#########################################

环境准备:正确配置IP地址,主机名,并且为每台主机配置yum源,不需要配置网关与DNS

|  |  |
| --- | --- |
| **主机名** | **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 |

**II-Nginx-Day01**

web服务器对比

Unix和Linux平台下:

Apache Nginx Tengine Lighttpd

Tomcat IBM WebSphere Jboss

Windows平台下:

微软公司的IIS(Internet Information Server)

Nginx简介

Nginx(“engine x”): 是俄罗斯编写的十分轻量级的HTTP服务器; 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; 官方网站: <http://nginx.org/>

**案例1：搭建Nginx服务器**

在IP地址为192.168.4.5的主机上安装部署Nginx服务，并可以将Nginx服务器，要求编译时启用如下功能：

1) 支持SSL加密功能

2) 设置Nginx账户及组名称均为nginx

3) Nginx服务器升级到更高版本。

然后客户端访问页面验证Nginx Web服务器：

1) 使用火狐浏览器访问

2) 使用curl访问

安装nginx-1.10.3版本时，需要使用如下参数：

--with-http\_ssl\_module：提供SSL加密功能

--user：指定账户

--group：指定组

支持加密

支持扩展正则

步骤一: 构建Nginx服务器

1) 使用源码包安装nginx软件包

[root@proxy ~]# yum -y install gcc pcre-devel openssl-devel #安装依赖包

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

[root@proxy ~]# tar -xf nginx-1.10.3.tar.gz

[root@proxy ~]# cd nginx-1.10.3/

[root@proxy nginx-1.10.3]# ./configure --prefix=/usr/local/nginx --user=nginx --group=nginx --with-http\_ssl\_module #指定安装路径;指定用户;指定组;开启SSL加密功能

[root@proxy nginx-1.10.3]# make && make install #编译并安装

1. nginx命令的用法

[root@proxy ~]# /usr/local/nginx/sbin/nginx #启动服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s stop #关闭服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload #重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -V #查看软件信息

[root@proxy ~]# ln -s /usr/local/nginx/sbin/nginx /sbin/ #创建软连接,方便以后使用

netstat命令可以查看系统中启动的端口信息，该命令常用选项如下：

-a显示所有端口的信息

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

nginx服务默认通过TCP 80端口监听客户端请求：

[root@proxy ~]# netstat -anptu | grep nginx

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 5852/nginx: master

1. 设置防火墙与SELinux(非必须的操作,如果有则关闭)

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

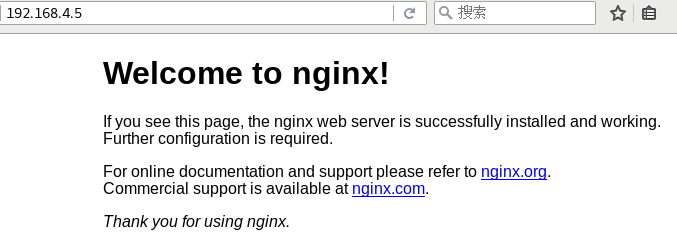
[root@proxy ~]# setenforce 0

1. 测试首页文件

Nginx Web服务默认首页文档存储目录为/usr/local/nginx/html/，在此目录下默认有一个名为index.html的文件，使用客户端访问测试页面：

[root@client ~]# curl <http://192.168.4.5> #客户机client测试

[root@client ~]# firefox 192.168.4.5



步骤二: 升级Nginx服务器

1. 编译新版本nginx软件

[root@proxy ~]# tar -xf nginx-1.12.2.tar.gz

[root@proxy ~]# cd nginx-1.12.2/

[root@proxy nginx-1.12.2]# ./configure --user=nginx --group=nginx --with-http\_ssl\_module

[root@proxy nginx-1.12.2]# make

1. 备份老的nginx主程序,并使用编译好的新版本nginx替换老版本

[root@proxy nginx-1.12.2]# mv /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginx.bak #备份老版本

[root@proxy nginx-1.12.2]# cp /root/nginx-1.12.2/objs/nginx /usr/local/nginx/sbin/ #拷贝新版本

[root@proxy nginx-1.12.2]# ls /usr/local/nginx/sbin/nginx

nginx nginx.bak

升级=make upgrade

不建议用make upgrade

[root@proxy nginx-1.12.2]# killall nginx

[root@proxy nginx-1.12.2]# nginx

[root@proxy nginx-1.12.2]# nginx -V #查看版本

nginx version: nginx/1.12.2

步骤三: 客户端访问测试

分别用使用浏览器和命令行工具curl测试服务器页面

[root@client ~]# curl <http://192.168.4.5>

[root@client ~]# firefox 192.168.4.5

模板配置文件框架如下：

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

全局配置(用户名,日志,进程)

http{

server{

listen 80;

server\_name localhost;

root html;

}

server{

listen 80;

server\_name www.xyz.com;

root www;

}

}

**案例2：用户认证**

沿用案例一，通过调整Nginx服务端配置，实现以下目标：

1) 访问Web页面需要进行用户认证

2) 用户名为：tom，密码为：123456

通过Nginx实现Web页面的认证，需要修改Nginx配置文件，在配置文件中添加auth语句实现用户认证。最后使用htpasswd命令创建用户及密码即可

步骤一:

1) 修改/usr/local/nginx/conf/nginx.conf

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

......

auth\_basic "Input Password"; #认证提示符

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

......

2) 生成密码文件,创建用户及密码

使用htpasswd命令创建账户文件，需要确保系统中已经安装了httpd-tools

[root@proxy nginx]# yum -y install httpd-tools

[root@proxy nginx]# htpasswd -c /usr/local/nginx/pass tom #创建密码文件,需要跟配置文件里的密码文件对应,不然会报错(-c = create)

New password:

Re-type new password:

Adding password for user tom

[root@proxy nginx]# cat /usr/local/nginx/pass

tom:$apr1$x2QneEDx$29iDl.VLWqpU.783x9bPB.

[root@proxy nginx]# htpasswd /usr/local/nginx/pass jerry #追加用户,不使用-c, 不然会覆盖tom密码文件

New password:

Re-type new password:

Adding password for user jerry

[root@proxy nginx]# cat /usr/local/nginx/pass #确认用户名与密码

tom:$apr1$x2QneEDx$29iDl.VLWqpU.783x9bPB.

jerry:$apr1$dKIr2a6v$7.0gt5pehsb1usrd33vj20

3) 重启nginx服务

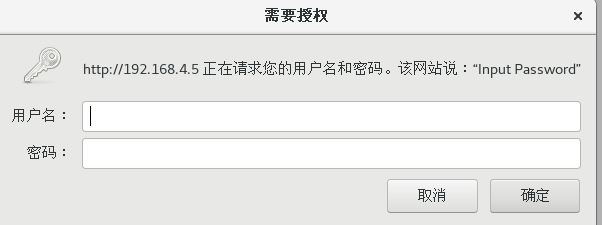
[root@proxy nginx]# nginx -s reload #重新加载配置文件

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

步骤二: 客户端测试

[root@client ~]# firefox http://192.168.4.5



**案例3：基于域名的虚拟主机**

沿用案例二，配置基于域名的虚拟主机，实现以下目标：

1) 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

2) 对域名为www.a.com的站点进行用户认证，用户名称为tom，密码为123456

步骤一: 修改配置文件

1. 修改nginx服务配置,添加相关虚拟主机配置

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

......

server {

listen 80; #端口

server\_name [www.a.com;](http://www.a.com;) #域名

auth\_basic "Input Password";

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

.......

server {

listen 80;

server\_name [www.b.com;](http://www.b.com;)

charset utf-8; #支持世界上所有语言,不然网站内容只能写英文

location / {

root www; #指定网站根路径

index index.html index.htm;

}

}

......

1. 创建网站根目录及对应首页文件

[root@proxy nginx]# mkdir /usr/local/nginx/www

[root@proxy nginx]# echo "<h1>今天是学习Nginx的第一天" > /usr/local/nginx/www/index.html

3) 重启nginx服务

[root@proxy nginx]# nginx -s reload

步骤二: 客户端测试

1. 修改客户端主机192.168.4.10的/etc/hosts文件,进行域名解析

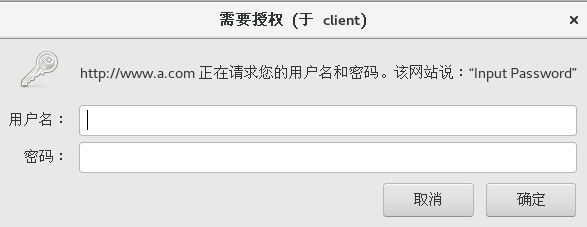
[root@client ~]# vim /etc/hosts

192.168.4.5 www.a.com www.b.com

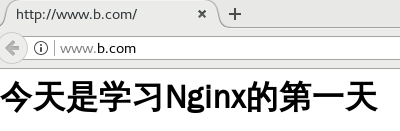
1. 登录192.168.4.10客户端主机进行测试

注意：请先关闭真实机的firefox，再SSH -X远程连接调用虚拟机的firefox

[root@client ~]# firefox <http://www.a.com>



[root@client ~]# firefox http://www.b.com



**扩展课外实验:其他类型的虚拟主机**

1 基于端口的虚拟主机

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

......

server {

listen 8080;

server\_name www.a.com;

auth\_basic "Input Password";

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

......

server {

listen 8000;

server\_name www.b.com;

charset utf-8;

location / {

root www;

index index.html index.htm;

}

}

......

[root@proxy ~]# nginx -s reload

[root@client ~]# firefox http://www.a.com:8080

[root@client ~]# firefox http://www.b.com:8000

2 基于IP地址的虚拟主机

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

......

server {

listen 192.168.4.5:80;

server\_name www.a.com;

auth\_basic "Input Password";

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

......

server {

listen 192.168.2.5:80;

server\_name www.b.com;

charset utf-8;

location / {

root www;

index index.html index.htm;

}

}

.......

[root@proxy ~]# nginx -s reload

[root@room9pc01 ~]# firefox 192.168.4.5

[root@room9pc01 ~]# firefox 192.168.2.5

案例4：SSL虚拟主机

沿用案例三，配置基于加密网站的虚拟主机，实现以下目标：

1) 域名为www.c.com

2) 该站点通过https访问

3) 通过私钥、证书对该站点所有数据加密

源码安装Nginx时必须使用--with-http\_ssl\_module参数，启用加密模块，对于需要进行SSL加密处理的站点添加ssl相关指令（设置网站需要的私钥和证书）。

加密算法一般分为对称算法、非对称算法、信息摘要。

对称算法有：AES、DES，主要应用在单机数据加密。 123<--->123

非对称算法有：RSA、DSA，主要应用在网络数据加密。私钥(解密) 公钥(加密)

信息摘要：MD5、sha256，sha512主要应用在数据完整性校验。

[root@proxy opt]# echo "今天学习了nginx" > /opt/a.txt

[root@proxy opt]# md5sum /opt/a.txt

5aa8da1ff714c87f67aebd6d06097458 /opt/a.txt

[root@proxy opt]# echo "明天继续学习nginx" >> /opt/a.txt

[root@proxy opt]# md5sum /opt/a.txt

44481baae4c5462063d3850d2ba96a29 /opt/a.txt

[root@proxy opt]# md5sum /opt/a.txt

44481baae4c5462063d3850d2ba96a29 /opt/a.txt

注释: ASCII跟权限,文件名称,文件类型等没有关系,只跟文件内容有关,只要文件内容不变,ASCII码永远不会变,sha256、sha512用法一样,就是ASCII码长度不一样

[root@proxy opt]# sha256sum a.txt

2f26f63d5dbb198d186d87e7d1535297b4c8085421d73a34bc98a1da05264d73 a.txt

#! /bin/bash #! /bin/bash

for i in `ls /etc/\*.conf` for i in `ls /etc/\*.conf`

do do

md5sum $i >> log1 md5sum $i >> log2

done done

步骤一: 配置SSL虚拟主机

1. 生成私钥与证书

[root@proxy ~]# cd /usr/local/nginx/conf/ #私钥和证书(公钥)必须在此目录

[root@proxy conf]# openssl genrsa > cert.key #生成私钥

[root@proxy conf]# openssl req -new -x509 -key cert.key > cert.pem #生成证书

......

Country Name (2 letter code) [XX]:CN #国家有要求,以下问题随便写

State or Province Name (full name) []:zj

Locality Name (eg, city) [Default City]:hz

Organization Name (eg, company) [Default Company Ltd]:abc

Organizational Unit Name (eg, section) []:abc

Common Name (eg, your name or your server's hostname) []:abc

Email Address []:abc

1. 修改Nginx配置文件,设置加密网站的虚拟主机

[root@proxy conf]# vim nginx.conf

.......

server {

listen 443 ssl;

server\_name www.c.com;

ssl\_certificate cert.pem;

ssl\_certificate\_key cert.key;

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout 5m;

ssl\_ciphers HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

location / {

root html;

index index.html index.htm;

}

}

}

1. 重启nginx服务

[root@proxy conf]# nginx

[root@proxy conf]# nginx -s reload

步骤二: 客户端验证

1. 修改客户端主机192.168.4.10的/etc/hosts文件，进行域名解析

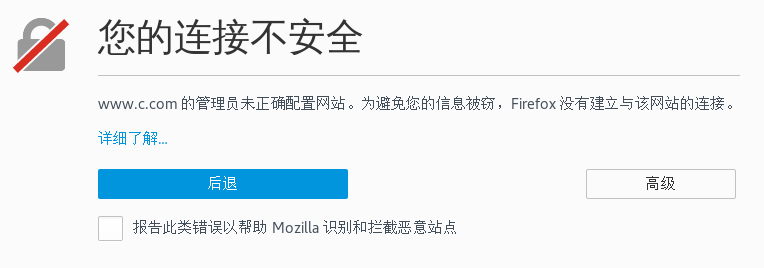
[root@client ~]# vim /etc/hosts

.......

192.168.4.5 www.a.com www.b.com www.c.com

2）登录192.168.4.10客户端主机进行测试

[root@client ~]# firefox <https://www.c.com> #信任证书后可以访问





#########################################

**II-Nginx-Day02**

**LNMP基础知识**

L: Linux操作系统

N: Nginx网站服务软件

M: MySQL、MariaDB数据库

P: 网站开发语言(PHP、Perl、Python)

静态网站:

动态网站:每次访问,代码需要在服务器运行一遍

LNMP（Linux、Nginx、MySQL、PHP）

在RHEL7系统中，源码安装Nginx，使用RPM包安装MariaDB、PHP、PHP-FPM软件。

操作过程中需要安装的软件列表如下：

nginx

mariadb、mariadb-server、mariadb-devel

php、php-fpm、php-mysql

**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;端口号为9000）、php-mysql（PHP的数据库扩展包）。

**案例1：部署LNMP环境**

安装部署Nginx、MariaDB、PHP环境

1) 安装部署Nginx、MariaDB、PHP、PHP-FPM；

2) 启动Nginx、MariaDB、FPM服务；

3) 并测试LNMP是否工作正常。

步骤一: 安装软件

1) Nginx前面课程已经装过,这里不再装包

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

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

[root@proxy ~]# cd /root/lnmp\_soft/

[root@proxy lnmp\_soft]# yum -y install php-fpm-5.4.16-42.el7.x86\_64.rpm #php-fpm包不在yum光盘中,所以需要切到lnmp\_soft目录下,再装包

步骤二: 启动服务

1) 启动Nginx服务（已经启动nginx，可以忽略这一步）

这里需要注意的是，如果服务器上已经启动了其他监听80端口的服务软件（如httpd），则需要先关闭该服务，否则会出现冲突。

[root@proxy ~]# systemctl stop httpd     #如果该服务存在则关闭该服务

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

2）启动MySQL服务

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

[root@proxy ~]# systemctl status mariadb #查看服务状态

[root@proxy ~]# systemctl enable mariadb #设置开机启动

3）启动PHP-FPM服务

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

[root@proxy ~]# systemctl status php-fpm #查看服务状态

[root@proxy ~]# systemctl enable php-fpm #设置开机启动

4）设置防火墙与SELinux（非必需操作，如果有则关闭）

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

[root@proxy ~]# setenforce 0

步骤三:测试lnmp

[root@proxy ~]# vim /usr/local/nginx/html/test.php

<?php

$i=33; #注意命令行分号结尾和赋值(跟shell的区别)

echo $i;

echo "\n"; #换行

?>

[root@proxy ~]# php /usr/local/nginx/html/test.php

33

[root@client ~]# firefox 192.168.4.5/test.php



#显示的是脚本内容,而不是脚本执行后的结果(即不支持动态网站)

nginx动静分离(默认请求什么,nginx直接返回什么)

如果是静态,直接返回

如果是动态,转发给9000

php-fpm是多进程的程序,一个PHP-FPM解释器将消耗约25M的内存。

程序 ----(硬盘)

进程 ----(内存)

线程 ----(共享进程的内存) 进程死了,进程下面的线程也挂

location : 匹配用户的地址栏(匹配域名后的地址栏)

server {

listen 80;

server\_name [www.baidu.com](http://www.baidu.com)

location / { #默认匹配所有,优先级最低

allow all;

}

location /test {

deny all;

}

location /abc{

allow 1.1.1.1;

deny all;

}

}

[root@proxy ~]#firefox www.baidu.com

[root@proxy ~]#firefox www.baidu.com/test

[root@proxy ~]#firefox www.baidu.com/abc

[root@proxy ~]#firefox [www.baidu.com/1234](http://www.baidu.com/1234) --->访问的是www.baidu.com

**案例2：构建LNMP平台**

沿用练习一，通过调整Nginx服务端配置，实现以下目标：

1) 配置Fast-CGI支持PHP网页

2) 创建PHP测试页面，测试使用PHP连接数据库的效果

步骤一: php-fpm配置文件

1. 查看php-fpm配置文件/etc/php-fpm.d/www.conf(实验中不需要修改该文件)

[root@proxy ~]# vim /etc/php-fpm.d/www.conf

......

[www]

listen = 127.0.0.1:9000            #PHP端口号

pm.max\_children = 32              #最大进程数量

pm.start\_servers = 15               #最小进程数量

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

pm.max\_spare\_servers = 32       #最多允许几个进程处于空闲状态

.......

步骤二:修改nginx配置文件并启动服务

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

......

正则表达式

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;

include fastcgi.conf;

}

.......

[root@proxy ~]# nginx -s reload

步骤三:创建php页面,测试LNMP架构能否解析PHP页面

1) 创建PHP测试页面1,可以参考lnmp\_soft/php\_scripts/test.php

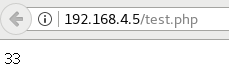
[root@proxy ~]# cp /root/lnmp\_soft/php\_scripts/test.php /usr/local/nginx/html/test1.php

2）创建PHP测试页面,连接并查询MariaDB数据库。可以参考lnmp\_soft/php\_scripts/mysql.php

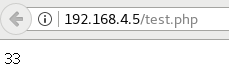
[root@proxy ~]# cp /root/lnmp\_soft/php\_scripts/mysql.php /usr/local/nginx/html/

3）客户端使用浏览器访问服务器PHP首页文档，检验是否成功：

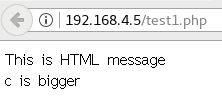
[root@client ~]# firefox 192.168.4.5/test.php



[root@client ~]# firefox [www.a.com/test.php](http://www.a.com/test.php)



[root@client ~]# firefox 192.168.4.5/test1.php



[root@client ~]# firefox www.a.com/mysql.php



4）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。

**案例3：地址重写(地址栏被重写)**

沿用练习二，通过调整Nginx服务端配置，实现以下目标：

1) 所有访问a.html的请求，重定向到b.html;

2) 所有访问192.168.4.5的请求重定向至www.tmooc.cn；

3) 所有访问192.168.4.5/下面子页面，重定向至www.tmooc.cn/下相同的页面；

4) 实现firefox与curl访问相同页面文件，返回不同的内容。

关于Nginx服务器的地址重写，主要用到的配置参数是rewrite：

rewrite regex replacement flag

rewrite 旧地址 新地址 [选项]

支持正则表达式

步骤一:

修改配置文件(访问a.html重定向到b.html)

1. 修改nginx服务配置

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

......

rewrite /a.html /b.html; #写在server里面

.......

[root@proxy ~]# echo "<h1>BBB" > /usr/local/nginx/html/b.html

1. 重新加载配置文件

[root@proxy html]# nginx -s reload

1. 客户端测试

[root@client ~]# firefox 192.168.4.5/a.html



步骤二: 访问a.html重定向到b.html(跳转地址栏)

1. 修改Nginx服务配置

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

......

rewrite /a.html /b.html redirect;

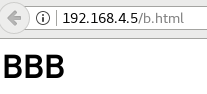
......

1. 重新加载配置文件

[root@proxy ~]# nginx -s reload

1. 客户端测试(仔细观察浏览器地址栏的变化)

[root@client ~]# firefox 192.168.4.5/a.html

 #地址栏地址被重写了

步骤三: 修改配置文件(访问192.168.4.5的请求重定向至www.tmooc.cn)

1. 修改nginx服务配置

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

......

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

......

1. 重新加载配置文件

[root@proxy ~]# nginx -s reload

1. 客户端测试(真实机测试,真实机才可以连接tmooc)

[root@room9pc01 ~]# firefox 192.168.4.5 #访问成功,出现tmooc页面

步骤四: 修稿配置文件(访问192.168.4.5/下面子页面,重定向至www.tmooc.cn/下相同的页面)

1. 修改Nginx服务配置

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

......

rewrite ^/(.\*)$ http://www.tmooc.cn/$1;

......

粘贴:(perl的正则)

1. 重新加载配置文件

[root@proxy ~]# nginx -s reload

3) 客户端测试(真实机测试,真实机才可以连接tmooc)

[root@room9pc01 ~]# firefox 192.168.4.5/vip #访问成功

[root@room9pc01 ~]# firefox 192.168.4.5/xxxx #页面未找到

步骤五：修改配置文件(实现curl和火狐访问相同链接返回的页面不同)

1) 创建网页目录以及对应的页面文件：

[root@proxy ~]# echo "I am Normal page" > /usr/local/nginx/html/test.html

[root@proxy ~]# mkdir -p /usr/local/nginx/html/firefox

[root@proxy ~]# echo "firefox page" > /usr/local/nginx/html/firefox/test.html

1. 修改Nginx服务配置

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

......

~符号代表正则匹配，\*符号代表不区分大小写

server {

listen 80;

server\_name www.a.com;

if ($http\_user\_agent ~\* firefox) { #识别客户端firefox浏览器

rewrite /(.\*) /firefox/$1;

}

......

3）重新加载配置文件

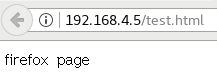
[root@proxy ~]# nginx -s reload #确保nginx是开启状态,不然会报错

4）客户端测试

[root@client ~]# curl http://192.168.4.5/test.html

I am Normal page

[root@client ~]# firefox http://192.168.4.5/test.html



5）地址重写格式【总结】

rewrite 旧地址 新地址 [选项];

1) last 不再读其他rewrite(匹配及停止)

2) break 不再读其他语句，结束请求

3) redirect 临时重定向

4) permanent 永久重定向

#########################################

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

# '$status $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

日志内容格式对应关系

192.168.4.254 - tom [18/Jan/2019:19:18:37 +0800] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x86\_64; rv:52.0) Gecko/20100101 Firefox/52.0"

#########################################

**II-Nginx-Day03**

**nginx调度算法**

nginx目前支持的调度算法

1. 轮询(默认的): 逐一循环调度
2. weight: 指定轮询几率,权重值和访问比率成正比
3. ip\_hash: 根据客户端ip分配固定的后端服务器

**服务器组主机状态**

状态类型:

1. down: 表示当前server暂时不参与负载
2. max\_fails: 允许请求失败的次数(默认为1)
3. fail\_timeout: max\_fails次失败后,暂时提供服务的时间

**案例1：Nginx反向代理**

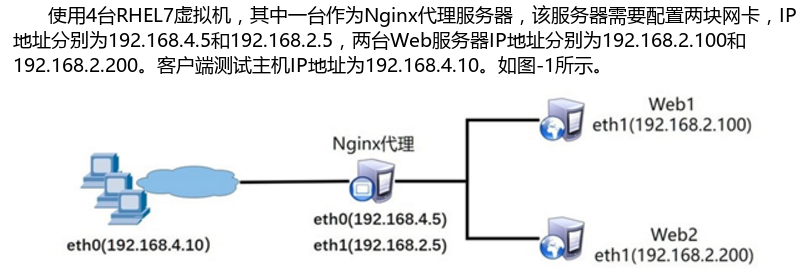
使用Nginx实现Web反向代理功能，实现如下功能：

1) 后端Web服务器两台，可以使用httpd实现

2) Nginx采用轮询的方式调用后端Web服务器

3) 两台Web服务器的权重要求设置为不同的值

4) 最大失败次数为1，失败超时时间为30秒



步骤一: 部署实施后端web服务器

1. 部署后端web1服务器

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.2.100" > /var/www/html/index.html

[root@web1 ~]# systemctl restart httpd

[root@web1 ~]# firewall-cmd --set-default-zone=trusted

[root@web1 ~]# setenforce 0

1. 部署后端web2服务器

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "192.168.2.200" > /var/www/html/index.html

[root@web2 ~]# systemctl restart httpd

[root@web2 ~]# firewall-cmd --set-default-zone=trusted

[root@web2 ~]# setenforce 0

步骤二: 配置nginx服务器,添加服务器池,实现反向代理功能

1) 修改/usr/local/nginx/conf/nginx.conf配置文件

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

......

#gzip on;

upstream webserver { #使用upstream定义后端服务器集群,集群名称任意(如webserver)

server 192.168.2.100:80; #使用server定义集群中的具体服务器和端口

server 192.168.2.200:80;

}

server {

listen 80;

server\_name www.a.com;

......

location / {

proxy\_pass <http://webserver;> #通过proxy\_pass将用户的请求转发给webserver集群

root html;

index index.html index.htm;

......

1. 重启nginx服务

[root@proxy ~]# nginx -s reload

1. 客户端使用浏览器访问代理服务器测试轮询效果

[root@client ~]# curl http://192.168.4.5

192.168.2.100

[root@client ~]# curl http://192.168.4.5

192.168.2.200

步骤二: 配置upstream服务器集群池属性

1. 设置失败次数,超时时间,权重

weight可以设置后台服务器的权重，max\_fails可以设置后台服务器的失败次数，fail\_timeout可以设置后台服务器的失败超时时间

#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

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

......

#gzip on;

upstream webserver {

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

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

#server 192.168.2.101 down;

}

server {

listen 80;

......

