目录：

[1、NFS服务使用指南 2](#_Toc418499242)

1.1NFS 介绍 2

1.2NFS在企业中的应用场景 2

1.3NFS系统原理介绍 4

1.4实战开始 5

1.5NFS系统应用优点缺点说明 23

2、Linux系统SSH服务24

2.1SSH介绍 24

2.2 SSH结构 25

2.3 SSH加密技术 25

2.4 SSH服务认证类型 27

2.5 远程链接SSH服务 29

2.6ssh客户端附带的远程拷贝scp命令 30

2.7 ssh服务附带的sftp功能服务 31

2.8ssh的企业生产应用场景 32

# 

# 1、NFS服务使用指南

**1.1 NFS介绍**

NFS是network file system的缩写，中文意思是网络文件系统，它的主要功能是通过网络（一般是局域网）让不同的主机系统之间可以共享文件或目录。NFS客户端（一般为应用服务器，例如web）可以通过挂载（mount）的方式将NFS服务器端共享的数据目录挂载到NFS客户端本地系统中（就是有一个挂载点下）。从NFS客户端的机器本地看，NFS服务器端共享的目录就好像是客户端自己的磁盘分区或者目录一样，而实际上却是远端的NFS服务器目录。

NFS网络文件系统很想Windows系统的网络共享、安全功能、网络驱动器映射，这也和Linux系统里的Samba服务类似。只不过一般情况，Windows网络共享服务或者Samba服务用于办公局域网共享，互联网中小型网站集群架构后端常用NFS作为数据共享，如果是大型网站，那么有可能还会用到更复杂的分布式文件系统，例如：moosefs(mfs)，glusterfs，fastDFS。

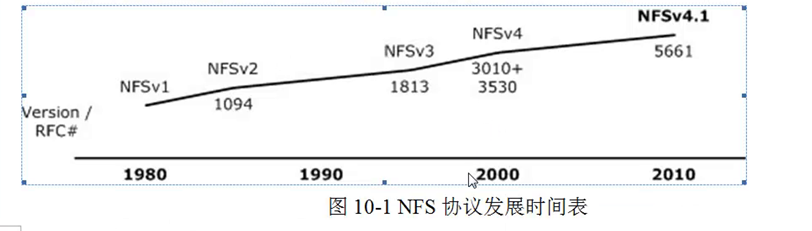
**NFS历史介绍**

第一个网络文件系统被称为 file Access Listener，由digital equipment corporation（DEC）在1976年开发。

NFS是第一个构建与IP协议之上的现代网络文件系统。在20世纪80年代，它首先作为实验的文件系统，由sun microsystems 在内部完成开发。NFS协议归为request for comments （RFC）标准，并烟花为NFSv2.作为一个标准，由于NFS与其他客户端和服务器的互操作能力很好而发展快速。

标准持续地演化为NFSv3，在RFC1813中有定义，这一新的协议比之前的版本具有更好的可扩展性，支持大文件（超过2GB），异步写入，以及将TCP作为传输协议，为文件系统在更广泛的网络中使用铺平了到了。在2000年，RFC3010（由RFC3530修订）将NFS带入企业级应用。Sun引入了具有较高安全性，带有状态协议的NFSv4（NFS之前的版本都是无状态的）。今天，NFS是版本4.1（由RFC5661定义），它增加了对跨越分布式服务器的并行访问的支持（称为PNFS extension）

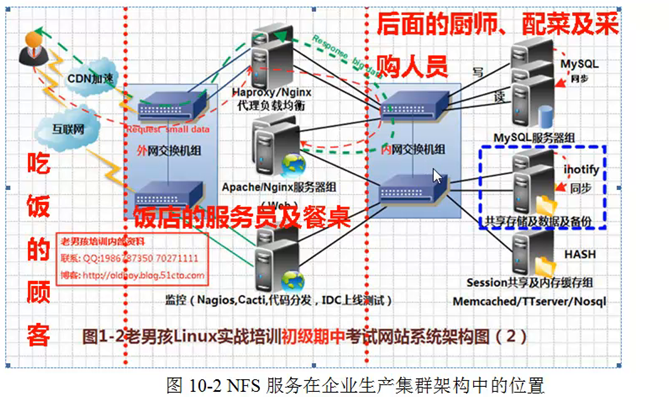
NFS系统发展的时间表，包括记录其特性的特定RFC，都在下图中展示



NFS系统已经历了近30年的发展，它代表了一个非常稳定的（及可移植）网络文件系统，盘、它具备可扩展、高性能等特性并达到企业级应用质量标准。由于忙完了速度的增加和延迟的降低，NFS系统移植是通过网络提供文件系统服务的有竞争力的选择，特别是中小型互联网企业，应用十分广泛。

