remote后面是与其他主机建立隧道的对方IP地址

2)启用该隧道

#ip link show //查看本机网卡信息

#ip link set tun0 up //tun0 up

#ip link show

3)为VPN配置隧道ip地址

#ip addr add 10.10.10.10/24 peer 10.10.10.5/24 dev tun0

#ip addr show //简写ip a s 查看IP地址

//为隧道tun0设置本地IP地址（10.10.10.10.10/24）

//隧道对面的主机IP的隧道IP为10.10.10.5/24

4)关闭防火墙

# firewall-cmd --set-default-zone=trusted

步骤三：

1)Proxy主机创建VPN隧道

#ip tunnel add tun0 mode gre remote 201.1.2.10 local 201.1.2.5 //创建隧道

# ip link set tun0 up

# ip addr add 10.10.10.5/24 peer 10.10.10.10/24 dev tun0 //配置隧道ip地址

2)开启路由转发,关闭反火墙

#echo "1" > /proc/sys/net/ipv4/ip\_forward

//此为开启linux转发功能

# firewall-cmd --set-default-zone=trusted

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例2:创建PPTP VPN

注:proxy做服务器,win2008作客户端

先要把win2008虚拟机桥网卡接到 public2.

设置配置: windows主机: 网卡桥接public2(201.1.2.20/24)

proxy eth0(192.168.4.5/24)

eth3(201.1.2.5/24)

测试windows与proxy通讯.

步骤一:部署VPN服务器

1)安装软件包

#cd /root/lnmp\_soft/vpn

#yum -y install pptpd //安装pptp软件

#rpm -qc pptp //查看pptp的配置文件,有三个

#tailf /var/log/message //该软件日志

2)修改配置文件

#vim /etc/pptp

... ...

localip 201.1.2.5 //服务器本地ip

remoteip 192.168.3.1-50 //分配给客户端的ip池

#vim /etc/ppp/option.pptpd

require-mppe-128 //使用mppe加密数据使用128位算法加密

ms-dns 8.8.8.8 //DNS服务器

name pptpd //vpn名字

#vim /etc/ppp/chap-secrets

hana pptpd 123456 \*

//用户名 服务器名字 密码 客户端ip

#echo "1" > /proc/sys/net/ipv4/ip\_forward

//开启路由转发 软路由 变身成路由器 只能 > 修改

\*需要服务器打开NAT功能才可能访问墙外网.

3)启动服务

# systemctl start pptpd

# systemctl enable pptpd

# firewall-cmd --set-default-zone=trusted

4)翻墙设置(非必须操作)

# iptables -t nat -A POSTROUTING -s 192.168.3.0/24

-j SNAT --to-source 201.1.2.5

//将私网ip转成公网ip

步骤二:客户端设置

windows2008

网络--->设置新的连接或网络---->连接到工作区-->使用我的internet

---->我将稍后连接---->internet地址-,目标名称 --->用户名,密码.

更改适配器设置---->pptpd---->输入用户名密码登录

#ping 192.168.4.1 //proxy可以ping通

案例三: XL2TP+lpsec VPN部署

windows主机: 网卡桥接public2(201.1.2.20/24)

client eth0(192.168.4.10/24)

eth3(201.1.2.10/24)

步骤一:

1)安装软件包

#yum -y install libreswan

2)新建IPSec秘钥验证配置文件

#cat /etc/ipsec.conf //仅查看该主配置文件

... ...

include /etc/ipsec.d/\*.conf //加载该目录下的所有配置文件

#vim /etc/ipsec.d/myipsec.conf //直接复制了

#cp /root/lnmp\_soft/vpn/myipsec.conf /etc/ipsec.d

//myipsec.conf为提前写好的配置文件.

conn IDC-PSK-NAT

rightsubnet=vhost:%priv //允许建立的VPN虚拟网络

also=IDC-PSK-noNAT

conn IDC-PSK-noNAT

authby=secret //加密认证

ike=3des-sha1;modp1024 //算法

phase2alg=aes256-sha1;modp2048 //算法

pfs=no

auto=add

keyingtries=3

rekey=no

ikelifetime=8h

keylife=3h

type=transport

left=201.1.2.10 //重要，服务器本机的外网IP

leftprotoport=17/1701

right=%any //允许任何客户端连接

rightprotoport=17/%any

3)创建IPSec预定义共享秘钥

#cat /etc/ipsec.secrets //仅查看

include /etc/ipsec.d/\*.secrets

#vim /etc/ipsec.d/mypass.secrets //新建该文件

201.1.2.10 %any: PSK "randpass"

//服务器ip 所有人可连 预共享秘钥(pre share key) 秘钥密码

4)启动IPSec服务

#systemctl start ipsec

#netstat -ntulp |grep pluto //查看状态,端口4500,500

步骤二:部署XL2TP服务

1)安装软件包

# cd /root/lnmp\_soft/vpn/ //包在lnmp\_soft中

# yum -y install xl2tpd-1.3.8-2.el7.x86\_64.rpm

2)修改xl2tp配置文件(修改3个配置文件的内容)

# vim /etc/xl2tpd/xl2tpd.conf //主配置文件

... ...

ip range = 192.168.3.128-192.168.3.254

local ip = 201.1.2.10 //在最底下修改ip池,

#vim /etc/ppp/options.xl2tpd //认证配置

require-mschap-v2 //添加一行，强制要求认证(默认注释掉,顶格写)

#crtscts //注释或删除该行 新版本不识别

#lock //注释或删除该行 新版本不识别

#vim /etc/ppp/chap-secrets //修改密码文件

hana \* 123456 \*

//账户名称 服务器标记 密码 客户端IP

3)启动服务

# systemctl start xl2tpd

# netstat -auntlp | grep xl2tpd

4)设置路由转发,防火墙

# echo "1" > /proc/sys/net/ipv4/ip\_forward

#echo "net.ipv4.ip\_forward=1" >> /etc/sysctl.conf

//转发功能永久生效 主配置文件 /etc/sysctl.conf

# firewall-cmd --set-default-zone=trusted

5)翻墙设置

# iptables -t nat -A POSTROUTING -s 192.168.3.0/24 \

> -j SNAT --to-source 201.1.2.10

步骤二:客户端设置

网络--->设置新的连接或网络---->连接到工作区-->使用我的internet

---->我将稍后连接---->internet地址-,目标名称 --->用户名,密码.

更改适配器设置---->右击网卡属性---->安全--->vpn类型--->高级属性预共享randpass

3)设置Windows注册表

\*单击"开始"，单击"运行"，键入"regedit"，然后单击"确定"

\*找到下面的注册表子项，然后单击它：

\*HKEY\_LOCAL\_MACHINE\ System\CurrentControlSet\Services\Rasman\Parameters

\*在"编辑"菜单上，单击"新建"->"DWORD值"

\*在"名称"框中，键入"ProhibitIpSec"

\*在"数值数据"框中，键入"1"，然后单击"确定"

\*退出注册表编辑器，然后重新启动计算机

4)ping 201.1.2.10

+++++++++++++++++++++++++++++++++++

day08

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

VPN: GRE,PPTP,xl2tp+ipsec, ssl,ssh....

GRE:

第一台:

linux内核模块:

modprobe 模块名称

lsmod //列出模块

rmmod

modinfo 名称 //查询模块信息

ip tunnel add 创建隧道 tun0

ip link set tun0 up //

ip addr add 网络IP地址

第二台也重新设置一遍,ip反一下.

pptpd:

3个配置文件

1.VPN的私有网络

localip

remoteip

2.ms-dns 8.8.8.8 //若要域名则设置

3.用户名密码:

//账户名称 服务器标记 密码 客户端IP

xl2tp+ipsec

/etc/ipsed.d/xxx.conf //加密算法

/etc/ipsec.d/xxx.secrets //PSK预共享秘钥

systemctl start ipsec

1.VPN的私有网络

localip

remoteip

2.ms-dns 8.8.8.8 //若要域名则设置

3.用户名密码:

//账户名称 服务器标记 密码 客户端IP

VPN--->网络

路由器 转发(NAT)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NTP时间同步:搭建一个NTP服务器,为整个网络环境中的所有主机提供时间校准服务

准备: client eth0(192.168.4.10/24)

proxy: eth0(192.168.4.5/24)

eth1(192.168.2.5/24)

NTP(闰秒) 59-->60--->00 不定时,为世界续一秒

步骤一:

1).安装软件包:

#yum -y install chrony

#rpm -qc chrony //查看配置文件列表

/etc/chrony.conf

/etc/chrony.keys

2)修改配置文件

#vim /etc/charony.conf

... ...

server 0.centos.pool.ntp.org iburst

//server用户客户端指向上层NTP服务器

allow 192.168.4.0/24 //允许那个IP或网络访问NTP

#deny 192.168.4.1 //拒绝那个IP或网络访问NTP

local stratum 10 //设置NTP服务器层数量

... ...

3)启动NTP服务

#systemctl restart chronyd

#systemctl enable chronyd

4)设置防火墙

# firewall-cmd --set-default-zone=trusted

步骤二:配置客户端

1)安装软件包

#yum -y install chrony

2)修改配置文件

#vim /etc/chrony.conf

server 192.168.4.5 iburst

3)将客户端时间修改为错误时间

#date -s "3:00" //时间改为3点

#date

4)重启chrony与服务器同步时间

#systemctl restart chronyd

5)确认时间是否已经同步

#date

++++++++++++++++++++++++++++++++++++++++++++++++++

CLUSTER集群

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

集群:将很多服务器集中起来一起,提供同一种功能,

任务调度是集群系统中的核心技术

集群目的:

提高性能

降低成本

提高可扩展性

增强可靠性

集群分类:

高性能计算集群HPC

通过以集群开发的并行应用程序,解决复杂的科学问题

负载均衡(LB)集群

客户端负载在计算机集群中尽可能平均分摊

高可用(HA)集群

避免单点 故障,当一个系统发生故障时,可以快速迁移

集群常用软件:nginx,LVS,haproxy,F5,big-ip

\*LVS可以实现高可用的,可伸缩的Web,Mail,Cache和Media等网络服务

\*最终目标是利用Linux操作系统和LVS集群软件实现一个高可用,高性能,低成本的服务应用集群

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

集群组成:

前端:负载均衡层

由一台或多台负载调度器构成

中间:服务器集群组

由一组实际运行应用服务的服务器

低端:数据共享存储层

提供共享存储空间的存储区域

web

LVS web mysql

web NFS/common

mount ip:/common /usr/local/nginx/html

mount ip:/common /usr/local/nginx/html

在后端存储修改网页,就能达到web更新效果

LVS术语:

Director Server:调度服务器有VIP,RIP

Real Server:真是服务器

VIP:虚拟IP地址 公布给用户访问的虚拟IP地址

RIP:真是IP地址 集群节点上使用的IP地址

DIP:调度器连接节点服务器的IP地址

LVS工作模式:

NAT模式:从哪进,从哪出,数据出去时会堵塞在LVS服务器上

TUN模式:用户访问北京的调度器,转发给上海的服务器.VPN

DR模式: 数据直接从web返回给用户,避免了LVS服务器上的数据堵塞

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例1:熟悉命令ipvsadm用法:

ipvsadm -A //创建集群

ipvsadm -E //修改集群

ipvsadm -D //删除集群

ipvsadm -C //删除所有集群

ipvsadm -L //查看LVS规则表

ipvsadm -a [-g|i|m] //添加真实服务器

-g(DR模式)-i(隧道模式)-m(NAT模式)

ipvsadm -e //修改真实服务器

ipvsadm -d //删除真实服务器

ipcsadm -s [rr|wrr|lc|wlc...] //指定算法

调度算法: source hash[sh]

roundrobin[rr] 轮询 不支持权重

weight roundrobin[wrr]加权轮询

least connect[lc]最少连接算法

weighted leadt connection[wlc] 加权最少连接

#yum -y install ipvsadm

#ipvsadm -A -t 192.168.4.5:80 -s wrr //创建集群

-A 创建集群 -t tcp -s 算法

#ipvsadm -Ln //-L查看 -n数字显示

#ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100:80 -w 2

//-a添加真实服务器 -r真实服务器地址 -w权重

#ipvsadm -A -t 192.168.4.5:3306 -s wlc

#ipvsadm -a -t 192.168.4.5:3306 -t 192.168.2.200 -w 2

#ipvsadm -E -t 192.168.4.5:80 -s rr

#ipvsadm -e -t 192.168.4.5:3306 -s rr

#ipvsadm -D -t 192.168.4.5:80

#ipvsadm -d -t 192.168.4.5:3306 -r 192.168.2.100:80

#ipvsadm -C

#ipvsadm -e -t 192.168.4.5:80 -r 192.168.2.100:80 -m NAT模式

#ipvsadm -save -n > /etc/sysconfig/ipvsadm //永久保存

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

---web1 (网关,网站,ip)

client------LVS----

---web2 (网关,网站,ip)

LVS工作原理与nginx不同

nginx代理服务器

LVS(NAT) 等同于 路由器(ip\_forward=1),web1,web2网关

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例2:部署LVS-NAT集群

主机配置: client:eth0:192.168.4.10/24

proxy: eth0:192.168.4.5/24

eth1:192.168.2.5/24

web1: eth1:192.168.2.100/24

网关:192.168.2.5

web2: eth1:192.168.2.200/24

网关:192.168.2.5

web1:

#nmcli connection modify eth1 ipv4.method manual ipv4.addresses 192.168.2.200/24 ipv4.gateway 192.168.2.5 connection.autoconnect yes

#nmcli connection up eth1

web2: 俺也一样. 或

#vim /etc/sysconfig/network-script/ifcfg-eth1

GATEWAY=192.168.2.5

#nmcli connection up eth1

步骤一:配置基础环境

1)设置web服务器(web1为例)

#systemctl start httpd

#curl 127.0.0.1

2)关闭防火墙与SELinux

#systemctl stop firewall

#setenforce 0

步骤二:部署LVS-NAT模式调度器 proxy做调度器

1)确认调度器的路由转发功能

#echo 1 > /proc/sys/net/ipv4/ip\_forward

#echo "net.ipv4.ip\_forward=1" >> /etc/sysctl.conf

//转发功能永久生效 主配置文件 /etc/sysctl.conf

2)创建集群服务器

#ipvsadm -A -t 192.168.4.5:80 -s wrr

//c创建集群

3)添加真实服务器(NAT)

#ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100 -m

#ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.200 -m

4)访问调度器测试

#curl 192.168.4.5

#curl 192.168.4.5

添加ssh的调度服务器,将proxy的2345口转到web1的22端口.

proxy:

#ipvsadm -A -t 192.168.4.5:2345 -s wrr

#ipvsadm -a -t 192.168.4.5:2345 -r 192.168.2.100:22 -m

#ipvsadm -a -t 192.168.4.5:2345 -r 192.168.2.200:22 -m

client:

#ssh 192.168.4.5 -p 2345 //多次访问

web1----web2 权重轮询

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例3:部署LVS-DR集群

网络基本原理:访问谁,谁给回应

说明:

CIP是客户端的IP地址；

VIP是对客户端提供服务的IP地址；

RIP是后端服务器的真实IP地址；

DIP是调度器与后端服务器通信的IP地址（VIP必须配置在虚拟接口）。

配置:client: eth0: 192.168.4.10/24

proxy: eth0: 192.168.4.5/24

eth0:0 192.168.4.15/24

web1: eth0 192.168.4.100/24

lo:0 192.168.4.15/32

web2 eth0 192.168.4.200/24

lo:0 192.168.4.15/32

步骤一:配置实验网络环境

1)设置proxy代理服务器的VIP和DIP

注意：为了防止冲突，VIP必须要配置在网卡的虚拟接口！！！

#cd /etc/sysconfig/network-scripts/

#cp ifcfg-eth0 ifcfg-eth0:0

#vim ifcfg-eth0 ifcfg-eth0:0

TYPE=Ethernet //以太接口

BOOTPROTO=none //ip怎么获得,none手动配IP,dhcp获取

NAME=eth0:0 //网卡名

DEVICE=eth0:0

ONBOOT=yes //开机时网卡激活

IPADDR=192.168.4.15 //IP地址

PREFIX=24 //子网掩码

#systemctl restart nerwork //若报错重启机器

#ifconfig //查看eth0:0网卡信息

2)设置web1,web2服务器网络参数

web1:

# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

# nmcli connection up eth0

也可以用]setip配置.

接下来给web1配置VIP地址。 192.168.4.15/32用户给客户发数据时伪装的ip

注意：这里的子网掩码必须是32（也就是全255），网络地址与IP地址一样，广播地址与IP地址也一样。

#cd /etc/sysconfig/network-scripts/

#cp ifcfg-lo ifcfg-lo:0

#vim ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255 //掩码必须32位,32位网络位

NETWORK=192.168.4.15 //网络地址自己

BROADCAST=192.168.4.15 //广播地址自己

ONBOOT=yes

NAME=lo:0

修改内核参数,防止ip冲突:

#vim /etc/sysctl.conf

#手动写入如下4行内容

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.lo.arp\_ignore = 1

net.ipv4.conf.lo.arp\_announce = 2

net.ipv4.conf.all.arp\_announce = 2

#当有arp广播问谁是192.168.4.15时，本机忽略该ARP广播，不做任何回应

#本机不要向外宣告自己的lo回环地址是192.168.4.15.

#sysctl -p //使修改内容生效

重启网络服务,设置防火墙与SELinux

#systemctl restart NetworkManager

#systemctl restart network

#ifconfig

#systemctl stop firewall

#setenforce 0

web2:俺也一样. web2同样设置.

步骤二:配置后端web服务器

步骤三:proxy调度器安装软件并部署LVS-DR模式调度器

1）安装软件

2)清理之前实验的规则,创建新的集群服务器规则

#ipvsadm -C

#ipvsadm -A -t 192.168.4.15:80 -s rr

#ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.100:80 -g

#ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.200:80 -g

#ipvsadm-save -n > /etc/sysconfig/ipvsadm //保存规则

步骤四:客户端测试 client

#curl 192.168.4.15 //反复连接

性能:nginx<LVS(内核)

功能:nginx>LVS

默认LVS不带健康检查功能，需要自己手动编写动态检测脚本，实现该功能：(参考脚本如下，仅供参考)

[root@proxy ~]# vim check.sh

#!/bin/bash

VIP=192.168.4.15:80

RIP1=192.168.4.100

RIP2=192.168.4.200

while :

do

for IP in $RIP1 $RIP2

do

curl -s http://$IP &>/dev/vnull

if [ $? -eq 0 ];then

ipvsadm -Ln |grep -q $IP || ipvsadm -a -t $VIP -r $IP

else

ipvsadm -Ln |grep -q $IP && ipvsadm -d -t $VIP -r $IP

fi

done

sleep 1

done

+++++++++++++++++++++++++++++++++++++++++++++++++++++

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

昨日回顾

nginx nginx是代理,帮client去上网,所以访问web1的是proxy,web1不需要网关

upstream a {

server web1 weight 2;

server web2;

}

LVS:

yum -y install ipvsadm

ipvsadm -A -t $VIP:80 -s rr

ipvsadm -a -t $VIP:80 -r $RIP

默认DR模式,

模式:NAT DR tunel fullnat(ospf)

ipvsadm -Ln //查看

1.NAT模式 -------------web1(client在访问

client ----lvs

------web2(网关)

lvs(ip\_forward=0)

NAT:

ipvsadm -A -t 192.168.4.5:80

ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100:80

2.LVS-DR模式

web1()4.100 lo:0 4.15/32(伪装)

client---------lvs

web2()4.200 lo:0 4.15/32(唯一主机位)

lvs:

cp ifcfg-eth0 ifcfg-eth0:0 //4.15放在虚拟接口eth0:0上

4.15/32 唯一主机位,独立的网

arp\_ignore = 1 //0 恢复任何本地ip,1只回本地网卡的一个ip,进站网卡

8不回任何消息

arp\_annoubce = 2 //0.所有ip,接口都告诉别人.1尝试避免冲突

2.每次只使用唯一的本地ip

yum -y install kernel-doc

vim /usr/share/doc/kernel-doc-3.10.0/Documentation/networking/ip-sysctl.txt //查看arp\_ignore解释

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

git init /var/git/abc //仓库无法共享给服务器,即单机

git init --bare /var/git/abc //可以共享给服务器

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Keepalived 高可用服务器

Keepalived:1.实现路由功能的VRRP(虚拟路由热备协议)协议.

两台路由器2.252 2.253---->虚拟网关2.254,路由器设置优先级master,

第一个功能:VRRP实现vip.

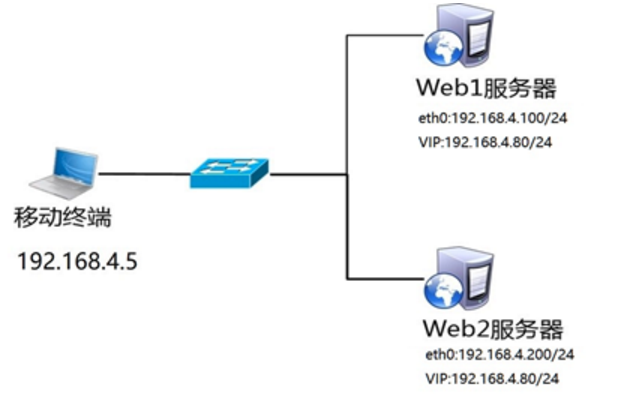
第二个功能:自动配置lvs,健康检查

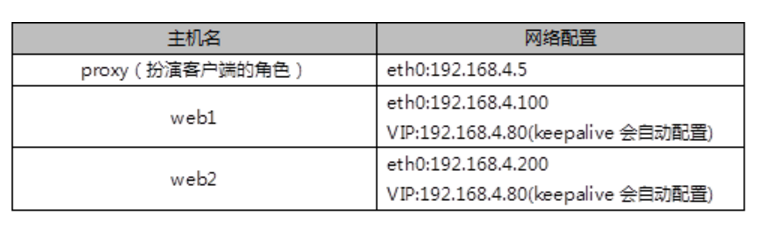
lvs|nginx(keepalivied)100,4.15 主服务器

用户

lvs|nginx(keepalivied)50,4.15

第二个功能:自动配置lvs,健康检查





**图一**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例1:Keepalive高可用服务器, ip设置如图1

准备: proxy eth0:192.168.4.5

web1 eth0:192.168.4.100

VIP:192.168.4.80(keepalive会自动分配)

web2 eth0:192.168.4.200

VIP:192.168.4.80(keepalive会自动分配)