1. 重启nginx服务

[root@proxy ~]# nginx -s reload

1. 关闭一台后端服务器(如web1)

[root@web1 ~]# systemctl stop httpd

1. 客户端使用浏览器访问代理服务器测试轮询效果

[root@client ~]# curl <http://192.168.4.5> #返回值一直都是200

192.168.2.200

1. 再次启动后端服务器的httpd(如web1)

[root@web1 ~]# systemctl start httpd

1. 客户端再次使用浏览器访问代理服务器测试轮询效果

[root@client ~]# curl http://192.168.4.5

192.168.2.200

[root@client ~]# curl http://192.168.4.5

192.168.2.200 #30秒之后web1服务器加入工作

[root@client ~]# curl http://192.168.4.5

192.168.2.100

步骤三: 配置upstream服务器集群的调度算法

1. 设置相同客户端访问相同web服务器(ip\_hash)

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

......

upstream webserver {

ip\_hash; #注意放置的位置,不能放在web1或web2后面

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

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

}

......

1. 重启nginx服务

[root@proxy ~]# nginx -s reload

1. 客户端使用浏览器访问代理服务器测试轮询效果

[root@client ~]# curl http://192.168.4.5

192.168.2.200

[root@client ~]# curl <http://192.168.4.5> #返回值一直是200

192.168.2.200

[root@web2 ~]# systemctl stop httpd

[root@client ~]# curl <http://192.168.4.5> #返回值变成100

192.168.2.100

[root@client ~]# curl http://192.168.4.5

192.168.2.100

[root@web2 ~]# systemctl start httpd

[root@client ~]# curl <http://192.168.4.5> #返回值重新变成200

192.168.2.200

[root@client ~]# curl http://192.168.4.5

192.168.2.200

ip\_hash工作原理:

[root@proxy ~]# echo "192.168.4.10" | md5sum

95e033fe68643fd1e4c11190191baead -

ip | md5sum 　　16进制数％２＝［０，１］

后端服务器的台数

7层代理,7层调度(http)

4层代理,4层调度(TCP,UDP)

**案例2：Nginx的TCP/UDP调度器**

使用Nginx实现TCP/UDP调度器功能，实现如下功能：

1) 后端SSH服务器两台

2) Nginx编译安装时需要使用--with-stream，开启

ngx\_stream\_core\_module模块

3) Nginx采用轮询的方式调用后端SSH服务器

步骤一: 部署支持4层TCP/UDP代理的nginx服务器

1. 部署安装nginx服务器

编译安装必须要使用--with-stream参数开启4层代理模块

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

[root@proxy ~]# tar -xf nginx-1.12.2.tar.gz

[root@proxy ~]# cd /root/nginx-1.12.2/

[root@proxy nginx-1.12.2]# ./configure --user=nginx --group=nginx --with-http\_ssl\_module --with-stream

[root@proxy nginx-1.12.2]# make && make install #编译并安装

步骤二: 配置nginx服务器,添加服务器池,实现TCP/UDP反向代理功能

1. 修改/usr/local/nginx/conf/nginx.conf配置文件

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

......

stream{

upstream backend {

server 192.168.2.100:80; #后端ssh服务器的ip和端口

server 192.168.2.200:80;

}

server {

listen 12345; #nginx监听的端口(端口不能跟其他端口重复)

# proxy\_connect\_timeout 1s;

         # proxy\_timeout 3s;

proxy\_pass backend;

}

} #不能写在http里面

http {

......

1. 重启nginx服务

[root@proxy ~]# nginx -s reload

1. 客户端使用访问代理服务器测试轮询效果

[root@client ~]# ssh -X root@192.168.4.5

[root@web1 ~]#

[root@web1 ~]# exit

登出

Connection to 192.168.4.5 closed.

[root@client ~]# ssh -X root@192.168.4.5

root@192.168.4.5's password:

Last login: Mon Jan 21 10:48:39 2019 from 192.168.2.254

[root@web2 ~]#

**案例3：Nginx常见问题处理**

本案例要求对Nginx服务器进行适当优化，解决如下问题，以提升服务器的处理性能：

1) 如何自定义返回给客户端的404错误页面

2) 如何查看服务器状态信息

3) 如果客户端访问服务器提示“Too many open files”如何解决

4) 如何解决客户端访问头部信息过长的问题

5) 如何让客户端浏览器缓存数据

6) 日志切割

7) 开启gzip压缩功能，提高数据传输效率

8) 开启文件缓存功能

然后客户机访问此Web服务器验证效果：

1) 使用ab压力测试软件测试并发量

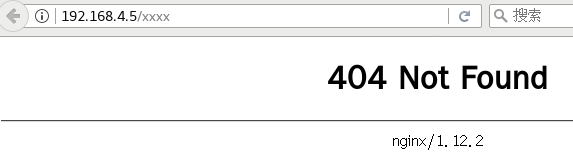
2) 编写测试脚本生成长头部信息的访问请求

3) 客户端访问不存在的页面，测试404错误页面是否重定向

步骤一：自定义报错页面

1) 优化前,客户端使用浏览器访问不存在的页面,会提示404文件未找到

[root@client ~]# firefox 192.168.4.5/xxxx



1. 修改nginx配置文件,自定义报错页面

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

......

charset utf-8; #为了支持中文

......

error\_page 404 /404.html; #去掉注释即可

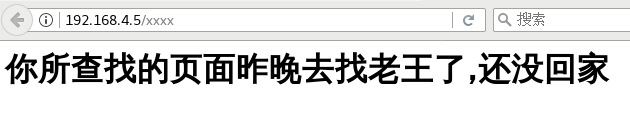
......

1. 优化后,客户端使用浏览器访问不存在的页面,会提示自己定义的40x.html页面

[root@proxy ~]# echo "<h1>你所查找的页面昨晚去找老王了,还没回家" > /usr/local/nginx/html/404.html

[root@proxy ~]# nginx -s reload

[root@client ~]# firefox 192.168.4.5/xxxx



1. 常见http状态码

|  |  |
| --- | --- |
| **状态码** | **功能描述** |
| 200 | 一切正常 |
| 301 | 永久重定向 |
| 302 | 临时重定向 |
| 400 | 请求语法错误 |
| 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 |

步骤二: 如何查看服务器状态信息(非常重要的功能)

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

[root@proxy ~]# nginx -s stop

[root@proxy ~]# rm -rf /usr/local/nginx/

[root@proxy ~]# cd nginx-1.12.2/

[root@proxy nginx-1.12.2]# ./configure --user=nginx --group=nginx --with-http\_ssl\_module --with-stream --with-http\_stub\_status\_module

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

[root@proxy nginx-1.12.2]# nginx

2）启用Nginx服务并查看监听端口状态

ss命令可以查看系统中启动的端口信息，该命令常用选项如下：

-a显示所有端口的信息

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

注意：在RHEL7系统中可以使用ss命令替代netstat命令，功能一样，选项一样。

[root@proxy ~]# ss -antulp | grep nginx

tcp LISTEN 0 128 \*:80 \*:\* users:(("nginx",pid=2200,fd=6),("nginx",pid=2199,fd=6))

[root@proxy ~]# netstat -antulp | grep nginx

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 2199/nginx: master

1. 修改nginx配置文件,定义状态页面

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

......

server {

listen 80;

server\_name localhost;

location /status{

stub\_status on;

#allow IP地址;

跟配置文件location后的名称一致

#deny IP地址;

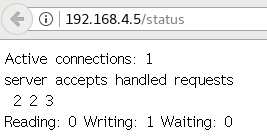
}

......

[root@proxy ~]# nginx -s reload

1. 优化后,查看状态页面信息

[root@client ~]# firefox 192.168.4.5/status



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。

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。

步骤三: 优化nginx并发量

常用压力测试工具: ab

格式: ab -c 并发数 -n 总请求数 URL #并发数不能大于总请求数

其他常见压力测试软件(需要额外下载)

http\_load webbench siege

1. 优化前使用ab高并发测试

[root@proxy ~]# ab -c 100 -n 100 <http://192.168.4.5/> #记得加/

[root@proxy ~]# ab -c 2000 -n 2000 http://192.168.4.5/

Benchmarking 192.168.4.5 (be patient)

socket: Too many open files (24) #提示打开文件数量过多

1. 修改nginx配置文件,增加并发量

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

worker\_processes 1; #与CPU核心数量一致(虚拟机为1)

......

events {

worker\_connections 65536; #每个worker最大并发连接数

#use epoll; #Nginx默认的一种工作模式,可写可不写,Apache是select

}

......

[root@proxy ~]# nginx -s reload

1. 优化Linux内核参数(最大文件数量)

[root@proxy ~]# ulimit -a | grep "open files" #查看所有属性值

open files (-n) 1024

[root@proxy ~]# ulimit -Hn 100000 #设置硬限制(临时规则)

[root@proxy ~]# ulimit -Sn 100000 #设置软限制(临时规则)

[root@proxy ~]# vim /etc/security/limits.conf #永久设置,需要重启才生效

.......

\* soft nofile 100000

\* hard nofile 100000

用户或组 硬限制或软限制 需要限制的项目 限制的值

1. 优化后测试服务器并发量(因为客户端没调内核参数,所以在proxy测试)

[root@proxy ~]# ab -c 2000 -n 2000 <http://192.168.4.5/> #瞬间完成

步骤四: 优化nginx数据包头缓存

1. 优化前,使用脚本测试长头部请求是否能获得响应

[root@proxy ~]# cat /root/lnmp\_soft/buffer.sh

#!/bin/bash

URL=http://192.168.4.5/index.html?

for i in {1..5000}

do

URL=${URL}v$i=$i

done

curl $URL

[root@proxy ~]# /root/lnmp\_soft/buffer.sh #运行脚本报错

<html>

<head><title>414 Request-URI Too Large</title></head>

<body bgcolor="white">

<center><h1>414 Request-URI Too Large</h1></center>

<hr><center>nginx/1.12.2</center>

</body>

</html>

1. 修改nginx配置文件,增加数据包头部缓存大小

[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 1M; #大请求包头部信息的缓存个数与容量

......

[root@proxy ~]# nginx -s reload

1. 优化后,使用脚本测试长头部请求是否能获得相应

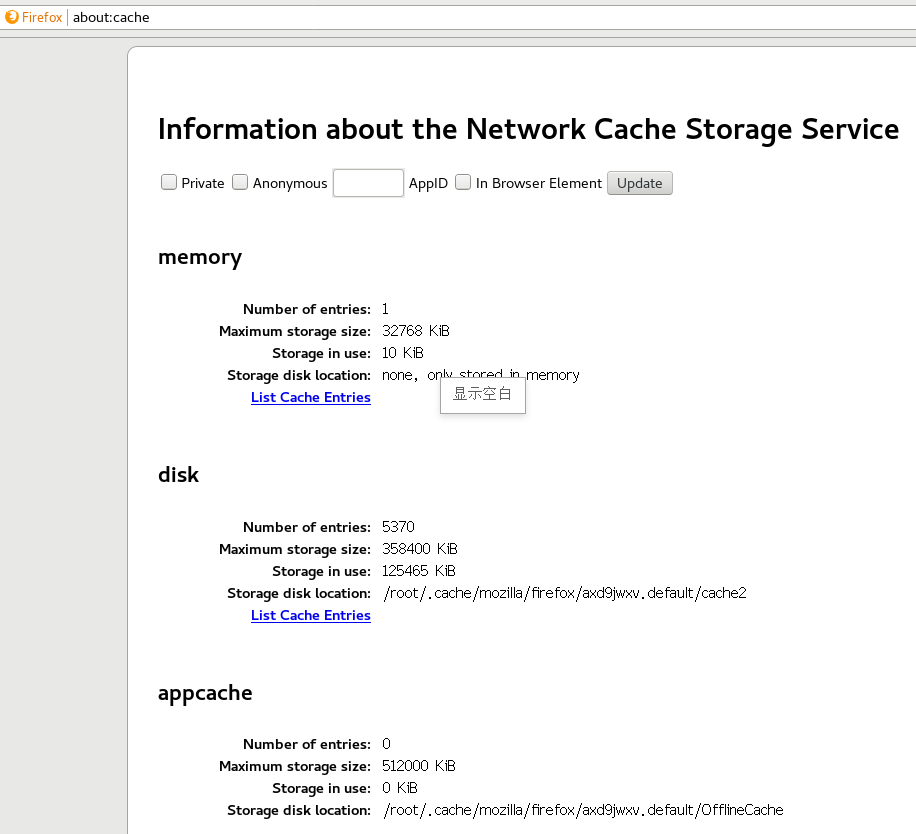
[root@proxy ~]# /root/lnmp\_soft/buffer.sh

步骤五: 浏览器本地缓存静态数据

用户浏览器缓存,适合静态数据,如图片,视频,音频等,不适合动态数据

1. 使用firefox浏览器查看缓存

以Firefox浏览器为例，在Firefox地址栏内输入about:cache将显示Firefox浏览器的缓存信息，如图所示，点击List Cache Entries可以查看详细信息。



1. 清空firefox本地缓存数据

手动清空缓存,是为了下面步骤方便验证

1. 修改Nginx配置文件,定义对静态页面的缓存时间

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

......

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

expires 30d; #定义客户端缓存时间为30天

} #写在server里面

......

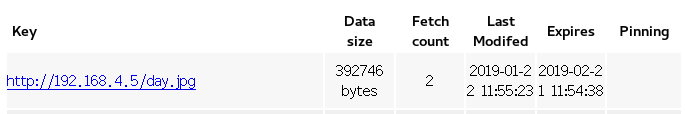
[root@proxy ~]# nginx -s reload

[root@proxy ~]# cp /usr/share/backgrounds/day.jpg /usr/local/nginx/html/

1. 优化后,使用firefox浏览器访问图片,再次查看缓存信息

[root@client ~]# firefox http://192.168.4.5/day.jpg

在firefox地址栏内输入about:cache，查看本地缓存数据，查看是否有图片以及过期时间是否正确。



步骤六: 日志切割

[root@proxy ~]# mv /usr/local/nginx/logs/access.log /usr/local/nginx/logs/access2.log #备份旧日志

1) 手动执行

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

2199

备注：/usr/local/nginx/logs/nginx.pid文件中存放的是nginx的进程PID号。

[root@proxy ~]# kill -USR1 2199

1. 自动完成

[root@proxy ~]# 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)

[root@proxy ~]# chmod +x /usr/local/nginx/logbak.sh

[root@proxy ~]# crontab -e

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

步骤七: 对页面进行压缩处理

1. 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(1~9之间,越大,压缩效果越好)

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/usr/local/nginx/conf/mime.types

.. ..

}

步骤八: 服务器内存缓存

1. 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

//关闭缓存报错功能

}

总结:

nginx web服务器

nginx代理服务器功能: 代理,调度,健康检查

7层代理:HTTP

4层代理:TCP/UDP

#########################################

**II-Day04-memcached**

SQL: Oracle MySQL SQL\_server 关系型数据库

NoSQL: memcached redis mongodb 非关系型数据库/kv数据库/缓存数据库/内存数据库

memcached数据会丢失

redis 存内存,自动同步到硬盘

memcached简介

memcached是高性能的分布式缓存服务器,用来集中缓存数据库查询结果,减少数据库访问次数,以提高动态web应用的响应速度

官方网站: http://memcached.org/

**案例1：构建memcached服务**

本案例要求先快速搭建好一台memcached服务器，并对memcached进行简单的增、删、改、查操作：

1) 安装memcached软件，并启动服务

2) 使用telnet测试memcached服务

3) 对memcached进行增、删、改、查等操作

步骤一: 构建memcached服务

1. 使用yum安装软件包memcached

[root@proxy ~]# yum -y install memcached

1. memcached配置文件(查看即可,不需要修改)

[root@proxy ~]# vim /usr/lib/systemd/system/memcached.service

......

EnvironmentFile=-/etc/sysconfig/memcached

ExecStart=/usr/bin/memcached -u $USER -p $PORT -m $CACHESIZE -c $MAXCONN $OPTIONS

......

[root@proxy ~]# vim /etc/sysconfig/memcached

PORT="11211"

USER="memcached"

MAXCONN="1024"

CACHESIZE="64"

OPTIONS=""

1. 启动服务并查看网络连接状态验证是否开启成功

[root@proxy ~]# systemctl start memcached.service

[root@proxy ~]# systemctl status memcached

[root@proxy ~]# ss -ntulp | grep 11211

步骤二: 使用telnet访问memcached服务器

1) 使用yum安装telnet

[root@proxy ~]# yum -y install telnet

2) 使用telnet连接服务器测试memcached服务器功能，包括增、删、改、查等操作。

[root@proxy ~]# telnet 192.168.2.5 11211

Trying 192.168.2.5...

Connected to 192.168.2.5.

Escape character is '^]'.

set name 0 180 3 #0表示不压缩,180为数据缓存时间,3为需要存储的数据字节数量

aaa #输入变量的值,值为aaa,如果输入的不是3字节的值,会报错

STORED

get name #获取变量的值

VALUE name 0 3 #输出结果

aaa

END

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

stats                        //查看状态

flush\_all                        //清空所有

quit                            //退出登录

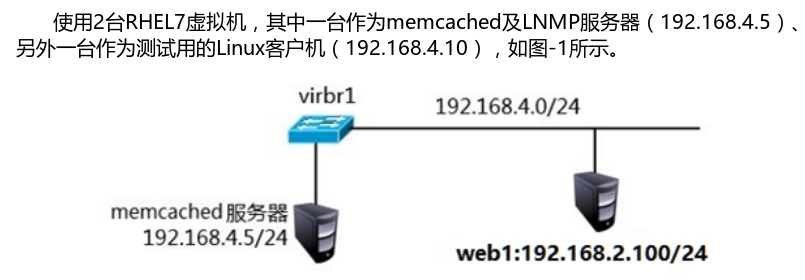
**案例2：LNMP+memcached**

沿用练习一，部署LNMP+memcached网站平台,通过PHP页面实现对memcached服务器的数据操作，实现以下目标：

1) 部署LNMP实现PHP动态网站架构

2) 为PHP安装memcache扩展

3) 创建PHP页面，并编写PHP代码，实现对memcached的数据操作



在RHEL7系统光盘中包含有我们需要的MariaDB、PHP，我们需要使用源码安装Nginx，使用RPM包安装FPM。另外如果希望使用PHP来操作memcached，注意必须要为PHP安装memcache扩展（php-pecl-memcache），否则PHP无法解析连接memcached的指令。客户端测试时需要提前安装telnet远程工具。

步骤一: 部署LNMP环境(如果环境中已经存在LNMP环境,本步骤可以忽略)

1）使用yum安装基础依赖包

[root@web1 ~]# yum -y install gcc openssl-devel pcre-devel zlib-devel

2）源码安装Nginx

[root@web1 ~]# tar -xf nginx-1.12.2.tar.gz

[root@web1 ~]# cd nginx-1.12.2

[root@web1 nginx-1.12.2]# ./configure --user=nginx --group=nginx --with-http\_ssl\_module

[root@web1 nginx-1.12.2]# make && make install

3）安装MariaDB数据库

[root@web1 ~]# yum -y install mariadb mariadb-server mariadb-devel

4）安装PHP

[root@web1 ~]# yum -y install php php-mysql

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

[root@web1 lnmp\_soft]# yum -y install php-fpm-5.4.16-42.el7.x86\_64.rpm

5）修改Nginx配置文件

[root@web1 ~]# vim /usr/local/nginx/conf/nginx.conf

......

location / {

root html;

index index.html index.htm;

}

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

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

include fastcgi.conf;

}

步骤二: 启动服务（如果所有服务已经启动，也可以忽略这一步骤）

1）启动Nginx服务

这里需要注意的是，如果服务器上已经启动了其他监听80端口的服务软件（如httpd），则需要先关闭该服务，否则会出现冲突。

[root@web1 ~]# systemctl stop httpd    //如果该服务存在，则关闭该服务

[root@web1 ~]# /usr/local/nginx/sbin/nginx

[root@web1 ~]# netstat -utnlp | grep :80

tcp    0    0 0.0.0.0:80        0.0.0.0:\*        LISTEN        32428/nginx

2）启动MySQL服务

[root@web1 ~]# systemctl start mariadb

[root@web1 ~]# systemctl status mariadb

3）启动PHP-FPM服务

[root@web1 ~]# systemctl start php-fpm

[root@web1 ~]# systemctl status php-fpm

4）关闭SELinux、防火墙

[root@web1 ~]# setenforce 0

[root@web1 ~]# firewall-cmd --set-default-zone=trusted

步骤三: 创建PHP页面,使用php语言测试memcached服务

1) 部署测试页面

创建PHP首页文档/usr/local/nginx/html/index.php，测试页面可以参考lnmp\_soft/php\_scripts/mem.php。

**注意**：192.168.2.5是memcached数据库。

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

[root@web1 php\_scripts]# cp mem.php /usr/local/nginx/html/

[root@web1 php\_scripts]# cat /usr/local/nginx/html/mem.php

<?php

$memcache=new Memcache; #创建memcache对象

$memcache->connect('192.168.2.5',11211) or die ('could not connect!! ');

$memcache->set('key', 'test'); #定义变量

$get\_values=$memcache->get('key'); #获取变量值

echo $get\_values;

?>

1. 客户端测试(结果会失败)

客户端使用浏览器访问服务器PHP首页文档，检验对memcached的操作是否成功：

[root@web1 ~]# firefox <http://192.168.2.100/mem.php> #失败

**注意**：这里因为没有给PHP安装扩展包，默认PHP无法连接memcached数据库，需要给PHP安装扩展模块才可以连接memcached数据库。

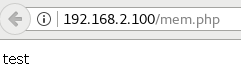
3) 为PHP添加memcache扩展

[root@web1 ~]# yum list | grep memcache

[root@web1 ~]# systemctl restart php-fpm

4) 客户端再次测试(结果会成功显示数据结果)

[root@web1 ~]# firefox http://192.168.2.100/mem.php



client----------------------->server[动态,登录]

用户名,密码---------------->touch id.txt[tom, logined]

<返回页面(包头:cookie:id号)

浏览器-------刷新访问------------>服务器

给服务器id

session文件存储的是用户的登录状态,不是储存账号和密码的

session和cookie:核心功能,防止重复登录

**案例3：PHP的本地Session信息**

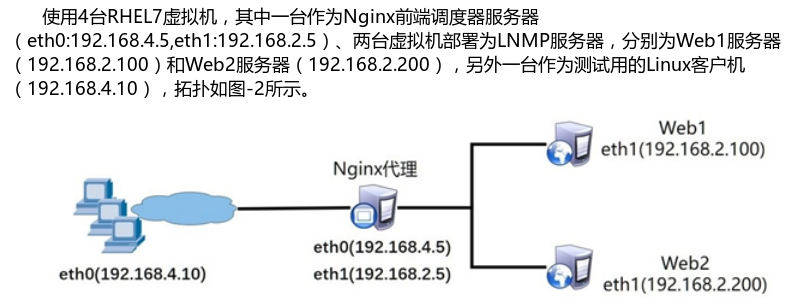
通过Nginx调度器负载后端两台Web服务器，实现以下目标：

1) 部署Nginx为前台调度服务器

2) 调度算法设置为轮询

3) 后端为两台LNMP服务器

4) 部署测试页面，查看PHP本地的Session信息



步骤一：部署后端LNMP服务器相关软件（如果已经安装完成，则忽略此步骤）

注意:以下部署LNMP服务器的操作，需要在两台后端服务器做相同的操作，下面我们以一台Web2服务器（192.168.2.200）为例，对Web1服务器执行相同操作即可。

1）使用yum安装基础依赖包

[root@web2 ~]# yum -y install gcc openssl-devel pcre-devel

2）源码安装Nginx

[root@web2 ~]# tar -xf nginx-1.12.2.tar.gz

[root@web2 ~]# cd nginx-1.12.2

[root@web2 nginx-1.12.2]# ./configure --user=nginx --group=nginx --with-http\_ssl\_module

[root@web2 nginx-1.12.2]# make && make install

3）安装MariaDB数据库

[root@web2 ~]# yum -y install mariadb mariadb-server mariadb-devel

4）安装PHP（php-fpm软件包在lnmp\_soft中有提供）

[root@web2 ~]# yum -y install php php-mysql

[root@web2 ~]# cd lnmp\_soft/

[root@web2 lnmp\_soft]# yum -y install php-fpm-5.4.16-42.el7.x86\_64.rpm

5）修改Nginx配置文件（修改默认首页与动静分离）

[root@web2 ~]# vim /usr/local/nginx/conf/nginx.conf

......

location / {

root html;

index index.html index.htm;

}

......

