**LNMP平台搭建**

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

Nginx环境搭建

yum -y install gcc make

yum -y install pcre-devel openssl-devel zlib-devel

./configure

--prefix=/usr/local/nginx

--user=nginx

--group=nginx

--with-http\_ssl\_module //ssl模块

--with-http\_stub\_status\_module //状态查看模块

--with-stream

make && make install

//关闭httpd服务，与nginx80端口冲突

//可以通过./configure --help | grep --with 查看可以加载的模块

Mariadb数据库安装

yum -y install mariadb mariadb-server mariadb-devel

Php环境搭建

yum -y install php

yum -y install php-fpm (个人rpm包提供)

yum -y install php-mysql

Nginx升级/添加模块

下载需要的echo模块

# wget https://github.com/openresty/echo-nginx-module/archive/v0.60.tar.gz

# tar -zxvf v0.60.tar.gz

# cp /echo-nginx-module-0.60 /home/usrdata/nginx-1.10/echo-nginx-module-0.60

下载nginx源码

# wget http://nginx.org/download/nginx-1.10.1.tar.gz

# tar -zxvf nginx-1.10.1.tar.gz

# cd nginx-1.10.1

# ./configure --add-module=/home/usrdata/nginx-1.10/echo-nginx-module-0.60 ... (复制之前已经安装好的模块)

# make （不要make install会直接覆盖安装）

# mv /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginx.old

# cp -f objs/nginx /usr/local/nginx/sbin/nginx

这里是平滑升级，如是全新安装请执行：

#make install# make upgrade # make clean （清除操作，可以忽略）

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

用户认证

生成密码文件

#yum -y install httpd-tools

#htpasswd -c /usr/local/nginx/pass tom //创建密码文件

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

修改配置文件

Vim /usr/local/nginx/conf/nginx.conf

server {

listen 80;

server\_name localhost;

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

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

location / {

root html;

index index.html index.htm;

}

}

基于域名的虚拟主机

server {

listen 80;                                      //端口

server\_name www.b.com;     //域名

location / {

root www;                                  //指定网站根路径

index index.html index.htm;

}

}

基于端口的虚拟主机

listen 80;                                      //端口

基于IP的虚拟主机

listen 192.168.4.254：80;                                      //端口

基于加密网站的虚拟主机（源码安装使用--with-http\_ssl\_module）

#生成私钥与证书

cd /usr/local/nginx/conf

openssl genrsa > cert.key //生成私钥

openssl refq -new -x509 -key cert.ket > cert.pem //生成证书

#修改配置文件

vim /usr/local/nginx/conf/nginx.conf //原文件被注释，取消注释即可

# HTTPS server

#

#server {

# listen 443 ssl;

# server\_name localhost;

# 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;

# }

#}

配置支持PHP动态网页

vim /usr/local/nginx/conf/nginx.conf //原文件被注释，取消注释并添加index.php

location ~ \.php$ {

root html;

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

fastcgi\_index index.php;

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

include fastcgi.conf; //模版错误，正确包含在fastcgi.conf

}

// ~支持正则表达式。Location路径首先选择匹配原则

// 日志错误文件路径 /usr/local/nginx/logs/error.log nginx

/var/log/php-fpm/www-error.log php

// tailf 查看错误日志

地址重写

rewrite 旧地址 新地址 [选项]

last 不再读其他rewrite

break 不再读其他语句，结束请求

redirect 临时重定向

permanent 永久重定向

1、访问a.html重定向到b.html

location / {

    root html;

index index.html index.htm;

rewrite /a.html /b.html;

}

2、跳转地址栏

location / {

    root html;

index index.html index.htm;

rewrite /a.html /b.html redirect;

}

3、访问192.168.4.5的请求重定向至www.tmooc.cn

server {

listen 80;

server\_name localhost;

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

location / {

    root html;

index index.html index.htm;

}

}

1. (访问192.168.4.5/下面子页面，重定向至www.tmooc.cn/下相同的页面)

servserver {

listen 80;

server\_name localhost;

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

5、curl和火狐访问相同链接返回的页面不同

server {

listen 80;

server\_name localhost;

location / {

    root html;

index index.html index.htm;

