1.HTTP状态码详情

1xx（临时响应）

2xx （成功）

3xx （重定向）

4xx（请求错误）

这些状态码表示请求可能出错，妨碍了服务器的处理。

5xx（服务器错误）

这些状态码表示服务器在处理请求时发生内部错误。这些错误可能是服务器本身的错误，而不是请求出错。



1. kill -HUP pid 或者 killall -HUP pName：

其中pid是进程标识，pName是进程的名称

如果想要更改配置而不需停止并重新启动服务，可以使用上面两个命令。在对配置文件作必要的更改后，发出该命令以动态更新服务配置。

根据约定，当你发送一个挂起信号(信号1或HUP)时，大多数服务器进程(所有常用的进程)都会进行复位操作并重新加载它们的配置文件。

USR1亦通常被用来告知应用程序重载配置文件；例如，向Apache HTTP服务器发送一个USR1信号将导致以下步骤的发生：停止接受新的连接，等待当前连接停止，重新载入配置文件，重新打开日志文件，重启服务器，从而实现相对平滑的不关机的更改。

kill -USR1 $PID

(location / 的优先级最低,最后匹配)

nginx动静分离:

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

location / {

root html;

index index.php index.html index.htm;

#设置默认首页为index.php，当用户在浏览器地址栏中只写域名或IP，不说访问什么页面时，服务器会把默认首页index.php返回给用户

}

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000; #将请求转发给本机9000端口，PHP解释器

fastcgi\_index index.php;

#fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi.conf;

}

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。

nginx开启用户认证:

配置文件加 (在需要的server中)

auth\_basic "Input Password:"; //认证提示符

auth\_basic\_user\_file "/usr/local/nginx/pass"; //认证密码文件

用htpasswd 生成所需密码文件

yum -y install httpd-tools

[root@proxy ~]# htpasswd -c /usr/local/nginx/pass tom //创建密码文件

htpasswd /usr/local/nginx/pass jerry //追加用户，不使用-c选项

重启服务.

1. web服务器反向代理

upstream服务器集群池属性:

#weight设置服务器权重值，默认值为1

#max\_fails设置最大失败次数

#fail\_timeout设置失败超时时间，单位为秒

#down标记服务器已关机，不参与集群调度

格式1:置于http之中

http {

upstream webserver {

ip\_hash; #设置相同客户端访问相同Web服务器,解决session问题

server 192.168.2.100:80;

server 192.168.2.200:80;

}

.. ..

server {

listen 80;

server\_name localhost;

location / {

#通过proxy\_pass将用户的请求转发给webserver集群

proxy\_pass http://webserver;

}

}

}

ip\_hash: IPmd5sum 之后%2 ,结果为0/1 从而进行匹配

另外,可设置后端服务器集群池属性:

upstream webserver {

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

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

server 192.168.2.101 down;

}

#weight设置服务器权重值，默认值为1

#max\_fails设置最大失败次数

#fail\_timeout设置失败超时时间，单位为秒

#down标记服务器已关机，不参与集群调度

.. ..

如server 192.168.2.100 weight=1 max\_fails=2 fail\_timeout=10;

1. TCP/UDP调度:

Nginx编译安装时需要使用--with-stream，开启ngx\_stream\_core\_module模块

单独设置一个stream

stream {

upstream backend {

server 192.168.2.100:22; //后端SSH服务器的IP和端口

server 192.168.2.200:22;

}

server {

listen 12345; //Nginx监听的端口,防止冲突

proxy\_connect\_timeout 1s;

proxy\_timeout 3s;

proxy\_pass backend;

}

}

http {

.. ..

}

客户端测试 [root@client ~]# ssh 192.168.4.5 -p 12345

1. 查看服务器状态信息（非常重要的功能）

编译安装时使用--with-http\_stub\_status\_module开启状态页面模块

找一个http的server,在之中添加

location /status {

stub\_status on;

#allow IP地址;

#deny IP地址;

}



Active connections：当前活动的连接数量。

Accepts：已经接受客户端的连接总数量。 #非实时,表示开启服务至今

Handled：已经处理客户端的连接总数量。 #非实时,表示开启服务至今

（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。 #非实时,表示开启服务至今

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。

1. Linux内核参数

配置文件 vim /etc/security/limits.conf

1. 修改Nginx配置文件，增加数据包头部缓存大小

解决414 URL过长错误

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

.. ..

http {

client\_header\_buffer\_size 1k; //默认请求包头信息的缓存

large\_client\_header\_buffers 4 4k; //大请求包头部信息的缓存个数与容量 (16k)

.. ..