步骤二：启动LNMP服务器相关的服务

1）启动Nginx服务

这里需要注意的是，如果服务器上已经启动了其他监听80端口的服务软件（如httpd），则需要先关闭该服务，否则会出现冲突。

[root@web2 ~]# systemctl stop httpd    //如果该服务存在，则关闭该服务

[root@web2 ~]# /usr/local/nginx/sbin/nginx

2）启动MySQL服务

[root@web2 ~]# systemctl start mariadb

[root@web2 ~]# systemctl status mariadb

3）启动PHP-FPM服务

[root@web2 ~]# systemctl start php-fpm

[root@web2 ~]# systemctl status php-fpm

4）关闭SELinux、防火墙

[root@web2 ~]# setenforce 0

[root@web2 ~]# firewall-cmd --set-default-zone=trusted

步骤三：部署前端Nginx调度服务器(代理服务器)

1）使用源码安装nginx软件（如果Nginx软件包已存在可以忽略此步骤）

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

[root@proxy ~]# tar -xf nginx-1.12.2.tar.gz

[root@proxy ~]# cd nginx-1.12.2

[root@proxy nginx-1.12.2]# ./configure --user=nginx --group=nginx --with-http\_ssl\_module

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

2）修改Nginx配置文件

Nginx配置文件中，通过upstream定义后端服务器地址池，默认调度策略为轮询，使用proxy\_pass调用upstream定义的服务器地址池：

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

......

upstream webserver {

server 192.168.2.100:80;

server 192.168.2.200:80;

}

server {

listen 80;

server\_name localhost;

location / {

             proxy\_pass http://webserver;

root html;

index index.html index.htm;

}

}

3）重新加载配置文件

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

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

4）关闭SELinux、防火墙

[root@proxy ~]# setenforce 0

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

步骤四：测试环境是否配置成功

1）浏览器访问测试页面验证。

[root@client ~]# curl http://192.168.4.5/index.html        //查看是否有数据

步骤五: 部署测试画面

1. 部署测试页面(web1服务器)

测试页面可以参考lnmp\_soft/php\_scripts/php-memcached-demo.tar.gz

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

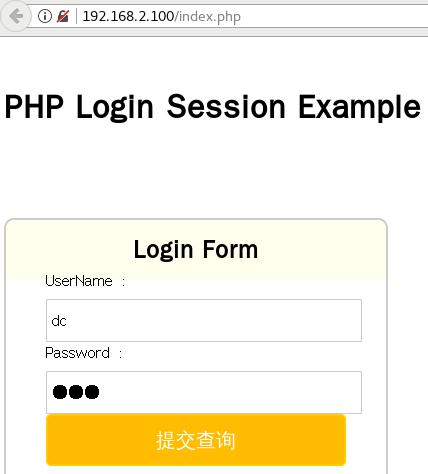
[root@web1 php\_scripts]# tar -xf php-memcached-demo.tar.gz

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

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

1. 浏览器直接访问后端服务器的测试页面(web1服务器)

[root@web1 php-memcached-demo]# firefox <http://192.168.2.100/index.php> #填写账户信息



[root@web1 php-memcached-demo]# cd /var/lib/php/session/

[root@web1 session]# ls #查看服务器本地的session信息

sess\_b6iqsu7mmcq0js8omepng3p366 #注意这里的ID是随机的

[root@web1 session]# cat sess\_b6iqsu7mmcq0js8omepng3p366

login\_user|s:2:"dc";logged\_in|s:4:"true";id|s:26:"b6iqsu7mmcq0js8omepng3p366";

注意:可以修改Index.php和home.php两个文件的内容,添加页面颜色属性,以区别后端两台不同的服务器; <body bgcolor=blue>

1. 部署测试页面(web2)

[root@web2 ~]# cd lnmp\_soft/php\_scripts/

[root@web2 php\_scripts]# ls

mem.php mysql.php php-memcached-demo.tar.gz test.php

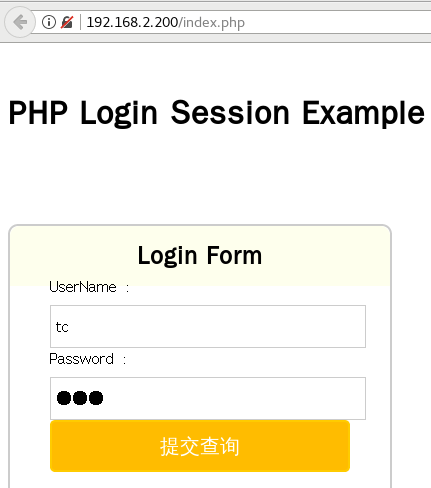
[root@web2 php\_scripts]# tar -xf php-memcached-demo.tar.gz

[root@web2 php\_scripts]# cd php-memcached-demo/

[root@web2 php-memcached-demo]# cp -r \* /usr/local/nginx/html/

1. 浏览器直接访问后端服务器的测试页面(web2)

[root@web2 php-memcached-demo]# firefox http://192.168.2.200/index.php





[root@web2 php-memcached-demo]# cd /var/lib/php/session/

[root@web2 session]# ls

sess\_ojkn637b8ffa8kjmfod97csh52

[root@web2 session]# cat sess\_ojkn637b8ffa8kjmfod97csh52

login\_user|s:2:"tc";logged\_in|s:4:"true";id|s:26:"ojkn637b8ffa8kjmfod97csh52";

注意:可以修改Index.php和home.php两个文件的内容,添加页面颜色属性,以区别后端两台不同的服务器; <body bgcolor=red>

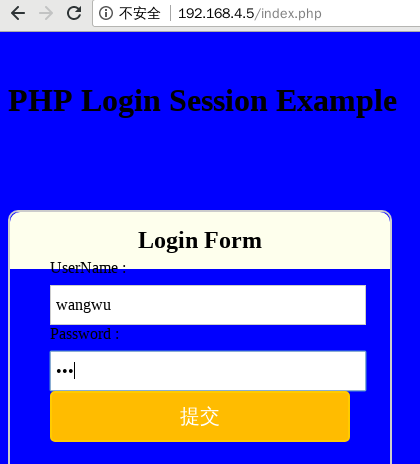
1. 浏览器访问前端调度器测试(不同后端服务器session不一致)

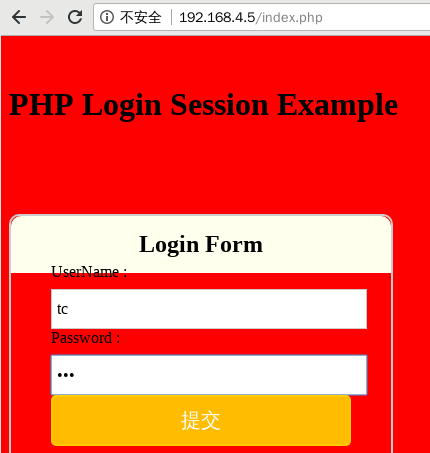
推荐使用google浏览器测试。

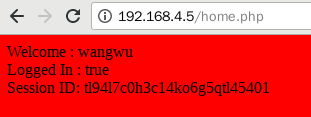
[root@client ~]# google-chrome http://192.168.4.5/index.php

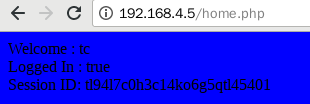
//填写注册信息后，刷新，还需要再次注册，说明两台计算机使用的是本地Session

//第二台主机并不知道你在第一台主机已经登录，第一台主机的登录信息也没有传递给第二台主机









**案例4：PHP实现session共享**

沿用练习三，通过修改PHP-FPM配置文件，实现session会话共享，本案例需要在练习三的基础上实现：

1) 配置PHP使用memcached服务器共享Session信息

2) 客户端访问两台不同的后端Web服务器时，Session 信息一致

在练习三拓扑的基础上，Nginx服务器除了承担调度器外，还需要担任memcached数据库的角色，并在两台后端LNMP服务器上实现PHP的session会话共享

步骤一: 构建memcached服务

1) 安装memcached服务(如果192.168.4.5上已经有本软件包,此步骤可以忽略)

[root@proxy ~]# yum -y install memcached

1. 启动服务并查看网络连接状态是否开启成功

[root@proxy ~]# systemctl restart memcached

[root@proxy ~]# netstat -anptu | grep memcached

1. 关闭防火墙、selinux

[root@proxy ~]# setenforce 0

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

步骤二: 在后端LNMP服务器上部署session共享

注意:这些操作在两台后端web服务器上均需要执行,以下操作以web1(192.168.2.100)服务器为例

1. 为PHP添加memcache扩展

注意，因为后端两台web服务器(web1,web2)都需要连接memcached数据库，所以两台主机都需要安装PHP扩展模块(下面也web1为例)。

[root@web1 ~]# yum -y install php-pecl-memcache

1. 修改PHP-FPM配置文件,并重启服务

注意，因为后端两台web服务器(web1,web2)都需要修改配置文件(下面以web1为例)。

[root@web1 ~]# vim /etc/php-fpm.d/www.conf        //修改该配置文件的两个参数

//文件的最后2行

修改前效果如下:

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

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

//原始文件，默认定义Sessoin会话信息本地计算机（默认在/var/lib/php/session）

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

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

修改后效果如下:

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

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

//定义Session信息存储在公共的memcached服务器上，主机参数中为memcache（没有d）

//通过path参数定义公共的memcached服务器在哪（服务器的IP和端口）

[root@web1 ~]# systemctl restart php-fpm

步骤三: 客户端测试

[root@client ~]# firefox 192.168.4.5/index.php 访问成功,登录后再次刷新不需要再次登录

**总结:**

1 单服务器有session和cookie,可以防止重复登录

2 搭建集群服务器,session默认不共享,导致重复登录

3 web1和web2都把session写入memcached数据库

在proxy安装memcached,启动服务,使用telnet测试

给web1和web2安装memcache扩展包(php-pecl-memcache)

4 默认php会把session写到/var/lib/php/session

修改配置/etc/php-fpm.d/www/conf

设置将session保存到memcached数据库中

#########################################

**II-Day05-Tomcat/Varnish**

PHP, Python + Nginx, httpd

Java + tomcat, Jboss, websphere, weblogic

**Java简介**

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体系

1) Java SE(标准版)

2) Java EE(企业版)

3) Java ME(移动版)

**JDK简介**

JDK(Java Development Kit)是Sun针对Java开发者推出的Java语言的软件开发工具包

JDK是整个Java的核心

1) 包括了Java运行环境

2) Java工具(如编译、排错、打包等工具)

3) Java基础的类库

JRE(Java Runtime Environment, Java运行环境), JRE是JDK的子集

JRE包括

1) Java虚拟机(jvm)

2) Java核心类库和支持文件

3) 不包括开发工具(JDK)--编译器、调试器和其他工具

Java Servlet

Servlet是Java扩展web服务器功能的组件规范

常见Servlet容器

IBM------------websphere

Oracle--------weblogic

Apache-------tomcat

RedHat-------Jboss

java里面所有的配置文件严格区分大小写,一个host代表一个网站

Tomcat是使用Java编写的

**案例1：安装部署Tomcat服务器**

本案例要求部署Tomcat服务器，具体要求如下：

1) 安装部署JDK基础环境

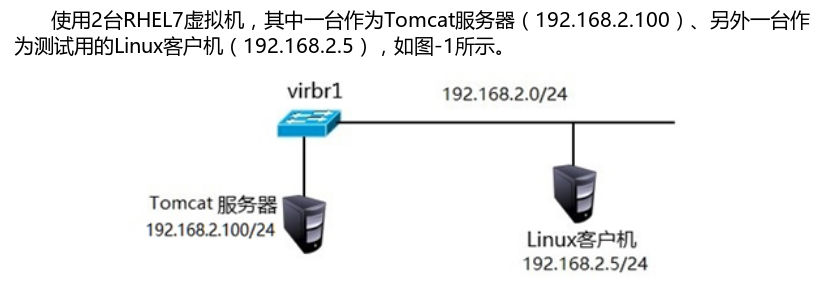
2) 安装部署Tomcat服务器

3) 创建JSP测试页面，文件名为test.jsp，显示服务器当前时间

然后客户机访问此Web服务器验证效果：

1) 使用火狐浏览器访问Tomcat服务器的8080端口，浏览默认首页

2) 使用火狐浏览器访问Tomcat服务器的8080端口，浏览默认测试页面



使用RPM安装JDK基础环境

使用源码安装部署Tomcat服务器

步骤一: 部署Tomcat服务器软件(192.168.2.100/24)

1. 使用rpm安装JDK环境 (默认已经安装)

[root@web1 ~]# yum -y install java-1.8.0-openjdk             //安装JDK

[root@web1 ~]# yum -y install java-1.8.0-openjdk-headless      //安装JDK

[root@web1 ~]# java -version                   //查看JAVA版本

2) 安装Tomcat（apache-tomcat-8.0.30.tar.gz软件包，在lnmp\_soft中有提供）

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

[root@web1 lnmp\_soft]# tar -xf apache-tomcat-8.0.30.tar.gz

[root@web1 lnmp\_soft]# mv apache-tomcat-8.0.30 /usr/local/tomcat

[root@web1 ~]# ls /usr/local/tomcat

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

1. 启动服务

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

[root@web1 ~]# firewall-cmd --set-default-zone=trusted

[root@web1 ~]# setenforce 0

1. 服务器验证端口信息

[root@web1 ~]# ss -antulp | grep java //查看java监听的端口

tcp LISTEN 0 100 :::8080 :::\* users:(("java",pid=9445,fd=46))

tcp LISTEN 0 1 ::ffff:127.0.0.1:8005 :::\* users:(("java",pid=9445,fd=67))

tcp LISTEN 0 100 :::8009 :::\* users:(("java",pid=9445,fd=51))

**提示**：如果检查端口时，8005端口启动非常慢，可用使用下面的命令用urandom替换random（非必须操作）。

[root@web1 ~]# mv /dev/random /dev/random.bak

[root@web1 ~]# ln -s /dev/urandom /dev/random

5）客户端浏览测试页面(proxy作为客户端)

[root@proxy ~]# firefox <http://192.168.2.100:8080> #访问成功

步骤二: 修改Tomcat配置文件

1) 创建测试JSP页面

[root@web1 ~]# vim /usr/local/tomcat/webapps/ROOT/test.jsp

<html>

<body>

<center>

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

</center>

</body>

</html>

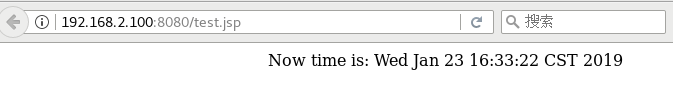
2）重启服务

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

步骤三: 验证测试

[root@proxy ~]# firefox 192.168.2.100:8080/test.jsp



**案例2：使用Tomcat部署虚拟主机**

沿用练习二，使用Tomcat部署加密虚拟主机，实现以下要求：

1) 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

2) 使用www.a.com域名访问的页面根路径为/usr/local/tomcat/a/ROOT

3) 使用www.b.com域名访问的页面根路径为/usr/local/tomcat/b/base

4) 访问www.a.com/test时，页面自动跳转到/var/www/html目录下的页面

5) 访问页面时支持SSL加密通讯

6) 私钥、证书存储路径为/usr/local/tomcat/conf/cert

7) 每个虚拟主机都拥有独立的访问日志文件

8) 配置tomcat集群环境

步骤一: 配置服务器虚拟主机

1. 修改server.xml配置文件,创建虚拟主机

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml #添加两个网站

......

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

unpackWARs="true" autoDeploy="true">

</Host>

Host name="localhost" appBase="webapps"

unpackWARs="true" autoDeploy="true">

.......

#appBase:网页根目录,类似于DocumentRoot

1. 创建虚拟主机对应的页面根路径

[root@web1 ~]# mkdir -p /usr/local/tomcat/{a,b}/ROOT

[root@web1 ~]# echo "<h1>AA" > /usr/local/tomcat/a/ROOT/index.html

[root@web1 ~]# echo "<h1>BB" > /usr/local/tomcat/b/ROOT/index.html

1. 重启Tomcat服务器

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

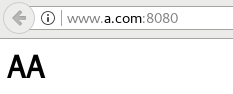
1. 客户端设置host文件,并浏览测试页面进行测试(proxy充当客户端角色)

注意: ssh远程连接时使用-X参数才可以

[root@proxy ~]# vim /etc/hosts

192.168.2.100 www.a.com www.b.com

[root@proxy ~]# firefox [www.a.com:8080](http://www.a.com:8080)



[root@proxy ~]# firefox [www.b.com:8080](http://www.b.com:8080)



步骤二: 修改www.b.com网站的首页目录为base

1）使用docBase参数可以修改默认网站首页路径(docBase默认为ROOT)

[root@web1 ~]# vim /usr/local/tomcat/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>

......

[root@web1 ~]# mkdir /usr/local/tomcat/b/base

[root@web1 ~]# echo "<h1>base" > /usr/local/tomcat/b/base/index.html

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

2）测试查看页面是否正确(proxy充当客户端角色)

[root@proxy ~]# firefox [www.b.com:8080](http://www.b.com:8080)



步骤三：跳转

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

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

......

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

unpackWARs="true" autoDeploy="true">

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

</Host>

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

unpackWARs="true" autoDeploy="true">

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

</Host>

.......

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

2）测试查看页面是否正确(proxy充当客户端角色)

[root@proxy ~]# firefox [www.a.com:8080/test](http://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

[root@web1 ~]# 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/keystore

输入密钥库口令: #输入密码123456

再次输入新口令: #再次输入密码123456

您的名字与姓氏是什么? #随便写

[Unknown]: sxk

您的组织单位名称是什么? #随便写

[Unknown]: sxk

您的组织名称是什么? #随便写

[Unknown]: sxk

您所在的城市或区域名称是什么? #随便写

[Unknown]: sxk

您所在的省/市/自治区名称是什么? #随便写

[Unknown]: sxk

该单位的双字母国家/地区代码是什么? #随便写

[Unknown]: sxk

CN=sxk, OU=sxk, O=sxk, L=sxk, ST=sxk, C=sxk是否正确?

[否]: y

输入 <tomcat> 的密钥口令

(如果和密钥库口令相同, 按回车): #输入密码123456

再次输入新口令: #再次输入密码123456

2) 再次修改server.xml配置文件,创建支持加密连接的Connector

[root@web1 ~]# vim /usr/local/tomcat/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"

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

.......

//备注,默认这段Connector被注释掉了,打开注释,添加要信息即可

3）重启Tomcat服务器

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

4）客户端设置host文件，并浏览测试页面进行测试(proxy充当客户端角色)

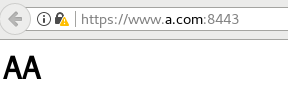
[root@proxy ~]# vim /etc/hosts

192.168.2.100     www.a.com www.b.com

[root@proxy ~]# firefox <https://www.a.com:8443>







[root@proxy ~]# firefox <https://www.b.com:8443/> #访问成功

[root@proxy ~]# firefox <https://192.168.2.100:8443/> #访问成功

步骤五: 配置Tomcat日志

1) 为每个虚拟主机设置不同的日志文件

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

......

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

unpackWARs="true" autoDeploy="true">

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

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="www\_a\_com\_log" suffix=".log"

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

</Host>

......

#从默认localhost虚拟主机中把Valve这段复制过来，适当修改下即可

1. 重启Tomcat服务器

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

1. 查看服务器日志文件

[root@web1 ~]# ls /usr/local/tomcat/logs/

catalina.2019-01-23.log localhost.2019-01-25.log

catalina.2019-01-25.log localhost\_access\_log.2019-01-23.txt

catalina.out localhost\_access\_log.2019-01-25.txt

host-manager.2019-01-23.log manager.2019-01-23.log

host-manager.2019-01-25.log manager.2019-01-25.log

localhost.2019-01-23.log www\_a\_com\_log.2019-01-25.log

步骤六: 扩展实验(配置Tomcat集群)----晚上做

1) 在192.168.4.5主机上配置nginx调度器(具体步骤参考前面的章节)

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

......

upstream web{

server 192.168.2.100:8080;

server 192.168.2.200:8080;

}

....

location / {

proxy\_pass http://web;

.....

[root@proxy ~]# nginx -s reload

1. 在192.168.2.100和192.168.2.200主机上配置Tomcat调度器

以下以web1为例:

[root@web1 ~]# yum -y install java-1.8.0-openjdk                //安装JDK

[root@web1 ~]# yum -y install java-1.8.0-openjdk-headless        //安装JDK

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

[root@web1 ~]# tar -xzf apache-tomcat-8.0.30.tar.gz

[root@web1 ~]# mv apache-tomcat-8.0.30 /usr/local/tomcat

3）启动服务

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

4) 客户端验证

为了防止有数据缓存，可以使用真实主机的google-chrome访问代理服务器，输入Ctrl+F5刷新页面。

CDN:内容分发网络

**案例3：使用Varnish加速Web**

通过配置Varnish缓存服务器，实现如下目标：

1) 使用Varnish加速后端Web服务

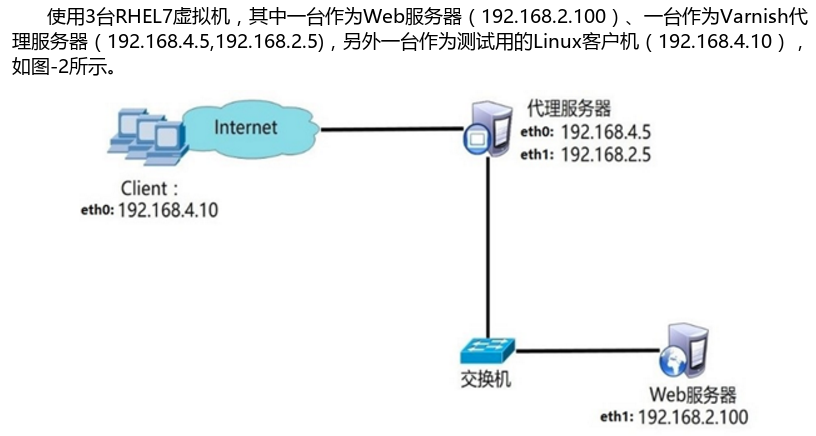
2) 代理服务器可以将远程的Web服务器页面缓存在本地

3) 远程Web服务器对客户端用户是透明的

4) 利用缓存机制提高网站的响应速度

5) 使用varnishadm命令管理缓存页面

6) 使用varnishstat命令查看Varnish状态



对于Web服务器的部署，此实验中仅需要安装nginx或者httpd软件、启动服务，并生成测试首页文件即可，默认httpd网站根路径为/var/www/html，首页文档名称为index.html，默认nginx网站根路径为/usr/local/nginx/html，默认首页为index.html。下面的实验我们以httpd为例作为Web服务器

步骤一:构建web服务器

1. 使用yum安装web软件包

[root@web1 ~]# yum -y install httpd

2）启用httpd服务（注意需要关闭nginx，否则端口冲突）

[root@web1 ~]# systemctl start httpd

[root@web1 ~]# setenforce 0

[root@web1 ~]# firewall-cmd --set-default-zone=trusted

3）为Web访问建立测试文件

在网站根目录/var/www/html下创建一个名为index.html的首页文件：

[root@web1 ~]# cat /var/www/html/index.html

192.168.2.100

4）测试页面是否正常（代理服务器测试后台web）

[root@proxy ~]# firefox http://192.168.2.100

步骤二：部署Varnish缓存服务器(192.168.4.5)

1）编译安装软件(python-docutils默认光盘中没有，需要在lnmp\_soft中找)

[root@proxy ~]# yum -y install gcc readline-devel ncurses-devel pcre-devel #安装软件依赖包

[root@proxy ~]# cd lnmp\_soft/

[root@proxy lnmp\_soft]# yum -y install python-docutils-0.11-0.2.20130715svn7687.el7.noarch.rpm

[root@proxy ~]# useradd -s /sbin/nologin varnish

[root@proxy ~]# cd lnmp\_soft/

[root@proxy lnmp\_soft]# ls varnish-5.2.1.tar.gz

varnish-5.2.1.tar.gz

[root@proxy lnmp\_soft]# tar -xf varnish-5.2.1.tar.gz

[root@proxy lnmp\_soft]# cd varnish-5.2.1/

[root@proxy varnish-5.2.1]# ./configure

[root@proxy varnish-5.2.1]# make && make install

2）复制启动脚本及配置文件

[root@proxy varnish-5.2.1]# cp /root/lnmp\_soft/varnish-5.2.1/etc/example.vcl /usr/local/etc/default.vcl

3）修改代理配置文件

[root@proxy varnish-5.2.1]# vim /usr/local/etc/default.vcl

......

backend default {

.host = "192.168.2.100";

.port = "80";

}

......

4）启动服务

[root@proxy varnish-5.2.1]# varnishd -f /usr/local/etc/default.vcl

//varnishd命令的其他选项说明如下：

//varnishd –s malloc,128M     定义varnish使用内存作为缓存，空间为128M

//varnishd –s file,/var/lib/varnish\_storage.bin,1G 定义varnish使用文件作为缓存

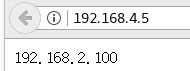
步骤三:客户端测试

1. 客户端开启浏览器访问

[root@client ~]# curl 192.168.4.5

192.168.2.100

[root@client ~]# firefox 192.168.4.5



[root@web1 ~]# cat /var/www/html/index.html

192.168.2.100

[root@web1 ~]# echo "<h1>今天是礼拜五,后天回家一趟" >> /var/www/html/index.html

[root@client ~]# curl 192.168.4.5

192.168.2.100

[root@client ~]# curl 192.168.4.5 #几分钟后会自动缓存到代理服务器上

192.168.2.100

<h1>今天是礼拜五,后天回家一趟

步骤四:其他操作

1. 查看varnish日志

[root@proxy ~]# varnishlog #varnish日志

[root@proxy ~]# varnishncsa #访问日志

1. 更新缓存数据,在后台web服务器更新页面内容后,用户访问代理服务器看到的还是之前的数据,说明缓存中的数据过期了,需要更新(默认也会自动更新,但非实时更新)

[root@proxy ~]# varnishadm

200

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

Varnish Cache CLI 1.0

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

Linux,3.10.0-693.el7.x86\_64,x86\_64,-junix,-smalloc,-smalloc,-hcritbit

varnish-5.2.1 revision 67e562482

Type 'help' for command list.

Type 'quit' to close CLI session.

varnish> ban req.url ~ .\* #清空缓存数据,支持正则表达式

200

#########################################

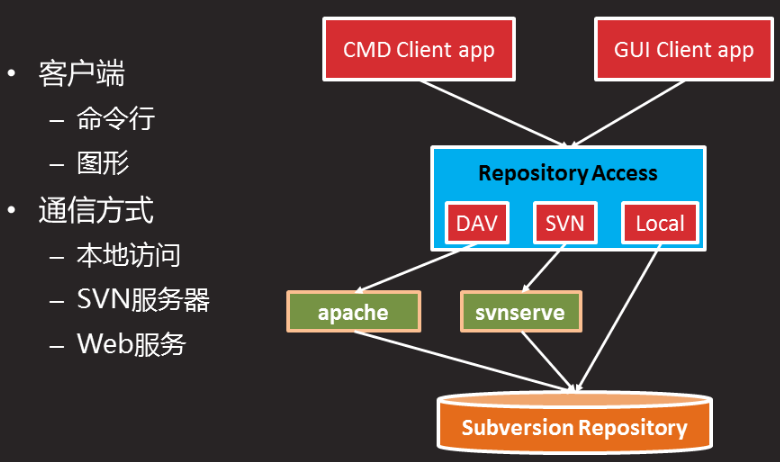
**II-Day-06(SVN、RPM)**

**Subversion (SVN)**

Subversion是一个自由/开源的版本控制系统

1. 在Subversion管理下,文件和目录可以超越时空
2. Subversion允许你数据恢复到早期版本
3. 或者是检查数据修改的历史
4. 许多人将版本控制系统当作一种神奇的”时间机器”
5. 允许你和别人协作文档并跟踪所做的修改

**Subversion架构**



**版本库**

典型的客户/服务器系统

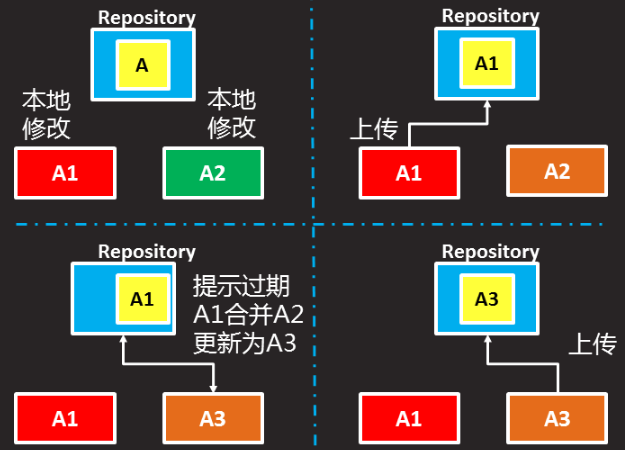
1) 版本库是版本控制的核心

2) 任意数量客户端

3) 客户端通过写数据库分享代码

Subversion特点: 记录每一次改变

**版本模型**



Subversion控制系统使用拷贝-修改-合并模型

1) 每个客户连接版本库,并建立个人个工作副本

2) 用户并行工作,修改自己的副本

3) 最终合并版本

4) 个别冲突问题,需要人为手动解决

**流程**

1) Harry拷贝副本,Sally拷贝副本

2) Harry修改A1,Sally修改A2

3) Harry上传A1后,Sally上传A2,会提示Sally的文件已过期

4) Sally更新文件(合并)后上传新的A3

**创建版本库**

格式: svnadmin 命令 /版本库路径 [选项]

命令: create 创建一个新的版本库

**导入项目代码**

使用svn命令将项目代码导入到版本库中

import指令执行导入操作

-m选项设置注释性的字符串

## **案例1：Subversion基本操作**

本案例要求先快速搭建好一台Subversion服务器，并测试该版本控制软件：

(web1做服务器, web2做客户端)

1) 创建版本库

2) 导入初始化数据

3) 检出数据至用户本地副本

4) 对本地副本进行增删改查等操作

步骤一：安装Subversion服务器

1）YUM安装subversion软件

[root@web1 ~]# yum -y install subversion

[root@web1 ~]# rpm -q subversion

2) 创建版本库

[root@web1 ~]# mkdir /var/svn

[root@web1 ~]# svnadmin create /var/svn/project #创建版本库

[root@web1 ~]# ls /var/svn/project

conf db format hooks locks README.txt

写什么都可以

1. 本地导入初始化数据

[root@web1 ~]# cd /usr/lib/systemd/system/

[root@web1 system]# svn import . file:///var/svn/project/ -m "注释"

1. 修改配置文件,创建账户与密码

所有配置文件,要求顶头写,开头不要有空格

[root@web1 system]# vim /var/svn/project/conf/svnserve.conf

#修改19行、20行、27行、34行

anon-access = none #匿名用户无任何权限

auth-access = write #有效账户可写

password-db = passwd #密码文件

authz-db = authz #ACL访问控制列表文件

[root@web1 system]# vim /var/svn/project/conf/passwd

......

harry = 123456 #用户名和密码

tom = 123456 #用户名和密码

[root@web1 system]# vim /var/svn/project/conf/authz

......