}

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

rewrite ^(.\*)$ /firefox/$1; //firefox浏览器用户访问firefox目录下内容

}

}

Nginx反向代理--集群

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

.. ..

http {

.. ..

upstream webserver {

  ip\_hash;

server 192.168.2.100:80 weight=1 max\_fails=2 fail\_timeout=10;

server 192.168.2.200:80 weight=2 max\_fails=2 fail\_timeout=10;

}

.. ..

server {

listen        80;

server\_name localhost;

location / {

proxy\_pass http://webserver;

}

}

//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

// ip\_hash 根据客户端ip分配固定的后端服务器

TCP/UDP调度器

--with-stream 开启ngx\_stream\_core\_module模块

配置在http外

stream {

upstream backend {

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

server 192.168.2.200:22;

}

server {

listen 12345; //Nginx监听的端口

proxy\_connect\_timeout 1s;

proxy\_timeout 3s;

proxy\_pass backend;

}

}

连接测试ssh 192.168.4.5 -p 12345

Nginx优化

1>优化并发量

# ab -n 2000 -c 2000 http://192.168.4.5/ //模拟多并发

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

.. ..

worker\_processes 2; //与CPU核心数量一致

events {

worker\_connections 65535; //每个worker最大并发连接数

use epoll;

}

.. ..

修改配置文件后仍然无法高并发需要修改linux内核参数

# ulimit -a //查看所有属性值

# ulimit -Hn 100000 //设置硬限制（临时规则）

# ulimit -Sn 100000 //设置软限制（临时规则）

# vim /etc/security/limits.conf //永久

.. ..

\* soft nofile 100000

\* hard nofile 100000

//该配置文件分4列，分别如下：

//用户或组 硬限制或软限制 需要限制的项目 限制的值

2>解决客户端访问头部信息过长的问题

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

.. ..

http {

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

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

.. ..

}

3>客户端浏览器缓存数据

Firefox地址栏内输入about:cache将显示Firefox浏览器的缓存信息

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

server {

...

location ~\* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ { //浏览器缓存图片等信息

expires 30d; //定义客户端缓存时间为30天

}

....

}

4>自定义报错页面

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

.. ..

error\_page 404 /40x.html;    //自定义错误页面

.. ..

常见http状态码

200 一切正常

301 永久重定向

302 临时重定向

401 用户名或密码错误

403 禁止访问（客户端ip地址被拒绝）

404 文件不存在

414 请求URL头部过长

500 服务器内部错误

502 Bad Gateway

5>查看服务器状态信息

使用模块 --with-http\_stub\_status\_module

# cat /usr/local/nginx/conf/nginx.conf

… …

location /status {

stub\_status on;

}

… …

查看

#curl http://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。

6>开启gzip压缩功能，提高数据传输效率

# cat /usr/local/nginx/conf/nginx.conf

http {

.. ..

gzip on; //开启压缩

gzip\_min\_length 1000; //小文件不压缩

gzip\_comp\_level 4; //压缩比率

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

//对特定文件压缩，类型参考mime.types

.. ..

}

7>服务器内存缓存

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

}

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）

**Memcached**

---------------------------------------------------------------------------------------------------------------------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，提高web应用的响应速度。

memcache的工作流程如下：先检查客户端的请求数据是否在memcached中，如有，直接把请求数据返回，不再对数据库进行任何操作；如果请求的数据不在memcached中，就去查数据库，把从数据库中获取的数据返回给客户端，同时把数据缓存一份到memcached中（memcached客户端不负责，需要程序明确实现）；

验证时需要客户端主机安装telnet，远程memcached来验证服务器的功能：

add name 0 180 10 //变量不存在则添加

set name 0 180 10 //添加或替换变量

replace name 0 180 10 //替换

get name //读取变量

append name 0 180 10 //向变量中追加数据

delete name //删除变量

stats //查看状态

flush\_all //清空所有

提示：0表示不压缩，180为数据缓存时间，10为需要存储的数据字节数量。

Memcached安装

yum -y install memcached

systemctl start memcached

systemctl enable memcached

//配置文件

# vim /etc/sysconfig/memcached

PORT="11211"

USER="memcached"

MAXCONN="1024"

CACHESIZE="64"

OPTIONS=""