**1.2 NFS在企业中的应用场景：**

在企业集群的工作场景中，NFS网络文件系统一般被用来存储共享视频、图片、附件等静态资源文件，一般是把网站用户上传的文件都放到NFS共享里，例如：BBS产品的图片、附件、头像，注意网站BBS程序不要放NFS共享里，然后前段所有的节点访问这些静态资源时都会读取NFS存储上的资源。NFS是当前互联网系统架构中最常用的数据存储服务之一，特别是中小型网站公司应用频率更高。大公司或门户除了使用NFS外，还可能会使用更为复杂的分布式文件系统moosefs（mfs），glusterfs，fastDFS等。

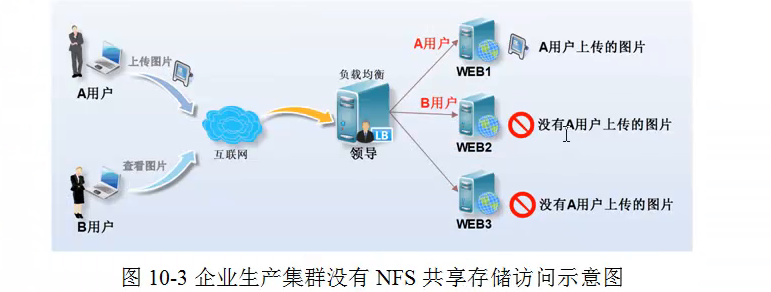


提示：上图中后面的虚线框里就是NFS系统工作的位置，NFS作为所有前段Web服务的共享存储，存储的内容一般有网站用户上传的图片、附件、头像等，注意，网站的程序代码不要放NFS共享里，网站程序是人工发布的，不存在延迟问题，直接力量发布到Web节点提供访问，这样访问效率更高。

**企业生产集群为什么需要共享存储角色**

通过图解展示集群架构需要共享存储服务的理由。例如：A用户传图片到Web1服务器，然后让B用户访问这张图片，结果B用户访问的请求分发到了Web2，因为Web2上没有这张图片，结果无法看到A用户传的图片，如果此时有一个共享存储，A用户上传图片无论分到Web1还是Web2上，最终都存储到共享存储上，此时，B用户访问图片时，无论分发到Web1还是Web2上，最终也都会去共享存储上访问，这样就可以访问到资源了。这个共享存储的位置可以通过开源软件和商业硬件实现，互联网中小型集群架构会用普通PC服务器和NFS文件系统实现。