[/]

harry = rw

tom = rw

.....

5) 启动服务

[root@web1 system]# svnserve -d -r /var/svn/project/

-d: 放入后台执行

-r: 指定共享的路径

[root@web1 system]# ss -ntulp | grep svn

tcp LISTEN 0 7 \*:3690 \*:\* users:(("svnserve",pid=6760,fd=3))

**备注**：启动服务也可以使用svnserve -d启动，但客户端访问时需要指定绝对路径（svn://服务器IP/var/svn/project）

步骤二：客户端测试(192.168.2.200)

1）将服务器上的代码下载到本地

[root@web2 ~]# cd /tmp

[root@web2 tmp]# svn --username harry --password 123456 co svn://192.168.2.100/ code

//建立本地副本,从服务器192.168.2.100上co下载代码到本地code目录(co=checkout,可以打全称)

//用户名harry,密码123456

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

注意! 你的密码，对于认证域:

<svn://192.168.2.100:3690> 9737e8c2-17a5-490f-a11e-bf57f351d186

只能明文保存在磁盘上! 如果可能的话，请考虑配置你的系统，让 Subversion

可以保存加密后的密码。请参阅文档以获得详细信息。

你可以通过在“/root/.subversion/servers”中设置选项“store-plaintext-passwords”为“yes”或“no”，

来避免再次出现此警告。

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

保存未加密的密码(yes/no)?yes #提示是否保存密码

[root@web2 tmp]# cd /tmp/code/

[root@web2 code]# ls

[root@web2 code]# vim user.slice #挑选任意文件修改其内容

[root@web2 code]# vim upower.service #挑选任意文件修改其内容

[root@web2 code]# svn ci -m "modify 2 file" #将本地修改的数据同步到服务器(ci=commit,可以写全称)

正在发送 upower.service

正在发送 user.slice

传输文件数据..

提交后的版本为 2。

[root@web2 code]# svn update #将服务器上新的数据同步到本地

正在升级 '.':

版本 3。

[root@web2 code]# rm -rf \*.service

[root@web2 code]# svn update #可以还原数据

[root@web2 code]# svn info svn://192.168.2.100 #查看版本仓库基本信息

路径: .

URL: svn://192.168.2.100

版本库根: svn://192.168.2.100

版本库 UUID: 9737e8c2-17a5-490f-a11e-bf57f351d186

版本: 3

节点种类: 目录

最后修改的作者: harry

最后修改的版本: 3

最后修改的时间: 2019-01-25 16:47:44 +0800 (五, 2019-01-25)

[root@web2 code]# svn log svn://192.168.2.100 #查看版本仓库的日志

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

r3 | harry | 2019-01-25 16:47:44 +0800 (五, 2019-01-25) | 1 行

modify umount.target

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

r2 | harry | 2019-01-25 16:43:26 +0800 (五, 2019-01-25) | 1 行

modify 2 file

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

r1 | root | 2019-01-25 15:39:13 +0800 (五, 2019-01-25) | 1 行

注释

[root@web2 code]# echo "test" > test.sh #本地新建一个文件

[root@web2 code]# svn ci -m "new file" #提交失败,该文件不被svn管理

[root@web2 code]# svn add test.sh #将文件或目录加入版本控制

A test.sh

[root@web2 code]# svn ci -m "new file" #再次提交,成功

正在增加 test.sh

传输文件数据.

提交后的版本为 4。

[root@web2 code]# svn mkdir subdir #创建子目录

A subdir

[root@web2 code]# svn rm test.sh #使用svn删除文件

D test.sh

[root@web2 code]# svn ci -m "xxx" #提交一次代码

正在增加 subdir

正在删除 test.sh

提交后的版本为 5。

[root@web2 code]# svn diff #查看所有文件的差异

[root@web2 code]# vim umount.target #任意修改本地一个文件

[root@web2 code]# svn diff umount.target #仅查看某一个文件的差异

[root@web2 code]# svn cat svn://192.168.2.100/umount.target #查看服务器文件的内容

[root@web2 code]# wc -l tmp.mount

26 tmp.mount

[root@web2 code]# sed -i 'd' tmp.mount #删除文件所有内容,但未提交

[root@web2 code]# cat tmp.mount

[root@web2 code]# svn revert tmp.mount #还原tmp.mount文件

已恢复“tmp.mount”

[root@web2 code]# sed -i '1a #test###' tuned.service #修改本地内容

[root@web2 code]# svn ci -m "xx" #提交代码

正在发送 tuned.service

正在发送 umount.target

传输文件数据..

提交后的版本为 6。

[root@web2 code]# svn merge -r6:5 tuned.service #将文件从版本6还原到版本5

--- 正在反向合并 r6 到 “tuned.service”:

U tuned.service

--- 记录反向合并 r6 到“tuned.service”的信息:

U tuned.service

--- 正在从 'tuned.service' 删除合并信息:

U tuned.service

[root@web2 code]# cd ..

[root@web2 tmp]# svn co -r3 svn://192.168.2.100 xyz #下载版本3

使用svn命令测试svnserver服务时可以使用的命令列表

|  |  |
| --- | --- |
| **命令** | **作用** |
| add | 添加文件 |
| commit (ci) | 提交更新 |
| checkout (co) | 检出代码 |
| cat | 查看代码文件内容 |
| del | 删除文件 |
| diff | 文件对比 |
| import | 导入代码 |
| info | 查看版本信息 |
| list | 查看文件列表 |
| log | 查看版本历史 |
| update | 更新 |
| mkdir | 创建目录 |

**案例2：使用Subversion协同工作**

沿用练习一，通过svn工具，对subversion版本库进行多人协同工作测试，要求如下：

1) 该版本库支持多个账户同时协作编辑文件

2) 测试演示多人协作编辑的具体操作

3) 手动解决版本冲突问题

4) 备份版本库数据

5) 注册使用Github

使用svn客户端工具连接subversion服务器并测试多人协同工作以及如何手动解决冲突问题，账户名称分别为harry和tom，最后使用svnadmin dump指令对版本库进行备份工作。

步骤一: 多人协同工作

1）远程连接两个终端，每个人下载代码本地副本，注意web1(192.168.2.100)和web2（192.168.2.200）代表了两个不同的主机，看清楚操作是在哪一台计算机上执行！

[root@web1 ~]# cd /tmp

[root@web1 tmp]# svn --username tom --password 123456 co svn://192.168.2.100/ code

[root@web2 ~]# cd /tmp

[root@web2 tmp]# svn --username harry --password 123456 co svn://192.168.2.100/ code

[root@web1 code]# cd code

[root@web2 code]# cd code

2) harry和tom修改不同的文件

[root@web1 code]# sed -i "3a ###tom modify#####" tmp.mount

[root@web1 code]# svn ci -m "has modified"

[root@web2 code]# sed -i "3a ###harry modify#####" umount.target

[root@web2 code]# svn ci -m "has modified"

[root@web2 code]# svn update

[root@web1 code]# svn update

3）harry和tom修改相同文件的不同行

[root@web1 code]# sed -i "3a ###tom modify#####" user.slice

[root@web1 code]# svn ci -m "modified"

[root@web2 code]# sed -i "6a ###harry modify#####" user.slice

[root@web2 code]# svn ci -m "modified"        //提交失败

Sending svnserve

Transmitting file data .svn: Commit failed (details follow):

svn: File '/user.slice' is out of date（过期）

[root@web2 code]# svn update            //提示失败后，先更新再提交即可

[root@web2 code]# svn ci -m "modified"        //提交成功

Sending user.slice

Transmitting file data .

4) harry和tom修改相同文件的相同行

[root@web1 code]# sed -i '1c [UNIT]' tuned.service

[root@web1 code]# svn ci -m "modified"

[root@web2 code]# sed -i '1c [unit]' tuned.service

[root@web2 code]# svn ci -m "modified"

Sending tuned.service

Transmitting file data .svn: Commit failed (details follow):

svn: File '/tuned.service' is out of date(过期)

[root@web2 code]# svn update               //出现冲突，需要解决

Conflict(冲突) discovered in 'tuned.service'.

Select: (p) postpone, (df) diff-full, (e) edit,

(mc) mine-conflict, (tc) theirs-conflict,

(s) show all options:p                    //选择先标记p，随后解决

[root@web2 code]# ls

tuned.service tuned.service.mine    tuned.service.r10   tuned.service.r9

[root@web2 code]# mv tuned.service.mine tuned.service

[root@web2 code]# rm -rf tuned.service.r10 tuned.service.r9

[root@web2 code]# svn ci -m "modified"    //解决冲突

步骤二：使用dump指令备份版本库数据

[root@web1 ~]# svnadmin dump /var/svn/project > project.bak //备份

\* Dumped revision 0.

\* Dumped revision 1.

\* Dumped revision 2.

\* Dumped revision 3.

\* Dumped revision 4.

\* Dumped revision 5.

\* Dumped revision 6.

\* Dumped revision 7.

\* Dumped revision 8.

\* Dumped revision 9.

\* Dumped revision 10.

\* Dumped revision 11.

[root@web1 ~]# svnadmin create /var/svn/project2 //新建空仓库

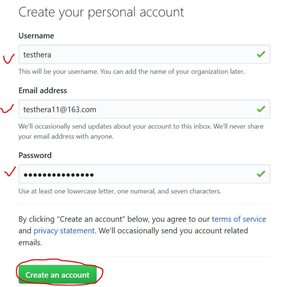
[root@web1 ~]# svnadmin load /var/svn/project2 < project.bak   //还原

步骤三：注册使用Github

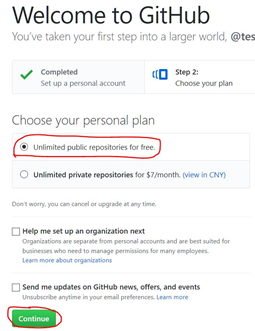
1.登陆网站https://github.com，点击Sign up（注册），如图-1所示。

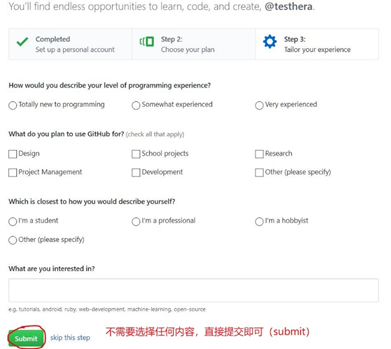
IMG_256

2.填写注册信息（用户名，邮箱，密码），如图-2所示。



3. 初始化操作，如图-3和图-4所示。

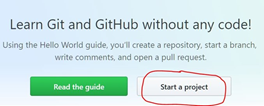




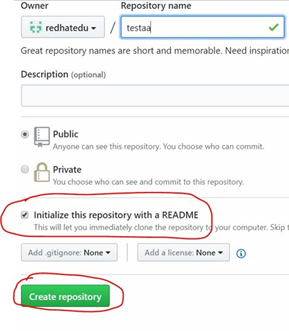
注意，初始化完成后，到邮箱中去激活Github账户。

4. 创建仓库、使用仓库

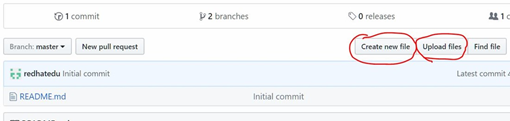
点击Start a project（如图-5所示），



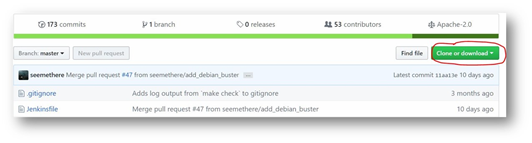
填写项目名称（项目名称任意），如图-6所示。



往仓库中上传文件或新建文件，如图-7所示



下载仓库中的代码，如图-8所示。



5. 命令行操作（需要联网的主机，如真实机）

[root@pc001 ~]# yum -y install git

[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 #add添加新文件

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

[root@pc001 ~]# git push #commit和push实现提交代码的功能

[root@pc001 ~]# git pull #pull更新，类似于svn update

## **案例3：制作nginx的RPM包**

本案例使用nginx-1.12.2版本的源码软件，生成对应的RPM包软件，具体要求如下：

1) 软件名称为nginx

2) 软件版本为1.12.2

3) RPM软件包可以查询描述信息

4) RPM软件包可以安装及卸载

步骤一：安装rpm-build软件

1）安装rpm-build软件包

[root@web1 ~]# yum -y install rpm-build

2）生成rpmbuild目录结构

[root@web1 ~]# rpmbuild -ba nginx.spec       //会报错，没有文件或目录

错误：stat /root/nginx.spec 失败：没有那个文件或目录

[root@web1 ~]# ls /root/rpmbuild                   //自动生成的目录结构

BUILD BUILDROOT RPMS SOURCES SPECS SRPMS

3）准备工作，将源码软件复制到SOURCES目录

[root@web1 ~]# cp nginx-1.12.2.tar.gz /root/rpmbuild/SOURCES/

4）创建并修改SPEC配置文件

[root@web1 ~]# vim /root/rpmbuild/SPECS/nginx.spec #必须以.spec结尾

Name:nginx #软件名

Version:1.12.2 #版本号

Release:10 #第几次做rpm包

Summary:nginx is a web server software #描述信息

#Group: #软件属于哪个组

License:GPL #协议,受法律保护,但是修改后必须开源

URL:www.douniwan.com #网站

Source0:nginx-1.12.2.tar.gz #来源

#BuildRequires: #源码安装时所需要的依赖包

#Requires: #yum安装时所需要的依赖包

%description #详细描述信息

nginx [enigine x] is an HTTP and reverse proxy server

%post #安装后脚本

useradd nginx #非必需操作:安装后脚本(创建用户),看./configure是否指定用户

%prep

%setup -q #自动解压源码包,并cd进入目录

%build

./configure --user=nginx --group=nginx

make %{?\_smp\_mflags}

%install

make install DESTDIR=%{buildroot}

%files

%doc

/usr/local/nginx/\* #打包的文件(默认路径是/usr/local/nginx,以./configure后的prefix的路径为准)

%changelog

步骤二: 使用配置文件创建RPM包

1. 安装依赖软件包

[root@web1 ~]# yum -y install gcc pcre-devel openssl-devel

1. rpmbuild创建RPM软件包

[root@web1 ~]# rpmbuild -ba /root/rpmbuild/SPECS/nginx.spec

[root@web1 ~]# ls /root/rpmbuild/RPMS/

x86\_64

[root@web1 ~]# ls /root/rpmbuild/RPMS/x86\_64/

nginx-1.12.2-10.x86\_64.rpm nginx-debuginfo-1.12.2-10.x86\_64.rpm

[root@web1 ~]# rpm -qpi /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm #查看软件包具体信息

[root@web1 ~]# rpm -qpl /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm #查看安装清单

步骤三: 安装、卸载软件

[root@web1 ~]# rpm -ivh /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

[root@web1 ~]# ls /usr/local/nginx/

conf html logs sbin

[root@web1 ~]# rpm -qa | grep nginx

nginx-1.12.2-10.x86\_64

[root@web1 ~]# /usr/local/nginx/sbin/nginx

[root@web1 ~]# ss -antulp | grep nginx

tcp LISTEN 0 128 \*:80 \*:\* users:(("nginx",pid=8235,fd=6),("nginx",pid=8234,fd=6))

[root@web1 ~]# ls /root/rpmbuild/SRPMS/

nginx-1.12.2-10.src.rpm

[root@web1 ~]# rpm -qpl /root/rpmbuild/SRPMS/nginx-1.12.2-10.src.rpm

nginx-1.12.2.tar.gz

nginx.spec

#########################################

**II-Day-07(VPN/NTP/PSSh)**

**VPN概述**

Virtual Private Network(虚拟专用网络)

1. 在公用网络上建立专用私有网络,进行加密通讯
2. 多用于为集团公司的各地子公司建立连接
3. 连接完成后,各个地区的子公司可以像区域网一样通讯
4. 在企业网络中有广泛应用
5. 偶尔可以用于翻墙
6. 目前主流的VPN技术(GRE,PPTP,L2TP+IPSec,SSL)

**案例1：配置GRE VPN(不支持Windows)**

本案例要求搭建一个GRE VPN环境，并测试该VPN网络是否能够正常通讯，要求如下：

1) 启用内核模块ip\_gre

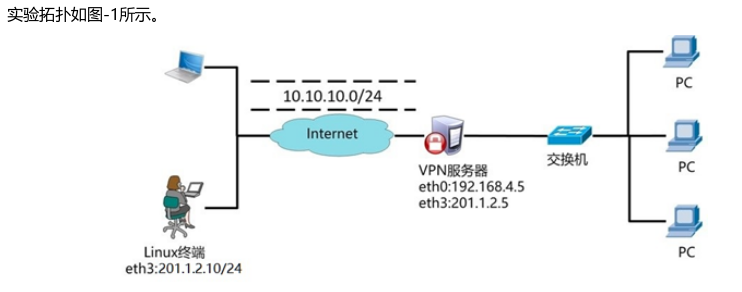
2) 创建一个虚拟VPN隧道(10.10.10.0/24)

3) 实现两台主机点到点的隧道通讯

使用lsmod查看当前计算机已经加载的模块，使用modprobe加载Linux内核模块，使用modinfo可以查看内核模块的信息。

准备实验所需的虚拟机环境，实验环境所需要的主机及对应的IP设置列表如表所示，正确配置IP地址、主机名称，并且为每台主机配置YUM源。

|  |  |
| --- | --- |
| **主机名** | **IP地址** |
| 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）查看计算机当前加载的模块

[root@client ~]# lsmod                            //显示模块列表

[root@client ~]# lsmod | grep ip\_gre     //确定是否加载了gre模块

2)加载模块ip\_gre

[root@client ~]# modprobe ip\_gre

[root@client ~]# lsmod | grep ip\_gre

ip\_gre 22707 0

ip\_tunnel 25163 1 ip\_gre

gre 13144 1 ip\_gre

[root@proxy ~]# modprobe ip\_gre

[root@proxy ~]# lsmod | grep ip\_gre

ip\_gre 22707 0

ip\_tunnel 25163 1 ip\_gre

gre 13144 1 ip\_gre

3）查看模块信息

[root@client ~]# modinfo ip\_gre

步骤二：Client主机创建VPN隧道

1）创建隧道

[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

1. 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

3) 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（10.10.10.10/24）

//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

4）关闭防火墙

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

步骤三：Proxy主机创建VPN隧道

1）查看计算机当前加载的模块

[root@client ~]# lsmod                           //显示模块列表

[root@client ~]# lsmod | grep ip\_gre     //确定是否加载了gre模块

2)加载模块ip\_gre

[root@client ~]# modprobe ip\_gre

3）创建隧道

[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（10.10.10.5/24）

//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

6）开启路由转发、关闭防火墙

[root@proxy ~]# echo "1" > /proc/sys/net/ipv4/ip\_forward

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

7)测试连通性

[root@client ~]# ping 10.10.10.5 #可以ping通

[root@proxy ~]# ping 10.10.10.10 #可以ping通

**案例2：创建PPTP VPN(支持Windows)**

本案例要求搭建一个PPTP VPN环境，并测试该VPN网络是否能够正常通讯，要求如下:

1) 使用PPTP协议创建一个支持身份验证的隧道连接

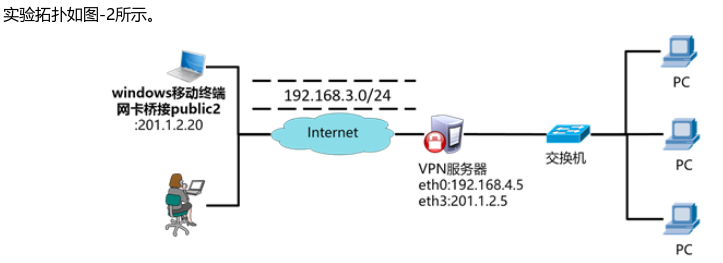
2) 使用MPPE对数据进行加密

3) 为客户端分配192.168.3.0/24的地址池

4) 客户端连接的用户名为jacob，密码为123456

准备实验所需的虚拟机环境，实验环境所需要的主机及对应的IP设置列表如表所示，正确配置IP地址、主机名称，并且为每台主机配置YUM源。

|  |  |
| --- | --- |
| **主机名** | **IP地址** |
| 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 |



步骤一：部署VPN服务器

1）安装软件包（软件包参考lnmp\_soft）

[root@proxy ~]# cd lnmp\_soft/vpn/

[root@proxy vpn]# yum -y install pptpd-1.4.0-2.el7.x86\_64.rpm

[root@proxy vpn]# rpm -qc pptpd #查看软件的配置文件

/etc/ppp/options.pptpd

/etc/pptpd.conf

/etc/sysconfig/pptpd

1. 修改配置文件

[root@proxy vpn]# vim /etc/pptpd.conf #主配置文件

......

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 vpn]# vim /etc/ppp/options.pptpd

......

name pptpd

require-mppe-128 #使用MPPE加密数据

ms-dns 8.8.8.8 #DNS服务器

......

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

# client server secret IP addresses

jacob \* 123456 \*

//用户名 服务器标记 密码 客户端

[root@proxy vpn]# echo "1" > /proc/sys/net/ipv4/ip\_forward #开启路由转发(默认已经做好的),为1,相当于路由器; 为0,相当于关闭路由功能

1. 启动服务

[root@proxy vpn]# systemctl start pptpd

[root@proxy vpn]# systemctl enable pptpd

[root@proxy vpn]# 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

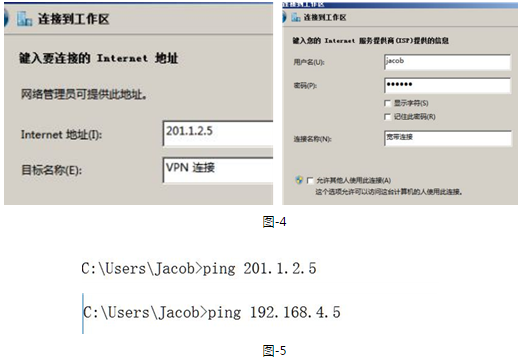
步骤二：客户端设置

启动一台Windows虚拟机，将虚拟机网卡桥接到public2，配置IP地址为201.1.2.20。

新建网络连接（具体操作如图-3所示），输入VPN服务器账户与密码（具体操作如图-4所示），连接VPN并测试网络连通性（如图-5所示）。



图三



## **案例3：创建L2TP+IPSec VPN**

本案例要求搭建一个L2TP+IPSec VPN环境，并测试该VPN网络是否能够正常通讯，具体要求如下：

1) 使用L2TP协议创建一个支持身份验证与加密的隧道连接

2) 使用IPSec对数据进行加密

3) 为客户端分配192.168.3.0/24的地址池

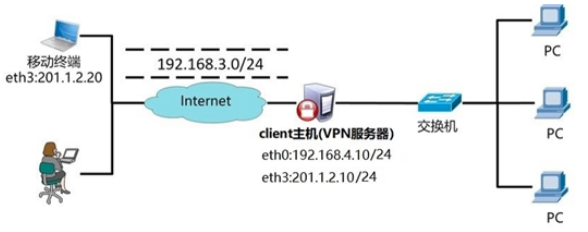
4) 客户端连接的用户名为：jacob，密码为：123456

5) 预共享密钥为：randpass

准备实验所需的虚拟机环境，实验环境所需要的主机及对应的IP设置列表如表所示，正确配置IP地址、主机名称，并且为每台主机配置YUM源。

|  |  |
| --- | --- |
| **主机名** | **IP地址** |
| Windows主机 | 网卡桥接public2 : 201.1.2.20/24 |
| client(作为vpn服务器) | eth0 : 192.168.4.10/24  eth3 : 201.1.2.10/24 |

实验拓扑如图所示:



步骤一: 部署IPSec服务

1. 安装软件包(默认已经安装)

[root@client ~]# yum -y install libreswan

2) 新建IPSec密钥验证配置文件

[root@client ~]# cat /etc/ipsec.conf #仅查看一下主配置文件,不修改内容

.....

include /etc/ipsec.d/\*.conf #加2. 设置Windows注册表（不修改注册表，连接VPN默认会报789错误），具体操作如下：

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

连接VPN并测试网络连通性（参考案例2）

载该目录下的所有配置文件

[root@client ~]# cp lnmp\_soft/vpn/myipsec.conf /etc/ipsec.d/

[root@client ~]# vim /etc/ipsec.d/myipsec.conf //新建该文件，参考lnmp\_soft/vpn/myipsec.conf

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 #服务器本机的外网IP地址

leftprotoport=17/1701

right=%any #允许任何客户端连接

rightprotoport=17/%any

3) 创建IPSec预定义共享密钥

[root@client ~]# cat /etc/ipsec.secrets  //仅查看，不要修改该文件

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

[root@client ~]# vim /etc/ipsec.d/mypass.secrets

201.1.2.10 %any: PSK "randpass" #randpass为预共享秘钥,201.1.2.10是VPN服务器的IP

4) 启动IPSec服务

[root@client ~]# systemctl start ipsec

[root@client ~]# netstat -ntulp | grep pluto #端口号为4500和500

步骤二：部署XL2TP服务

1）安装软件包（软件包参考lnmp\_soft）

[root@client ~]# cd lnmp\_soft/vpn/

[root@client vpn]# 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）

[root@client vpn]# vim /etc/xl2tpd/xl2tpd.conf

.....

ip range = 192.168.3.128-192.168.3.254 #分配给客户端的IP池

local ip = 201.1.2.10 #VPN服务器的IP地址

.....

[root@client vpn]# vim /etc/ppp/options.xl2tpd #认证配置

.....

#crtscts #注释或者删除该行

#lock #注释或者删除该行

require-mschap-v2 #去掉此行的注释, 强制要求认证

......