客户端访问：telnet ip address 11211

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

LNMP+Memcached

yum -y install php-pecl-memcache //默认php无法连接memcache数据库

PHP的服务器本地Session信息

ls /var/lib/php/session/

PHP实现session共享,将session信息存放在memcache数据库

为实现客户端访问两台不同的后端Web服务器时，Session 信息一致

修改web服务器配置文件：

//web服务器都需要安装php-pecl-memcache

# 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）

**Tomcat安装**

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

1. 安装jdk环境(默认已经安装)

yum -y install java-1.8.0-openjdk

yum -y install java-1.8.0-openjdk-headless

1. 安装Tomcat

tar -xf apache-tomcat-8.0.30.tar.gz

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

1. 启动Tomcat

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

1. 访问测试

Firefox http//:localhost:8080 (文件地址/usr/local/tomcat/webapps/ROOT/index.jsp)

//监控网络端口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）。

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

# ln -s /dev/urandom /dev/random

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

部署虚拟主机

Vim /usr/local/tomcat/conf/server.xml

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

</Host>

Name:虚拟主机名称

appbase:定义基础目录，基础目录下默认项目ROOT

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

DocBase 指定页面存储位置，默认ROOT

Path 指定访问URL 默认index.html

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

跳转

配置SSL加密网站

1. 创建私钥和证书

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

//jdk软件包安装生成工具

//提示输入密码为:123456, 交互界面信息输入完成输入是或者y。

//-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

//y或是 确认输入信息正确

1. 修改配置文件server.xml(取消注释添加密钥地址及密码即可)

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

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

clientAuth="false" sslProtocol="TLS"keystorefile="/usr/local/tomcat

/keystore" keystroePass="123456"

/>

//打开注释添加加密文件配置信息

虚拟主机设置独立日志文件

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

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

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

prefix=" **bb\_access**" suffix=".txt"

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

</Host>

查看日志：ls /usr/local/tomcat/logs

利用Nginx作为调度器------------配置Tomcat集群

http...

upstream toms {

server 192.168.2.100:8080;

server 192.168.2.200:8080;

}

Server...

...

Location...{

proxy\_pass http://toms;

}

Varnish缓存服务器加速web服务

作用：使用Varnish加速后端Web服务

代理服务器可以将远程的Web服务器页面缓存在本地

远程Web服务器对客户端用户是透明的

利用缓存机制提高网站的响应速度

使用varnishadm命令管理缓存页面

使用varnishstat命令查看Varnish状态

部署

yum -y install gcc readline-devel //安装软件依赖包

yum -y install ncurses-devel //安装软件依赖包

yum -y install pcre-devel //安装软件依赖包

yum -y install python-docutils-0.11-0.2.20130715svn7687.el7.noarch.rpm //个人下载

useradd -s /sbin/nologin varnish //创建账户

源码安装

tar -xf varnish-5.2.1.tar.gz

cd varnish-5.2.1

./configure

make && make install

修改配置

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

vim /usr/local/etc/default.vcl

backend default {

.host = "192.168.2.100"; //后段web服务器地址

.port = "80";

}

启动服务：varnishd -f /usr/local/etc/default.vcl

客户端测试：curl http://192.168.4.5

//varnishd 默认使用内存缓存

//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

查看varnish日志

#varnishlog //varnish日志

#varnishncsa //访问日志

更新数据缓存-----后台web服务更新页面后，代理服务器缓存更新（默认会自动更新，但非实时更新）

#varnishadm

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

**Subversion**

记录每一个客户的更改，最终合并版本。（个别冲突问题，需要人为手动解决）

服务器配置

1. 安装软件包

yum -y install subversion

2、创建版本库

mkdir /var/svn

svnadmin create /var/svn/project

1. 导入项目代码

cd /usr/lib/systemd/system

svn import ./ file:///var/svn/project -m “Init’