**当集群中没有NFS共享存储，用户访问图片的情况：**



**如果集群中有NFS共享存储，用户访问图片的情况：**

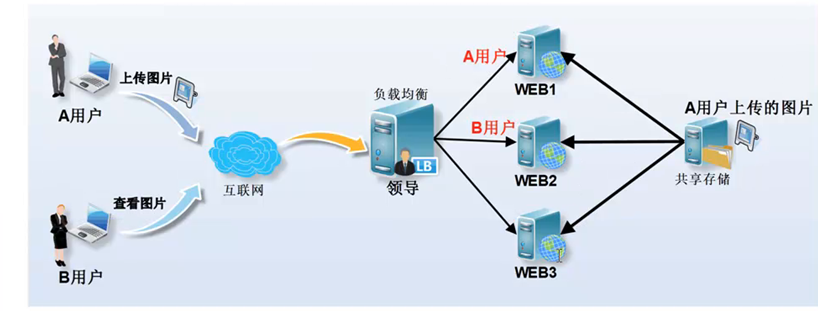


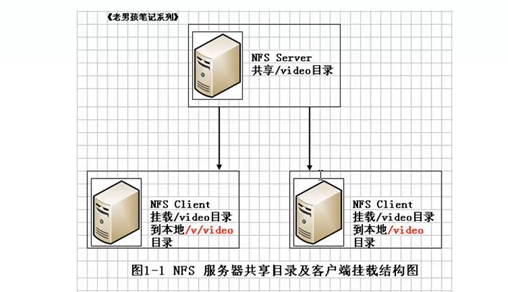
图10-4企业生产集群有NFS共享存储访问示意图

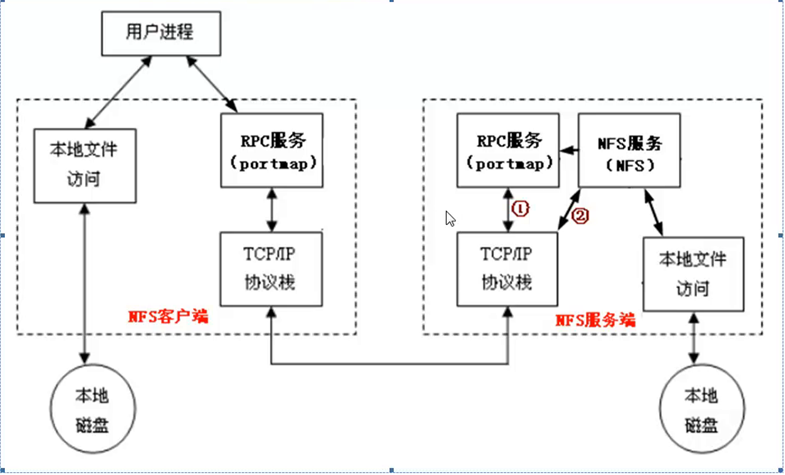
提示：中小型互联网企业一般不会买硬件存储因为太贵，大公司如果业务发展很快的话，可能会临时买硬件存储顶一下网站的压力，当网站并发继续加大后，硬件存储扩展就相对很费劲了，且价格成几何级数增加。例如：淘宝网就替换掉了很多硬件设备集群软件，用lvs+haproxy替换了netscaler负载均衡设备，用FastDFS.TFS配合PC服务器替换了NetApp、EMC商业存储设备，去IOE正在成为互联网公司的主流。

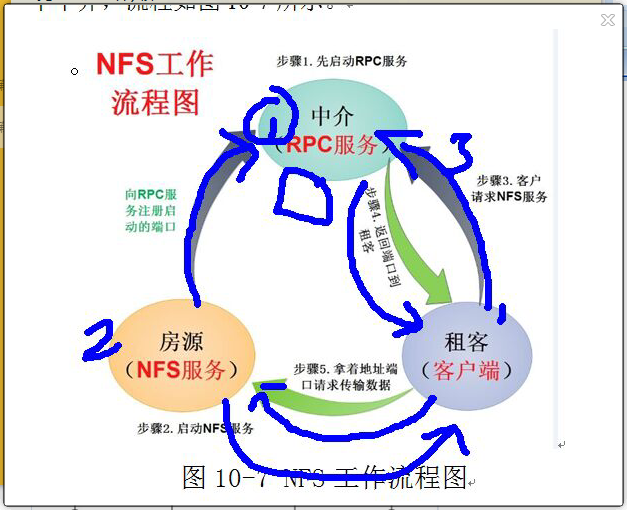
**1.3 NFS系统原理介绍**

NFS系统挂载结构图解与介绍

下图是企业工作中的NFS服务器与客户端挂载情况结构图解







**工作原理：**  
启动NFS服务时，会把信息和端口发送给RPC服务，当客户端访问时，RPC会查找对应的信息，反馈给客户端，用户会根据信息查找。  
注：NFS的RPC服务，在contos5.x 名称为portmap,在centos6.x下名称为rpbind

**1.4 实战开始**

NFS服务端操作系统及内核版本信息：

[root@lnmp ~]# cat /etc/redhat-release

CentOS release 6.7 (Final)