步骤一:

1)web1,web2安装keepalive

web1~]#yum -y install keepalive

web2~]#yum -y install keepalive

2)修改web1服务器keepalive配置文件

vim /etc/keepalived/keepalived.conf

\*星号为需要修改的参数

global\_defs {

notification\_email {

admin@tarena.com.cn //设置报警收件人邮箱

}

notification\_email\_from ka@localhost //设置发件人

smtp\_server 127.0.0.1 //定义邮件服务器

smtp\_connect\_timeout 30

\* router\_id web1 //设置路由ID号（实验需要修改）

}

vrrp\_instance VI\_1 {

state MASTER //主服务器为MASTER（备服务器需要修改为BACKUP）

interface eth0 //VIP设置在eth0上

virtual\_router\_id 51 //主备服务器VRID号必须一致

priority 100 //服务器优先级,优先级高优先获取VIP（实验需要修改）

advert\_int 1

authentication {

auth\_type pass //安全

auth\_pass 1111 //主备服务器密码必须一致

}

\* virtual\_ipaddress { //谁是主服务器谁获得该VIP（实验需要修改）

192.168.4.80

}

}

3)启动服务:

#systemctl start keepalived

#systemctl start keepalived

#ip a s eth0 //查看eth0是否 有192.168.4.80ip

4)配置防火墙和SELinux

启动keepalive会自动添加一个drop防火墙规则,需要清空

#iptables -F //清空防火墙规则

#setenforce 0

web2修改配置:

#scp /etc/keepalived/keepalived.conf 192.168.4.200:/etc/keepalived/

#vim /etc/keepalived/keepalived.conf

router\_id web2

state BACKUP

priority 50

#systemcrl start keepalived

//关闭web1的keepalived,web2的eth0网卡上会有4.80ip.

//注意每次更改了keepalive配置文件或重启需关闭防火墙.

//web1开启keepalive后,4.80ip回到web1的eth0上.

5)验证

client]#curl 192.168.4.15

keepalive 日志 /var/log/message

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

extend:两台nginx代理高可用

nginx代理+keepalived 100 vip 4.80

client--VIP 4.80

nginx代理+keepalived 50 vip 4.80

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Keepalive 自动配置LVS,健康检查 (支持对端口检查,md5检查)

#ping -c3 192.168.4.100 &> /dev/null //ping 3个包给100

健康检查:

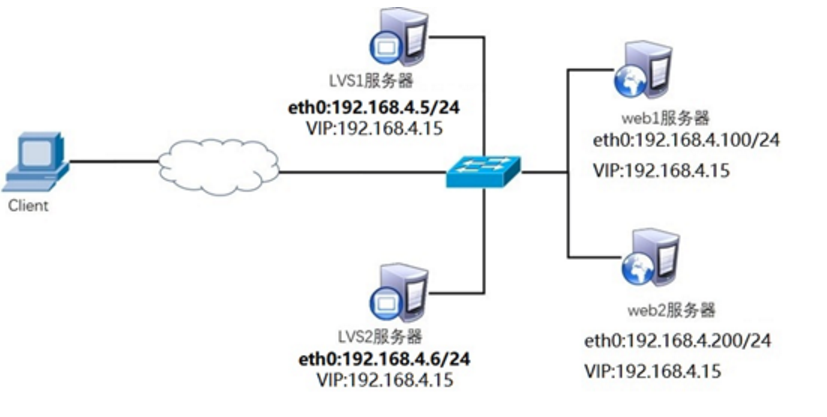
for i in 100 200

do

a=`curl http://192.168.4.$i | md5sum`//将网页的md5值赋值给a

if [ $a == "md5值" ]

done



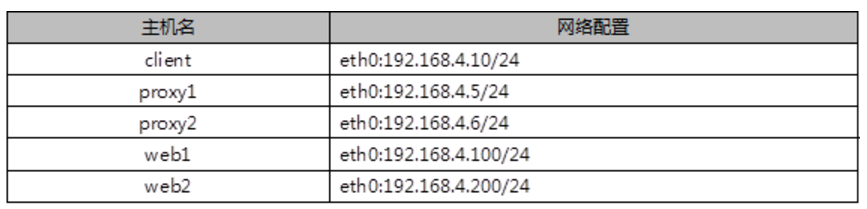


图2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

方案二:Keepalived+LVS服务器 ip配置如图2

前置条件: client eth0 192.168.4.10/24

proxy1 eth0: 192.168.4.5/24

proxy2 eth0: 192.168.4.6/24

web1 eth0: 192.168.4.100/24

web2 eth0: 192.168.4.200/24

此处延续昨日DR集群:

web1 web2 VIP 保留

proxy1,proxy2的VIP删除,即删除ifcfg-eth0:0文件,并重启

#rm -f /etc/sysconfig/network-scripts/ifcfg-eth0:0

#systemctl restart Network

#systemctl restart network

1)给proxy1,proxy2安装keepalived

proxy1]yum -y install keepalived

#systemctl enable keepalived

#yum -y install ipvsadm

#ipvsadm -C //清空规则

proxy2]#yum -y install keepalived

proxy2]#systemctl enable keepalived

proxy2]#yum -y install ipvsadm

proxy2]#ipvsadm -C

2)修改proxy1,proxy2的keepalive 配置文件

vim /etc/keepalived/keepalived.conf

12行 router\_id lvs1

29行 virtual\_ipaddress {

30行 192.168.4.15 }

virtual\_server 192.168.4.15 80 {

delay\_loop 6

lb\_algo rr //算法 rr

lb\_kind NAT //DR模式

#persistence\_timeout 50 //50秒之内始终访问

protocol TCP //协议tcp

real\_server 192.168.201.100 443 {

weight 1

SSL\_GET { //HTTP\_GET

url {

path / //访问根页面

digest 17c6084d3d40df800a536166e4a9ca29

}

connect\_timeout 3 //连接等待时间3秒

nb\_get\_retry 3 //重试3次

delay\_before\_retry 3 //每隔3秒做一次健康检查

}

}

real\_server 192.168.4.200 80 {

weight 1

HTTP\_GET {

url {

path /

digest a008d320d8fb97191584e72684967c19

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

} //上述为HTTP\_GET健康检查的格式.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

下面为TCP\_CHECK只检测端口的健康检查:

virtual\_server 192.168.4.15 80 {

//设置ipvsadm的VIP规则（实验需要修改）

delay\_loop 6

lb\_algo wrr //设置LVS调度算法为WRR

lb\_kind DR //设置LVS的模式为DR

#persistence\_timeout 50

#注意这样的作用是保持连接，开启后，客户端在一定时间内始终访问相同服务器

protocol TCP

real\_server 192.168.4.100 80 { //设置后端web服务器真实IP（实验需要修改）

weight 1 //设置权重为1

TCP\_CHECK {

//对后台real\_server 做健康检查

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.200 80 { //设置后端web服务器真实IP（实验需要修改）

weight 2 //设置权重为2

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

# systemctl start keepalived.service //重启服务,多重启一次

#ipvsadm -Ln //查看lvs集群

#ip a s eth0 //查看eth0ip

#iptables -F //清空防火墙规则 keepalived 配置改了,需要重新清防火墙

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

proxy2上:

proxy1]#scp /etc/keepalived/keepalived.conf root@192.168.4.6:/etc/keepalived/keepalived.conf

proxy2]#vim /etc/keepalived/keepalived.conf

....

router\_id lvs2

state BACKUP

priority 70

...

#systemctl restart keepalived.service //重启服务

# iptables -F //关防火墙

#ipvsadm -Ln //列出lvs集群

把proxy1关了,在client验证:

#curl 192.168.4.15

#curl 192.168.4.15

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例三: 配置HAProxy负载平衡集群 .与nginx调度一样

nginx分析:

优点:

工作在7层,可以针对http做分流策略

正则表达式比HAProxy强大 //基本,扩展,Posix,perl正则 \d \w

安装/配置/测试简单,通过日志可以解决多数问题

并发量可以达到几万次

nginx还可以作为Web服务器使用

缺点:

仅支持http.https.mail协议,应用面小

监控检查仅通过端口,无法使用url检查

LVS分析

优点

负载能力强,工作在第4层,对内存,cpu消耗低

配置性低,没有太多可配置性,减少认为错误

应用面广,几乎可以为所有应用提供负载均衡

缺点

不支持正则表达式,不能实现动静分离

如果网站架构庞大,LVS-DR配置比较繁琐

HAProxy分析:

优点

支持session,cookie功能

可以通过url进行健康检查

效率,负载均衡速度,高于nginx,低于LVS

HAProxy支持TCP,可以对mysql进行负载均衡

调度算法丰富

缺点:

正则弱于nginx

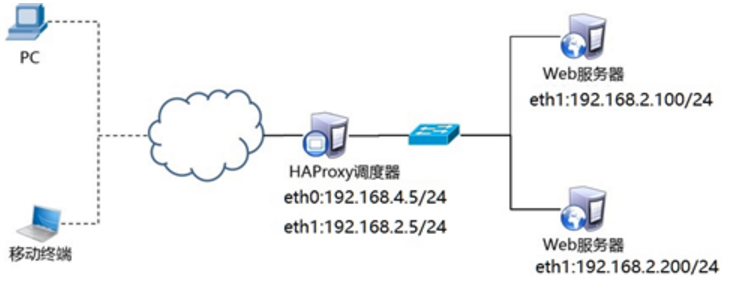
日志依赖于sysloqd,不支持Apache日志

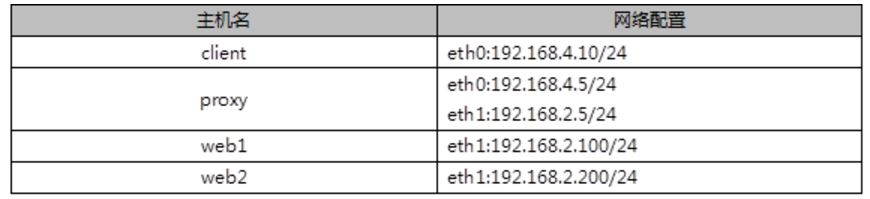
决定使用什么集群的因素:

1.企业现有的集群是什么,就用什么.

2.新的业务取决于你会什么.