[root@client vpn]# vim /etc/ppp/chap-secrets

jacob \* 123456 \* #账户名称 服务器标记 密码 客户端IP

3) 启动服务

[root@client vpn]# systemctl start xl2tpd

[root@client vpn]# netstat -nutlp | grep xl2tpd

udp 0 0 0.0.0.0:1701 0.0.0.0:\* 3465/xl2tpd

4) 设置路由转发,防火墙

[root@client vpn]# echo "1" > /proc/sys/net/ipv4/ip\_forward

[root@client vpn]# firewall-cmd --set-default-zone=trusted

5) 翻墙设置(非必要操作)

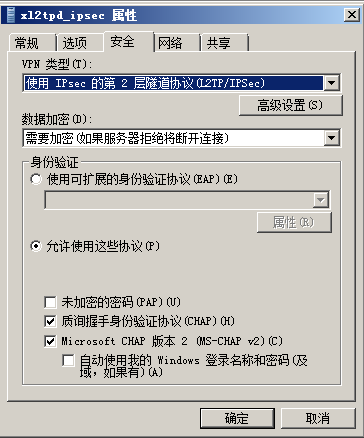
[root@client vpn]# iptables -t nat -A POSTROUTING -s 192.168.3.0/24 -j SNAT --to-source 201.1.2.10

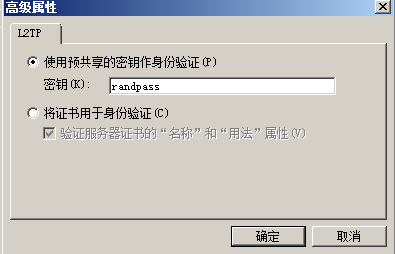
步骤二：客户端设置

启动一台Windows虚拟机，将虚拟机网卡桥接到public2，配置IP地址为201.1.2.20。

1. 新建网络连接（参考案例2），输入VPN服务器账户与密码（参考案例2）。

设置VPN连接的属性，预共享密钥是IPSec配置文件中填写的randpass，具体操作如图所示。





2. 设置Windows注册表（不修改注册表，连接VPN默认会报789错误），具体操作如下：(Windows8 及以上的版本不需要操作此步骤)

1) 单击"开始"，单击"运行"，键入"regedit"，然后单击"确定"

2) 找到下面的注册表子项，然后单击它：

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

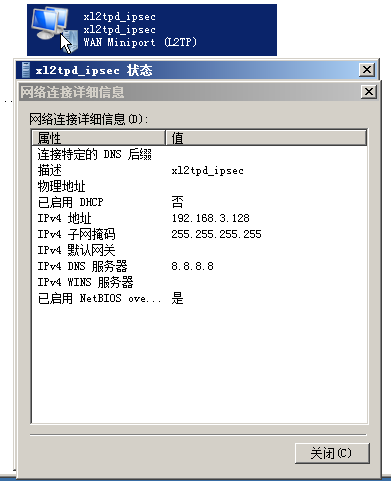
3) 在"编辑"菜单上，单击"新建"->"DWORD值"

4) 在"名称"框中，键入"ProhibitIpSec"

5) 在"数值数据"框中，键入"1"，然后单击"确定"

6) 退出注册表编辑器，然后重新启动计算机

连接VPN并测试网络连通性（参考案例2）



[root@client vpn]# ping -c 1 192.168.3.128

PING 192.168.3.128 (192.168.3.128) 56(84) bytes of data.

64 bytes from 192.168.3.128: icmp\_seq=1 ttl=128 time=0.560 ms

--- 192.168.3.128 ping statistics ---

1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 0.560/0.560/0.560/0.000 ms

**案例4：NTP时间同步**

本案例要求搭建一个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，具体要求如下：

1) 部署一台NTP时间服务器

2) 设置时间服务器上层与0.rhel.pool.ntp.org同步

3) 设置本地服务器层级数量为10

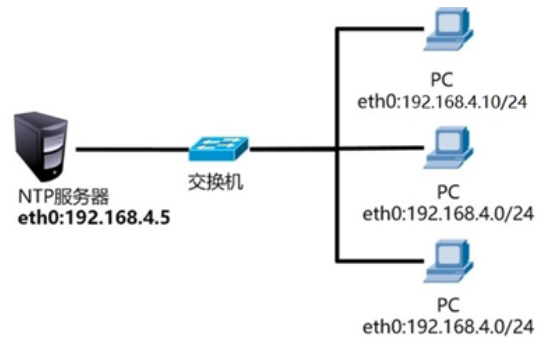
4) 允许192.168.4.0/24网络的主机同步时间

5) 客户端验证时间是否同步

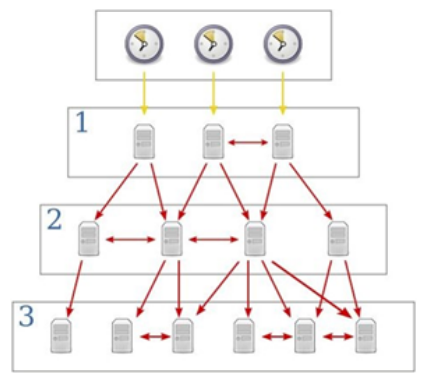
准备实验所需的虚拟机环境，实验环境所需要的主机及对应的IP设置列表如表所示，正确配置IP地址、主机名称，并且为每台主机配置YUM源。

|  |  |
| --- | --- |
| **主机名** | **IP地址** |
| client | eth0: 192.168.4.10/24 |
| proxy | eth0: 192.168.4.5/24  eth1: 192.168.2.5/24 |

实验拓扑如图所示:



Network Time Protocol（网络时间协议）采用的是**分层设计**，如图所示;Stratum层的总数限制在15以内（包括15）



步骤一：部署NTP服务

1）安装软件包

[root@proxy ~]# yum -y install chrony #默认已安装

[root@proxy ~]# rpm -qc chrony #查看配置文件列表

/etc/chrony.conf

/etc/chrony.keys

/etc/logrotate.d/chrony

/etc/sysconfig/chronyd

1. 修改配置文件

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

.....

server 0.rhel.pool.ntp.org iburst #server用户客户端指向上层NTP服务器

......

# Allow NTP client access from local network.

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

......

1. 启动NTP服务

[root@proxy ~]# systemctl restart chronyd

[root@proxy ~]# systemctl enable chronyd

4) 设置防火墙

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

步骤二: 配置客户端

1）安装软件包

[root@client ~]# yum -y install chrony

1. 修改配置文件

[root@client ~]# vim /etc/chrony.conf

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

3) 将客户端时间修改为错误的时间

[root@client ~]# date -s "2000-12-12 8:8:8"

2000年 12月 12日 星期二 08:08:08 CST

1. 重启chrony与服务器同步时间

[root@client ~]# systemctl restart chronyd

1. 确认时间是否已经同步

[root@client ~]# date

2019年 01月 28日 星期一 11:13:27 CST

**案例5：pssh远程套件工具**

本案例要求使用pssh套件工具并发远程其他主机，具体要求如下：

1) 使用密码批量、多并发远程其他主机

2) 使用密钥批量、多并发远程其他主机

3) 批量、多并发拷贝数据到其他主机

4) 批量、多并发从其他主机下载数据到本机

5) 批量、多并发杀死其他主机的进程

准备实验所需的虚拟机环境，实验环境所需要的主机及对应的IP设置列表如表所示，正确配置IP地址、主机名称，并且为每台主机配置YUM源。

|  |  |
| --- | --- |
| **主机名** | **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 |

安装pssh后会提供如下命令：

/usr/bin/pnuke

/usr/bin/prsync

/usr/bin/pscp.pssh

/usr/bin/pslurp

/usr/bin/pssh

步骤一：准备工作

1）安装软件包

[root@proxy ~]# cd lnmp\_soft/

[root@proxy lnmp\_soft]# yum -y install pssh-2.3.1-5.el7.noarch.rpm 2)修改/etc/hosts本地解析文件

[root@proxy ~]# vim /etc/hosts

… …

192.168.2.100 host1 #名字随便写(host1)

192.168.2.200 host2

192.168.4.10 host3

1. 创建主机列表文件

[root@proxy ~]# vim /root/host.txt #每行一个用户名、IP或域名

root@host1

host2

host3

步骤二: 使用密码批量、多并发远程其他主机

1. 语法格式

[root@proxy ~]# man pssh           //通过man帮助查看工具选项的作用

pssh提供并发远程连接功能:

-A               使用密码远程其他主机（默认使用密钥）

-i                将输出显示在屏幕

-H               设置需要连接的主机

-h                设置主机列表文件

-p                设置并发数量

-t                 设置超时时间

-o dir            设置标准输出信息保存的目录

-e dir            设置错误输出信息保存的目录

-x                 传递参数给ssh

2) 使用密码远程多台主机执行命令，屏幕显示标准和错误输出信息

[root@proxy ~]# pssh -A -i -H "host1 host2 host3" -x '-o StrictHostKeyChecking=no' echo hello

Warning: do not enter your password if anyone else has superuser

privileges or access to your account.

Password:

[1] 11:45:33 [SUCCESS] host1

hello

Stderr: Warning: Permanently added 'host1,192.168.2.100' (ECDSA) to the list of known hosts.

[2] 11:45:33 [SUCCESS] host2

hello

Stderr: Warning: Permanently added 'host2,192.168.2.200' (ECDSA) to the list of known hosts.

[3] 11:45:33 [SUCCESS] host3

3) 使用密码远程多台主机执行命令，不显示标准和错误输出信息，通过读取host.txt读取主机信息

[root@proxy ~]# pssh -A -h /root/host.txt echo hello

Warning: do not enter your password if anyone else has superuser

privileges or access to your account.

Password:

[1] 11:52:00 [SUCCESS] root@host1

[2] 11:52:00 [SUCCESS] host2

[3] 11:52:00 [SUCCESS] host3

步骤三:使用秘钥批量、多并发远程其他主机

1）生成密钥并发送密钥到其他主机

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

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

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

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

2) 使用密钥远程其他主机

[root@proxy ~]# pssh -h host.txt echo hello

[1] 12:02:08 [SUCCESS] host2

[2] 12:02:08 [SUCCESS] root@host1

[3] 12:02:08 [SUCCESS] host3

3)使用密钥远程其他主机，将标准输出信息写入到/tmp目录

[root@proxy ~]# pssh -h host.txt -o /tmp/ echo hello

步骤四：批量、多并发拷贝数据到其他主机

1）语法格式

[root@proxy ~]# man pscp.pssh       //通过man帮助查看工具选项的作用

pscp.pssh提供并发拷贝文件功能

-r    递归拷贝目录

其他选项基本与pssh一致

2)将本地的/etc/hosts拷贝到远程主机的/tmp目录下

[root@proxy ~]# pscp.pssh -h host.txt /etc/hosts / tmp

3)递归将本地的/etc目录拷贝到远程主机的/tmp目录下

[root@proxy ~]# pscp.pssh -r -h host.txt /etc /tmp

步骤五：批量、多并发从其他主机下载数据到本机

1）语法格式

[root@proxy ~]# man pslurp            //通过man帮助查看工具选项的作用

pslurp提供远程下载功能

选项与pscp.pssh基本一致

2)将远程主机的/etc/passwd，拷贝到当前目录下，存放在对应IP下的pass文件中

[root@proxy ~]# pslurp -h host.txt /etc/passwd /pass

注意：最后的pass是文件名

[root@proxy ~]# ls /root/host1

代表的是当前路径,不是根目录

pass

3)将远程主机的/etc/passwd目录，拷贝到media下，存放在对应IP下的pass文件

[root@proxy ~]# pslurp -h host.txt -L /media /etc/passwd /pass

[root@proxy ~]# ls /media/

host1 host2 host3 #目录

代表的/media,不是根目录

[root@proxy ~]# ls /media/host1

pass #文件

步骤六：批量、多并发杀死其他主机的进程

1）语法格式

[root@proxy ~]# man pnuke             //通过man帮助查看工具选项的作用

pnuke提供远程杀死进程的功能

选项与pssh基本一致

2)将远程主机上的sleep进程杀死

[root@proxy ~]# pnuke -h host.txt sleep

[1] 14:23:56 [SUCCESS] root@host1

[2] 14:23:56 [SUCCESS] host3

[3] 14:23:56 [SUCCESS] host2

3)将远程主机上的test相关脚本都杀死（如：test1,testtt,test2等等）

[root@proxy ~]# pnuke -h host.txt test

4)将远程主机上的test.sh脚本杀死

[root@proxy ~]# pnuke -h host.txt test.sh

####################################

**II-Cluster-Day01**

**存储概述**

存储是根据不同的应用环境通过采取合理.安全,有效的方式将数据保存到某些介质上并能保证有效的时间;

一方面它是数据临时或长期驻留的物理媒介;

另一方面,它是保证数据完整安全存放的方式或行为;

存储就是把这两个方面结合起来,向客户提供一套数据存放解决方案

**存储技术分类**

SCSI小型计算机系统接口

DAS直连式存储; 存储容量有限

NAS网络技术存储(文件系统) Samba NFS httpd

SAN存储区域网络(块存储) iscsi

FC光纤通道

**SCSI技术**

Small Computer System Interface的简称

作为输入/输出接口

主要用于硬盘,光盘,磁带机等设备

**DAS技术**

Direct-Attached Storage的简称

将存储设备通过SCSI接口或光纤通道直接连接到计算机上

不能实现数据与其他主机的共享

占用服务器操作系统资源,如CPU, IO等

数据量越大,性能越差

**NAS技术**

Network-Attached Storage的简称

一种专用数据存储服务器,以数据为中心,将存储设备与服务器彻底分离,集中管理数据,从而释放带宽,提高性能,降低总拥有成本,保护投资

用户通过TCP/IP协议访问数据,采用标准的NFS/HTTP/CIFS等

**SAN技术**

Storage Area Network的简称

通过光纤交换机,光纤路由器,光纤集线器等设备将磁盘阵列,磁带等存储设备与相关服务器连接起来,形成高速专网网络

组成部分:

如路由器、光纤交换机

接口: 如SCSI、FC

通信协议: 如IP、SCSI

**FC光纤通道**

Fibre Channel

一种适用于千兆数据传输的,成熟而安全解决方案

与传统的SCSI相比,FC提供更高的数据传输速率,更远的传输距离,更多的设备连接支持以及更稳定的性能,更简易的安装

主要组件:

光纤

HBA(主机总线适配器)

FC交换机

FC交换机交换拓扑

1) 点到点:point-to-point :简单将两个设备互连

2) 已裁定的环路:arbitrated loop 可多达126个设备共享一段信道或环路

3) 交换式拓扑:switched fabric 所有设备通过光纤交换机互连

**案例1：配置iSCSI服务**

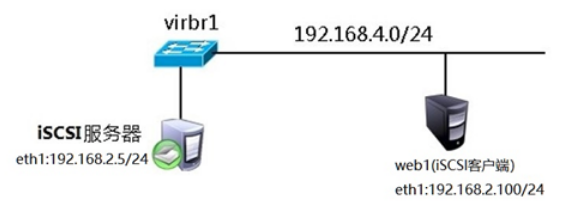
本案例要求先搭建好一台iSCSI服务器，并将整个磁盘共享给客户端：

1) 服务器上要额外配置一块硬盘

2) 服务端安装target，并将新加的硬盘配置为iSCSI 的共享磁盘

3) 在客户端上安装initiator，挂在服务器iSCSI，要求实现开机自动挂载

使用2台RHEL7虚拟机，其中一台作为iSCSI服务器（192.168.2.5）、另外一台作为测试用的客户机（192.168.2.100），如图所示，主机网络地址配置如表所示。



|  |  |
| --- | --- |
| **主机名** | **网络参数配置** |
| proxy(作为iscsi服务器) | eth1:192.168.2.5/24 |
| web1(作为iscsi客户端) | eth1:192.168.2.100/24 |

步骤一：安装iSCSI服务器软件

1）使用yum安装targetcli软件包

[root@proxy ~]# yum -y install targetcli

[root@proxy ~]# yum info targetcli

步骤二：通过命令行配置iSCSI服务

1）真实主机准备底层存储磁盘

手动添加一个新的20G的分区

[root@proxy ~]# parted /dev/vdb mklabel gpt #非交互式分区(mklabel=mktable)

[root@proxy ~]# parted /dev/vdb mkpart primary 1 100%

2)使用targetcli定义后端存储

设置需要将哪个设备共享给其他主机，这里将/dev/vdb1设置为后端共享磁盘。

[root@proxy ~]# targetcli

/> ls

/> backstores/block create store /dev/vdb1 #store为任意名称

3）创建iqn对象

给iSCSI共享设置一个共享名称，客户端访问时需要使用该共享名称。

/> /iscsi create iqn.2018-01.cn.tedu:server1

4) 授权客户机访问

类似于一个密码，设置ACL访问控制，拥有iqn.2018-01.cn.tedu:client1这个字符串的客户端才有权限访问服务器。

/> iscsi/iqn.2018-01.cn.tedu:server1/tpg1/acls create iqn.2018-01.cn.tedu:client1

5) 绑定存储

将iqn共享名称（iqn.2018-01.cn.tedu:server1）与后端实际的存储设备（vdb）绑定。

/>iscsi/iqn.2018-01.cn.tedu:server1/tpg1/luns create /backstores/block/store # block后面的store必须与前面步骤2定义后端存储create创建的名称一致。

6) 存储绑定服务监听的地址，并保存配置

/> iscsi/iqn.2018-01.cn.tedu:server1/tpg1/portals/ create 0.0.0.0

/> saveconfig #保存

/> exit

步骤三：服务管理

1. 启动服务

[root@proxy ~]# systemctl start target.service

1. 查看端口信息

[root@proxy ~]# ss -antulp | grep :3260

3）关闭防火墙与SELinux

[root@proxy ~]# systemctl stop firewalld

[root@proxy ~]# setenforce 0

步骤四：客户端访问(web1作为客户端的角色)

1. 客户端安装软件并启动服务

[root@web1 ~]# yum -y install iscsi-initiator-utils

2）设置本机的iqn名称

[root@web1 ~]# vim /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.2018-01.cn.tedu:client1 #必须跟服务器上配置的ACL一致！

3）发现远程target存储 提示：参考man iscsiadm

[root@web1 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.5 --discover

[root@web1 ~]# iscsiadm --mode node --targetname iqn.2018-01.cn.tedu:server1 --portal 192.168.2.5:3260 --login

[root@web1 ~]# systemctl restart iscsi

Warning: iscsi.service changed on disk. Run 'systemctl daemon-reload' to reload units.

[root@web1 ~]# systemctl daemon-reload

[root@web1 ~]# systemctl restart iscsi

1. 分区,格式化,挂载

[root@web1 ~]# parted /dev/sda mklabel gpt

[root@web1 ~]# parted /dev/sda mkpart primary 1 800

[root@web1 ~]# mkfs.xfs /dev/sda1

[root@web1 ~]# mount /dev/sda1 /mnt

[root@web1 ~]# umount /mnt

步骤四：**附加课外实验：**多台FTP或者http主机使用共享存储。

这里以FTP为例，web1和web2主机都安装vsftpd软件，使用统一的后端共享存储设备。

1) web1操作(延续前面步骤三的实验)：

[root@web1 ~]# mkdir /var/ftp/

[root@web1 ~]# mount /dev/sda1 /var/ftp/

[root@web1 ~]# yum -y install vsftpd

[root@web1 ~]# sed -i 's/^#anon/anon/' /etc/vsftpd/vsftpd.conf #修改vsftpd配置文件，开启匿名上传功能。将下面2行默认的注释行打开。

#anon\_upload\_enable=YES #anon\_mkdir\_write\_enable=YES

[root@web1 ~]# chmod 777 /var/ftp/pub

[root@web1 ~]# systemctl start vsftpd

[root@web1 ~]# systemctl enable vsftpd

2) 真实主机访问web1的FTP共享，并任意上传一个文件到FTP服务器。

连接到服务器后，进入pub共享目录，将真实主机的任意文件拖拽到FTP的共享目录下（pub目录）。注意：仅pub目录有读写权限。

3）当web1宕机后，web2主机可以继续使用iscsi提供FTP共享服务。

Web1关闭vsftpd服务，卸载iscsi挂载。

[root@web1 ~]# systemctl stop vsftpd

[root@web1 ~]# umount /var/ftp

添加iSCSI共享（web2操作）。

[root@web2 ~]# vim /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.2018-01.cn.tedu:client1 #必须跟服务器上配置的ACL一致！

[root@web2 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.5 --discover

[root@web2 ~]# iscsiadm --mode node --targetname iqn.2018-01.cn.tedu:server1 --portal 192.168.2.5:3260 --login

安装部署vsftpd软件(web2操作)。

[root@web2 ~]# yum -y install vsftpd

[root@web2 ~]# sed -i 's/^#anon/anon/' /etc/vsftpd/vsftpd.conf

[root@web2 ~]# chmod 777 /var/ftp/pub/

[root@web2 ~]# systemctl start vsftpd

[root@web2 ~]# systemctl enable vsftpd

4) 真实主机访问web2的FTP共享，查看共享里现有的数据，并任意上传一个新文件到FTP服务器

**案例2：部署Multipath多路径环境**

通过Multipath，实现以下目标：

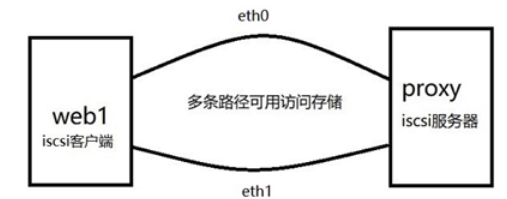
1) 在共享存储服务器上配置iSCSI，为应用服务器共享存储空间

2) 应用服务器上配置iSCSI，发现远程共享存储

3) 应用服务器上配置Multipath，将相同的共享存储映射为同一个名称

|  |  |
| --- | --- |
| **节点及网卡** | **地址及应用** |
| proxy(iscsi服务器) | eth0:192.168.4.5/24 eth1:192.168.2.5/24 |
| web1(iscsi客户端) | eth0:192.168.4.100/24 eth1:192.168.2.100/24 |

多路径示意图:



步骤一：存储服务器上添加额外的磁盘（如果已经完成案例1，此步骤可以忽略）

使用KVM软件新建（或修改）虚拟机，为虚拟机额外添加一块硬盘。

步骤二：存储服务器上安装并配置共享存储（如果已经完成案例1，此步骤可用忽略）

1. 定义后端存储

[root@proxy ~]# targetcli

/> ls

/> backstores/block create store /dev/vdb1

2）创建iqn对象

/> /iscsi create iqn.2018-01.cn.tedu:server1

3) 授权客户机访问

/> iscsi/iqn.2018-01.cn.tedu:server1/tpg1/acls create iqn.2018-01.cn.tedu:client1

4) 绑定存储

/>iscsi/iqn.2018-01.cn.tedu:server1/tpg1/luns create /backstores/block/store

5) 绑定存储绑定监听地址，并保存配置

/> iscsi/iqn.2018-01.cn.tedu:server1/tpg1/portals/ create 0.0.0.0

/> saveconfig

/> exit

步骤三：在client服务器上安装并配置iSCSI客户端

1）安装客户端软件（前面的案例1已经完成的情况下，可以忽略此步骤）

[root@web1 ~]# yum install -y iscsi-initiator-utils

2）发现存储服务器的共享磁盘

因为有两条链路都可以连接到共享存储，所以需要在两条链路上都发现它。

注意：两次发现使用的IP地址不同！

[root@web1 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.5 --discover

[root@web1 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.5 --discover

3）登陆共享存储

只需要将iscsi服务重启就可以自动登陆（就不需要再login了）。

在login之前，只能看到本地的存储，登陆之后，将会多出两块新的硬盘。

[root@web1 ~]# systemctl restart iscsi

[root@web1 ~]# lsblk | head -5

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 20G 0 disk

└─sda1 8:1 0 762M 0 part

sdb 8:16 0 20G 0 disk

└─sdb1 8:17 0 762M 0 part

提示：登陆的是同一个服务器的同一个iSCSI，但客户端看到的是两个独立的设备，sda和sdb。其实，这两个设备是同一个设备。

4）设置开机自启动

iscsi用于自动login远程存储，iscsid是守护进程。

[root@web1 ~]# systemctl enable iscsid

[root@web1 ~]# systemctl enable iscsi

步骤四：配置Multipath多路径

1. 安装多路径软件包

[root@web1 ~]# yum -y install device-mapper-multipath

2）生成配置文件

[root@web1 ~]# cp /usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf /etc

3）获取wwid(设备号)

登陆共享存储后，系统多了两块硬盘，这两块硬盘实际上是同一个存储设备。应用服务器使用哪个都可以，但是如果使用sdb时，sdb对应的链路出现故障，它不会自动切换到sda。

为了能够实现系统自动选择使用哪条链路，需要将这两块磁盘绑定为一个名称。

通过磁盘的wwid来判定哪些磁盘是相同的。

取得一块磁盘wwid的方法如下：

[root@web1 ~]# /usr/lib/udev/scsi\_id --whitelisted --device=/dev/sdb

360014058d3cbef2283c4cf192d4f4876

[root@web1 ~]# /usr/lib/udev/scsi\_id --whitelisted --device=/dev/sda

360014058d3cbef2283c4cf192d4f4876

4）修改配置文件

首先声明自动发现多路径：

[root@web1 ~]# vim /etc/multipath.conf

......

multipaths {

multipath {

wwid 360014058d3cbef2283c4cf192d4f4876

alias mpatha

}

} #添加上面的几行

......

