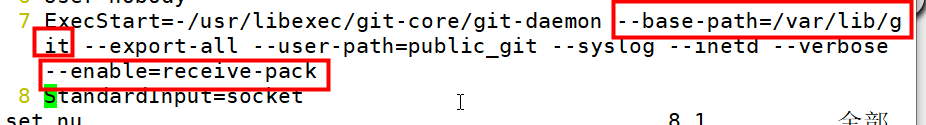
实验一：配置git服务端

1， 在服务端安装支持git协议的软件包

[root@proxy ~]# yum -y install git-daemon

[root@proxy ~]# vim [/usr/lib/systemd/system/git@.service](mailto:/usr/lib/systemd/system/git@.service)

修改第7行，--base-path是git服务器仓库的上级目录，--enable=receive-pack是支持对仓库写入数据



[root@proxy ~]# chmod -R 777 /var/lib/git/ 开放目录权限

[root@proxy ~]# systemctl start git.socket 开启服务

systemctl stop firewalld

setenforce 0

2，在web1

cd

rm -rf project 先把之前的仓库删除

git clone git://192.168.2.5/project 使用git协议克隆仓库

再测试新建文件或者修改现有文件

git add .

git commit -m "xxxx"

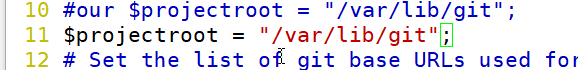
git push

3，测试使用http方式的git

[root@proxy ~]# yum -y install httpd gitweb

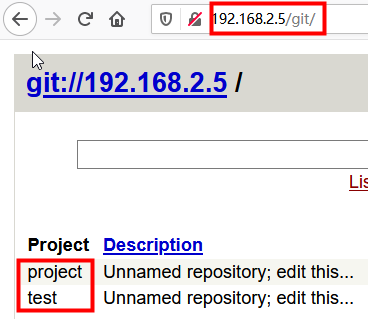
[root@proxy ~]# ss -ntulp | grep :80 检查80端口，如果被httpd之外的服务占用，一律kill

[root@proxy ~]# vim +11 /etc/gitweb.conf 在11行添加路径



[root@proxy ~]# systemctl restart httpd

然后使用浏览器访问git服务器



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

rpm包 使用方便 更新速度慢

源码包 使用繁琐 更新速度快

实验二：通过源码包制作rpm软件包

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

[root@web1 ~]# rpmbuild -ba nginx.spec 会报错，同时也会帮创建所需目录

[root@web1 ~]# cd rpmbuild/SPECS/ 进入配置文件目录

[root@web1 SPECS]# vim nginx.spec 创建制作rpm的配置文件，必须是.spec结尾

Name:nginx 软件名字，不能随意写

Version:1.12.2 软件版本，不能随意写

Release:1 作者发布的rpm包版本

Summary:test test 简单描述

#Group:

License:GPL 软件使用协议

<URL:www.test.com> 软件的官方网站

Source0:nginx-1.12.2.tar.gz tar包名，不能随意写

#BuildRequires:

#Requires:

%description 详细描述

test test test

%post 可选配置，表示安装rpm的同时还执行的命令

useradd nginx

%prep

%setup -q

%build

./configure 稍作修改

make %{?\_smp\_mflags}

%install

make install DESTDIR=%{buildroot}

%files

%doc

/usr/local/nginx/\* 制作rpm的所有内容

%changelog

之后保存退出

cp lnmp\_soft/nginx-1.12.2.tar.gz rpmbuild/SOURCES/ 拷贝tar包到sources目

yum -y install gcc pcre-devel openssl-devel 安装nginx依赖包

rpmbuild -ba rpmbuild/SPECS/nginx.spec 制作rpm包

ls rpmbuild/RPMS/x86\_64/ 查看结果

rpm -qpi rpmbuild/RPMS/x86\_64/nginx-1.12.2-1.x86\_64.rpm 查看制作号的rpm包的信息

实验三：vpn 虚拟专用网

1. 在web1使用linux主机建立vpn（gre vpn）

[root@web1 ~]# modprobe ip\_gre 在内核开启gre

[root@web1 ~]# lsmod | grep ip\_gre 查看

[root@web1 ~]#ip tunnel add tun0 mode gre remote 192.168.2.200 local 192.168.2.100 创建vpn隧道，名字叫tun0，使用的是gre技术，远程主机ip是2.200，我自己的ip是2.100

[root@web1 ~]# ip addr show 查看结果，通常最后可以看到创建的名叫tun0的隧道

[root@web1 ~]# ip tunnel del tun0 如果配置错误，删除tun0

[root@web1 ~]# ip addr add 10.10.10.10/8 peer 10.10.10.5/8 dev tun0 在专用网（隧道）中配置内部ip，本机是10.10，远程主机是10.5，可以自由定义

[root@web1 ~]# ip link set tun0 up 开启隧道

关闭防火墙和selinux

web2主机的配置基本与上述一致，但ip要相反

最后使用web1去ping 10.10.10.5 能通则说明vpn建立成功

实验四 ：使用L2TP+IPSec VPN

1. 在proxy主机配置

cd /root/lnmp\_soft/vpn

[root@proxy vpn]# yum -y install xl2tpd-1.3.8-2.el7.x86\_64.rpm

[root@proxy vpn]# yum -y install libreswan

[root@proxy vpn]# cp myipsec.conf /etc/ipsec.d/

[root@proxy vpn]# vim /etc/ipsec.d/myipsec.conf

修改16行为vpn服务器地址，如果是真实环境需要填写公网ip

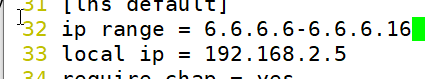


[root@proxy vpn]# vim /etc/ipsec.secrets 再次修改配置文件，在第2行添加本机ip、any是允许任何用户访问，PSK是预共享密钥，randpass是密码