3.企业愿不愿意花钱,F5 bigip





**图3**

前置条件: web1关闭多余的网卡与VIP，配置本地真实IP地址。

Web2关闭多余的网卡与VIP，配置本地真实IP地址

client eth0:192.168.4.10/24

proxy eth0:192.168.4.5/24

eth1:192.168.2.5/24

web1 eth1:192.168.2.100/24

web2 eth1:192.168.2.200/24

web1

#rm -rf /etc/sysconfig/network-scripts/ifcfg-lo:0

#systemctl stop Networkmanage

#systemctl restart network

#ifconfig //lo:0 删除了. eth1:192.168.2.100

web2

#rm -rf /etc/sysconfig/network-scripts/ifcfg-lo:0

#systemctl stop Networkmanage

#systemctl restart network

#ifconfig //lo:0 删除了. eth1:192.168.2.200

步骤二:部署HAProxy服务器

1)配置网络,安装软件

#yum -y install haproxy

2)修改配置文件

#vim /etc/haproxy/haproxy.cfg

global //全局设置

\*maxconn 4000 //整个软件总并发量4000

\*daemon //起来后把服务放入后台

defaults //默认设置

\*option redispatch ###serverid服务器挂掉后强制定向到其他健康服务器

\*maxconn 60000 ###一个集群最大连接数

listen stats 0.0.0.0:1080 #监听端口 监控页面配置

stats refresh 30s #统计页面自动刷新时间

stats uri /stats #统计页面url

stats realm Haproxy Manager #进入管理解面查看状态信息

stats auth admin:admin #统计页面用户名和密码设置

#stats hide-version #隐藏统计页面上HAProxy的版本信息

listen websrv-rewrite 0.0.0.0:80 //集群配置

balance roundrobin //轮询全称

server web1 192.168.2.100:80 check inter 2000 rise 2 fall 5

server web2 192.168.2.200:80 check inter 2000 rise 2 fall 5

//每隔2000ms做一次健康检查,失败5次删除,两次访问成功添加回集群

3)client上验证

#firefox 192.168.4.5 //测试调度器是否正常

#firefox 192.168.4.5:1080/stats //测试状态监控页面

++++++++++++++++++++++++++++++++++++++++++++++++++++

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

昨日回顾

DAS

NAS

SAN

分布式存储ceph

默认:

osd osd osd

mon mon mon

node1:

yum -y install ceph-deploy

mkdir ceph-cluster

ceph-deploy new node1 node2 node3

ceph-deploy mon create-initial //自动安装启动mon

ceph -s

3 mon

0 osds

vdb1 vdb2 chown 权限 将vdb隔成两个区

ceph-deploy disk zap node1:vdc node1:vdd //格式化node1.3

ceph-dploy osd create node1:vdc:/dev/vdb1 node1:vdd:/dev/vdb2 //node1-3自动格式化并挂载

Ceph -s

3 mons

6 osds

ceph osd lapools //查看存储池

rbd create image feature layering --size 10G //创建镜像

用户rbd map

ceph 存储数据默认切成4M大小

dd if=/dev/zero //创建1000M的数据

du -sh /etc/passwd //查看文件大小

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

镜像---快照(保护)----克隆镜像 操作均在node1上进行

步骤四:创建快照克隆

1)查看昨日快照 我的是snap2

#rbd snap ls image //查看image镜像

#rbd snap protect image --snap image-snap1 //保护快照

#rbd snap rm image --snap image-snap1 //提示快照被保护

# rbd clone image --snap image-snap1 image-clone --image-feature layering

//使用image的快照image-snap1克隆一个新的名称为image-clone镜像

#rbd list //查看镜像信息

2)查看克隆镜像与父镜像快照的关系

#rbd info image-clone //相当于做了image-snap1的快捷方式

parent: rbd/image@image-snap1

#rbd flatten image-clone //做image-snap1镜像文件的拷贝

# rbd info image-clone //查看克隆镜像的状态

[root@node1 ~]# rbd snap unprotect image --snap image-snap1 #取消快照保护

[root@node1 ~]# rbd snap rm image --snap image-snap1

//可以删除快照

步骤四: 其他操作

客户端撤销磁盘映射 client上操作

client~]#umount /mnt

client~]#rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

client~]#rbd unmap /dev/rbd0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NSD CLUSTER DAY04

虚拟机本质就是文件:磁盘镜像cow 和 XML描述文件

存放位置:/var/lib/libvirt/images/

/etc/libvirt/qemu/

存放位置:网盘ceph,192.168.4.11

真实主机---->XML(磁盘镜像--->ceph)

虚拟机的迁移 阿里云 华为云 腾讯云

虚拟机: 性能差80-90%,,

文件 3副本,

迁移

ceph 实现秒级别迁移

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例1:块存储应用案例:

Ceph创建块存储镜像

客户端安装部署ceph软件

客户端部署虚拟机

客户端创建secret

设置虚拟机配置文件，调用ceph存储

1)创建磁盘镜像 node1上操作

#rbd create vm1-image --image-feature layering --size 10G //创建镜像vm1-image 大小10G

#rbd list //列出镜像列表

#rbd info vm1-image //查看vm1-image镜像信息

2)ceph认证账户(仅查看) node1操作

#cat /etc/ceph/ceph.conf

# cat /etc/ceph/ceph.client.admin.keyring

[client.admin] //用户名全称,admin为简称

key = AQBtz+9cTpLUERAAJMyeaulpwf+v1k/O9coodA==

3)创建KVM虚拟机 注意:真实机当客户端

\*虚拟机管理---创建新虚拟机----pxe---内存2048---名称nb---完成---关机

虚拟机磁盘默认路径 /var/lib/libvirt/image/.

4)配置libvirt secret 注意:真实机当客户端

~]#vim secret.xml

<secret ephemeral='no' private='no'>

<usage type='ceph'>

<name>client.admin secret</name>

</usage>

</secret> //秘钥文件,用来访问ceph用的, 名字client.admin

# virsh secret-define secret.xml

//随机的UUID，这个UUID对应的有账户信息

# virsh --help | grep secret //查看virsh secret命令

# virsh secret-list //uuid对应的ceph的用户名

UUID 用量

--------------------------------------------------------------------------------

14a4ffc9-1a53-40e8-b8b1-e1b3827ed26f ceph client.admin secret

给secret绑定admin账户的密码，密码参考ceph.client.admin.keyring文件。

#virsh secret-set-value --secret 14a4ffc9-1a53-40e8-b8b1-e1b3827ed26f --base64 AQCYD/Fc0+XWCRAAlYbQa/xKyCyimTuj2rvnCA==

secret 值设定

//这里secret后面是之前创建的secret的UUID

//base64后面是client.admin账户的密码

//现在secret中既有账户信息又有密钥信息

6)修改虚拟机的XML文件 不推荐vim修改xml

#virsh edit nb //虚拟机名字nb

... ...

<disk type='file' device='disk'>

<driver name='qemu' type='qcow2'/>

<source file='/var/lib/libvirt/images/nb.qcow2'/>

<target dev='hda' bus='ide'/>

<address type='drive' controller='0' bus='0' target='0' unit='0'/>

</disk>

... ...

改为----------------------------->

<disk type='network' device='disk'> //网盘

<driver name='qemu' type='raw'/>

<auth username='admin'> //用户名admin

<secret type='ceph' uuid='733f0fd1-e3d6-4c25-a69f-6681fc19802b'/> //是secret的uuid

</auth>

<source protocol='rbd' name='rbd/vm1-image'> <host name='192.168.4.11' port='6789'/> </source>

<target dev='vda' bus='virtio'/>

<address type='pci' domain='0x0000' bus='0x09' slot='0x08' function='0x0'/>

</disk>

//镜像vm1-image,更改secret的uuid[virsh secret-list]

结果:虚拟机设置里, 磁盘路径变为:

rbd://192.168.4.11:6789/rbd/vm1-image

只需要有 nb.xml文件, virsh define nb.xml 就能创建虚拟机.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ceph

1.快共享

直接在一个操作系统上rbd,map

给kvm虚拟机使用

格式化动作: 分两个区,左边inode默认每个格子256字节大小,右边block每个格子4k大小,并给每个格子编号. 参见截图1

eg:写入5M大小的a.mp3,随机读写[若不是连续写则叫:磁盘碎片],磁盘碎片整理

2.文件系统共享 9(格式化完成后是文件系统)

fat32,ntfs,ext3.ext4,xfs....

左边inode------>元数据(描述信息) 默认大小256字节

右边block------->数据 默认分配单元大小4k

小文件 block设置小,大文件block设置大,杂文件默认4k大小

1G数据写入磁盘,3分钟,删除文件10秒.

删文件时,inode信息增加flag:delete标记,1G空间被算为可用空间.,没有实际删除数据,硬盘任何数据删除都可以恢复,但如果该1G可用空间被覆盖,则无法恢复.

数据恢复:扫描inode上的delete,把标记的flag:delete去掉,数据就恢复了.

eg: #mkfs.xfs -i -b format unformat

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例2:ceph文件系统存储

部署MDSs节点

创建Ceph文件系统

客户端挂载文件系统

ceph-deploy mon create node2 //拷贝配置文件 并起mon服务

ceph-deploy osd create node2

ceph-deploy mds create node2

ceph-deploy rgw create node2

本次实验为挑选node3为mds服务器.

1)启动node3的mds服务. [之前已经安装了ceph-mds软件]

node1]#cd /root/ceph-cluster

node1]#ceph-deploy mds create node3 //远程node3,起动mds服务

2)创建共享池 node3

[root@node3 ~]# ceph osd pool create cephfs\_data 128

[root@node3 ~]# ceph osd pool create cephfs\_metadata 128

#ceph osd lspools

//创建两个共享池 对应128个PG(逻辑概念),对应2^n

PG(逻辑概念) 把六个osd对应128个PG. 类似虚拟概念文件夹

3)创建ceph文件系统

#ceph mds stat //查看mds状态

e2:, 1 up:standby

[root@node3 ~]#ceph fs new myfs1 cephfs\_metadata cephfs\_data

new fs with metadata pool 2 and data pool 1

//注意，先写medadata池，再写data池

//默认，只能创建1个文件系统，多余的会报错

[root@node3 ~]#ceph fs ls

name: myfs1, metadata pool: cephfs\_metadata, data pools:...

[root@node3 ~]# ceph mds stat //查看mds状态

e5: 1/1/1 up {0=node3=up:active}

4)客户端挂载

client~]mkdir /mnt/cephfs

[client~]#mount -t ceph 192.168.4.11:6789:/ /mnt/cephfs/ -o name=admin,secret=AQBtz+9cTpLUERAAJMyeaulpwf+v1k/O9coodA==

//-t挂载类型 -o username=,secret=

//ceph只有一个文件系统.192.168.4.11:6789:/ 挂载到/mnt/cephfs

#df -h

//注意:文件系统类型为ceph

//192.168.4.11为MON节点的IP（不是MDS节点）

//admin是用户名,secret是密钥

//密钥可以在/etc/ceph/ceph.client.admin.keyring中找到

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ceph文件系统 整合命令

[root@node3 ~]# ceph osd pool create cephfs\_data 128

[root@node3 ~]# ceph osd pool create cephfs\_metadata 128

[root@node3 ~]#ceph osd lspools

[root@node3 ~]#ceph fs new myfs1 cephfs\_metadata cephfs\_data

[root@node3 ~]#ceph fs ls

[root@node3 ~]#mkdir /mnt/cephfs

[root@node3 ~]#mount -t ceph 192.168.4.11:6789:/ /mnt/cephfs/ -o name=admin,secret=AQBtz+9cTpLUERAAJMyeaulpwf+v1k/O9coodA==

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

快存储,文件系统,对象存储, 华为:hdfs(hadoop大数据)

mysql中的数据为结构化数据,文件是非结构化数据.