}

1. 日志切割脚本

kill -l 传送信号

1. 压缩页面

修改Nginx配置文件

**[**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

gzip\_comp\_level **4;**                //压缩比率

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

**..** **..**

**}**

9.**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

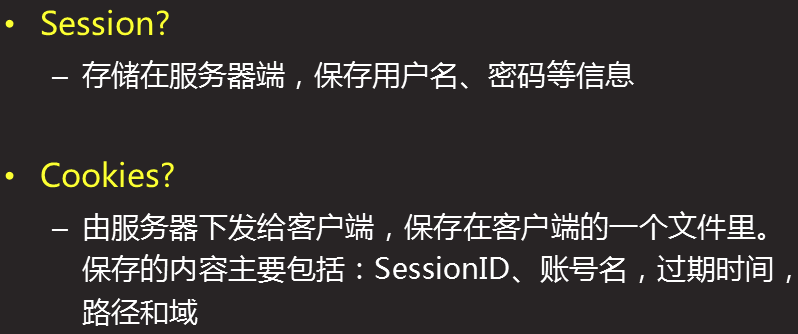
//只有访问次数超过5次会被缓存

**}**

补充:

nginx地址重写

10. session cookies 主要防止重复登录



1. memcached 非关系型数据库 key-value
2. ss -antpu

13.利用memcached实现共享session

* 装php-memcache 的扩展包
* 启动服务 nginx mariadb php-fpm
* 启动服务并查看网络连接状态验证是否开启成功

netstat -anptu | grep memcached

* 关闭SELinux、防火墙
* 修改PHP-FPM配置文件，并重启服务



1. git步骤: add ,commoit -m, push(推送代码,上传) , pull(拉取代码,下载)

root@pc001 ~]# git clone https://github.com/账户名称/仓库名称

#clone指令用于将服务器仓库中的资料打包下载到本地

[root@pc001 ~]# cd 仓库名称

[root@pc001 ~]# 任意修改文件，或新建文件

[root@pc001 ~]# git add . .可以是某些文件

#add添加新文件

[root@pc001 ~]# git commit -m "test" 提交更改

[root@pc001 ~]# git push 推送请求

#commit和push实现提交代码的功能

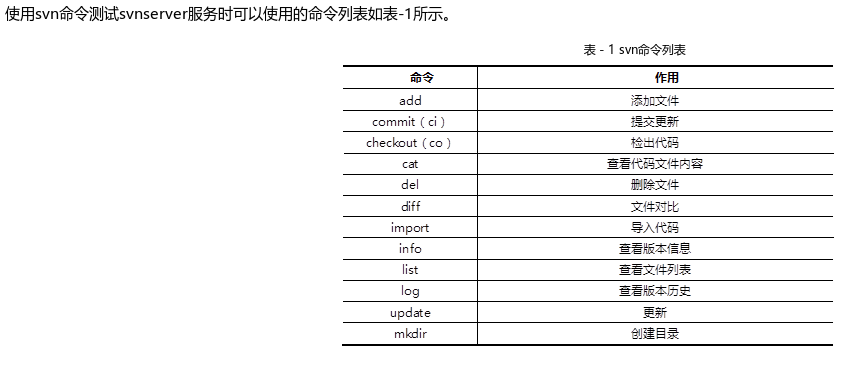
[root@pc001 ~]# git pull 更新为网络仓库中的版本

#pull更新，类似于svn update

svn:

checkout co 检出,从服务器下载

commit ci 提交.上传到服务器



1. NTP时间服务器的搭建

①服务端,装包

改配置(rpm -qc chronyd 可以查看配置文件)

[root@proxy ~]# cat /etc/chrony.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服务器的层数量

.. ..

启动服务

②客户端

装包.

改配置:

server 192.168.4.5 iburst //设置与哪台服务器同步数据

//iburst参数设置重启服务后尽快同步时间

修改系统时间后启动服务,验证是否同步.