步骤五：启用Multipath多路径，并测试

注意：如果做案例1时，已经挂载了iSCSI设备，一定要先umount卸载掉再启动多路径。

1）启动Multipath，并设置为开机启动

[root@web1 ~]# systemctl start multipathd.service

[root@web1 ~]# systemctl enable multipathd

2）检查多路径设备文件

如果多路径设置成功，那么将在/dev/mapper下面生成名为mpatha的设备文件：

[root@web1 ~]# ls /dev/mapper/

control mpatha mpatha1 rhel-root rhel-swap

3）对多路径设备文件执行分区、格式化、挂载操作

提示：如果前面已经对iscsi做过分区操作，则这里可以直接识别到mpatha1（就不需要再次分区了）。

新的分区名称应该是/dev/mapper/mpathap1，如果该文件不存在，则执行以下命令进行配置的重新载入：

[root@web1 ~]# ls /dev/mapper/ ＃再次查看，将会看到新的分区

创建目录并挂载（如果已经格式化，这里就不需要再次格式化，直接挂载即可）：

[root@web1 ~]# mkfs.xfs /dev/mapper/mpatha1

[root@web1 ~]# mkdir /data

[root@web1 ~]# mount /dev/mapper/mpatha1 /data/

[root@web1 ~]# df -h /data/

4）验证多路径

查看多路径，sda和sdb都是running状态。

[root@web1 ~]# multipath -rr

reload: mpatha (360014058d3cbef2283c4cf192d4f4876) undef LIO-ORG ,store

size=20G features='0' hwhandler='0' wp=undef

|-+- policy='service-time 0' prio=1 status=undef

| `- 2:0:0:0 sda 8:0 active ready running

`-+- policy='service-time 0' prio=1 status=undef

`- 3:0:0:0 sdb 8:16 active ready running

关闭某个链路后，再次查看效果，此时会发现sdb为运行失败状态。

[root@web1 ~]# nmcli connection down eth0 (=ifconfig eth0 down)

[root@web1 ~]# multipath -rr

reload: mpatha (360014058d3cbef2283c4cf192d4f4876) undef LIO-ORG ,store

size=20G features='0' hwhandler='0' wp=undef

|-+- policy='service-time 0' prio=1 status=undef

| `- 2:0:0:0 sda 8:0 active ready running

`-+- policy='service-time 0' prio=0 status=undef

`- 3:0:0:0 sdb 8:16 active faulty running

使用-ll选项查看，仅sda为有效运行状态。

[root@web1 ~]# multipath -ll

mpatha (360014058d3cbef2283c4cf192d4f4876) dm-2 LIO-ORG ,store

size=20G features='0' hwhandler='0' wp=rw

`-+- policy='service-time 0' prio=1 status=active

`- 2:0:0:0 sda 8:0 active ready running

**案例3：配置并访问NFS共享**

服务器利用NFS机制发布2个共享目录，要求如下：

1) 将目录/root共享给192.168.2.100，客户机的root用户有权限写入

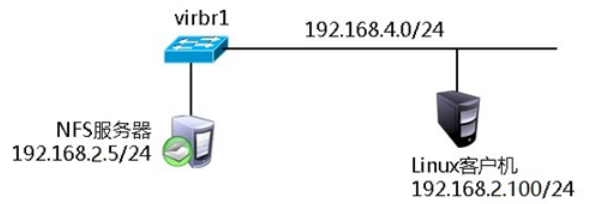
2) 将/usr/src目录共享给192.168.2.0/24网段，只开放读取权限

从客户机访问NFS共享：

1) 分别查询/挂载上述NFS共享目录

2) 查看挂载点目录，并测试是否有写入权限

使用2台RHEL7虚拟机，其中一台作为NFS共享服务器（192.168.2.5）、另外一台作为测试用的Linux客户机（192.168.2.100），如图所示。



步骤一：配置NFS服务器，发布指定的共享

1）确认服务端程序、准备共享目录

软件包nfs-utils用来提供NFS共享服务及相关工具，而软件包rpcbind用来提供RPC协议的支持，这两个包在RHEL7系统中一般都是默认安装的：

[root@proxy ~]# rpm -q nfs-utils rpcbind

nfs-utils-1.3.0-0.48.el7.x86\_64

rpcbind-0.2.0-42.el7.x86\_64

根据本例的要求，需要作为NFS共享发布的有/root、/usr/src这两个目录：

[root@proxy ~]# ls -ld /root /usr/src/

dr-xr-x---. 35 root root 4096 1月 15 18:52 /root

drwxrwxr-x+ 4 root root 4096 1月 15 17:35 /usr/src/

2）修改/etc/exports文件，添加共享目录设置

默认情况下，来自NFS客户端的root用户会被自动降权为普通用户，若要保留其root权限，注意应添加no\_root\_squash控制参数(没有该参数，默认root会被自动降级为普通账户)；另外，限制只读的参数为ro、可读可写为rw，相关配置操作如下所示：

[root@proxy ~]# vim /etc/exports

/root 192.168.2.100(rw,no\_root\_squash)

/usr/src 192.168.2.0/24(ro)

3）启动NFS共享相关服务，确认共享列表

依次启动rpcbiind、nfs服务：

[root@proxy ~]# systemctl restart rpcbind ; systemctl enable rpcbind

[root@proxy ~]# systemctl restart nfs ; systemctl enable nfs

使用showmount命令查看本机发布的NFS共享列表：

[root@proxy ~]# showmount -e localhost

Export list for localhost:

/usr/src 192.168.2.0/24

/root 192.168.2.100

步骤二：从客户机访问NFS共享

1）启用NFS共享支持服务

客户机访问NFS共享也需要rpcbind服务的支持，需确保此服务已开启：

[root@web1 ~]# systemctl restart rpcbind ; systemctl enable rpcbind

2）查看服务器提供的NFS共享列表

[root@web1 ~]# showmount -e 192.168.2.5

Export list for 192.168.2.5:

/usr/src 192.168.2.0/24

/root 192.168.2.100

3）从客户机192.168.2.100访问两个NFS共享，并验证权限

将远程的NFS共享/root挂载到本地的/root5文件夹，并验证可读可写：

[root@web1 ~]# mkdir /root5                       //建立挂载点

[root@web1 ~]# mount 192.168.2.5:/root /root5 //挂载NFS共享目录

[root@web1 ~]# df -hT /root5                  //确认挂载结果

Filesystem Type Size Used Avail Use% Mounted on

192.168.2.5:/root nfs 50G 15G 33G 31% /root5

[root@web1 ~]# cd /root5                  //切换到挂载点

[root@web1 root5]# echo "NFS Write Test" > test.txt     //测试写入文件

[root@web1 root5]# cat test.txt                      //测试查看文件

NFS Write Test

将远程的NFS共享/usr/src挂载到本地的/mnt/nfsdir，并验证只读：

[root@web1 ~]# mkdir /mnt/nfsdir          //建立挂载点

[root@web1 ~]# mount 192.168.2.5:/usr/src /mnt/nfsdir/    //挂载NFS共享目录

[root@web1 ~]# df -hT /mnt/nfsdir/        //确认挂载结果

Filesystem Type Size Used Avail Use% Mounted on

192.168.2.5:/usr/src nfs 50G 15G 33G 31% /mnt/nfsdir

[root@web1 ~]# cd /mnt/nfsdir/                  //切换到挂载点

[root@web1 nfsdir]# ls                                 //读取目录列表

debug install.log kernels test.txt

[root@web1 nfsdir]# echo "Write Test." > pc.txt //尝试写入文件失败

-bash: pc.txt: 只读文件系统

**！！！！** 如果从未授权的客户机访问NFS共享，将会被拒绝。比如从NFS服务器本机尝试访问自己发布的/root共享（只允许192.168.2.100访问），结果如下所示：

[root@proxy ~]# mkdir /root5

[root@proxy ~]# mount 192.168.2.5:/root /root5

mount.nfs: access denied by server while mounting 192.168.2.5:/root

4）设置永久挂载

[root@web1 ~]# vim /etc/fstab

192.168.2.5:/usr/src /mnt/nfsdir         nfs default,ro 0 0

192.168.2.5:/root /root5          nfs default 0 0

**案例4：编写udev规则(真机操作)**

编写udev规则，实现以下目标：

1) 当插入一个U盘时，该U盘自动出现一个链接称为udisk

2) U盘上的第1个分区名称为udisk1，以此类推

3) 终端上出现提示信息”udisk plugged in”

问题：加载一个USB设备后，系统可能识别为sda也可能识别为sdb，能不能固定呢？

对于Linux kernel 2.6及更新的操作系统版本会将设备的相关信息动态写入/sys文件系统中，而udev程序可以通过读取这些设备系信息，并根据自己的udev规则进行设备管理器，实现如下功能：

处理设备命名

决定要创建哪些设备文件或链接

决定如何设置属性

决定触发哪些事件

udev默认规则存放在/etc/udev/rules.d目录下，通过修改此目录下的规则实现设备的命名、属性、链接文件等。

1）查看设备属性

加载USB设备的同时实时查看设备的相关属性，可以使用monitor指令。

[root@room9pc01 ~]# udevadm monitor --property

如果设备已经加载则无法使用monitor查看相关属性。可以使用下面的命令查看设备属性。

[root@room9pc01 ~]# udevadm info --query=path --name=/dev/sdb

/devices/pci0000:00/0000:00:14.0/usb1/1-9/1-9:1.0/host6/target6:0:0/6:0:0:0/block/sdb

[root@room9pc01 ~]# udevadm info --query=property --path=/block/sdb

单独查看某个磁盘分区的属性信息。

[root@room9pc01 ~]# udevadm info --query=property --path=/block/sdb/sdb1

2）编写udev规则文件（实现插拔USB设备时有屏幕提示信息）

注意：修改规则文件不能照抄，这里的变量都是需要根据实际情况而修改的！！！

每个设备的属性都有所不同！！！一定要根据前面查询的info信息填写。

[root@room9pc01 ~]# cd /etc/udev/rules.d/

[root@room9pc01 rules.d]# vim usb.rules #必须以.rules结尾

ENV{ID\_VENDOR}=="KINGSTON",ENV{ID\_SERIAL\_SHORT}=="001CC05FE92CAC50791D22D0",RUN+="/usr/bin/wall usbusb"

在virt-manager中删除、添加USB设备，测试自己的udev规则是否成功。

排错方法：通过查看/var/log/messages日志文件排错。

3）继续修改规则文件（实现给分区命名）

[root@room9pc01 ~]# cd /etc/udev/rules.d/

[root@room9pc01 rules.d]# vim usb.rules

ENV{ID\_VENDOR}=="KINGSTON",ENV{DEVTYPE}=="partition",SYMLINK+="usb%n"

[root@room9pc01 rules.d]# ls /dev/sdb1

/dev/sdb1

[root@room9pc01 rules.d]# ls /dev/usb1

/dev/usb1

在virt-manager中删除、添加USB设备，测试自己的udev规则是否成功。

4）继续修改规则文件（修改设备所有者和权限）

[root@room9pc01 rules.d]# vim usb.rules

ENV{ID\_VENDOR}=="KINGSTON",ENV{DEVTYPE}=="partition",SYMLINK+="usb%n",OWNER="dc",GROUP="dc",MODE="0644"

[root@room9pc01 rules.d]# ll /dev/sdb1

brw-r--r-- 1 dc dc 8, 17 1月 29 11:44 /dev/sdb1

5）继续修改规则文件（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@room9pc01 rules.d]# vim usb.rules

ACTION=="add",ENV{ID\_VENDOR}=="KINGSTON",RUN+="/usr/bin/systemctl start httpd"

ACTION=="remove",ENV{ID\_VENDOR}=="KINGSTON",RUN+="/usr/bin/systemctl stop httpd"

在virt-manager中删除、添加USB设备，测试自己的udev规则是否成功。

[root@room9pc01 rules.d]# ss -antulp | grep httpd #服务关闭

**总结知识点:**

**udev规则文件,常见指令操作符如下表所示**

|  |  |
| --- | --- |
| **选项** | **值** |
| == | 表示匹配 |
| != | 表示不匹配 |
| = | 指定赋予的值 |
| += | 添加新值 |
| := | 指定值(赋值),且不允许被替换 |
| NAME=”udisk” | 定义设备名称 |
| SYMLINK+=”data1” | 定义设备的别名(快捷方式) |
| OWNER=”student” | 定义设备的所有者 |
| GROUP=”student” | 定义设备的所属组 |
| MODE=”0600” | 定义设备的权限 |
| ACTION==”add” | 判断设备的操作动作(添加或删除设备等) |
| KERNEL==”sd[a-z]1” | 判断设备的内核名称 |
| RUN+=程序 | 为设备添加程序 |

udev常用替代变量：

%k：内核所识别出来的设备名，如sdb1

%n：设备的内核编号，如sda3中的3

%p：设备路径，如/sys/block/sdb/sdb1

####################################

**II-Cluster-Day02**

**集群简介**

1. 一组通过高速网络互联的计算组,并以单一系统的模式加以管理
2. 将很多服务器集中起来一起,提供同一种服务,在客户端看来就像是只有一个服务器
3. 可以在付出较低成本的情况下获得在性能、可靠性、灵活性方面的相对较高的收益
4. 任务调度是集群系统中的核心技术

**集群的目的**

1. 提高性能,如计算密集型应用: 天气预报 核试验模拟
2. 降低成本,相对百万美元级的超级计算机,价格便宜
3. 提高可扩展性,只要增加集群节点即可
4. 增强可靠性,多个节点完成相同功能,避免单点失败

**集群分类**

1) 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

1. 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

1. 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

**LVS概述**

1. Linux虚拟服务器(LVS)是章文嵩在国防科技大学就读博士期间创建的
2. LVS可以实现高可用、可伸缩的web、Mail、Cache和Media等网络服务
3. 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、低成本的服务器应用集群

**LVS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:服务器群组层,由一组实际运行应用服务的服务器组成

底端:数据共享存储层,提供共享存储空间的存储区域

**LVS术语**

Director Server:调度服务器, 将负载分发到Real Server的服务器

Real Server:真实服务器, 真正提供应用服务的服务器

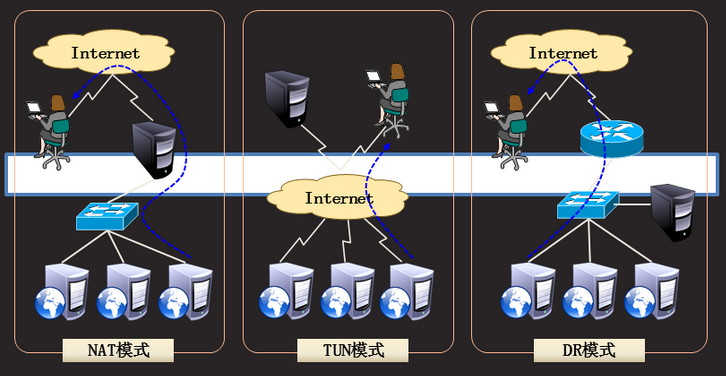
VIP:虚拟IP地址, 公布给用户访问的虚拟IP地址 (192.168.4.5)

RIP:真实IP地址, 集群节点上使用的IP地址 (192.168.2.100, 192.168.2.200)

DIP:调度器连接节点服务器的IP地址 (192.168.2.5)

CIP:客户端计算机的IP地址 (192.168.4.10)

**LVS工作模式:**



**VS/NAT**

1) 通过网络地址转换实现的虚拟服务器

2) 大并发访问时,调度器的性能成为瓶颈

**VS/DR**

1. 直接使用路由技术实现虚拟服务器
2. 节点服务器需要配置VIP,注意MAC地址广播

**VS/TUN**

通过隧道方式实现虚拟服务器

**负载均衡调度算法**

LVS目前实现了10种调度算法

常用调度算法有4种:

1. 轮询(Round Robin)

将客户端请求平均分发到Real Server

1. 加权轮询(Weighted Round Robin)

根据Real Server权重值进行轮询调度

1. 最少连接(Least Connections)

选择连接数最少的服务器

1. 加权最少连接(Weighted Least Connections)

根据Real Server权重值,选择连接数最少的服务器

1. 源地址散列(Source Hashing)

根据请求的目标IP地址,作为散列键(Hash Key)从静态分配的散列表找出对应的服务器------ip\_hash

**其他调度算法:**

1. 基于局部性的最少链接
2. 带复制的基于局部性最少链接
3. 目标地址散列(Destination Hashing)
4. 最短的期望的延迟
5. 最少队列调度

## **案例1：ipvsadm命令用法**

准备一台Linux服务器，安装ipvsadm软件包，练习使用ipvsadm命令，实现如下功能：

1) 使用命令添加基于TCP一些的集群服务

2) 在集群中添加若干台后端真实服务器

3) 实现同一客户端访问，调度器分配固定服务器

4) 会使用ipvsadm实现规则的增、删、改

5) 保存ipvsadm规则

常用ipvsadm命令语法格式:

|  |  |
| --- | --- |
| **命令选项** | **含义** |
| ipvsadm -A | 添加虚拟服务器 |
| ipvsadm -E | 修改虚拟服务器 |
| ipvsadm -D | 删除虚拟服务器 |
| ipvsadm -C | 清空所有 |
| ipvsadm -a | 添加真实服务器 |
| ipvsadm -e | 修改真实服务器 |
| ipvsadm -d | 删除真实服务器 |
| ipvsadm -L | 查看LVS规则表 |
| -s [rr|wrr|lc|wlc] | 指定集群算法 |
| -r | 指定真实服务器(Real Server)的地址 |
| -w | 为节点服务器设置权重,默认为1 |

rr: 轮询 wrr:加权轮询 lc:最少链接 wlc:加权最少链接

|  |  |
| --- | --- |
| **命令** | **含义** |
| ipvsadm -A -t|u 192.168.4.5:80 -s [算法] | 添加虚拟服务器,协议为tcp(-t)或者udp(-u) |
| ipvsadm -E -t|u 192.168.4.5:80 -s [算法] | 修改虚拟服务器,协议为tcp或者udp |
| ipvsadm -D -t|u 192.168.4.5:80 | 删除虚拟服务器,协议为tcp或者udp |
| ipvsadm -C | 清空所有 |
| ipvsadm -a -t|u 192.168.4.5:80 -r 192.168.2.100 [-g|i|m] [-w 权重] | 添加真实服务器  -g : DR模式(默认)  -i : 隧道模式  -m : NAT模式 |
| ipvsadm -e -t|u 192.168.4.5:80 -r 192.168.2.100 [-g|i|m] [-w 权重] | 修改真实服务器 |
| ipvsadm -d -t|u 192.168.4.5:80 -r 192.168.2.100 | 删除真实服务器 |
| ipvsadm -Ln | 查看LVS规则表 |

步骤一：使用命令增、删、改LVS集群规则

1）创建LVS虚拟集群服务器（算法为加权轮询：wrr）

[root@proxy ~]# yum -y install ipvsadm

[root@proxy ~]# ipvsadm -A -t 192.168.4.5:80 -s wrr

[root@proxy ~]# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.5:80 wrr

1. 为集群添加若干real server

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100 -m -w 1

[root@proxy ~]# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.5:80 wrr

-> 192.168.2.100:80 Masq 1 0 0

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.200 -m -w 2

[root@proxy ~]# ipvsadm –a -t 192.168.4.5:80 -r 192.168.2.201 -m -w 3

[root@proxy ~]# ipvsadm –a -t 192.168.4.5:80 -r 192.168.2.202 -m -w 4

3）修改集群服务器设置(修改调度器算法，将加权轮询修改为轮询)

[root@proxy ~]# ipvsadm -E -t 192.168.4.5:80 -s rr

[root@proxy ~]# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.5:80 rr

-> 192.168.2.100:80 Masq 1 0 0

-> 192.168.2.200:80 Masq 2 0 0

-> 192.168.2.201:80 Masq 2 0 0

-> 192.168.2.202:80 Masq 1 0 0

4）修改read server（使用-g选项，将模式改为DR模式）

[root@proxy ~]# ipvsadm -e -t 192.168.4.5:80 -r 192.168.2.202 -g

5）查看LVS状态

[root@proxy ~]# ipvsadm -Ln

6）创建另一个集群（算法为最少连接算法；使用-m选项，设置工作模式为NAT模式）

[root@proxy ~]# ipvsadm -A -t 192.168.4.5:3306 -s lc

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:3306 -r 192.168.2.100 -m

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:3306 -r 192.168.2.200 -m

6）永久保存所有规则

[root@proxy ~]# ipvsadm-save -n > /etc/sysconfig/ipvsadm

7）清空所有规则

[root@proxy ~]# ipvsadm -C

**案例2：部署LVS-NAT集群**

使用LVS实现NAT模式的集群调度服务器，为用户提供Web服务：

1) 集群对外公网IP地址为192.168.4.5

2) 调度器内网IP地址为192.168.2.5

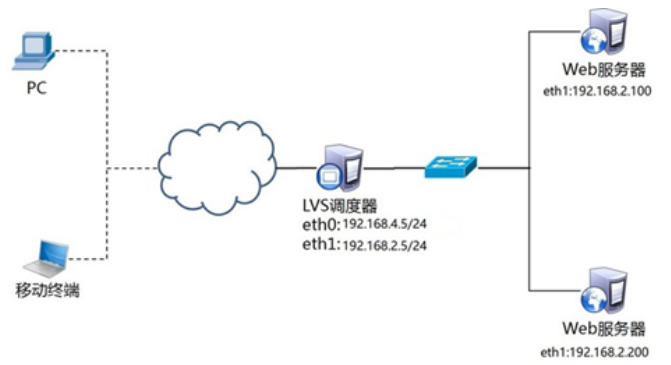
3) 真实Web服务器地址分别为192.168.2.100、192.168.2.200

4) 使用加权轮询调度算法，真实服务器权重分别为1和2

实验拓扑结构主机配置细节如表所示。

|  |  |
| --- | --- |
| **主机名** | **IP地址** |
| client | eth0:192.168.4.10/24 |
| proxy | eth0:192.168.4.5/24  eth1:192.168.2.5/24 |
| web1 | 关闭eth0:192.168.4.100(之前实验的配置)  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 |

使用4台虚拟机，1台作为Director调度器、2台作为Real Server、1台客户端，拓扑结构如图-1所示，注意：web1和web2必须配置网关地址。



步骤一：配置基础环境

1）设置Web服务器(以web1为例), web2也要操作

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "<h1>192.168.2.100" > /var/www/html/index.html

2）启动Web服务器软件

[root@web1 ~]# systemctl restart httpd

3)关闭防火墙与SELinux

[root@web1 ~]# systmctl stop firewalld

[root@web1 ~]# setenforce 0

步骤二：部署LVS-NAT模式调度器

1)确认调度器的路由转发功能(如果已经开启，可以忽略)

[root@proxy ~]# echo 1 > /proc/sys/net/ipv4/ip\_forward

[root@proxy ~]# cat /proc/sys/net/ipv4/ip\_forward

1

[root@proxy ~]# echo "net.ipv4.ip\_forward = 1" >> /etc/sysctl.conf

#修改配置文件，设置永久规则

2）创建集群服务器

[root@proxy ~]# yum -y install ipvsadm

[root@proxy ~]# ipvsadm -A -t 192.168.4.5:80 -s wrr

2）添加真实服务器

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100 -w 1 -m

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.200 -w 1 -m

3）查看规则列表，并保存规则

[root@proxy ~]# ipvsadm -Ln

[root@proxy ~]# ipvsadm-save -n > /etc/sysconfig/ipvsadm

步骤三：客户端测试

客户端使用curl命令反复连接http://192.168.4.5，查看访问的页面是否会轮询到不同的后端真实服务器

[root@client ~]# curl 192.168.4.5

<h1>192.168.2.100

[root@client ~]# curl 192.168.4.5

<h1>192.168.2.200

**案例3：部署LVS-DR集群**

使用LVS实现DR模式的集群调度服务器，为用户提供Web服务：

1) 客户端IP地址为192.168.4.10

2) LVS调度器VIP地址为192.168.4.15

3) LVS调度器DIP地址设置为192.168.4.5

4) 真实Web服务器地址分别为192.168.4.100、192.168.4.200

5) 使用加权轮询调度算法，web1的权重为1，web2的权重为2

说明：

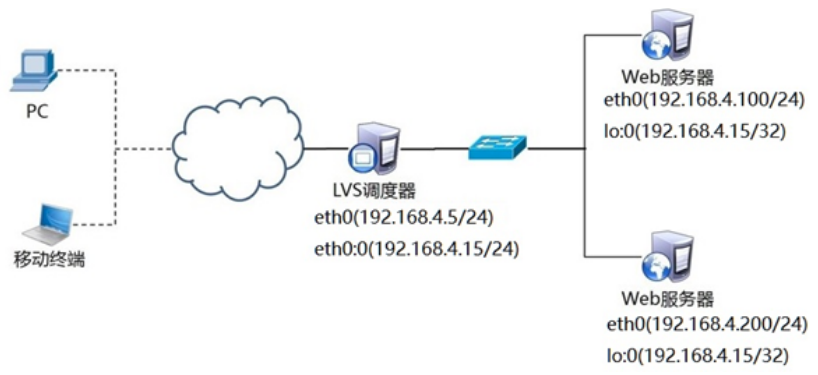
CIP是客户端的IP地址；

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）。

使用4台虚拟机，1台作为客户端、1台作为Director调度器、2台作为Real Server，拓扑结构如图所示。实验拓扑结构主机配置细节如表所示。



|  |  |
| --- | --- |
| **主机名** | **网络配置** |
| 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  注意子网掩码必须是32 |
| web2 | eth0:192.168.4.200/24  lo:0 192.168.4.15/32  注意子网掩码必须是32 |

步骤一：配置实验网络环境

1）设置Proxy代理服务器的VIP(4.15)和DIP(4.5)

注意：为了防止冲突，VIP必须要配置在网卡的虚拟接口！！！

[root@proxy ~]# cd /etc/sysconfig/network-scripts/

[root@proxy network-scripts]# cp ifcfg-eth0{,:0}

[root@proxy network-scripts]# ls ifcfg-eth0\*

ifcfg-eth0 ifcfg-eth0:0

[root@proxy network-scripts]# vim ifcfg-eth0:0

TYPE=Ethernet

BOOTPROTO=none

NAME=eth0:0

DEVICE=eth0:0

ONBOOT=yes

IPADDR=192.168.4.15

PREFIX=24

[root@proxy network-scripts]# systemctl restart network

[root@proxy network-scripts]# ip a s eth0 #查看IP地址

2: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

link/ether 52:54:00:15:1f:27 brd ff:ff:ff:ff:ff:ff

inet 192.168.4.5/24 brd 192.168.4.255 scope global eth0

valid\_lft forever preferred\_lft forever

inet 192.168.4.15/24 brd 192.168.4.255 scope global secondary eth0:0

valid\_lft forever preferred\_lft forever

inet6 fe80::1d36:1d47:870e:f93/64 scope link

valid\_lft forever preferred\_lft forever

2）设置Web1服务器网络参数

[root@web1 ~]# nmcli connection modify eth0 ipv4.method manual

ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth0

接下来给web1配置VIP地址。

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。

[root@web1 ~]# cd /etc/sysconfig/network-scripts/

[root@web1 network-scripts]# cp ifcfg-lo{,:0}

[root@web1 network-scripts]# vim ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255

NETWORK=192.168.4.15

BROADCAST=192.168.4.15

ONBOOT=yes

NAME=lo:0

防止地址冲突的问题：

这里因为web1也配置与代理一样的VIP地址，默认肯定会出现地址冲突；

sysctl.conf文件写入这下面四行的主要目的就是访问192.168.4.15的数据包，只有调度器会响应，其他主机都不做任何响应，这样防止地址冲突的问题。

[root@web1 network-scripts]# 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

arp\_ignore (定义回复ARP广播的方式)

0(默认值),回应所有的本地地址ARP广播,本地地址可以配置在任意网络接口

1:只回应配置在入站网卡接口上的任意IP地址的ARP广播

arp\_announce

0(默认),使用配置在任意网卡接口上的本地IP地址

2: 对查询目标使用最适当的本地地址,在此模式下将忽略这个IP数据包的源地址并尝试选择与能该地址通信的本地地址,首要是选择所有的网络接口的子网中外出访问子网中包含该目标IP地址的本地地址.如果没有合适的地址被发现,将选择当前的发送网络接口或其他的有可能接受到该ARP回应的网络接口进行发送