案例3 创建对象存储服务器 需要Restful API接口

对象存储:基于对象的存储 软件包:ceph-radosgw

\*ceph osd mon

块

文件系统 ceph-mds

对象存储 ceph-radoswg 百度云盘 app[]--调用--->ceph

步骤一:部署对象存储服务器

1）准备实验环境，要求如下：

IP地址:192.168.4.15

主机名:node5

配置yum源（包括rhel、ceph的源）

与Client主机同步时间

node1允许无密码远程node5

修改node1的/etc/hosts，并同步到所有node主机

2)部署RGW软件包

node3]#yum -y install ceph-radosgw //安装对象存储软件

node1]#cd /etc/ceph-cluster

node1]#ceph-deploy rgw create node3 //远程node3开启radosw服务

3)查看radosgw的端口信息 默认7480端口

node3]# ss -auntpl | grep rado

tcp LISTEN 0 128 \*:7480 \*:\* users:(("radosgw",pid=3396,fd=29))

4)修改服务端口

node3]#vim /etc/ceph/ceph.conf

....

[client.rgw.node3]

host = node3

rgw\_frontends = "civetweb port=8000"

node3]#systemctl restart ceph-radoswg@rwg,node3.service

node3]#ss -auntpl | grep rado

tcp LISTEN 0 128 \*:8000 \*:\* users:(("radosgw",pid=3752,fd=30))

案例4:扩展实验

https://s3tools.org/s3cmd 下载对象服务器的访问程序

需要扩展容量的话:

ceph-deploy osd create node4

++++++++++++++++++++++++++++++++++++++++++++++++

PROJECT(项目)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第二阶段末,项目: 基于LNMP的高可用负载均衡的网页架构

day01:

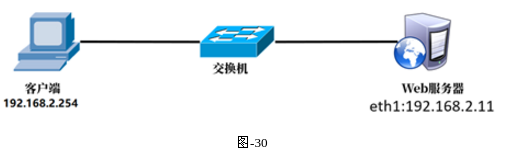


图1:

2 案例2：部署LNMP动态网站

步骤一：安装部署LNMP软件

备注：mariadb（数据库客户端软件）、mariadb-server（数据库服务器软件）、mariadb-devel（其他客户端软件的依赖包）、php（解释器）、php-fpm（进程管理器服务）、php-mysql（PHP的数据库扩展包）

1)安装软件包:

[root@centos7 ~]# yum -y install gcc openssl-devel pcre-devel

[root@centos7 ~]# useradd -s /sbin/nologin nginx

[root@centos7 ~]# tar -xvf nginx-1.12.2.tar.gz

[root@centos7 ~]# cd nginx-1.12.2

[root@centos7 nginx-1.12.2]# ./configure \

--user=nginx --group=nginx \

--with-http\_ssl\_module \

--with-http\_stub\_status\_module

[root@centos7 nginx-1.12.2]# make && make install

[root@centos7 ~]# yum -y install mariadb mariadb-server mariadb-devel

[root@centos7 ~]# yum -y install php php-mysql php-fpm

2)启动服务(nginx、mariadb、php-fpm)

[root@centos7 ~]# /usr/local/nginx/sbin/nginx #启动Nginx服务

[root@centos7 ~]# echo "/usr/local/nginx/sbin/nginx" >> /etc/rc.local

[root@centos7 ~]# chmod +x /etc/rc.local

[root@centos7 ~]# ss -utnlp | grep :80 #查看端口信息

[root@centos7 ~]# systemctl start mariadb #启动mariadb服务器

[root@centos7 ~]# systemctl enable mariadb

[root@centos7 ~]# systemctl start php-fpm #启动php-fpm服务

[root@centos7 ~]# systemctl enable php-fpm

2）修改Nginx配置文件，实现动静分离

修改配置文件，通过两个location实现动静分离，一个location匹配动态页面，一个loation匹配其他所有页面。

注意修改默认首页为index.php!

[root@centos7 ~]# vim /usr/local/nginx/conf/nginx.conf

...省略部分配置文件内容...

location / {

root html;

index index.php index.html index.htm;

}

...省略部分配置文件内容...

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf;

}

...省略部分配置文件内容...

[root@centos7 ~]# /usr/local/nginx/sbin/nginx -s reload

3）配置数据库账户与权限

为网站提前创建一个数据库、添加账户并设置该账户有数据库访问权限。

[root@centos7 ~]# mysql

MariaDB [(none)]> create database wordpress character set utf8mb4;

MariaDB [(none)]> grant all on wordpress.\* to wordpress@'%' identified by 'wordpress';

MariaDB [(none)]> flush privileges;

MariaDB [(none)]> exit

提示：在mysql和mariadb中%代表匹配所有，这里是授权wordpress用户可以从任意主机连接数据库服务器，生产环境建议仅允许特定的若干主机访问数据库服务器

软件包 wordpress.zip

步骤二：上线wordpress代码

1）上线PHP动态网站代码

[root@centos7 ~]# unzip wordpress.zip

[root@centos7 ~]# cd wordpress

[root@centos7 wordpress]# tar -xf wordpress-5.0.3-zh\_CN.tar.gz

[root@centos7 wordpress]# cp -r wordpress/\* /usr/local/nginx/html/

[root@centos7 wordpress]# chown -R apache.apache /usr/local/nginx/html/\*

提示：动态网站运行过程中，php脚本需要对网站目录有读写权限，而php-fpm默认启动用户为apache

2)初始化网站配置

[root@client ~]# firefox http://192.168.4.11/

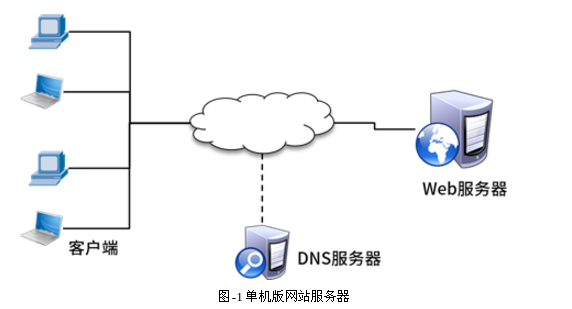
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

1 案例1：网站架构演变

1.2 步骤

此案例主要是学习网站架构演变的过程，以拓扑图和理论为主，具体实现还需要结合具体的软件。

步骤一：单机版LNMP



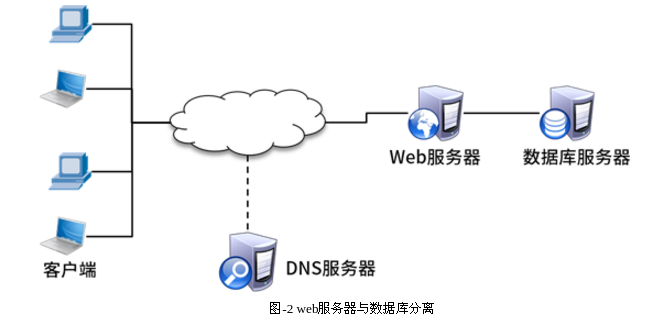
单机版网站，拓扑如图-1所示。

用户量少时使用，简单、成本低、存在单点故障。

步骤二：独立数据库服务器

独立数据库服务器是将网站静态文件、代码文件等资料与数据库分离的架构，当用户量增加时单机的处理能力有限，PHP或JAVA代码的执行需要消耗大量CPU资源，数据库的增删改查需要调用大量的内存资源，将两者分离可以减轻服务器的压力，其拓扑

结构

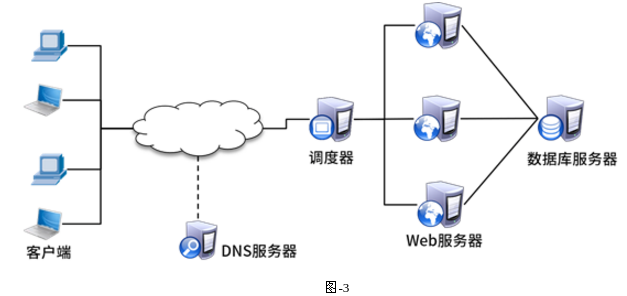


如图-2所示

Web服务器和数据库服务器的压力都可以得到有效改善，访问量有所增加。但是服务器依然存在单点故障问题。

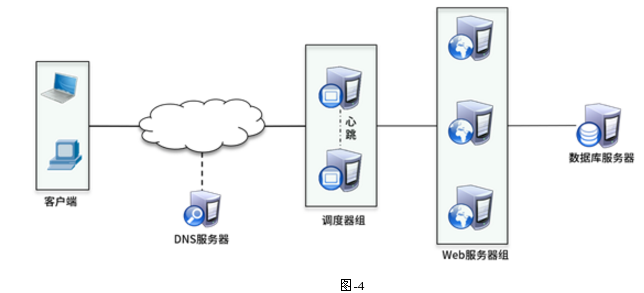
步骤三：Web服务器集群与Session保持

我们可以通过Nginx、Haproxy代理服务器实现Web负载均衡集群，也可以使用LVS调度器实现Web负载均衡集群。部署完Web集群后还需要考虑如何进行Session会话保持，方法很多，如：根据源IP保持，代理服务器重写Cookie信息，共享文件系统保存session，使用数据库共享session等等。

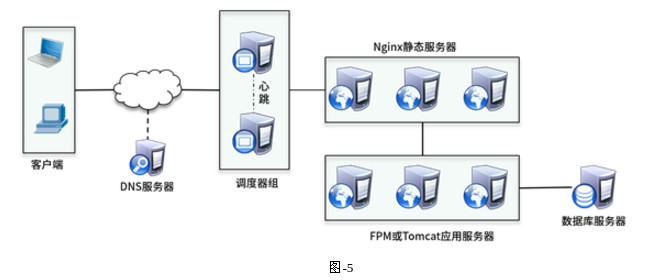


该架构拓扑如图-3所示。

但是如果只有一台调度器依然会导致单点故障的问题，因此还需要使用Keepalived或Heartbeat之类的软件进行高可用配置.如图-4所示。

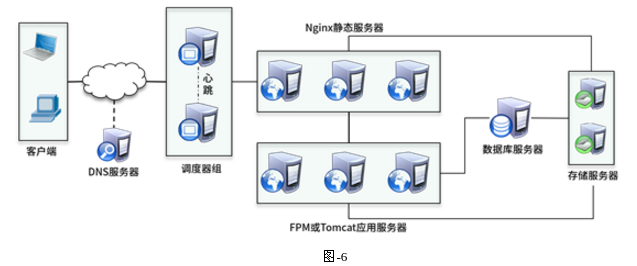


对于网站内容而言可以分离为动态页面和静态页面，静态页面就需要数据文件，动态页面则需要CPU解析代码，需要消耗大量的CPU资源，因此可以将静态和动态分离为两组服务器，动态页面有脚本代码组成，是一种基于网页的应用程序，因此这一组服务器也称为应用服务器，其架构如图-5所示。