[root@lnmp ~]# uname -r

2.6.32-573.el6.x86\_64

NFS软件列表：

要部署NFS服务，需要安装下面的软件包：

NFS-utils：这个服务主程序

包括rpc.nfsd、rpc.mountd两个daemons和相关文档说明及执行命令文件等。

Rpcbind：centos6.X下面RPC的主程序（centos5.X下面为portmap）

首先查看系统中是否有NFS安装程序：

[root@lamp ~]# rpm -qa nfs-utils Rpcbind

如果没有的话安装一下

[root@lamp ~]# yum install nfs-utils rpcbind –y

此时NFS服务已安装完成

[root@lnmp ~]# rpm -qa nfs-utils rpcbind

nfs-utils-1.2.3-64.el6.x86\_64

rpcbind-0.2.0-11.el6.x86\_64

出现上述两个软件包，表示NFS服务端软件安装完毕。

**提示：**

**知识拓展：NFS软件的3种安装方法：**

检查：rpm –qa nfs-utils rpcbind 最佳

1）方法1：yum install nfs-utils prcbind -y

2) 方法2：通过系统光盘里的rpm包安装，命令如：rpm –ivh nfs-utils-1.2.3-36.el6.x86\_64.rpm

3) 方法3：LANG=en

Yum grouplist|grep –I nfs

Yum groupinstall “NFS file server” -y

**实战开始：**

**首先在服务端开启rpcbind服务**

[root@NFS ~]# /etc/init.d/rpcbind status

rpcbind 已停

[root@NFS ~]# /etc/init.d/rpcbind start

正在启动 rpcbind： [确定]

[root@NFS ~]# lsof -i :111 ###rpcbind主端口为111

COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME

rpcbind 1622 rpc 6u IPv4 11343 0t0 UDP \*:sunrpc

rpcbind 1622 rpc 8u IPv4 11346 0t0 TCP \*:sunrpc (LISTEN)

rpcbind 1622 rpc 9u IPv6 11348 0t0 UDP \*:sunrpc

rpcbind 1622 rpc 11u IPv6 11351 0t0 TCP \*:sunrpc (LISTEN)

[root@NFS ~]# netstat -lntup|grep rpcbind

tcp 0 0 0.0.0.0:111 0.0.0.0:\* LISTEN 1622/rpcbind

tcp 0 0 :::111 :::\* LISTEN 1622/rpcbind

udp 0 0 0.0.0.0:949 0.0.0.0:\* 1622/rpcbind

udp 0 0 0.0.0.0:111 0.0.0.0:\* 1622/rpcbind

udp 0 0 :::949 :::\* 1622/rpcbind

udp 0 0 :::111 :::\* 1622/rpcbind

[root@NFS ~]# chkconfig --list rpcbind ####检查rpcbind是否设置为开机自启动

rpcbind 0:关闭 1:关闭 2:启用 3:启用 4:启用 5:启用 6:关闭

**查看NFS服务向rpc服务注册的端口信息，因为NFS没起，因此，没太多注册的端口映射信息，请注意和后面启动NFS服务后这里的对比。**

[root@NFS ~]# rpcinfo -p localhost

program vers proto port service

100000 4 tcp 111 portmapper

100000 3 tcp 111 portmapper

100000 2 tcp 111 portmapper

100000 4 udp 111 portmapper

100000 3 udp 111 portmapper

100000 2 udp 111 portmapper

**开启NFS服务，NFS的主端口为2049**

[root@NFS ~]# /etc/init.d/nfs start

启动 NFS 服务： [确定]

关掉 NFS 配额： [确定]

启动 NFS mountd： [确定]

启动 NFS 守护进程： [确定]

正在启动 RPC idmapd： [确定]