1. 修改配置文件，创建账户与密码

#vim /var/svn/project/conf/svnserve.conf

anon-access = none //19行，匿名无任何权限（write\read\none）

auth-access = write //20行，有效账户可写

password-db = passwd //27行，密码文件

authz-db = authz //34行，ACL访问控制列表文件

Vim /var/svn/project/conf/passwd

[users]

harry = pass //用户名和密码

tom = pass //用户名和密码

Vim /var/svn/project/conf/authz

[/] //定义ACL访问控制

harry = rw //用户对项目根路径可读可写

tom = rw

\* = r //其他人只读

1. 启动服务

Svnserve -d -r /var/svn/project

客户端操作

1. 安装软件包

yum -y install subversion

1. 下载服务器代码

cd /tmp

svn --username harry --password pass co svn://192.168.4.5/ code

//建立本地副本,从服务器192.168.4.5上co下载代码到本地code目录

//用户名harry,密码pass

1. 客户端操作

#svn ci -m "modify user" //将本地修改的数据同步到服务器

#svn update //将服务器上新的数据同步到本地

#svn info     svn://192.168.4.5    //查看版本仓库基本信息

#svn log svn://192.168.2.100 //查看版本仓库的日志

# echo "test" > test.sh //本地新建一个文件

# svn ci -m "new file" //提交失败，该文件不被svn管理

# svn add test.sh //将文件或目录加入版本控制

# svn ci -m "new file" //再次提交，成功

# svn mkdir subdir //创建子目录

# svn rm timers.target //使用svn删除文件

# svn ci -m "xxx" //提交一次代码

#vim umount.target //任意修改本地的一个文件

# svn diff //查看所有文件的差异

# svn diff umount.target //仅查看某一个文件的差异

# svn cat svn://192.168.2.100/reboot.target //查看服务器文件的内容

# sed -i 'd' tmp.mount //删除文件所有内容，但未提交

# svn revert tmp.mount //还原tmp.mount文件

# rm -rf \*.target //任意删除若干文件

# svn update //还原

# sed -i '1a #test###' tuned.service //修改本地副本中的代码文件

# svn ci -m "xxx" //提交代码

# svn merge -r7: 2 tuned.service //将文件从版本7还原到版本2

//-m “xxx” -m指定日志信息ARG 为注释内容

修改相同文件的相同行，冲突解决

Conflict discovered in 'tuned.service'. //出现冲突需要解决

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

# svnadmin dump /var/svn/project > project.bak //版本库存备份

# svnadmin load /var/svn/project2 < project.bak //还原

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

访问方式：

File:/// http:// https:// svn:// svn+ssh://

svn命令测试svnserver服务命令列表

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

//Windows 图形客户端软件Tortoise svn

**Git**

-----------------------------------------------------------------------------------------------------------------------网站注册github账户

emal: 787560291@qq.com 账户名:songweihong132731 密码:boedt132731

安装软件包

#yum -y install git

#git config --global user.name “用户名”

#git config --global user.email “用户邮件地址”

#git config --global core.editor vim //设置默认编辑器为vim

#git config --list //查看用户配置

#cat ~/.gitconfig

创建仓库

#mkdir /devops

#cd /devops

#git init

添加文件到暂存区

#echo “print(“hello world!”)” > hello.py

#git add hello.py //添加所有文件 git add .

#git status

确认到仓库

#git commit -m “提交信息”

#git status

将本地仓库推送到服务器

#git remote rename orign old-origin

Git clone https://github.com/boedt132731/1803

Git常用命令:

add 添加文件内容至索引

bisect 通过二分查找定位引入 bug 的变更

branch 列出、创建或删除分支

checkout 检出一个分支或路径到工作区

clone 克隆一个版本库到一个新目录

commit 记录变更到版本库

diff 显示提交之间、提交和工作区之间等的差异

fetch 从另外一个版本库下载对象和引用

grep 输出和模式匹配的行

init 创建一个空的 Git 版本库或重新初始化一个已存在的版本库

log 显示提交日志

merge 合并两个或更多开发历史

mv 移动或重命名一个文件、目录或符号链接

pull 获取并合并另外的版本库或一个本地分支

push 更新远程引用和相关的对象

rebase 本地提交转移至更新后的上游分支中

reset 重置当前HEAD到指定状态

rm 从工作区和索引中删除文件

show 显示各种类型的对象

status 显示工作区状态

tag 创建、列出、删除或校验一个GPG签名的 tag 对象

**RPM打包**

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

将源码包打包为RPM包

准备源码软件----->安装rpm-build----->编写编译配置文件----->编译RPM包

1、安装rpm-build软件

Yum -y install rpm-build

1. 生成rpmbuild目录

rpmbuild -ba nginx.spec //报错信息不用处理，会生成rpmbuild文件目录

ls /root/rpmbuild

1. 将源码软件复制到SOURCES目录

cp nginx-1.12.2.tar.gz /root/rpmbuild/SOURCES/

1. 创建并修改SPEC配置文件

Vim /root/rpmbuild/SPECS/nginx.spec

Name: 软件名称

Version: 软件版本

Release: 发布次数 如: 1%{?dist}

Summary: 软件说明

Group: 软件分组

License: 授权模式,例如 GPL，即自由软件

URL: 源码包的URL地址，可随意填写

Source0: 源码包,可指定多个,下面可用%{SOURCE0}变量引用

BuildRoot: 编译过程中的中间存档目录，考虑到多用户的环境，一般定义为：

%{\_tmppath}/%{name}-%{version}-%{release}-root ，

后面可使用$RPM\_BUILD\_ROOT 方式引用

BuildArch: 平台 %{\_arch}

BuildRequires: 编译过程依赖的工具

Requires: 打包生成的rpm包安装时所依赖的软件包

%description 说明文档

%prep 准备部分,比如创建目录,解压源码包等,可使用%setup内部函数

%build 在BUILD目录编译,可使用%configure内部函数,或者其他编译工具,如cmake, perl等

%install 安装到BUILDROOT虚拟目录

%clean 清理文件

%files 将指定的文件添加到rpm包中,文档类型可用%doc,配置文件可 用%config

%changelog 更新记录.格式: 第一行 "\* 日期 作者 " 第二行 "- 更新内容"

最终的生成的rpm名称: {Name}-{Version}-{Relesae}-{BuildArch}.rpm

Nginx实例：

Name:nginx

Version:1.12.2

Release: 1%{?dist}

Summary:nginx

Group: nginx

License:GPL

URL: www.test.com

Source0: nginx-1.12.2.tar.gz

#BuildRequires:

#Requires:

%description

nginx 1.12.2 Version

%prep

%setup -q

%build

./configure

make %{?\_smp\_mflags}

%install

make install DESTDIR=%{buildroot}

%files

%doc

/usr/local/nginx/\*

%changelog

1. rpmbuild创建RPM软件包

rpmbuild -ba /root/rpmbuild/SPECS/nginx.spec

ls /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

**VPN**

Virtual Private Network（虚拟专用网络）

方式：GRE、PPTP、L2TP+IPSec、SSL

1. GRE方式---------实现点到点的隧道通信（缺少加密机制）

1> 加载模块

Modprobe ip\_gre

// lsmod 显示模块列表

// modprobe ip\_gre 加载内核模块

// modinfo ip\_gre 查看模块信息

2>服务器创建VPN隧道

Ip tunnel add tun0 mode gre remote IP local IP

Ip link set tun0 up

Ip addr add 10.10.10.10/24 peer 10.10.10.5/24 dev tun0 //设置隧道ip

3>客户端创建VPN隧道

Ip tunnel add tun0 mode gre remote IP local IP

Ip link set tun0 up

Ip addr add 10.10.10.5/24 peer 10.10.10.10./24 dev tun0

4>ping测试连通性

1. PPTP（Point to Point Tunneling Protocol）------支持身份验证、数据加密、地址分配

1>部署服务器

Yum -y install pptpd-1.4.0-2.el7.x86\_64.rpm

修改配置文件

#Vim /etc/pptpd.conf

localip 201.1.2.5 //服务器本地IP

remoteip 192.168.3.1-50 //分配给客户端的IP池

#Vim /etc/ppp/options.pptpd

require-mppe-128 //使用MPPE加密数据

ms-dns 8.8.8.8 //DNS服务器

#Vim /etc/ppp/chap-secrets

jacob \* 123456 \* //用户名 服务器标记 密码 客户端

#echo 1 > /proc/sys/net/ipv4/ip\_forward //开启路由转发

#systemctl start pptpd

#systemctl enable pptpd

#Iptables -t nat -A POSTROUTING -s 192.168.3.0/24 -j SNAT --to-source 201.1.2.5 //翻墙设置(非必须操作)

2>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，输入VPN服务器账户与密码，连接VPN并测试网络连通性。