VPN:

1. GRE-VPN (服务器,客户端都要配置)

lsmod //显示模块列表

1) lsmod | grep ip\_gre //确定是否加载了gre模块

2)加载模块ip\_gre

modprobe ip\_gre

1）创建隧道 client

[root@client ~]# 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（类似与设置网卡up）

[root@client ~]# ip link show

[root@client ~]# ip link set tun0 up //设置UP

[root@client ~]# ip link show

2）为VPN配置隧道IP地址

[root@client ~]# ip addr add 10.10.10.10/24 peer 10.10.10.5/24 \

> dev tun0

//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

[root@client ~]# ip a s //查看IP地址

proxy

proxy的地址要反过来

[root@proxy ~]# ~]# ip tunnel add tun0 mode gre \

> remote 201.1.2.10 local 201.1.2.5

//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

4）启用该隧道（类似与设置网卡up）

[root@proxy ~]# ip link show

[root@proxy ~]# ip link set tun0 up //设置UP

[root@proxy ~]# ip link show

5）为VPN配置隧道IP地址

[root@proxy ~]# ip addr add 10.10.10.5/24 peer 10.10.10.10/24 \

> dev tun0

//为隧道tun0设置本地IP地址（10.10.10.10.5/24）

//隧道对面的主机IP的隧道IP为10.10.10.10/24

[root@proxy ~]# ip a s //查看IP地址

1. PPTP VPN

步骤一：部署VPN服务器

1）安装软件包（软件包参考lnmp\_soft）

[root@proxy ~]# yum localinstall pptpd-1.4.0-2.el7.x86\_64.rpm

[root@proxy ~]# rpm -qc pptpd

/etc/ppp/options.pptpd

/etc/pptpd.conf

/etc/sysconfig/pptpd

2)修改配置文件

[root@proxy ~]# vim /etc/pptpd.conf

.. ..

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

[root@proxy ~]# vim /etc/ppp/options.pptpd

require-mppe-128 //使用MPPE加密数据

ms-dns 8.8.8.8 //DNS服务器

[root@proxy ~]# vim /etc/ppp/chap-secrets //修改账户配置文件

jacob \* 123456 \*

//用户名 服务器标记 密码 客户端

[root@proxy ~]# echo "1" > /proc/sys/net/ipv4/ip\_forward //开启路由转发

3）启动服务

[root@proxy ~]# systemctl start pptpd

[root@proxy ~]# systemctl enable pptpd

[root@proxy ~]# firewall-cmd --set-default-zone=trusted

4）翻墙设置（非必需操作）

[root@proxy ~]# iptables -t nat -A POSTROUTING -s 192.168.3.0/24 \

> -j SNAT --to-source 201.1.2.5

让外地员工访问到内网资源，利用VPN的解决方法就是在内网中架设一台VPN服务器。外地员工在当地连上互联网后，通过互联网连接VPN服务器，然后通过VPN服务器进入企业内网。为了保证数据安全，VPN服务器和客户机之间的通讯数据都进行了加密处理。有了数据加密，就可以认为数据是在一条专用的数据链路上进行安全传输，就如同专门架设了一个专用网络一样，但实际上VPN使用的是互联网上的公用链路，因此VPN称为虚拟专用网络，其实质上就是利用加密技术在公网上封装出一个数据通讯隧道。

pssh:

[root@proxy ~]# ssh-keygen -N '' -f /root/.ssh/id\_rsa //非交互生成密钥文件

[root@proxy ~]# ssh-copy-id host1

[root@proxy ~]# ssh-copy-id host2

[root@proxy ~]# ssh-copy-id host3

udev:

单独查看某个磁盘分区的属性信息。

[root@proxy ~]# udevadm info --query=property --path=/block/sda/sda1

写规则

vim /etc/udev/rules.d/\*.rules

定义变量ENV{}==, ACTION==”add|remove”, RUN+=”绝对路径”

设备别名(快捷方式) SYMLINK=””(或者+=) ,磁盘名称NAME=””(或者+=)

如:①修改规则文件（设置别名,修改设备所有者和权限）

/etc/udev/rules.d/70-usb.rules

ACTION=="add",ENV{ID\_VENDOR}=="TOSHIBA",ENV{DEVTYPE}=="partition",ENV{ID\_SERIAL\_SHORT}=="60A44CB4665EEE4133500001",SYMLINK="usb%n",OWNER="root",GROUP="root",MODE="0644