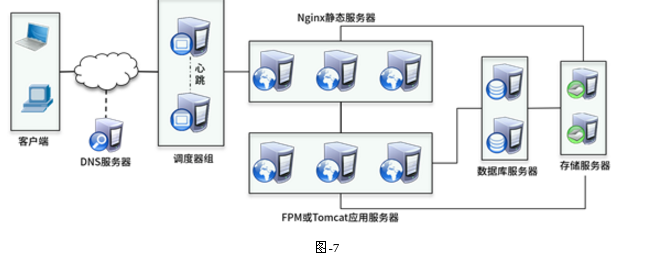


步骤四：动静分离、数据库集群

随着服务器的增加，虽然性能与并发量得到了明显的提升，但是数据的一致性、管理的便利性成为了新的问题，因此就需要增加统一的存储服务器，实现数据的同步一致，可以使用NFS，GlusterFS、Ceph等软件实现该功能，其架构如图-6所示。



此时所有应用服务器都连接一台数据库服务器进行读写操作，而且后期随着数据库中的数据不断增加，会导致数据库成为整个网站的瓶颈！这就需要我们对数据进行分库分表，创建数据库主从或者数据库集群，实现读写分离，其拓扑如图-7所示。



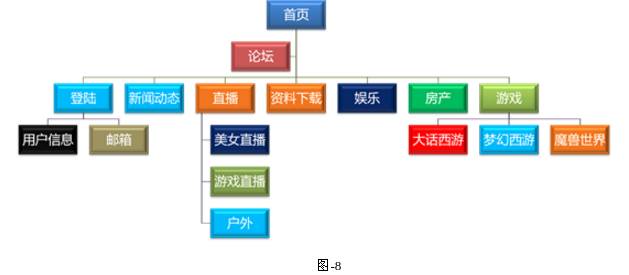
步骤四：缓存服务器与业务模型

对于静态数据我们可以通过varnish、squid或者nginx进行缓存，将数据缓存到距离用户更近的位置，构建CDN（内容分发网络）架构。

对于传统的SQL数据库而言，我们也可以通过增加NoSQL数据库，实现数据缓存的功能，提升数据库的访问速度。

备注：数据库相关知识在第三阶段课程有详细介绍，第二阶段项目暂时不做数据库优化。

最后，基于前面的架构，我们还可以将网站按照公司的业务进行分离，每个业务都可以是一个独立的集群，如图-8所示。



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第二阶段项目: 第二天

2 案例2：LNP+Mariadb数据库分离

2.1 问题

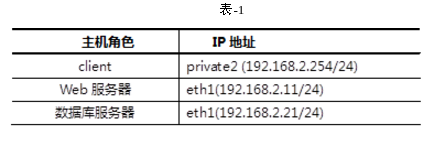
部署LNP+Mariadb实现数据库与Web服务器分离，实现以下目标：

将旧的数据库备份，迁移到新的服务器

修改配置调用新的数据库服务器



图1



主机配置如表-1所示

步骤一：部署数据库服务器

1）准备一台独立的服务器，安装数据库软件包

[root@database ~]# yum -y install mariadb mariadb-server mariadb-devel

[root@database ~]# systemctl start mariadb

[root@database ~]# systemctl enable mariadb

2)将之前单机版LNMP网站中的数据库迁移到新的数据库服务器。

登陆192.168.2.11主机，备份数据库并拷贝给新的服务器，关闭旧的数据库服务

[root@centos7 ~]# mysqldump wordpress > wordpress.bak

[root@centos7 ~]# scp wordpress.bak 192.168.2.21:/root/

[root@centos7 ~]# systemctl stop mariadb

[root@centos7 ~]# systemctl disable mariadb

登陆192.168.2.21主机，使用备份文件还原数据库。

创建空数据库：

[root@database ~]# mysql

MariaDB [(none)]> create database wordpress character set utf8mb4;

MariaDB [(none)]> exit

使用备份文件还原数据：

[root@database ~]# mysql wordpress < wordpress.bak

重新创建账户并授权访问：

[root@database ~]# mysql

MariaDB [(none)]> grant all on wordpress.\* to wordpress@'%' identified by 'wordpress';

MariaDB [(none)]> flush privileges;

MariaDB [(none)]> exit

3）修改wordpress网站配置文件，调用新的数据库服务器。

Wordpress在第一次初始化操作时会自动生产配置文件：wp-config.php，登陆192.168.2.11修改该文件即可调用新的数据库服务

[root@centos7 ~]# vim /usr/local/nginx/html/wp-config.php

修改前内容如下：

define('DB\_HOST', '192.168.2.11');

修改后内容如下：

define('DB\_HOST', '192.168.2.21');

步骤二：客户端测试

1）客户端使用浏览器访问wordpress网站

[root@client ~]# firefox http://192.168.2.11

3 案例3：Web服务器集群

3.1 问题

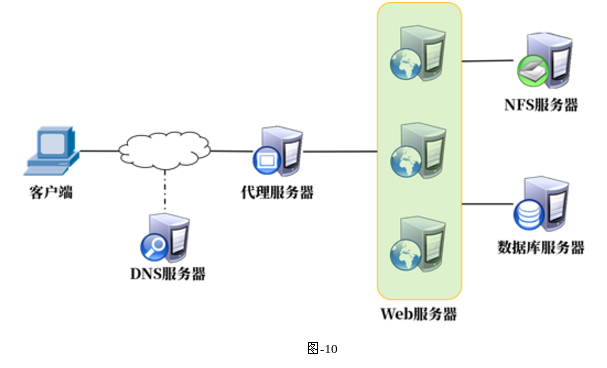
使用HAProxy部署Web服务器集群，实现以下目标：

部署三台Web服务器

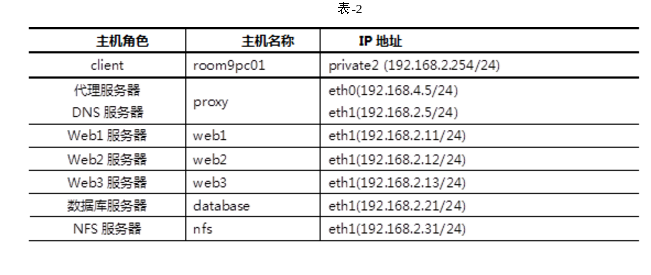
迁移网站数据，使用NFS实现数据共享

部署HAProxy代理服务器实现负载均衡

部署DNS域名解析服务器



方案拓扑图3



主机配置表 图4

3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：部署web2和web3服务器

1）安装LNP软件包

[root@web2 ~]# yum -y install gcc pcre-devel openssl-devel

[root@web2 lnmp\_soft]# tar -xf nginx-1.12.2.tar.gz

[root@web2 lnmp\_soft]# cd nginx-1.12.2/

[root@web2 nginx-1.12.2]# ./configure \

--with-http\_ssl\_module \

--with-http\_stub\_status\_module

[root@web2 nginx-1.12.2]# make && make instal

[root@web2 ~]# yum -y install php php-fpm php-mysql mariadb-devel

[root@web3 ~]# yum -y install gcc pcre-devel openssl-devel

[root@web3 lnmp\_soft]# tar -xf nginx-1.12.2.tar.gz

[root@web3 lnmp\_soft]# cd nginx-1.12.2/

[root@web3 nginx-1.12.2]# ./configure \

--with-http\_ssl\_module \

--with-http\_stub\_status\_module

[root@web3 nginx-1.12.2]# make && make instal

[root@web3 ~]# yum -y install php php-fpm php-mysql mariadb-devel

2）修改nginx配置实现动静分离（web2和web3操作）

web2修改默认首页index.php，配置两个location实现动静分离。

[root@web2 ~]# vim /usr/local/nginx/conf/nginx.conf

location / {

root html;

index index.php index.html index.htm;

}

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf;

}

web3修改默认首页index.php，配置两个location实现动静分离.

[root@web3 ~]# vim /usr/local/nginx/conf/nginx.conf

location / {

root html;

index index.php index.html index.htm;

}

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf;

}

3）启动相关服务

[root@web2 ~]# echo "/usr/local/nginx/sbin/nginx" >> /etc/rc.local

[root@web2 ~]# chmod +x /etc/rc.local

[root@web2 ~]# /usr/local/nginx/sbin/nginx

[root@web2 ~]# systemctl start php-fpm #启动php-fpm服务

[root@web2 ~]# systemctl enable php-fpm

[root@web3 ~]# echo "/usr/local/nginx/sbin/nginx" >> /etc/rc.local

[root@web3 ~]# chmod +x /etc/rc.local

[root@web3 ~]# /usr/local/nginx/sbin/nginx

[root@web3 ~]# systemctl start php-fpm #启动php-fpm服务

[root@web3 ~]# systemctl enable php-fpm

步骤二：部署NFS，将网站数据迁移至NFS共享服务器

1）部署NFS共享服务器

[root@nfs ~]# yum install nfs-utils

[root@nfs ~]# mkdir /web\_share

[root@nfs ~]# vim /etc/exports

/web\_share 192.168.2.0/24(rw,no\_root\_squash)

[root@nfs ~]# systemctl restart rpcbind

[root@nfs ~]# systemctl eanble rpcbind

NFS使用的是随机端口，每次启动NFS都需要将自己的随机端口注册到rpcbind服务，这样客户端访问NFS时先到rpcbind查询端口信息，得到端口信息后再访问NFS服务.

[root@nfs ~]# systemctl restart nfs

[root@nfs ~]# systemctl enable nfs

2）迁移旧的网站数据到NFS共享服务器

将web1（192.168.2.11）上的wordpress代码拷贝到NFS共享

[root@web1 ~]# cd /usr/local/nginx/

[root@web1 nginx]# tar -czpf html.tar.gz html/

[root@web1 nginx]# scp html.tar.gz 192.168.2.31:/web\_share/

登陆nfs服务器，将压缩包解压

[root@nfs ~]# cd /web\_share/

[root@nfs web\_share]# tar -xf html.tar.gz

3)所有web服务器访问挂载NFS共享数据。

[root@web1 ~]# rm -rf /usr/local/nginx/html/\*

[root@web1 ~]# yum -y install nfs-utils

[root@web1 ~]# echo "192.168.2.31:/web\_share/html /usr/local/nginx/html/ nfs defaults 0 0" >> /etc/fstab

[root@web1 ~]# mount -a

[root@web2 ~]# yum -y install nfs-utils

[root@web2 ~]# echo "192.168.2.31:/web\_share/html /usr/local/nginx/html/ nfs defaults 0 0" >> /etc/fstab

[root@web2 ~]# mount -a

[root@web3 ~]# yum -y install nfs-utils

[root@web3 ~]# echo "192.168.2.31:/web\_share/html /usr/local/nginx/html/ nfs defaults 0 0" >> /etc/fstab

[root@web3 ~]# mount -a

步骤三：部署HAProxy代理服务器

1）部署HAProxy

安装软件，手动修改配置文件，添加如下内容。

[root@proxy ~]# yum -y install haproxy

[root@proxy ~]# vim /etc/haproxy/haproxy.cfg

listen wordpress \*:80

balance roundrobin

server web1 192.168.2.11:80 check inter 2000 rise 2 fall 3

server web2 192.168.2.12:80 check inter 2000 rise 2 fall 3

server web3 192.168.2.13:80 check inter 2000 rise 2 fall 3

[root@proxy ~]# systemctl start haproxy

[root@proxy ~]# systemctl enable haproxy

步骤三：部署DNS域名服务器

1）安装DNS相关软件（192.168.4.5操作）。

[root@proxy ~]# yum -y install bind bind-chroot

2）修改主配置文件，添加zone。

[root@proxy ~]# vim /etc/named.conf

options {

listen-on port 53 { any; }; #服务监听的地址与端口

directory "/var/named"; #数据文件路径

allow-query { any; }; #允许任何主机访问DNS服务

... ...