[root@NFS ~]# netstat -lntup|grep 2049

tcp 0 0 0.0.0.0:2049 0.0.0.0:\* LISTEN -

tcp 0 0 :::2049 :::\* LISTEN -

udp 0 0 0.0.0.0:2049 0.0.0.0:\* -

udp 0 0 :::2049 :::\*

[root@NFS ~]# rpcinfo -p localhost

program vers proto port service

100000 4 tcp 111 portmapper

100000 3 tcp 111 portmapper

100000 2 tcp 111 portmapper

100000 4 udp 111 portmapper

100000 3 udp 111 portmapper

100000 2 udp 111 portmapper

100024 1 udp 35270 status

100024 1 tcp 10603 status

100011 1 udp 875 rquotad

100011 2 udp 875 rquotad

100011 1 tcp 875 rquotad

100011 2 tcp 875 rquotad

100005 1 udp 36811 mountd

100005 1 tcp 18897 mountd

100005 2 udp 30933 mountd

100005 2 tcp 55190 mountd

100005 3 udp 4205 mountd

100005 3 tcp 37620 mountd

100003 2 tcp 2049 nfs

100003 3 tcp 2049 nfs

100003 4 tcp 2049 nfs

100227 2 tcp 2049 nfs\_acl

100227 3 tcp 2049 nfs\_acl

100003 2 udp 2049 nfs

100003 3 udp 2049 nfs

100003 4 udp 2049 nfs

100227 2 udp 2049 nfs\_acl

100227 3 udp 2049 nfs\_acl

100021 1 udp 5982 nlockmgr

100021 3 udp 5982 nlockmgr

100021 4 udp 5982 nlockmgr

100021 1 tcp 34935 nlockmgr

100021 3 tcp 34935 nlockmgr

100021 4 tcp 34935 nlockmgr

**重启rpcbind之后，NFS都要想rpc重新注册。**

[root@NFS ~]# /etc/init.d/rpc

rpcbind rpcgssd rpcidmapd rpcsvcgssd

[root@NFS ~]# /etc/init.d/rpc

rpcbind rpcgssd rpcidmapd rpcsvcgssd

[root@NFS ~]# /etc/init.d/rpcbind restart

停止 rpcbind： [确定]

正在启动 rpcbind： [确定]

[root@NFS ~]# rpcinfo -p localhost

program vers proto port service

100000 4 tcp 111 portmapper

100000 3 tcp 111 portmapper

100000 2 tcp 111 portmapper

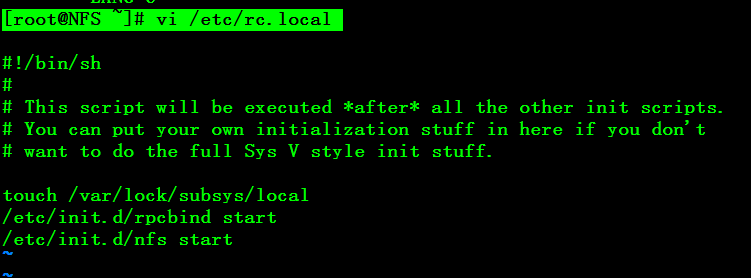
100000 4 udp 111 portmapper

100000 3 udp 111 portmapper

100000 2 udp 111 portmapper

**在开机自启动时，要让rpc比nfs先启动的方法：**

[root@NFS ~]# vi /etc/rc.local



将两个文件放到rc.local配置文件中。###注意前后顺序。

**查看NFS详细信息**

[root@NFS ~]# ps -ef|egrep "rpc|nfs"

rpcuser 1392 1 0 09:33 ? 00:00:00 rpc.statd #检查文件一致性

root 1567 2 0 09:41 ? 00:00:00 [rpciod/0]

root 1748 1 0 09:44 ? 00:00:00 rpc.rquotad #磁盘配额进程（remote quota server）

root 1753 1 0 09:44 ? 00:00:00 rpc.mountd

root 1760 2 0 09:44 ? 00:00:00 [nfsd4]

root 1761 2 0 09:44 ? 00:00:00 [nfsd4\_callbacks]

root 1762 2 0 09:44 ? 00:00:00 [nfsd] #NFS主进程

root 1763 2 0 09:44 ? 00:00:00 [nfsd] #NFS主进程

root 1764 2 0 09:44 ? 00:00:00 [nfsd]

root 1765 2 0 09:44 ? 00:00:00 [nfsd] #NFS主进程，管理登入，ID身份判别等。

root 1766 2 0 09:44 ? 00:00:00 [nfsd]

root 1767 2 0 09:44 ? 00:00:00 [nfsd]

root 1768 2 0 09:44 ? 00:00:00 [nfsd]

root 1769 2 0 09:44 ? 00:00:00 [nfsd]