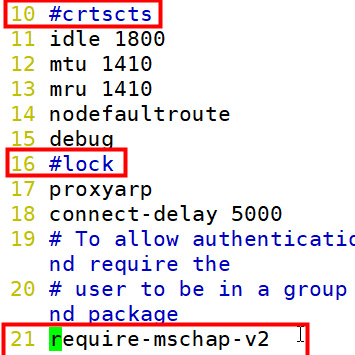


[root@proxy vpn]# vim /etc/xl2tpd/xl2tpd.conf

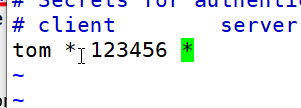
32行配置将来给windows客户端分配的ip，33行配置vpn服务器ip



[root@proxy vpn]# vim /etc/ppp/options.xl2tpd



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



[root@proxy vpn]# systemctl restart ipsec

[root@proxy vpn]# systemctl restart xl2tpd

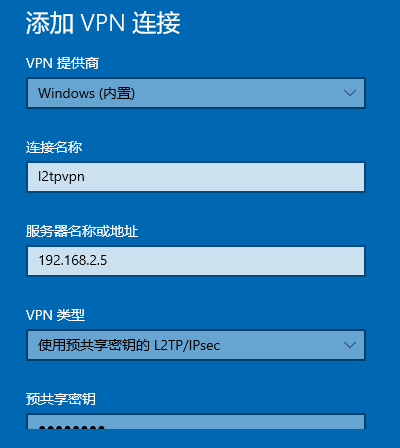
[root@proxy vpn]# ss -ntulp | grep 500

[root@proxy vpn]# ss -ntulp | grep xl2tpd

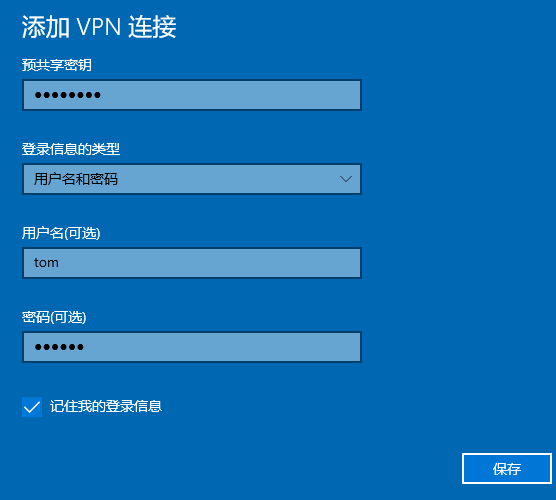
关闭防火墙 和 selinux ！！

1. 在windows主机

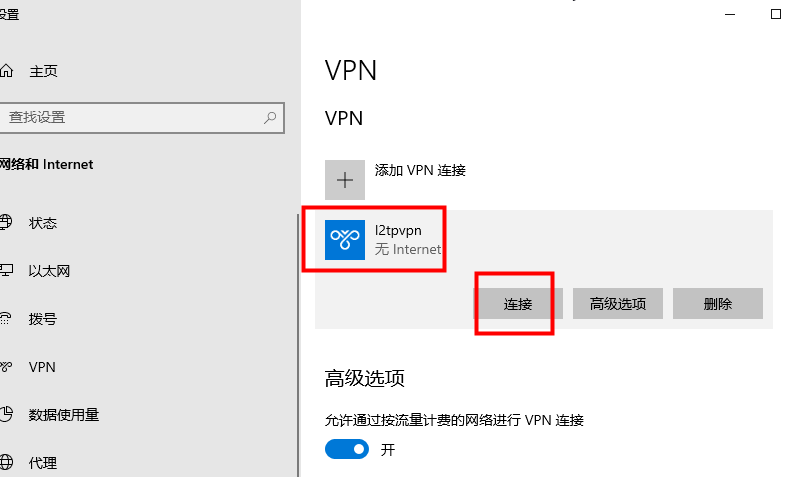




预共享密钥是randpass



tom账户的密码是123456





如果是win7系统，需要修改注册表，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

实验五：编写systemd Unit文件，使nginx支持systemctl命令

cat /usr/lib/systemd/system/httpd.service 查看httpd的unit文件

cd /usr/lib/systemd/system/

cp httpd.service nginx.service 复制httpd服务的unit文件作为模板给nginx服务

[root@proxy system]# vim nginx.service

[Unit]

Description=nginx

After=network.target remote-fs.target nss-lookup.target

[Service]

Type=forking 此类型代表nginx可以开启多进程，如果是单进程服务则使用simple

ExecStart=/usr/local/nginx/sbin/nginx 当输入systemctl start nginx时执行的指令

ExecReload=/usr/local/nginx/sbin/nginx -s reload 当输入systemctl reload nginx时执行的指令

ExecStop=/bin/kill -s QUIT ${MAINPID} 当输入systemctl stop nginx时执行的指令 -s QUIT是使用kill命令发送退出信号，${MAINPID}是进程号

[Install]

WantedBy=multi-user.target 在使用systemctl enable nginx时可以实现开机自启，原理是会在/etc/systemd/system/multi-user.target.wants/目录下创建一个此unit文件的软件连接，开机之后系统会将此目录的服务自动运行