1. L2TP+IPSec 方式-------支持身份认证、加密、分配地址

服务器设置：

1>部署ipsec服务

#yum -y install libreswan

#vim /etc/ipsec.conf

Include /etc/ipsec.d/\*.conf //加载配置文件

#vim /etc/ipsec.d/myipsec.conf //创建秘钥验证配置文件

conn IDC-PSK-NAT

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

also=IDC-PSK-noNAT

conn IDC-PSK-noNAT

authby=secret //加密认证

ike=3des-sha1;modp1024 //算法

phase2alg=aes256-sha1;modp2048 //算法

pfs=no

auto=add

keyingtries=3

rekey=no

ikelifetime=8h

keylife=3h

type=transport

left=201.1.2.200 //重要，服务器本机的外网IP

leftprotoport=17/1701

right=%any //允许任何客户端连接

rightprotoport=17/%any

#vim /etc/ipsec.d/mypass.secrets //定义共享秘钥

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

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

201.1.2.200 %any: PSK "randpass" //randpass为密钥 //201.1.2.200是VPN服务器的IP

#systemctl start ipsec //启动服务

2>部署XL2TP服务

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

#vim /etc/xl2tpd/xl2tpd.conf //修改主配置文件

[global]

.. ..

[lns default]

.. ..

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

local ip = 201.1.2.200 //VPN服务器的IP地址

#vim /etc/ppp/options.xl2tpd //认证配置

require-mschap-v2 //添加一行，强制要求认证

#crtscts //注释或删除该行

#lock //注释或删除该行

#vim /etc/ppp/chap-secrets //密码文件

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

#systemctl start xl2tpd //启动服务

#echo 1 > /proc/sys/net/ipv4/ip\_forward //设置路由转发

#iptables -t nat -A POSTROUTING -s 192.168.3.0/24 -j SNAT --to-source 201.1.2.200 //翻墙设置（非必须操作）

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，输入VPN服务器账户与密码

**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。

Network Time Protocol（网络时间协议）采用的是分层设计，总数限制在15以内（包括15）。

服务器

#yum -y install chrony

#vim /etc/chrony.conf

.. ..

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

allow 192.168.4.0/24 //允许那个IP或网络访问NTP

#deny 192.168.4.1 //拒绝那个IP或网络访问NTP

local stratum 10 //设置NTP服务器的层数量

.. ..

# systemctl restart chronyd

# systemctl enable chronyd

客户端

#yum -y install chrony

#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

# systemctl restart chronyd

**PSSH远程工具**

并发远程操作其他主机,需安装python2.4以上版本才可以使用。

1>安装软件

#yum -y install pssh-2.3.1-5.el7.noarch.rpm

2>修改/etc/hosts本地解析文件

#vim /etc/hosts

3>实际应用

生成密钥并部署在远程主机

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

# ssh-copy-id -i /root/.ssh/id\_rsa.pub host1 //发送密钥，其他客户机相同操作

# pssh -h host.txt -o /tmp/ -x '-o StrictHostKeyChecking=no' echo hello

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

多并发远程其他主机并执行命令

#pssh -i -A -H 'host1 host2 host3' -x '-o StrictHostKeyChecking=no' echo hello

命令选项：

--version：查看版本

--help：查看帮助，即此信息

-h：主机文件列表，内容格式”[user@]host[:port]”

-H：主机字符串，内容格式”[user@]host[:port]”

-：登录使用的用户名

-p：并发的线程数【可选】

-o：输出的文件目录【可选】

-e：错误输入文件【可选】

-t：TIMEOUT 超时时间设置，0无限制【可选】

-O：SSH的选项

-v：详细模式

-A：手动输入密码模式

-x：额外的命令行参数使用空白符号，引号，反斜线处理

-X：额外的命令行参数，单个参数模式，同-x

-i：每个服务器内部处理信息输出

-P：打印出服务器返回信息

批量、多并发copy数据到其他主机

#pscp.pssh -h host.txt -x '-o StrictHostKeyChecking=no' etc.tar.gz /

批量、多并发从其他主机下载数据到本机

# pslurp -r -h host.txt -x '-o StrictHostKeyChecking=no' /etc /tmp

批量、多并发杀死其他主机的进程

# pnuke -h host.txt -x '-o StrictHostKeyChecking=no' test

使用rsync协议本地文件同步到远程多台主机上