root 1800 1 0 09:44 ? 00:00:00 rpc.idmapd #名称映射

rpc 1825 1 0 09:44 ? 00:00:00 rpcbind

root 1862 1501 0 09:57 pts/0 00:00:00 grep -E rpc|nfs

**实战配置NFS服务端**

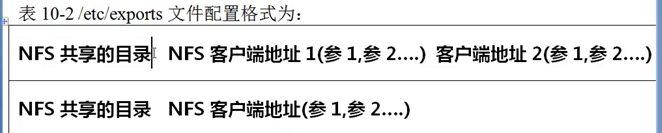
NFS服务的默认配置文件路径为：/etc/exports，并且默认是空的。

[root@NFS ~]# ls -l /etc/exports

-rw-r--r--. 1 root root 0 1月 12 2010 /etc/exports

[root@NFS ~]# cat /etc/exports

[root@NFS ~]#



**使用的例子：**



第一列为要共享的目录

第二列为客户端的地址（如果是一个IP就写一个IP，若一个网段，就写一个网段。“10.0.0.0/24”；还可以使用主机名。）

**实例一：共享/data目录给10.0.0.0/24整个网段的主机读写即实现将nfs server上的/data目录共享给10.0.0./24整个网段的主机可读写。**

如果想让/data变为共享的目录，使用：

[root@NFS data]# vi /etc/exports #/data目录必须为存在的

/data 10.0.0.\*(rw,sync) #目录、IP段、给读写权限、可以共享

[root@NFS data]# cat /etc/exports

/data 10.0.0.\*(rw,sync)

[root@NFS data]# /etc/init.d/nfs reload #平滑生效

（reloade==exportfs –rv）

[root@NFS data]# exportfs -rv

exporting 10.0.0.\*:/data

[root@NFS data]# showmount -e 127.0.0.1 #查看本机共享的记录

clnt\_create: RPC: Program not registered #当出现此报错时，是rpc与nfs前后重启顺序颠倒导致

[root@NFS data]# rpcinfo -p localhost

program vers proto port service

100000 4 tcp 111 portmapper

100000 3 tcp 111 portmapper

100000 2 tcp 111 portmapper

100000 4 udp 111 portmapper

100000 3 udp 111 portmapper

100000 2 udp 111 portmapper

[root@NFS data]# /etc/init.d/nfs restart

关闭 NFS 守护进程： [确定]

关闭 NFS mountd： [确定]

关闭 NFS quotas： [确定]

关闭 NFS 服务： [确定]

Shutting down RPC idmapd: [确定]

启动 NFS 服务： [确定]

关掉 NFS 配额： [确定]

启动 NFS mountd： [确定]

启动 NFS 守护进程： [确定]

正在启动 RPC idmapd： [确定]

[root@NFS data]# rpcinfo -p localhost

program vers proto port service

100000 4 tcp 111 portmapper

100000 3 tcp 111 portmapper

100000 2 tcp 111 portmapper

100000 4 udp 111 portmapper

100000 3 udp 111 portmapper

100000 2 udp 111 portmapper

100011 1 udp 875 rquotad

100011 2 udp 875 rquotad

100011 1 tcp 875 rquotad

100011 2 tcp 875 rquotad

100005 1 udp 23538 mountd

100005 1 tcp 58110 mountd

100005 2 udp 29894 mountd

[root@NFS data]# showmount -e 127.0.0.1

Export list for 127.0.0.1:

/data 10.0.0.\*

**检查完毕之后，开始挂载共享目录**

[root@NFS data]# mount -t nfs 10.0.0.31:/data /mnt

[root@NFS data]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/sda3 6.9G 1.5G 5.1G 22% /

tmpfs 118M 0 118M 0% /dev/shm

/dev/sda1 190M 36M 145M 20% /boot

10.0.0.31:/data 6.9G 1.5G 5.1G 22% /mnt

测试挂载是否成功

[root@NFS data]# touch /data/oldboy.txt

[root@NFS data]# ls /mnt/

oldboy.txt

[root@NFS data]# touch /mnt/test.txt #为什么不可向目录里写文件呢？

touch: 无法创建"/mnt/test.txt": 权限不够

[root@NFS data]# ll -d /mnt/

drwxr-xr-x 2 root root 4096 11月 9 10:31 /mnt/

####此时，切换到客户端服务器开始进行测试#####

客户端服务器需要开启rpcbind服务。

[root@lamp ~]# rpm -qa nfs-utils rpcbind

nfs-utils-1.2.3-64.el6.x86\_64

rpcbind-0.2.0-11.el6.x86\_64

[root@lamp ~]# /etc/init.d/rpcbind restart

停止 rpcbind： [确定]

正在启动 rpcbind： [确定]

[root@lamp ~]# LANG=en

[root@lamp ~]# chkconfig rpcbind on #在客户端将rpc设为开机自启动

[root@lamp ~]# vi /etc/rc.local

/etc/init.d/rpcbind start

[root@lamp ~]# /etc/init.d/rpcbind status

rpcbind (pid 1654) is running...