②插拔U盘等于启停服务

注意：启动服务的程序systemctl，必须使用绝对路径。

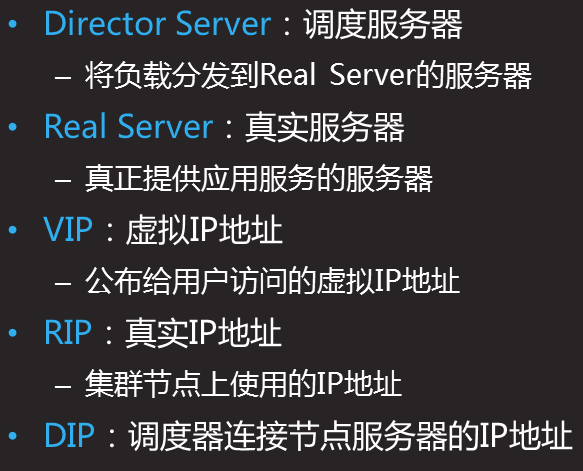
[root@proxy ~]# /etc/udev/rules.d/70-usb.rules

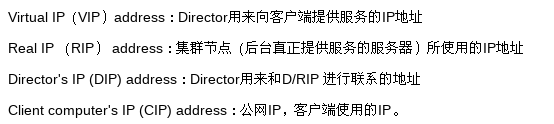
ACTION=="add",ENV{ID\_VENDOR}=="TOSHIBA",ENV{ID\_SERIAL\_SHORT}=="60A44CB4665EEE4133500001",RUN+="/usr/bin/systemctl start httpd"

ACTION=="remove",ENV{ID\_VENDOR}=="TOSHIBA",ENV{ID\_SERIAL\_SHORT}=="60A44CB4665EEE4133500001",RUN+="/usr/bin/systemctl stop httpd"

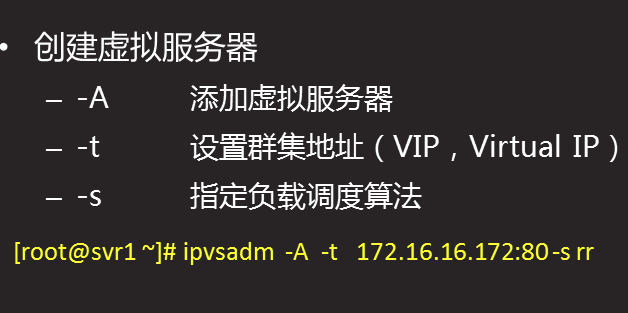
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

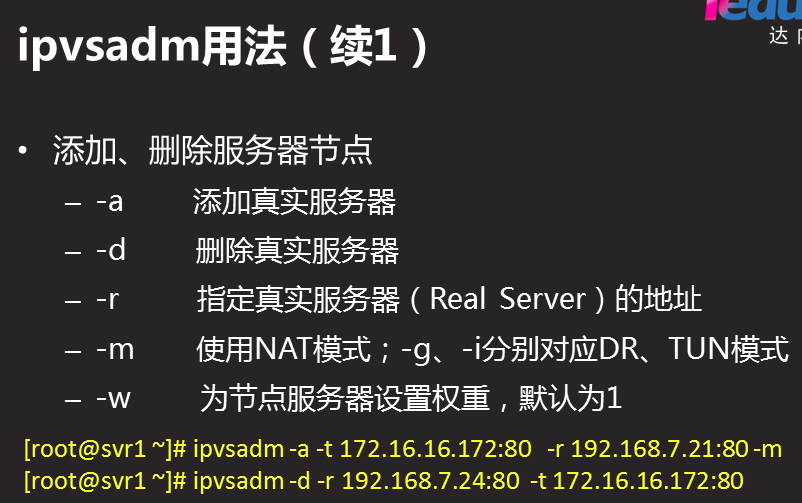
集群,LVS





ipvsadm的用法!





加权最小连接调度:

【（活动的连接数+1）/除以权重】（谁的小，挑谁）

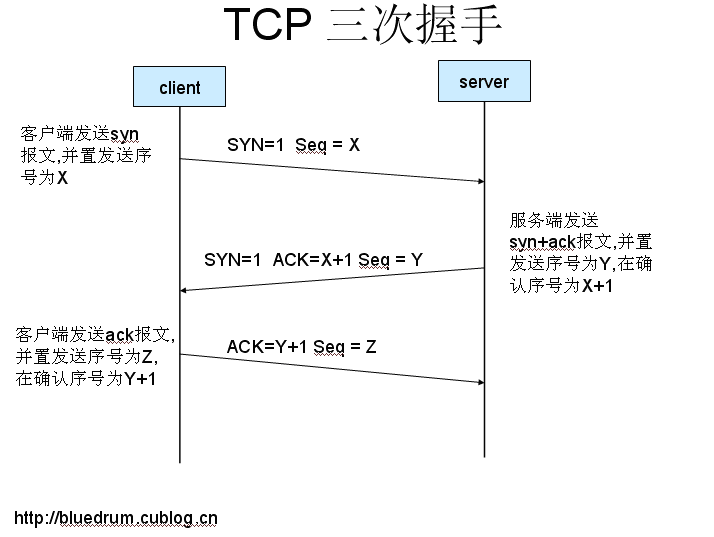
**Tomcat**

TCP三次握手四次断开

TCP三次握手

所谓三次握手(Three-way Handshake)，是指建立一个TCP连接时，需要客户端和服务器总共发送3个包。

三次握手的目的是连接服务器指定端口，建立TCP连接,并同步连接双方的序列号和确认号并交换 TCP 窗口大小信息.在socket编程中，客户端执行connect()时。将触发三次握手。



**（客户端）我要连你--》嗯，好的，请确认连接（服务端）--》连吧！（客户端）**

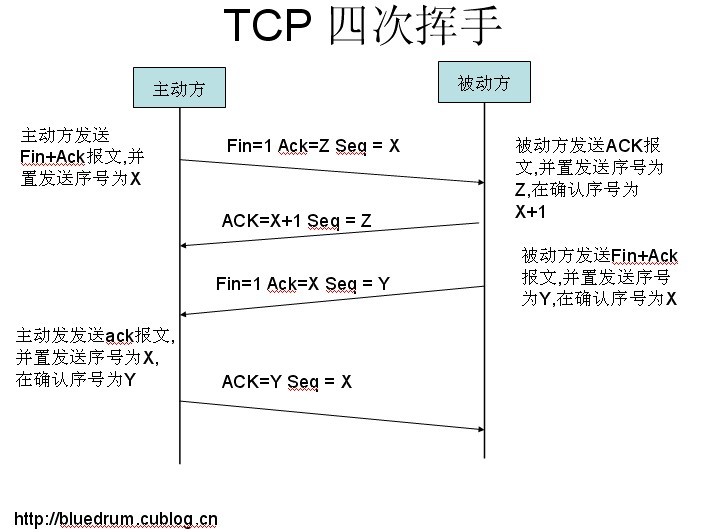
(1) 第一次握手：建立连接时，客户端A发送SYN包(SYN=j)到服务器B，并进入SYN\_SEND状态，等待服务器B确认。

(2) 第二次握手：服务器B收到SYN包，必须确认客户A的SYN(ACK=j+1)，同时自己也发送一个SYN包(SYN=k)，即SYN+ACK包，此时服务器B进入SYN\_RECV状态。

(3) 第三次握手：客户端A收到服务器B的SYN＋ACK包，向服务器B发送确认包ACK(ACK=k+1)，此包发送完毕，客户端A和服务器B进入ESTABLISHED状态，完成三次握手。

完成三次握手，客户端与服务器开始传送数据。

**TCP 四次挥手**  
TCP的连接的拆除需要发送四个包，因此称为四次挥手(four-way handshake)。客户端或服务器均可主动发起挥手动作，在socket编程中，任何一方执行close()操作即可产生挥手操作。



由于TCP连接是**全双工的，因此每个方向都必须单独进行关闭**。（所以会有四次，而不是对应的三次）这个原则是当一方完成它的数据发送任务后就能发送一个FIN来终止这个方向的连接。收到一个 FIN只意味着这一方向上没有数据流动，一个TCP连接在收到一个FIN后仍能发送数据。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。

**（主动方）完成了，断开吧--》（被动方）嗯，请确认要断开---》（主动方）我确认**

**被动方同时要自动断开自己的 3+1=4**

（1）客户端A发送一个FIN，用来关闭客户A到服务器B的数据传送(报文段4)。

（2）服务器B收到这个FIN，它发回一个ACK，确认序号为收到的序号加1(报文段5)。和SYN一样，一个FIN将占用一个序号。

（3）服务器B关闭与客户端A的连接，发送一个FIN给客户端A(报文段6)。

（4）客户端A发回ACK报文确认，并将确认序号设置为收到序号加1(报文段7)。