[root@web1 network-scripts]# sysctl -p

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

重启网络服务，设置防火墙与SELinux

[root@web1 network-scripts]# systemctl restart network

[root@web1 network-scripts]# systemctl stop firewalld

[root@web1 network-scripts]# setenforce 0

3）设置Web2服务器网络参数

[root@web2 ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth0

接下来给web2配置VIP地址。

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。

[root@web2 ~]# cd /etc/sysconfig/network-scripts/

[root@web2 network-scripts]# cp ifcfg-lo{,:0}

[root@web2 network-scripts]# vim ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255

NETWORK=192.168.4.15

BROADCAST=192.168.4.15

ONBOOT=yes

NAME=lo:0

防止地址冲突的问题：

这里因为web1也配置与代理一样的VIP地址，默认肯定会出现地址冲突；

sysctl.conf文件写入这下面四行的主要目的就是访问192.168.4.15的数据包，只有调度器会响应，其他主机都不做任何响应，这样防止地址冲突的问题。

[root@web2 network-scripts]# vim /etc/sysctl.conf

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

[root@web2 network-scripts]# sysctl -p

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

重启网络服务，设置防火墙与SELinux

[root@web2 network-scripts]# systemctl restart network

[root@web2 network-scripts]# systemctl stop firewalld

[root@web2 network-scripts]# setenforce 0

步骤二：配置后端Web服务器

1）自定义Web页面

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "<h1>192.168.4.100" > /var/www/html/index.html

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "<h1>192.168.4.200" > /var/www/html/index.html

2）启动Web服务器软件

[root@web1 ~]# systemctl restart httpd

[root@web2 ~]# systemctl restart httpd

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（如果已经安装，此步骤可以忽略）

[root@proxy ~]# yum -y install ipvsadm

2）清理之前实验的规则，创建新的集群服务器规则

[root@proxy ~]# ipvsadm -C #清空所有规则

[root@proxy ~]# ipvsadm -A -t 192.168.4.15:80 -s wrr

3）添加真实服务器(-g参数设置LVS工作模式为DR模式，-w设置权重)

[root@proxy ~]# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.100 -g -w 1

[root@proxy ~]# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.200 -g -w 2

4）查看规则列表，并保存规则

[root@proxy ~]# ipvsadm -Ln

TCP 192.168.4.15:80 wrr

-> 192.168.4.100:80 Route 1 0 0

-> 192.168.4.200:80 Route 2 0 0

[root@proxy ~]# ipvsadm-save -n > /etc/sysconfig/ipvsadm

步骤四：客户端测试

客户端使用curl命令反复连接http://192.168.4.15，查看访问的页面是否会轮询到不同的后端真实服务器。

[root@client ~]# curl 192.168.4.15

<h1>192.168.2.200

[root@client ~]# curl 192.168.4.15

<h1>192.168.2.100

**扩展知识**：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

####################################

**II-Cluster-Day03**

keepalived的功能:

1) 写配置文件,自定执行ipvsadm配置集群

2) 健康检查

3) 浮动VIP

VRRP:虚拟路由冗余协议(Virtual Router Redundancy Protocol) HSRP

**Keepalived概述**

keepalived实现了高可用集群

Keepalived最初是为LVS设计的,专门监控各服务器节点的状态

Keepalived后来加入了VRRP功能,防止单点故障

**Keepalived运行原理**

Keepalived检测每个服务器节点状态

服务器节点异常或工作出现故障,keepalived将故障节点从集群系统中剔除

故障节点恢复后,keepalived再将其加入到集群系统中

所有工作自动完成,无需人工干预

**案例1：Keepalived高可用服务器**

准备三台Linux服务器，两台做Web服务器，并部署Keepalived高可用软件，一台作为客户端主机，实现如下功能：

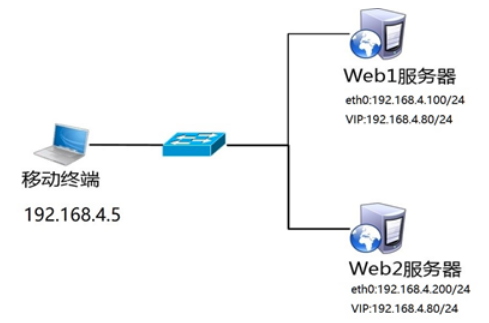
1) 使用Keepalived实现web服务器的高可用

2) Web服务器IP地址分别为192.168.4.100和192.168.4.200

3) Web服务器的浮动VIP地址为192.168.4.80

4) 客户端通过访问VIP地址访问Web页面

使用3台虚拟机，2台作为Web服务器，并部署Keepalived、1台作为客户端，拓扑结构如图所示，主机配置如表所示。



|  |  |
| --- | --- |
| **主机名** | **网络配置** |
| proxy(扮演客户端的角色) | eth0:192.168.4.5/24 |
| web1 | eth0: 192.168.4.100/24  VIP: 192.168.4.80(keppalive会自动配置) |
| web2 | eth0:192.168.4.200/24  VIP: 192.168.4.80(keppalive会自动配置) |

步骤一：配置网络环境（如果在前面课程已经完成该配置，可以忽略此步骤）

1）设置Web1服务器网络参数、配置Web服务

[root@web1 ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth0

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "<h1>192.168.4.100" > /var/www/html/index.html

[root@web1 ~]# systemctl restart httpd

2）设置Web2服务器网络参数、配置Web服务

[root@web2 ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth0

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "<h1>192.168.4.200" > /var/www/html/index.html

[root@web2 ~]# systemctl restart httpd

3）配置proxy主机的网络参数（如果已经设置，可以忽略此步骤）

[root@proxy ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

[root@proxy ~]# nmcli connection up eth0

步骤二：安装Keepalived软件

注意：两台Web服务器做相同的操作。

[root@web1 ~]# yum install -y keepalived

[root@web2 ~]# yum install -y keepalived

步骤三：部署Keepalived服务

1）修改web1服务器Keepalived配置文件

[root@web1 ~]# vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

[sysadmin@firewall.loc](mailto:sysadmin@firewall.loc) #设置报警收件人邮箱

}

notification\_email\_from [Alexandre.Cassen@firewall.loc](mailto:Alexandre.Cassen@firewall.loc) #设置发件人

smtp\_server 192.168.200.1 #定义邮件服务器

smtp\_connect\_timeout 30 #连接超时时间

router\_id web1 #设置路由ID号(实验需要修改)

vrrp\_skip\_check\_adv\_addr

vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

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 #主备服务器VRID必须一致,类似于分组

priority 100 #服务器优先级,优先级高先获取VIP(实验需要改)

advert\_int 1 #每隔一秒对比下优先级谁高

authentication {

auth\_type PASS

auth\_pass 1111 #主备服务器密码必须一致

}

virtual\_ipaddress {

192.168.4.80 #谁是主服务器谁获得该VIP(实验需要修改)

}

}

2）修改web2服务器Keepalived配置文件

[root@web2 ~]# vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 192.168.200.1

smtp\_connect\_timeout 30

router\_id web2

vrrp\_skip\_check\_adv\_addr

vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0

virtual\_router\_id 51

priority 90

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.4.80 #注意换行,不然验证时会出错

}

}

3）启动服务

[root@web1 ~]# systemctl start keepalived

[root@web2 ~]# systemctl start keepalived

4）配置防火墙和SELinux

启动keepalived会自动添加一个drop的防火墙规则，需要清空！

[root@web1 ~]# iptables -F 或者把vrrp\_strict注释掉,即可不用清防火墙规则

[root@web1 ~]# setenforce 0

[root@web2 ~]# iptables -F

[root@web1 ~]# setenforce 0

步骤四：测试

1）登录两台Web服务器查看VIP信息

[root@web1 ~]# ip addr show eth0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

link/ether 52:54:00:8c:24:86 brd ff:ff:ff:ff:ff:ff

inet 192.168.4.100/24 brd 192.168.4.255 scope global eth0

valid\_lft forever preferred\_lft forever

inet 192.168.4.80/32 scope global eth0

valid\_lft forever preferred\_lft forever

inet6 fe80::95b7:c341:8fbc:cc88/64 scope link

valid\_lft forever preferred\_lft forever

[root@web2 ~]# ip addr show eth0

2) 客户端访问

客户端使用curl命令连接http://192.168.4.80，查看Web页面；关闭Web1服务器的网卡，客户端再次访问http://192.168.4.80，验证是否可以正常访问服务。

[root@proxy1 ~]# curl 192.168.4.80

<h1>192.168.2.100

[root@web1 ~]# systemctl stop keepalived.service

[root@proxy1 ~]# curl 192.168.4.80

<h1>192.168.2.200

## **案例2：Keepalived+LVS服务器**

使用Keepalived为LVS调度器提供高可用功能，防止调度器单点故障，为用户提供Web服务：

1) LVS1调度器真实IP地址为192.168.4.5

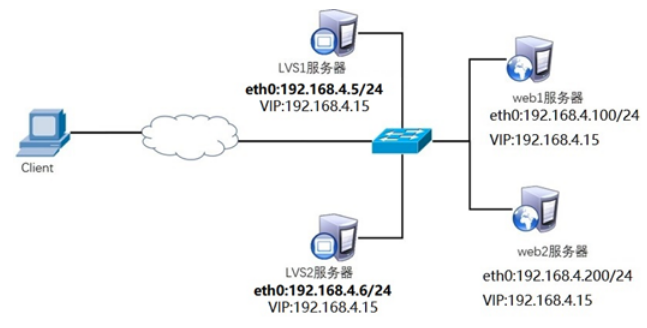
2) LVS2调度器真实IP地址为192.168.4.6

3) 服务器VIP地址设置为192.168.4.15

4) 真实Web服务器地址分别为192.168.4.100、192.168.4.200

5) 使用加权轮询调度算法，真实web服务器权重不同

使用5台虚拟机，1台作为客户端主机、2台作为LVS调度器、2台作为Real Server，实验拓扑环境结构如图所示，基础环境配置如表所示。



|  |  |
| --- | --- |
| **主机名** | **网络配置** |
| 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 |

注意: 所有主机都需要配置IP地址与有效的yum源

步骤一：配置网络环境

1）设置Web1服务器的网络参数

[root@web1 ~]# nmcli connection modify eth0 ipv4.method manual

ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth0

接下来给web1配置VIP地址

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。

[root@web1 ~]# cd /etc/sysconfig/network-scripts/

[root@web1 network-scripts]# cp ifcfg-lo{,:0}

[root@web1 network-scripts]# vim ifcfg-lo:0}

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255

NETWORK=192.168.4.15

BROADCAST=192.168.4.15

ONBOOT=yes

NAME=lo:0

注意：这里因为web1也配置与调度器一样的VIP地址，默认肯定会出现地址冲突。

写入这四行的主要目的就是访问192.168.4.15的数据包，只有调度器会响应，其他主机都不做任何响应。

[root@web1 ~]# 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

重启网络服务，设置防火墙与SELinux

[root@web1 ~]# systemctl restart network

[root@web1 ~]# systemctl stop firewalld

[root@web1 ~]# setenforce 0

2）设置Web2服务器的网络参数

[root@web2 ~]# nmcli connection modify eth0 ipv4.method manual

ipv4.addresses 192.168.4.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth0

接下来给web2配置VIP地址

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。

[root@web2 ~]# cd /etc/sysconfig/network-scripts/

[root@web2 network-scripts]# cp ifcfg-lo{,:0}

[root@web2 network-scripts]# vim ifcfg-lo:0}

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255

NETWORK=192.168.4.15

BROADCAST=192.168.4.15

ONBOOT=yes

NAME=lo:0

注意：这里因为web2也配置与代理一样的VIP地址，默认肯定会出现地址冲突。

写入这四行的主要目的就是访问192.168.4.15的数据包，只有调度器会响应，其他主机都不做任何响应。

[root@web2 ~]# 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

重启网络服务，设置防火墙与SELinux

[root@web2 ~]# systemctl restart network

[root@web2 ~]# systemctl stop firewalld

[root@web2 ~]# setenforce 0

3）配置proxy1主机的网络参数(不配置VIP，由keepalvied自动配置)

[root@proxy1 ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

[root@proxy1 ~]# nmcli connection up eth0

4）配置proxy2主机的网络参数(不配置VIP，由keepalvied自动配置)

注意：按照前面的课程环境，默认没有该虚拟机，需要重新建一台虚拟机proxy2。

[root@proxy2 ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.6/24 connection.autoconnect yes

[root@proxy2 ~]# nmcli connection up eth0

步骤二：配置后台web服务

1）安装软件，自定义Web页面（web1和web2主机）

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "<h1>192.168.4.100" > /var/www/html/index.html

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "<h1>192.168.4.200" > /var/www/html/index.html

3）启动Web服务器软件(web1和web2主机)

[root@web1 ~]# systemctl start httpd ; systemctl enable httpd

[root@web2 ~]# systemctl start httpd ; systemctl enable httpd

步骤三：调度器安装Keepalived与ipvsadm软件

注意：两台LVS调度器执行相同的操作（如果已经安装软件，可用忽略此步骤）。

安装软件

[root@proxy1 ~]# yum -y install keepalived

[root@proxy1 ~]# yum install -y ipvsadm

[root@proxy2 ~]# yum -y install keepalived.x86\_64

[root@proxy2 ~]# yum install -y ipvsadm

步骤四：部署Keepalived实现LVS-DR模式调度器的高可用

1）LVS1调度器设置Keepalived，并启动服务

[root@proxy1 ~]# vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

[sysadmin@firewall.loc](mailto:sysadmin@firewall.loc) #设置报警收件人邮箱

}

notification\_email\_from [Alexandre.Cassen@firewall.loc](mailto:Alexandre.Cassen@firewall.loc) #设置发件人

smtp\_server 192.168.200.1 #定义邮件服务器

smtp\_connect\_timeout 30

router\_id LVS1 #设置路由ID号(实验需要修改)

vrrp\_skip\_check\_adv\_addr

vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state MASTER #主服务器为MASTER

interface eth0 #定义网络接口

virtual\_router\_id 51 #主辅VRID号必须一致

priority 100 #服务器优先级

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111 //主辅服务器密码必须一致

}

virtual\_ipaddress {

192.168.4.15 #配置VIP(实验需要设置)

}

}

virtual\_server 192.168.4.15 80 { #设置ipvsadm的VIP规则(实验需要设置)

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的服务器

protocol TCP

real\_server 192.168.4.100 80 { #设置后端web服务器真实ip

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

weight 2 #设置权重为2

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

[root@proxy1 ~]# systemctl start keepalived

[root@proxy1 ~]# iptables -F

[root@proxy1 ~]# ipvsadm -Ln #查看LVS规则

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.15:80 wrr

-> 192.168.4.100:80 Route 1 0 0

-> 192.168.4.200:80 Route 1 0 0

[root@proxy1 ~]# ip a s eth0 #查看VIP配置

2: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

link/ether 52:54:00:15:1f:27 brd ff:ff:ff:ff:ff:ff

inet 192.168.4.5/24 brd 192.168.4.255 scope global eth0

valid\_lft forever preferred\_lft forever

inet 192.168.4.15/32 scope global eth0

valid\_lft forever preferred\_lft forever

inet6 fe80::1d36:1d47:870e:f93/64 scope link

valid\_lft forever preferred\_lft forever

2）LVS2调度器设置Keepalived

[root@proxy2 ~]# yum -y install ipvsadm

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

[root@proxy2 ~]# vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 192.168.200.1

smtp\_connect\_timeout 30

router\_id LVS2

vrrp\_skip\_check\_adv\_addr

vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0

virtual\_router\_id 51

priority 90

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.4.15

}

}

virtual\_server 192.168.4.15 80 {

delay\_loop 6

lb\_algo wrr

lb\_kind DR

# persistence\_timeout 50

protocol TCP

real\_server 192.168.4.100 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.200 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

[root@proxy2 ~]# systemctl start keepalived.service

[root@proxy2 ~]# iptables -F

[root@proxy2 ~]# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.15:80 wrr

-> 192.168.4.100:80 Route 1 0 0

-> 192.168.4.200:80 Route 1 0 0

步骤五：客户端测试

客户端使用curl命令反复连接http://192.168.4.15，查看访问的页面是否会轮询到不同的后端真实服务器。

[root@client ~]# curl 192.168.4.15

<h1>192.168.2.200

[root@client ~]# curl 192.168.4.15

<h1>192.168.2.100

**案例3：配置HAProxy负载平衡集群**

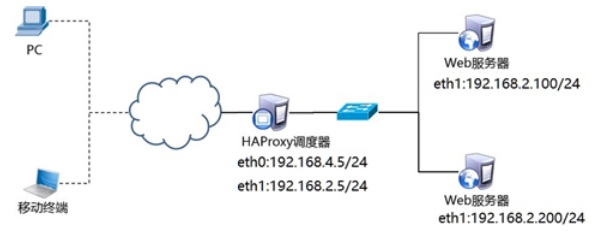
准备4台Linux服务器，两台做Web服务器，1台安装HAProxy，1台做客户端，实现如下功能：

1) 客户端访问HAProxy，HAProxy分发请求到后端Real Server

2) 开启HAProxy监控页面，及时查看调度器状态

3) 设置HAProxy为开机启动

使用4台虚拟机，1台作为HAProxy调度器、2台作为Real Server、1台作为客户端，拓扑结构如图所示，具体配置如表所示。



|  |  |
| --- | --- |
| **主机名** | **网络配置** |
| 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 |
| web2 | eth1:192.168.2.200/24 |

实现此案例需要按照如下步骤进行。

注意事项：

将前面实验VIP、LVS等实验的内容清理干净！！！！！！

删除所有设备的VIP，清空所有LVS设置，关闭keepalived！！！

web1关闭多余的网卡与VIP，配置本地真实IP地址。

[root@web1 ~]# ifdown eth0

[root@web1 ~]# ifdown lo:0

[root@web1 ~]# nmcli connection modify eth1 ipv4.method manual

ipv4.addresses 192.168.2.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth1

Web2关闭多余的网卡与VIP，配置本地真实IP地址。

[root@web2 ~]# ifdown eth0

[root@web2 ~]# ifdown lo:0

[root@web2 ~]# nmcli connection modify eth1 ipv4.method manual

ipv4.addresses 192.168.2.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth1

proxy关闭keepalived服务，清理LVS规则。

[root@proxy ~]# systemctl stop keepalived

[root@proxy ~]# systemctl disable keepalived

[root@proxy ~]# ipvsadm -C

[root@proxy ~]# nmcli connection modify eth0 ipv4.method manual

ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

[root@proxy ~]# nmcli connection up eth0

[root@proxy ~]# nmcli connection modify eth1 ipv4.method manual

ipv4.addresses 192.168.2.5/24 connection.autoconnect yes

[root@proxy ~]# nmcli connection up eth1

步骤一：配置后端Web服务器

设置两台后端Web服务（如果已经配置完成，可用忽略此步骤）

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# systemctl start httpd

[root@web1 ~]# echo "<h1>192.168.2.100" > /var/www/html/index.html

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# systemctl start httpd

[root@web2 ~]# echo "<h1>192.168.2.200" > /var/www/html/index.html

步骤二：部署HAProxy服务器

1）配置网络，安装软件

[root@proxy ~]# echo 'net.ipv4.ip\_forward = 1' >> /etc/sysctl.conf #开启路由转发

[root@proxy ~]# sysctl -p

[root@proxy ~]# yum -y install haproxy

2）修改配置文件

[root@proxy ~]# vim /etc/haproxy/haproxy.cfg

......

global #全局设置

# local2.\* /var/log/haproxy.log

#

log 127.0.0.1 local2 #日志

chroot /var/lib/haproxy #工作的目录

pidfile /var/run/haproxy.pid #haproxy的pid存放路径

maxconn 4000 #最大连接数,默认4000

user haproxy #运行服务的用户,原理跟nginx一样

group haproxy

daemon #创建1个进程进入daemon模式运行

......

defaults #默认的设置

mode http #默认的模式 mode tcp|http|health

log global #采用全局定义的日志

option httplog #日志类别http日志格式

option dontlognull #不记录健康检查的日志信息

option http-server-close #每次请求完毕后主动关闭http通道

option forwardfor except 127.0.0.0/8

option redispatch #强制转换(web1坏了,转给web2)

retries 3 #3次连接失败就认为服务不可用,也可以通过后面设置

timeout http-request 10s

timeout queue 1m

timeout connect 10s

timeout client 1m #客户端连接超时

timeout server 1m #服务器连接超时

timeout http-keep-alive 10s

timeout check 10s #健康检查的超时时间

maxconn 3000 #最大连接数

# main frontend which proxys to the backends

#---------------------------------------------------------------------

listen webs :80

balance roundrobin

名字随便写

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

listen stats

bind 0.0.0.0:1080 #监听端口

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 #统计页面密码框上提示文本

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

3）启动服务器并设置开机启动

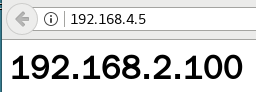
[root@haproxy ~]# systemctl start haproxy

[root@haproxy ~]# systemctl enable haproxy

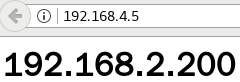
步骤三：客户端验证

客户端配置与HAProxy相同网络的IP地址，并使用火狐浏览器访问http://192.168.4.5，测试调度器是否正常工作，客户端访问http://192.168.4.5:1080/stats测试状态监控页面是否正常。

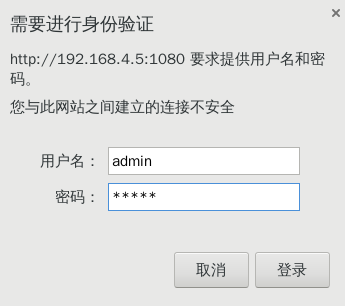
[root@client ~]# firefox 192.168.4.5

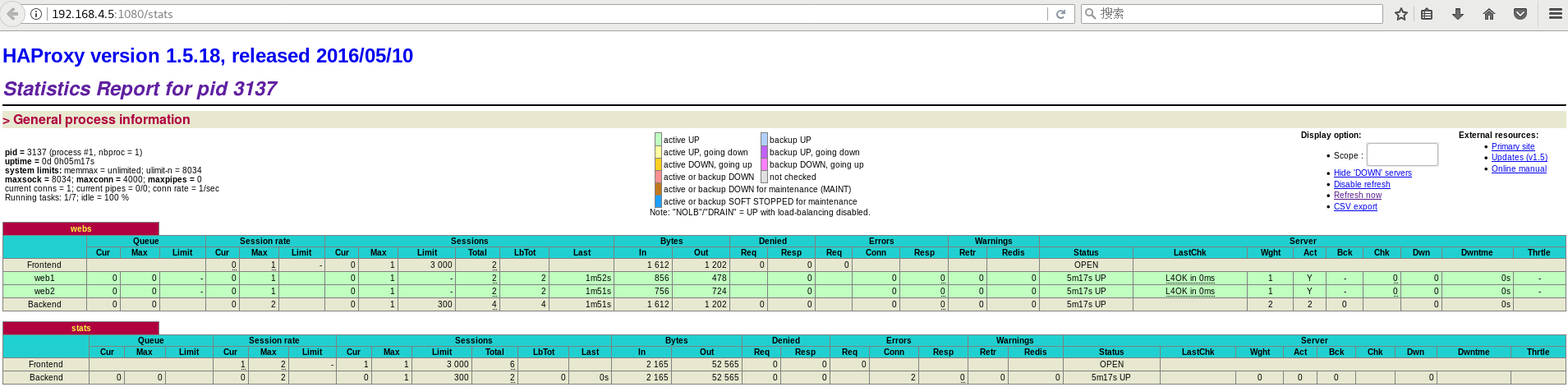


[root@client ~]# firefox 192.168.4.5



[root@client ~]# firefox 192.168.4.5:1080/stats





**备注**：

Queue队列数据的信息（当前队列数量，最大值，队列限制数量）；

Session rate每秒会话率（当前值，最大值，限制数量）；

Sessions总会话量（当前值，最大值，总量，Lbtot: total number of times a server was selected选中一台服务器所用的总时间）；

Bytes（入站、出站流量）；

Denied（拒绝请求、拒绝回应）；

Errors（错误请求、错误连接、错误回应）；

Warnings（重新尝试警告retry、重新连接redispatches）；

Server(状态、最后检查的时间（多久前执行的最后一次检查）、权重、备份服务器数量、down机服务器数量、down机时长)。

**集群调度软件对比**

**Nginx分析**

优点:

1. 工作在7层,可以针对http做分流策略
2. 正则表达式必HAProxy强大
3. 安装、配置、测试简单,通过日志可以解决多数问题
4. 并发量可以达到几万次
5. Nginx还可以作为web服务器使用

缺点:

1. 仅支持http、https、mail协议,应用面小
2. 监控检查仅通过端口,无法使用url检查

**LVS分析**

优点:

1. 负载能力强,工作在4层,对内存、CPU消耗低
2. 配置性低,没有太多可配置性,减少人为错误
3. 应用面广,几乎可以为所有应用提供负载均衡

缺点:

1. 不支持正则表达式,不能实现动静分离
2. 如果网站架构庞大,LVS-DR配置比较繁琐

**HAProxy分析**

优点:

1. 支持session、cookie功能
2. 可以通过url进行健康检查
3. 效率、负载均衡速度,高于Nginx,低于LVS
4. HAProxy支持TCP,可以对MySQL进行负载均衡
5. 调度算法丰富

缺点:

1. 正则弱于Nginx
2. 日志依赖于syslogd,不支持apache日志

####################################

**II-Cluster-Day04**

**分布式文件系统**

分布式文件系统(Distributed File System)是指文件系统管理的物理存储资源不一定直接连接在本地节点上,而是通过计算机网络与节点相连

分布式文件系统的设计基于客户机/服务器模式

**常用分布式文件系统**

Lustre

Hadoop

FastDFS

Ceph

GlusterFS

**Ceph** 使用的是CRUSH算法 数据会被打散为4M的小文件,并行写入

1) Ceph是一个分布式文件系统

2) 具有高扩展、高可用、高性能的特点

3) Ceph可以提供对象存储、块存储、文件系统存储

4) Ceph可以提供EB级别的存储空间(EB<--PB<--TB<--GB)

1024G \* 1024G=1048576G

1. 软件定义存储(Software Defined Storage)作为存储行业的一大发展趋势,已经越来越受到市场的认可

**帮助文档:http://docs.ceph.org/start/intro(所有知识来于此网站)**

**http://docs.ceph.org.cn(中文网站)**

**Ceph组件**

OSD : 存储设备 默认3副本

Monitors : 集群监控组件 默认3副本(最少3台mon服务器)

RBD : 块存储

radosgw : 对象存储(访问时,必须写程序APP)

MDS : 存放文件系统的元数据(对象存储和块存储不需要该组件)

Client : ceph-common客户端

## **案例1：实验环境**

准备四台KVM虚拟机，其三台作为存储集群节点，一台安装为客户端，实现如下功能：

1) 创建1台客户端虚拟机

2) 创建3台存储集群虚拟机

3) 配置主机名、IP地址、YUM源

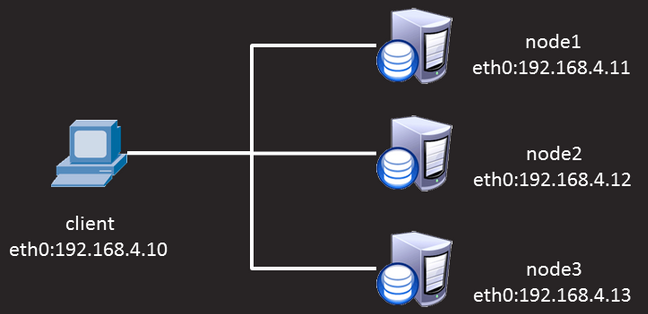
4) 修改所有主机的主机名

5) 配置无密码SSH连接

6) 配置NTP时间同步

7) 创建虚拟机磁盘

使用4台虚拟机，1台客户端、3台存储集群服务器，拓扑结构如图所示。



所有主机的主机名及对应的IP地址如表所示

|  |  |
| --- | --- |
| **主机名** | **IP地址** |
| client | eth0:192.168.4.10/24 |
| node1 | eth0:192.168.4.11/24 |
| node2 | eth0:192.168.4.12/24 |
| node3 | eth0:192.168.4.13/24 |

步骤一：安装前准备

1）真实机为所有节点配置yum源服务器。

[root@room9pc01 ~]# yum -y install vsftpd

[root@room9pc01 ~]# mkdir /var/ftp/ceph

[root@room9pc01 ~]# mount -o loop

rhcs2.0-rhosp9-20161113-x86\_64.iso /var/ftp/ceph

[root@room9pc01 ~]# systemctl restart vsftpd

2）修改所有节点都需要配置YUM源（这里仅以node1为例）。

[root@node1 ~]#vim /etc/yum.repos.d/ceph.repo

[mon]

name=mon

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/MON

gpgcheck=0

enabled=1

[osd]

name=osd

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/OSD

gpgcheck=0

enabled=1

[tools]

name=tools

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/Tools

gpgcheck=0

enabled=1

3）修改/etc/hosts并同步到所有主机。

警告：/etc/hosts解析的域名必须与本机主机名一致！！！！

[root@node1 ~]#vim /etc/hosts

... ...