};

zone "lab.com" IN { #定义正向区域

type master;

file "lab.com.zone";

};

#include "/etc/named.rfc1912.zones"; #注释掉改行

#include "/etc/named.root.key"; #注释掉改行

[root@proxy ~]# named-checkconf /etc/named.conf #检查语法

3）修改正向解析记录文件。

注意：保留文件权限。

[root@proxy named]# cp -p /var/named/named.localhost /var/named/lab.com.zone

[root@proxy named]# vim /var/named/lab.zone

$TTL 1D

@ IN SOA @ rname.invalid. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

@ NS dns.lab.com.

dns A 192.168.4.5

www A 192.168.4.5

4）启动服务

[root@proxy named]# systemctl start named

[root@proxy named]# systemctl enable named

5）客户端修改DNS解析文件

提示：做完实验修改回原始内容。

[root@room9pc01 data]# cat /etc/resolv.conf

# Generated by NetworkManager

search tedu.cn

nameserver 192.168.4.5

nameserver 172.40.1.10

nameserver 192.168.0.220

步骤四：修改wordpress配置文件

1）修改wp-config.php

在define('DB\_NAME', 'wordpress')这行前面添加如下两行内容：

[root@web3 html]# vim /usr/local/nginx/html/wp-config.php

define('WP\_SITEURL', 'http://www.lab.com');

define('WP\_HOME', 'http://www.lab.com');

如果不添加这两行配置，浏览器访问网站某个子页面后，URL会固定到某一台后端服务器不轮询。

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 案例1：Keepalived高可用

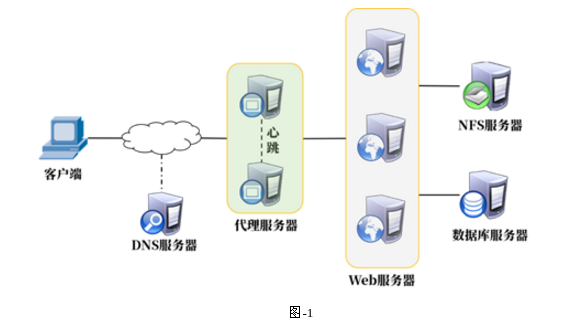
1.1问题

部署两台代理服务器，实现如下效果：

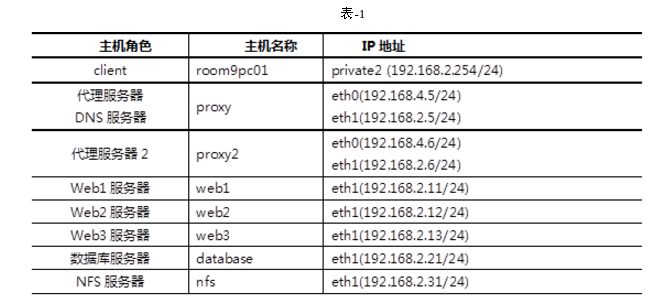
利用keepalived实现两台代理服务器的高可用

配置VIP为192.168.4.80

修改对应的域名解析记录



1.2实验拓扑如图-1所示，做具体实验前请先配置好环境



主机配置如表-1所示。

步骤一：配置第二台代理服务器

1）部署HAProxy

安装软件，手动修改配置文件，添加如下内容

[root@proxy2 ~]# yum -y install haproxy

[root@proxy2 ~]# vim /etc/haproxy/haproxy.cfg

listen wordpress \*:80

balance roundrobin

server web1 192.168.2.11:80 check inter 2000 rise 2 fall 3

server web2 192.168.2.12:80 check inter 2000 rise 2 fall 3

server web3 192.168.2.13:80 check inter 2000 rise 2 fall 3

[root@proxy2 ~]# systemctl start haproxy

[root@proxy2 ~]# systemctl enable haproxy

步骤二：为两台代理服务器配置keepalived

1）配置第一台代理服务器proxy（192.168.4.5）。

[root@proxy ~]# yum install -y keepalived

[root@proxy ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

router\_id proxy1 //设置路由ID号

}

vrrp\_instance VI\_1 {

state MASTER //主服务器为MASTER（备服务器需要修改为BACKUP）

interface eth0 //定义网络接口

virtual\_router\_id 51

priority 100 //服务器优先级,优先级高优先获取VIP（实验需要修改）

advert\_int 1

authentication {

auth\_type pass

auth\_pass 1111 //主备服务器密码必须一致

}

virtual\_ipaddress { //谁是主服务器谁获得该VIP（实验需要修改）

192.168.4.80

}

}

[root@proxy ~]# systemctl start keepalived

[root@proxy ~]# iptables -F #清空防火墙规则

2）配置第二台代理服务器proxy（192.168.4.6）。

[root@proxy2 ~]# yum install -y keepalived

[root@proxy2 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

router\_id proxy2 //设置路由ID号

}

vrrp\_instance VI\_1 {

state BACKUP //主服务器为MASTER（备服务器需要修改为BACKUP）

interface eth0 //定义网络接口

virtual\_router\_id 51

priority 50 //服务器优先级,优先级高优先获取VIP

advert\_int 1

authentication {

auth\_type pass

auth\_pass 1111 //主备服务器密码必须一致

}

virtual\_ipaddress { //谁是主服务器谁获得该VIP

192.168.4.80

}

}

[root@proxy2 ~]# systemctl start keepalived

[root@proxy2 ~]# iptables -F #清空防火墙规则

步骤三：修改DNS服务器

1）修改网站域名对应的解析记录，解析到新的VIP地址。

192.168.4.5为DNS服务器。

[root@proxy ~]# vim /var/named/lab.com.zone

$TTL 1D

@ IN SOA @ rname.invalid. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

@ NS dns.lab.com.

dns A 192.168.4.5

www A 192.168.4.80

2)重启DNS服务

[root@proxy ~]# systemctl restart named

2 案例2：部署Ceph分布式存储

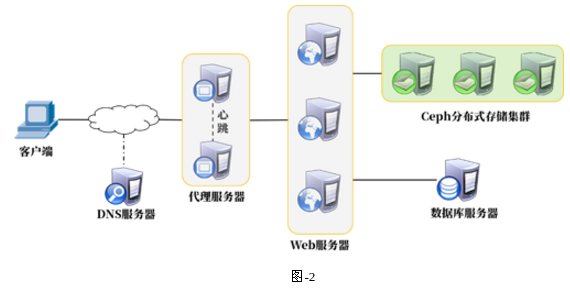
2.1 问题

部署Ceph分布式存储，实现如下效果：

使用三台服务器部署Ceph分布式存储

实现Ceph文件系统共享

将网站数据从NFS迁移到Ceph存储



实验拓扑如图-2所示

备注：实际操作中DNS服务代理服务器部署在同一台主机上（节约虚拟机资源）。



主机配置如表-2所示。

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：准备实验环境

1）物理机为所有节点配置yum源服务器。

提示：ceph10.iso在/linux-soft/02目录。

[root@room9pc01 ~]# mkdir /var/ftp/ceph

[root@room9pc01 ~]# mount ceph10.iso /var/ftp/ceph/

2）在node1配置SSH密钥，让node1可用无密码连接node1,node2,node3

[root@node1 ~]# ssh-keygen -f /root/.ssh/id\_rsa -N ''

[root@node1 ~]# for i in 41 42 43

do

ssh-copy-id 192.168.2.$i

done

3)修改/etc/hosts域名解析记录（不要删除原有的数据），同步给所有ceph节点。

[root@node1 ~]# vim /etc/hosts

192.168.2.41 node1

192.168.2.42 node2

192.168.2.43 node3

[root@node1 ~]# for i in 41 42 43

do

scp /etc/hosts 192.168.2.$i:/etc

done

4）为所有ceph节点配置yum源，并将配置同步给所有节点

[root@node1 ~]# cat /etc/yum.repos.d/ceph.repo

[mon]

name=mon

baseurl=ftp://192.168.2.254/ceph/MON

gpgcheck=0

[osd]

name=osd

baseurl=ftp://192.168.2.254/ceph/OSD

gpgcheck=0

[tools]

name=tools

baseurl=ftp://192.168.2.254/ceph/Tools

gpgcheck=0

[root@node1 ~]# yum repolist #验证YUM源软件数量

源标识 源名称 状态

Dvd redhat 9,911

Mon mon 41

Osd osd 28

Tools tools 33

repolist: 10,013

[root@node1 ~]# for i in 41 42 43

do

scp /etc/yum.repos.d/ceph.repo 192.168.2.$i:/etc/yum.repos.d/

done

5）所有节点主机与真实主机的NTP服务器同步时间。

提示：默认真实物理机已经配置为NTP服务器

[root@node1 ~]# vim /etc/chrony.conf

… …

server 192.168.2.254 iburst

[root@node1 ~]# for i in 41 42 43

do

scp /etc/chrony.conf 192.168.2.$i:/etc/

ssh 192.168.2.$i "systemctl restart chronyd"

done

6）使用virt-manager为三台ceph虚拟机添加磁盘。

每台虚拟机添加3块20G的磁盘。

步骤二：部署ceph集群

1）给node1主机安装ceph-deploy，创建工作目录，初始化配置文件

[root@node1 ~]# yum -y install ceph-deploy

[root@node1 ~]# mkdir ceph-cluster

[root@node1 ~]# cd ceph-cluster

[root@node1 ceph-cluster]# ceph-deploy new node1 node2 node3

2）给所有ceph节点安装ceph相关软件包

[root@node1 ceph-cluster]# for i in node1 node2 node3

do

ssh $i "yum -y install ceph-mon ceph-osd ceph-mds"

done

[root@node1 ceph-cluster]# ceph -s #查看结果

cluster 9f3e04b8-7dbb-43da-abe6-b9e3f5e46d2e

health HEALTH\_ERR

monmap e2: 3 mons at

{node1=192.168.2.41:6789/0,node2=192.168.2.42:6789/0,node3=192.168.2.43:6789/0}

3）准备磁盘分区，创建journal盘，并永久修改设备权限。

[root@node1 ceph-cluster]# for i in node1 node2 node3

do

ssh $i "parted /dev/vdb mklabel gpt"

ssh $i "parted /dev/vdb mkpart primary 1 50%"

ssh $i "parted /dev/vdb mkpart primary 50% 100%"

don

提示：下面的步骤在所有主机都需要操作（node1，node2，node3）

#临时修改权限：

[root@node1 ceph-cluster]# chown ceph.ceph /dev/vdb1

[root@node1 ceph-cluster]# chown ceph.ceph /dev/vdb2

#永久修改权限：

[root@node1 ceph-cluster]# vim /etc/udev/rules.d/70-vdb.rules

ENV{DEVNAME}=="/dev/vdb1",OWNER="ceph",GROUP="ceph"

ENV{DEVNAME}=="/dev/vdb2",OWNER="ceph",GROUP="cep

4）使用ceph-deploy工具初始化数据磁盘（仅node1操作）。

[root@node1 ceph-cluster]# ceph-deploy disk zap node1:vdc node1:vdd

[root@node1 ceph-cluster]# ceph-deploy disk zap node2:vdc node2:vdd

[root@node1 ceph-cluster]# ceph-deploy disk zap node3:vdc node3:vdd

5）初始化OSD集群

[root@node1 ceph-cluster]# ceph-deploy osd create \

node1:vdc:/dev/vdb1 node1:vdd:/dev/vdb2

//创建osd存储设备，vdc为集群提供存储空间，vdb1提供JOURNAL缓存，

//一个存储设备对应一个缓存设备，缓存需要SSD，不需要很大

[root@node1 ceph-cluster]# ceph-deploy osd create \

node2:vdc:/dev/vdb1 node2:vdd:/dev/vdb2

[root@node1 ceph-cluster]# ceph-deploy osd create \

node3:vdc:/dev/vdb1 node3:vdd:/dev/vdb2

[root@node1 ceph-cluster]# ceph -s #查看集群状态.

步骤三：部署ceph文件系统

1）启动mds服务

[root@node1 ceph-cluster]# ceph-deploy mds create node3

2）创建存储池（文件系统由inode和block组成

[root@node1 ceph-cluster]# ceph osd pool create cephfs\_data 128

[root@node1 ceph-cluster]# ceph osd pool create cephfs\_metadata 128

[root@node1 ceph-cluster]# ceph osd lspools

0 rbd,1 cephfs\_data,2 cephfs\_metadata

3）创建文件系统

[root@node1 ceph-cluster]# ceph fs new myfs1 cephfs\_metadata cephfs\_data

[root@node1 ceph-cluster]# ceph fs ls

name: myfs1, metadata pool: cephfs\_metadata, data pools: [cephfs\_data ]

步骤四：迁移网站数据到ceph集群

1）卸载web1，web2，web3的NFS共享。

暂停服务防止有人实时读写文件

[root@web1 ~]# /usr/local/nginx/sbin/nginx -s stop

[root@web2 ~]# /usr/local/nginx/sbin/nginx -s stop

[root@web3 ~]# /usr/local/nginx/sbin/nginx -s stop

[root@web1 ~]# umount /usr/local/nginx/html

[root@web2 ~]# umount /usr/local/nginx/html

[root@web3 ~]# umount /usr/local/nginx/html

[root@web1 ~]# vim /etc/fstab

#192.168.2.31:/web\_share/html /usr/local/nginx/html/ nfs defaults 0 0

[root@web2 ~]# vim /etc/fstab

#192.168.2.31:/web\_share/html /usr/local/nginx/html/ nfs defaults 0 0

[root@web3 ~]# vim /etc/fstab

#192.168.2.31:/web\_share/html /usr/local/nginx/html/ nfs defaults 0

2）web服务器永久挂载Ceph文件系统（web1、web2、web3都需要操作）。

在任意ceph节点，如node1查看ceph账户与密码

[root@node1 ~]# cat /etc/ceph/ceph.client.admin.keyring

[client.admin]

key = AQA0KtlcRGz5JxAA/K0AD/uNuLI1RqPsNGC7zg==

/etc/rc.local是开机启动脚本，任何命令放在该文件中都是开机自启。

[root@web1 ~]# mount -t ceph 192.168.2.41:6789:/ /usr/local/nginx/html/ \

-o name=admin,secret=AQA0KtlcRGz5JxAA/K0AD/uNuLI1RqPsNGC7zg==

[root@web1 ~]# echo 'mount -t ceph 192.168.2.41:6789:/ /usr/local/nginx/html/ \

-o name=admin,secret=AQA0KtlcRGz5JxAA/K0AD/uNuLI1RqPsNGC7zg==' >> /etc/rc.local

[root@web1 ~]# chmod +x /etc/rc.local

[root@web2 ~]# mount -t ceph 192.168.2.41:6789:/ /usr/local/nginx/html/ \

-o name=admin,secret=AQA0KtlcRGz5JxAA/K0AD/uNuLI1RqPsNGC7zg==

[root@web2 ~]# echo 'mount -t ceph 192.168.2.41:6789:/ /usr/local/nginx/html/ \

-o name=admin,secret=AQA0KtlcRGz5JxAA/K0AD/uNuLI1RqPsNGC7zg==' >> /etc/rc.local

[root@web2 ~]# chmod +x /etc/rc.local

[root@web3 ~]# mount -t ceph 192.168.2.41:6789:/ /usr/local/nginx/html/ \

-o name=admin,secret=AQA0KtlcRGz5JxAA/K0AD/uNuLI1RqPsNGC7zg==

[root@web3 ~]# echo 'mount -t ceph 192.168.2.41:6789:/ /usr/local/nginx/html/ \

-o name=admin,secret=AQA0KtlcRGz5JxAA/K0AD/uNuLI1RqPsNGC7zg==' >> /etc/rc.local

[root@web3 ~]# chmod +x /etc/rc.local

3)迁移NFS服务器中的数据到Ceph存储

登陆NFS服务器备份数据，将备份数据拷贝给web1或web2或web3，tar备份数据时注意使用-f选项保留文件权限。

[root@nfs ~]# cd /web\_share/html/

[root@nfs html]# tar -czpf /root/html.tar.gz ./\*

[root@nfs html]# scp /root/html.tar.gz 192.168.2.11:/usr/local/nginx/html/

登陆web1将数据恢复到Ceph共享目录

[root@web1 html]# tar -xf html.tar.gz

[root@web1 html]# rm -rf html.ta

3）恢复web服务

[root@web1 ~]# /usr/local/nginx/sbin/nginx

[root@web2 ~]# /usr/local/nginx/sbin/nginx

[root@web3 ~]# /usr/local/nginx/sbin/nginx

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 案例1：部署Git版本控制系统

1.1 问题

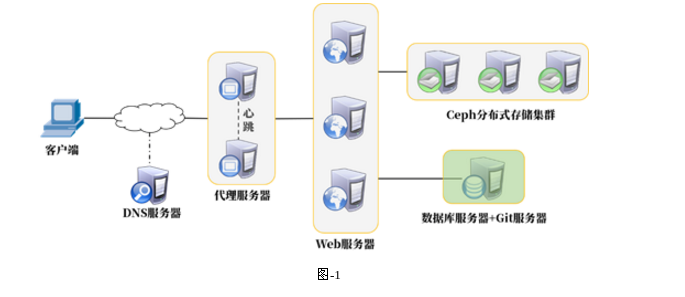
部署Git版本控制系统，管理网站代码，实现如下效果：

基于SSH协议的服务器

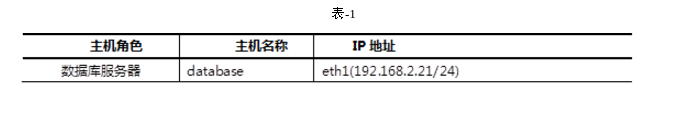
基于Git协议的服务器

基于HTTP协议的服务器

上传代码到版本仓库



生产环境应该有一台独立的Git服务器，这里为了节约主机资源，我们使用数据库主机同时做完Git服务器，如图-1所示



主机配置如表-1所示

1.3步骤

实现此案例需要按照如下步骤进行。

步骤一：部署SSH协议的版本控制服务器

1）安装软件包，创建空仓库

[root@database ~]# yum -y install git

[root@database ~]# git init --bare /var/git/wordpress.git #创建空仓

2）登陆web1服务器克隆git仓库，上传网站代码到git服务器

[root@web1 var]# git config --global push.default simple

[root@web1 var]# git config --global user.email you@example.com

[root@web1 var]# git config --global user.name "Your Name"

[root@web1 var]# cd /var/

[root@web1 var]# git clone root@192.168.2.21:/var/git/wordpress.git

[root@web1 var]# cd /var/wordpress

[root@web1 wordpress]# cp -a /usr/local/nginx/html/\* ./

[root@web1 wordpress]# git add .

[root@web1 wordpress]# git commit -m "wordpress code"

[root@web1 wordpress]# git push

root@192.168.2.21's password:<输入192.168.2.21主机root的密码>

3)客户端测试（使用web2做完客户端主机，192.168.2.12）

在web2执行clone等同于是把代码又备份了一份

[root@web2 ~]# cd /var/

[root@web2 var]# git clone git://192.168.2.21/wordpress.git

步骤三：部署HTTP协议的版本控制服务器

1）安装软件包（192.168.2.21操作)

[root@database ~]# yum -y install httpd gitweb

2）修改配置文件

[root@database ~]# vim /etc/gitweb.conf

$projectroot = "/var/git"; #添加一行

3）启动服务

[root@database ~]# systemctl start httpd

4）客户端验证

[root@room9pc01 ~]# firefox http://192.168.2.21/g

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 案例2：优化Web服务器

2.1 问题

优化Web服务器，实现如下效果：

自定义网站404错误页面

升级nginx至1.15.8版本，开启status模块

编写日志切割脚本，实现每周五备份日志

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：自定义404错误页面

1）优化前测试（客户端访问一个不存在的页面）。

[root@room9pc01 ~]# firefox http://www.lab.com/git

2) 修改Nginx配置文件，自定义错误页面

[root@web1 ~]# vim /usr/local/nginx/conf/nginx.conf

error\_page 404 /404.html; //自定义错误页面

[root@web2 ~]# vim /usr/local/nginx/conf/nginx.conf

error\_page 404 /404.html; //自定义错误页面

[root@web3 ~]# vim /usr/local/nginx/conf/nginx.conf

error\_page 404 /404.html; //自定义错误页面

3） 重启nginx

[root@web1 ~]# /usr/local/nginx/sbin/nginx -s reload

[root@web2 ~]# /usr/local/nginx/sbin/nginx -s reload

[root@web3 ~]# /usr/local/nginx/sbin/nginx -s reloa

步骤二：升级nginx版本，开启status模块

1）配置、编译新的nginx（web1、web2、web3做相同操作，下面以web1为例）

[root@web1 ~]# tar -xf nginx-1.15.8.tar.gz

[root@web1 ~]# cd nginx-1.15.8

[root@web1 ~]# ./configure \

--with-http\_ssl\_module \

--with-http\_stub\_status\_module

[root@web1 ~]# make

2) 备份老版本nginx，更新新版本nginx

[root@web1 ~]# mv /usr/local/nginx/sbin/nginx{,.bak}

[root@web1 ~]# cp objs/nginx /usr/local/nginx/sbin/

3）修改配置文件

[root@web1 ~]# vim /usr/local/nginx/conf/nginx.conf

... ...

location /status {

stub\_status on;

allow 192.168.2.0/24;

deny all;

}

... ...

4) 重启服务

[root@web1 ~]# killall nginx

[root@web1 ~]# /usr/local/nginx/sbin/nginx

步骤三：编写日志切割脚本

1）编写脚本（以web1为例）

[root@web1 ~]# vim /usr/local/nginx/logbak.sh

#!/bin/bash

date=`date +%Y%m%d`

logpath=/usr/local/nginx/logs

mv $logpath/access.log $logpath/access-$date.log

mv $logpath/error.log $logpath/error-$date.log

kill -USR1 $(cat $logpath/nginx.pid)

2）创建计划任务

[root@web1 ~]# crontab -e

03 03 \* \* 5 /usr/local/nginx/logbak.sh