192.168.4.10 client

192.168.4.11     node1

192.168.4.12     node2

192.168.4.13     node3

[root@node1 ~]# for i in 10 11 12 13

> do

> scp /etc/hosts 192.168.4.$i:/etc/

> done

[root@node1 ~]# for i in 10 11 12 13

> do

> scp /etc/yum.repos.d/ceph.repo 192.168.4.$i:/etc/yum.repos.d/

> done

4）配置无密码连接(包括自己远程自己也不需要密码)。

[root@node1 ~]# ssh-keygen -f /root/.ssh/id\_rsa -N ''

[root@node1 ~]# for i in 10 11 12 13

> do

> ssh-copy-id 192.168.4.$i

> done

步骤二：配置NTP时间同步

1）真实物理机创建NTP服务器。

[root@room9pc01 ~]# yum -y install chrony

[root@room9pc01 ~]# vim /etc/chrony.conf

......

server 0.centos.pool.ntp.org iburst

allow 192.168.4.0/24

local stratum 10

......

[root@room9pc01 ~]# systemctl restart chronyd

如果有防火墙规则，需要清空所有规则

[root@room9pc01 ~]# iptables -F

2）其他所有节点与NTP服务器同步时间（以node1为例）。

[root@node1 ~]# vim /etc/chrony.conf

......

server 192.168.4.254 iburst

......

[root@node1 ~]# systemctl restart chronyd

步骤三：准备存储磁盘

1）物理机上为每个虚拟机准备3块磁盘。（可以使用命令，也可以使用图形直接添加）

[root@room9pc01 ~]# virt-manager

给node1 node2 node3分别再加3块20G的磁盘

[root@node1 ~]# lsblk | tail -3

vdb 252:16 0 20G 0 disk

vdc 252:32 0 20G 0 disk

vdd 252:48 0 20G 0 disk

**案例2：部署ceph集群**

沿用练习一，部署Ceph集群服务器，实现以下目标：

1) 安装部署工具ceph-deploy

2) 创建ceph集群

3) 准备日志磁盘分区

4) 创建OSD存储空间

5) 查看ceph状态，验证

步骤一：部署软件

1）在node1安装部署工具，学习工具的语法格式。

[root@node1 ~]# yum -y install ceph-deploy

[root@node1 ~]# ceph-deploy --help

2）创建目录

[root@node1 ~]# mkdir ceph-cluster #名字随便起

[root@node1 ~]# cd ceph-cluster/ #执行ceph-deploy必须在此目录下

步骤二：部署Ceph集群

1）创建Ceph集群配置。

[root@node1 ceph-cluster]# ceph-deploy new node1 node2 node3

[root@node1 ceph-cluster]# ls

ceph.conf ceph-deploy-ceph.log ceph.mon.keyring

[root@node1 ceph-cluster]# cat ceph.conf

[global]

fsid = edcf423e-8929-41fc-8e15-4b925a30456a

mon\_initial\_members = node1, node2, node3

mon\_host = 192.168.4.11,192.168.4.12,192.168.4.13

auth\_cluster\_required = cephx

[root@node1 ceph-cluster]# cat ceph.mon.keyring

[mon.]

key = AQD1b1JcAAAAABAA6xgUyrzz5EMqKC9PN1KJVg==

caps mon = allow \*

2）给所有节点安装软件包。此命令是装所有有关ceph的软件包,不起服务

[root@node1 ceph-cluster]# ceph-deploy install node1 node2 node3

[root@node1 ceph-cluster]# rpm -qa | grep ceph

libcephfs1-10.2.2-38.el7cp.x86\_64

ceph-osd-10.2.2-38.el7cp.x86\_64

ceph-deploy-1.5.33-1.el7cp.noarch

ceph-mon-10.2.2-38.el7cp.x86\_64

ceph-selinux-10.2.2-38.el7cp.x86\_64

python-cephfs-10.2.2-38.el7cp.x86\_64

ceph-radosgw-10.2.2-38.el7cp.x86\_64

3）初始化所有节点的mon服务（主机名解析必须对）

[root@node1 ceph-cluster]# ceph-deploy mon create-initial

#起ceph-mon@node\*.service服务,并把配置文件拷贝到/etc/ceph/目录下

[root@node1 ceph-cluster]# ls /etc/ceph/

ceph.client.admin.keyring ceph.conf rbdmap tmpN\_OmMz

[root@node2 ~]# ls /etc/ceph/

ceph.client.admin.keyring ceph.conf rbdmap tmpexQjgd

[root@node3 ~]# ls /etc/ceph/

ceph.client.admin.keyring ceph.conf rbdmap tmp2zz8I6

[root@node1 ceph-cluster]# ceph -s #查看集群状态

常见错误及解决方法（非必要操作，有错误可以参考）：

如果提示如下错误信息：

[node1][ERROR ] admin\_socket: exception getting command descriptions: [Error 2] No such file or directory

解决方案如下（在node1操作）：

先检查自己的命令是否是在ceph-cluster目录下执行的！！！！如果确认是在该目录下执行的create-initial命令，依然保存，可以使用如下方式修复。

[root@node1 ceph-cluster]# vim ceph.conf #文件最后追加以下内容

public\_network = 192.168.4.0/24

修改后重新推送配置文件:

[root@node1 ceph-cluster]# ceph-deploy --overwrite-conf config push node1 node2 node3

步骤三：创建OSD

1）准备磁盘分区（node1、node2、node3都做相同操作）

[root@node1 ceph-cluster]# parted /dev/vdb mklabel gpt

[root@node1 ceph-cluster]# parted /dev/vdb mkpart primary 1M 50%

[root@node1 ceph-cluster]# parted /dev/vdb mkpart primary 50% 100%

[root@node1 ceph-cluster]# chown ceph.ceph /dev/vdb1 #.跟:一样

[root@node1 ceph-cluster]# chown ceph:ceph /dev/vdb2 #临时修改

//这两个分区用来做存储服务器的日志journal盘

2）初始化清空磁盘数据（仅node1操作即可）-----相当于分区

[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

3）创建OSD存储空间（仅node1操作即可）

[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

4）常见错误（非必须操作）

使用osd create创建OSD存储空间时，如提示run 'gatherkeys'，可以使用如下命令修复：

[root@node1 ceph-cluster]# ceph-deploy gatherkeys node1 node2 node3

步骤四：验证测试

1) 查看集群状态

[root@node1 ceph-cluster]# ceph -s

cluster edcf423e-8929-41fc-8e15-4b925a30456a

health HEALTH\_OK

monmap e1: 3 mons at {node1=192.168.4.11:6789/0,node2=192.168.4.12:6789/0,node3=192.168.4.13:6789/0}

election epoch 6, quorum 0,1,2 node1,node2,node3

osdmap e39: 6 osds: 6 up, 6 in

flags sortbitwise

pgmap v80: 64 pgs, 1 pools, 0 bytes data, 0 objects

203 MB used, 119 GB / 119 GB avail

64 active+clean

2）常见错误（非必须操作）

如果查看状态包含如下信息：

health: HEALTH\_WARN

clock skew detected on node2, node3…

clock skew表示时间不同步，解决办法：请先将所有主机的时间都使用NTP时间同步！！！

Ceph要求所有主机时差不能超过0.05s，否则就会提示WARN，如果使用NTP还不能精确同步时间，可以手动修改所有主机的ceph.conf，在[MON]下面添加如下一行：

mon clock drift allowed = 1

如果状态还是失败，可以尝试执行如下命令，重启ceph服务：

[root@node1 ~]# systemctl restart ceph\\*.service ceph\\*.target

**案例3：创建Ceph块存储**

沿用练习一，使用Ceph集群的块存储功能，实现以下目标：

1) 创建块存储镜像

2) 客户端映射镜像

3) 创建镜像快照

4) 使用快照还原数据

5) 使用快照克隆镜像

6) 删除快照与镜像

步骤一：创建镜像

1）查看存储池。

[root@node1 ceph-cluster]# ceph osd lspools

0 rbd, #默认有个rbd的存储池

[root@node2 ~]# ceph osd lspools

0 rbd,

[root@node3 ~]# ceph osd lspools

0 rbd,

2）创建镜像、查看镜像(创建共享盘)

名字可以随便起

[root@node1 ~]# rbd create demo-image --image-feature layering --size 10G

[root@node1 ~]# rbd create rbd/image --image-feature layering --size 10G

[root@node1 ~]# rbd list #查看镜像列表

demo-image

image

[root@node1 ~]# rbd info demo-image #查看具体一个镜像

rbd image 'demo-image':

size 10240 MB in 2560 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.104c238e1f29

format: 2

features: layering

flags:

步骤二：动态调整

1）缩小容量

[root@node1 ~]# rbd resize --size 7G image --allow-shrink

Resizing image: 100% complete...done.

[root@node1 ~]# rbd info image

rbd image 'image':

size 7168 MB in 1792 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.104f238e1f29

format: 2

features: layering

flags:

2）扩容容量

[root@node1 ~]# rbd resize --size 15G image

Resizing image: 100% complete...done.

[root@node1 ~]# rbd info image

rbd image 'image':

size 15360 MB in 3840 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.104f238e1f29

format: 2

features: layering

flags:

步骤三：通过KRBD访问

1）集群内将镜像映射为本地磁盘

[root@node1 ~]# rbd map demo-image

/dev/rbd0

[root@node1 ~]# lsblk

… …

rbd0 251:0 0 10G 0 disk

[root@node1 ~]# mkfs.xfs /dev/rbd0

[root@node1 ~]# mount /dev/rbd0 /mnt

2）客户端通过KRBD访问

#客户端需要安装ceph-common软件包

#拷贝配置文件（否则不知道集群在哪）

#拷贝连接密钥（否则无连接权限）

[root@client ~]# yum -y install ceph-common

[root@client ~]# scp 192.168.4.11:/etc/ceph/ceph.conf /etc/ceph/

[root@client ~]# scp 192.168.4.11:/etc/ceph/ceph.client.admin.keyring /etc/ceph/

[root@client ~]# ls /etc/ceph

ceph.client.admin.keyring ceph.conf rbdmap

[root@client ~]# rbd map image

/dev/rbd0

[root@client ~]# rbd map demo-image

/dev/rbd1

[root@client ~]# lsblk | tail -2

rbd0 251:0 0 15G 0 disk

rbd1 251:16 0 10G 0 disk

[root@client ~]# rbd unmap demo-image

[root@client ~]# lsblk | tail -1

rbd0 251:0 0 15G 0 disk

[root@client ~]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

3) 客户端格式化、挂载分区

[root@client ~]# mkfs.xfs /dev/rbd0

[root@client ~]# mount /dev/rbd0 /mnt/

[root@client ~]# echo "回家" > /mnt/test.txt

步骤四：创建镜像快照

快照: 基于COW(copy on write)写(编辑)时复制

1) 查看镜像快照

[root@node1 ~]# rbd snap ls image

2) 创建镜像快照

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

[root@node1 ~]# rbd snap ls image #查看镜像image的快照

SNAPID NAME SIZE

4 image-snap1 15360 MB

[root@node1 ~]# rbd snap ls demo-image #空值,无结果

3) 删除客户端写入的测试文件

[root@client ~]# rm -rf /mnt/test.txt

[root@client ~]# umount /mnt/

[root@client ~]# rbd unmap image

1. 还原快照 (ceph不支持在线还原,所以需要先卸载再还原快照)

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

Rolling back to snapshot: 100% complete...done.

[root@client ~]# rbd map image

/dev/rbd0

[root@client ~]# mount /dev/rbd0 /mnt/

[root@client ~]# ls /mnt/

test.txt

[root@client ~]# cat /mnt/test.txt

回家

步骤五：创建快照克隆

1）克隆快照

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

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

rbd: snapshot 'image-snap1' is protected from removal.

2019-02-01 09:36:04.093181 7f28b9f5bd80 -1 librbd::Operations: snapshot is protected

[root@node1 ~]# 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image-clone镜像,具有快照和分层的功能

[root@node1 ~]# rbd list

demo-image

image

image-clone

2）查看克隆镜像与父镜像快照的关系

[root@node1 ~]# rbd info image-clone

rbd image 'image-clone':

size 15360 MB in 3840 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.10603d1b58ba

format: 2

features: layering

flags:

parent: rbd/image@image-snap1

overlap: 15360 MB

#克隆镜像很多数据都来自于快照链

#如果希望克隆镜像可以独立工作，就需要将父快照中的数据，全部拷贝一份，但比较耗时！！！

[root@node1 ~]# rbd flatten image-clone

Image flatten: 100% complete...done.

[root@node1 ~]# rbd info image-clone #注意，父快照信息没了！

rbd image 'image-clone':

    size 15360 MB in 3840 objects

    order 22 (4096 kB objects)

    block\_name\_prefix: rbd\_data.d3f53d1b58ba

    format: 2

    features: layering

    flags:

步骤六：其他操作

1）客户端撤销磁盘映射

[root@client ~]# umount /mnt

[root@client ~]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

[root@client ~]# rbd unmap /dev/rbd0

2）删除快照与镜像

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

[root@node1 ~]# rbd list

[root@node1 ~]# rbd rm image

####################################

**II-Cluster-Day05**

虚拟机:由镜像文件(虚拟机磁盘)+配置文件组成

真机virt-manger新建虚拟机xyz,不安装操作系统

[root@room9pc01 ~]# cd /var/lib/libvirt/images/

[root@room9pc01 images]# ls xyz.qcow2

xyz.qcow2 #镜像文件

[root@room9pc01 images]# ll -h xyz.qcow2

-rw------- 1 root root 21G 2月 1 10:28 xyz.qcow2

[root@room9pc01 images]# du -h xyz.qcow2

4.1M xyz.qcow2

[root@room9pc01 ~]# cd /etc/libvirt/qemu/

[root@room9pc01 qemu]# ls xyz.xml

xyz.xml #配置文件

**clone-vm7的制作**

1 虚拟机镜像快照

[root@room9pc01 qemu]# cd /var/lib/libvirt/images/

[root@room9pc01 images]# ls .rh7\_template.img

.rh7\_template.img #镜像文件(模板)

[root@room9pc01 images]# ls .rhel7.xml

.rhel7.xml #配置文件(模板)

[root@room9pc01 images]# qemu-img create -f qcow2 -b .rh7\_template.img new.qcow2 10G

[root@room9pc01 images]# ll -h new.qcow2

-rw-r--r-- 1 root root 193K 2月 1 11:21 new.qcow2

2拷贝配置文件,修改

[root@room9pc01 images]# cp .rhel7.xml /etc/libvirt/qemu/new.xml

[root@room9pc01 images]# vim /etc/libvirt/qemu/new.xml

......

<name>new</name>

......

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

......

[root@room9pc01 images]# virsh define /etc/libvirt/qemu/new.xml

#虚拟机new创建成功

#新建的虚拟机,虚拟机的磁盘是真实主机上的一个文件

#新建的虚拟机,虚拟机的磁盘调用的是ceph的镜像image

**案例1：块存储应用案例**

延续Day04的实验内容，演示块存储在KVM虚拟化中的应用案例，实现以下功能：

1) Ceph创建块存储镜像

2) 客户端安装部署ceph软件

3) 客户端部署虚拟机

4) 客户端创建secret

5) 设置虚拟机配置文件，调用ceph存储

1）创建磁盘镜像。

[root@node1 ~]# rbd create vm1-image --image-feature layering --size 10G

[root@node1 ~]# rbd create vm2-image --image-feature layering --size 10G

[root@node1 ~]# rbd list

[root@node1 ~]# rbd info vm1-image

[root@node1 ~]# qemu-img info rbd:rbd/vm1-image

image: rbd:rbd/vm1-image

file format: raw

virtual size: 10G (10737418240 bytes)

disk size: unavailable

2）Ceph认证账户。

Ceph默认开启用户认证，客户端需要账户才可以访问，

默认账户名称为client.admin，key是账户的密钥，

可以使用ceph auth添加新账户（案例我们使用默认账户）。

[root@node1 ~]# cat /etc/ceph/ceph.conf //配置文件

[global]

mon\_initial\_members = node1, node2, node3

mon\_host = 192.168.2.10,192.168.2.20,192.168.2.30

auth\_cluster\_required = cephx //开启认证

auth\_service\_required = cephx //开启认证

auth\_client\_required = cephx //开启认证

[root@node1 ~]# cat /etc/ceph/ceph.client.admin.keyring //账户文件

[client.admin]

    key = AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg==  
 3）部署客户端环境。

注意：这里使用真实机当客户端！！！

客户端需要安装ceph-common软件包，拷贝配置文件（否则不知道集群在哪），

拷贝连接密钥（否则无连接权限）。

[root@room9pc01 ~]# yum -y install ceph-common

[root@room9pc01 ~]# scp 192.168.4.11:/etc/ceph/ceph.conf

/etc/ceph/

[root@room9pc01 ~]# scp 192.168.4.11:/etc/ceph/ceph.client.admin.keyring /etc/ceph/

4）创建KVM虚拟机。

使用virt-manager创建2台普通的KVM虚拟机。

5）配置libvirt secret。

编写账户信息文件（真实机操作）

[root@room9pc01 ~]# vim secret.xml //新建临时文件，内容如下

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

<usage type='ceph'>

<name>client.admin secret</name>

</usage>

</secret>

#使用XML配置文件创建secret

[root@room9pc01 ~]# virsh secret-define --file secret.xml

733f0fd1-e3d6-4c25-a69f-6681fc19802b //随机的UUID，这个UUID对应的有账户信息

编写账户信息文件（真实机操作）

[root@room9pc01 ~]# cat /etc/ceph/ceph.client.admin.keyring

设置secret，添加账户的密钥

[root@room9pc01] virsh secret-set-value --secret 733f0fd1-e3d6-4c25-a69f-6681fc19802b --base64 AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg

//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）虚拟机的XML配置文件。

每个虚拟机都会有一个XML配置文件，包括：

虚拟机的名称、内存、CPU、磁盘、网卡等信息

[root@room9pc01 ~]# vim /etc/libvirt/qemu/vm1.xml

//修改前内容如下

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

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

<source file='/var/lib/libvirt/images/vm1.qcow2'/>

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

<address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>

</disk>

[root@room9pc01] virsh edit vm1 //vm1为虚拟机名称

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

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

<auth username='admin'>

<secret type='ceph'

uuid='733f0fd1-e3d6-4c25-a69f-6681fc19802b'/>

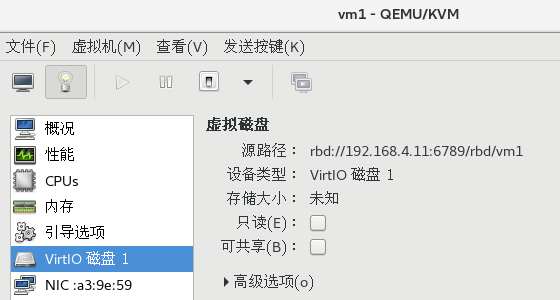
</auth>

<source protocol='rbd' name='rbd/vm1-image'>\_x000B\_ <host name='192.168.4.11' port='6789'/>\_x000B\_</source>

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

<address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x0'/>

</disk>



## **案例2：Ceph文件系统**

延续前面的实验，实现Ceph文件系统的功能。具体实现有以下功能：

1) 部署MDS节点

2) 创建Ceph文件系统

3) 客户端挂载文件系统

添加一台虚拟机，部署MDS节点。

主机的主机名及对应的IP地址如表所示。

|  |  |
| --- | --- |
| **主机名称** | **值** |
| node4 | 192.168.4.14 |

实现此案例需要按照如下步骤进行。

1）添加一台新的虚拟机，要求如下：

IP地址:192.168.4.14

主机名:node4

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node4

2）部署元数据服务器

登陆node4，安装ceph-m ds软件包

[root@node4 ~]# yum -y install ceph-mds

登陆node1部署节点操作

[root@node1 ~]# cd /root/ceph-cluster

//该目录，是最早部署ceph集群时，创建的目录

[root@node1 ceph-cluster]# ceph-deploy mds create node4

//给nod4拷贝配置文件，启动mds服务

同步配置文件和key

[root@node1 ceph-cluster]# ceph-deploy admin node4

3）创建存储池

[root@node4 ~]# ceph osd pool create cephfs\_data 128

//创建存储池，对应128个PG(数字128随便写,尽量是2的n次方)

[root@node4 ~]# ceph osd pool create cephfs\_metadata 128

//创建存储池，对应128个PG

4）创建Ceph文件系统

[root@node4 ~]# ceph mds stat //查看mds状态

e2:, 1 up:standby

[root@node4 ~]# 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@node4 ~]# ceph fs ls

name: myfs1, metadata pool: cephfs\_metadata, data pools: [cephfs\_data ]

[root@node4 ~]# ceph mds stat

e4: 1/1/1 up {0=node4=up:creating}

5）客户端挂载

[root@client ~]# mount -t ceph 192.168.4.11:/ /mnt/cephfs/ -o name=admin,secret=AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg==

//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

[root@client ~]# df -Th /mnt/cephfs/

文件系统 类型 容量 已用 可用 已用% 挂载点

192.168.4.11:6789:/ ceph 120G 588M 120G 1% /mnt/cephfs

**案例3：创建对象存储服务器**

延续前面的实验，实现Ceph对象存储的功能。具体实现有以下功能：

1) 安装部署Rados Gateway

2) 启动RGW服务

3) 设置RGW的前端服务与端口

4) 客户端测试

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

[root@node1 ~]# ceph-deploy install --rgw node5

同步配置文件与密钥到node5

[root@node1 ~]# cd /root/ceph-cluster

[root@node1 ceph-cluster]# ceph-deploy admin node5

3）新建网关实例

启动一个rgw服务

[root@node1 ceph-cluster]# ceph-deploy rgw create node5

登陆node5验证服务是否启动

[root@node5 ~]# ps aux | grep radosgw

ceph 4109 0.2 1.4 2289196 14972 ? Ssl 22:53 0:00 /usr/bin/radosgw -f --cluster ceph --name client.rgw.node4 --setuser ceph --setgroup ceph

[root@node5 ~]# systemctl status ceph-radosgw@rgw.node5.service

4）修改服务端口

登陆node5，RGW默认服务端口为7480，修改为8000或80更方便客户端记忆和使用

[root@node5 ~]# vim /etc/ceph/ceph.conf

......

[client.rgw.node5]

host = node5

rgw\_frontends = "civetweb port=8000"

//node5为主机名

//civetweb是RGW内置的一个web服务

[root@node5 ~]# systemctl restart ceph-radosgw@rgw.node5.service

[root@node5 ~]# ss -antulp | grep 8000

tcp LISTEN 0 128 \*:8000 \*:\* users:(("radosgw",pid=3193,fd=30))

步骤二：客户端测试

1）curl测试

[root@client ~]# curl 192.168.4.15:8000

<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><Owner><ID>anonymous</ID><DisplayName></DisplayName></Owner><Buckets></Buckets></ListAllMyBucketsResult>

2）使用第三方软件访问

登陆node5（RGW）创建账户

[root@node5 ~]# radosgw-admin user create --uid="testuser" --display-name="First User"

… …

"keys": [

{

"user": "testuser",

"access\_key": "5E42OEGB1M95Y49IBG7B",

"secret\_key": "i8YtM8cs7QDCK3rTRopb0TTPBFJVXdEryRbeLGK6"

}

],

... ...

#

[root@node5 ~]# radosgw-admin user info --uid=testuser

//testuser为用户，key是账户访问密钥

3）客户端安装软件

[root@client ~]# yum install s3cmd-2.0.1-1.el7.noarch.rpm #此包不在光盘中,需要自己上网下载

修改软件配置（注意，除了下面设置的内容，其他提示都默认回车）

[root@client ~]# s3cmd --configure

Access Key: 5E42OEGB1M95Y49IBG7B\_x000B\_Secret Key: i8YtM8cs7QDCK3rTRopb0TTPBFJVXdEryRbeLGK6

S3 Endpoint [s3.amazonaws.com]: 192.168.4.15:8000

[%(bucket)s.s3.amazonaws.com]: %(bucket)s.192.168.4.15:8000

Use HTTPS protocol [Yes]: No

Test access with supplied credentials? [Y/n] n

Save settings? [y/N] y

//注意，其他提示都默认回车  
4）创建存储数据的bucket（类似于存储数据的目录）

[root@client ~]# s3cmd ls

[root@client ~]# s3cmd mb s3://my\_bucket

Bucket 's3://my\_bucket/' created

[root@client ~]# s3cmd ls

2018-05-09 08:14 s3://my\_bucket

[root@client ~]# s3cmd put /var/log/messages s3://my\_bucket/log/

[root@client ~]# s3cmd ls

2018-05-09 08:14 s3://my\_bucket

[root@client ~]# s3cmd ls s3://my\_bucket

DIR s3://my\_bucket/log/

[root@client ~]# s3cmd ls s3://my\_bucket/log/

2018-05-09 08:19 309034 s3://my\_bucket/log/messages

测试下载功能

[root@client ~]# s3cmd get s3://my\_bucket/log/messages /tmp/

测试删除功能

[root@client ~]# s3cmd del s3://my\_bucket/log/messages