[root@lamp ~]# showmount -e 10.0.0.31 #检查是否已连接

Export list for 10.0.0.31: #如果连接不上的话，多是被防火墙挡住了。

/data 10.0.0.\*

[root@lamp ~]# mount -t nfs 10.0.0.31:/data /mnt 开始进行挂载

[root@lamp ~]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/sda3 6.9G 1.5G 5.1G 22% /

tmpfs 118M 0 118M 0% /dev/shm

/dev/sda1 190M 36M 145M 20% /boot

10.0.0.31:/data 6.9G 1.5G 5.1G 22% /mnt

[root@lamp ~]# mount #查看挂载的结果

10.0.0.31:/data on /mnt type nfs (rw,vers=4,addr=10.0.0.31,clientaddr=10.0.0.7)

[root@lamp ~]# cat /proc/mounts #另一种查询挂载的结果命令

10.0.0.31:/data/ /mnt nfs4 rw,relatime,vers=4,rsize=32768,wsize=32768,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=10.0.0.7,minorversion=0,local\_lock=none,addr=10.0.0.31 0 0

此时切换到/mnt目录下，查询中发现有服务端创建的文件，在服务端删除该文件之后，客户端查询结果为空。

[root@lamp ~]# cd /mnt/

[root@lamp mnt]# ls -l /mnt/

total 0

-rw-r--r-- 1 root root 0 Nov 9 10:31 oldboy.txt

[root@lamp mnt]# ls -l /mnt/

total 0

[root@lamp mnt]# touch test.txt #在客户端创建文件时，发现无法创建。

touch: cannot touch `test.txt': Permission denied

######系统默认将文件权限赋予65534这个用户，所以无法修改目录信息。######

注意此时已切换到NFS服务器上。

[root@NFS data]# cat /var/lib/nfs/etab ###NFS服务器端记录的所有参数。

/data 10.0.0.\*(rw,sync,wdelay,hide,nocrossmnt,secure,root\_squash,no\_all\_squash,no\_subtree\_check,secure\_locks,acl,anonuid=65534,anongid=65534,sec=sys,rw,root\_squash,no\_all\_squash)

[root@NFS data]# cat /etc/exports

/data 10.0.0.\*(rw,sync)

[root@NFS data]# grep 65534 /etc/passwd ###查询该用户的名称

nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin

[root@NFS data]# chown -R nfsnobody /data ###将目录属主修改为nfsnobody

[root@NFS data]# ll -d /data/ ####进行检查

drwxr-xr-x 2 nfsnobody root 4096 11月 9 10:47 /data/

###此时将服务器切换到客户服务器。

[root@lamp mnt]# touch test.txt #即可创建文件

[root@lamp mnt]# ls

test.txt

[root@lamp mnt]# ll

total 0

-rw-r--r-- 1 root root 0 Nov 9 10:58 lifen.txt #此文件是在服务端创建的

-rw-r--r-- 1 nfsnobody nfsnobody 0 Nov 9 10:56 test.txt #此文件是在客户端创建的

mount -t nfs 10.0.0.31:/data /mnt 将此条命令放到客户端rc.local配置文件中。

此条命令不可存放到/etc/fstab文件中，因为NFS是依靠网络的，而fstab启动时，网卡还未启动。

#####切换到另外一台服务器之后，将rpcbind服务开启，并放入到开机自启动文件中，进行测试。####

[root@lnmp ~]# chkconfig --list rpcbind

rpcbind 0:off 1:off 2:on 3:on 4:on 5:on 6:off

[root@lnmp ~]# vi /etc/rc.local

/etc/init.d/rpcbind start

mount -t nfs 10.0.0.31:/data /mnt

[root@lnmp ~]# mount -t nfs 10.0.0.31:/data /mnt

[root@lnmp ~]# cd /mnt/

[root@lnmp mnt]# ls

lifen.txt test.txt

[root@lnmp mnt]# ll

total 0

-rw-r--r-- 1 root root 0 Nov 9 10:58 lifen.txt

-rw-r--r-- 1 nfsnobody nfsnobody 0 Nov 9 10:56 test.txt

[root@lnmp mnt]# rm -f test.txt ###客户端可以删除服务端与另一台客户端创建的文件。

[root@lnmp mnt]# ll

total 0

-rw-r--r-- 1 root root 0 Nov 9 10:58 lifen.txt

[root@lnmp mnt]# rm -f lifen.txt

[root@lnmp mnt]# ll

total 0

**NFS客户端挂载排错思路：**

1. 首先确认NFS服务端配置和服务是OK的。

[root@NFS data]# showmount -e localhost

Export list for localhost:

/data 10.0.0.\*

最好服务端自己挂自己看看可以不可以。

[root@NFS data]# mount -t nfs 10.0.0.31:/data /mnt

[root@NFS data]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/sda3 6.9G 1.5G 5.1G 22% /

tmpfs 118M 0 118M 0% /dev/shm

/dev/sda1 190M 36M 145M 20% /boot

10.0.0.31:/data 6.9G 1.5G 5.1G 22% /mnt

###这步主要检查服务端的NFS服务是否是OK的。

1. 确认NFS客户端showmount是否OK

[root@lamp mnt]# showmount -e 10.0.0.31 正常情况。

Export list for 10.0.0.31:

/data 10.0.0.\*

两个报错：

[root@lamp mnt]# showmount -e 10.0.0.31

Clnt\_create: RPC: Port mapper failure – Unable to receive:errno 113 (No route to host)

[root@lamp mnt]# showmount -e 10.0.0.31

Clnt\_create: RPC: Program not registered

如果连接不到的话，一般根据提示解决，总体通用思路为：

1.ping以下NFS服务端IP ###看物理链路是否通。

2.telnet服务端IP端口检查 ###测试NFS服务是否通。

[root@lamp ~]# netstat -an|grep -i est ##查看是否有连接

Active Internet connections (servers and established)

tcp 0 52 10.0.0.7:52113 10.0.0.1:52693 ESTABLISHED

tcp 0 0 10.0.0.7:751 10.0.0.31:2049 ESTABLISHED

Active UNIX domain sockets (servers and established)

**NFS服务端配置过程：**

1、安装软件

Yum install nfs-utils rpcbind –y

2、启动服务（注意先后顺序）

/etc/init.d/rpcbind start

rpcinfo –p localhost

/etc/init.d/nfs start

rpcinfo –p localhost

3、设置开机自启动

Chkconfig nfs on

Chkconfig rpcbind on

4、配置nfs服务

Echo “/data 10.0.0.0/24（rw,sync）” >>/etc/exports

Mkdir –p /data

Chown –R nfsnobody.nfsnodboy /data

（查看NFS默认使用的用户以及共享的参数cat /var/lib/nfs/etab）

5、重新加载服务（优雅重启）

/etc/init.d/nfs reload ====exportfs –rv

6、检查或测试挂载

Showmount –e localhost

Mount –t nfs 127.0.0.1:/data /mnt

**NFS客户端配置过程：**

1、安装软件

yum install nfs-utils rpcbind –y

2、启动rpcbind

/etc/init.d/rpcbind start

3、配置开机自启动

Chkconfig rpcbind on

4、测试服务端共享情况

chowmount –e server\_ip（yum install nfs-utils -y）

5、挂载

mkdir –p /data

moun –t nfs server\_ip:/data /data

6、测试读、写

7、tail –l /etc/rc.local

/etc/init.d/rpcbind start

/bin/mount –t 10.0.0.31:/data /mnt

**NFS排错：**

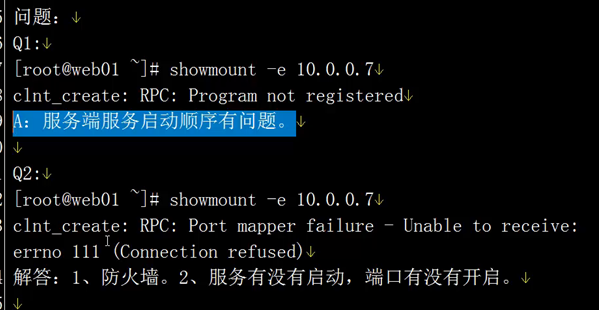
1、前提：NFS原理以及部署的步骤很熟练。

2、先在客户端排查

ping server\_ip

telnet server\_ip 111

showmount –e server\_ip



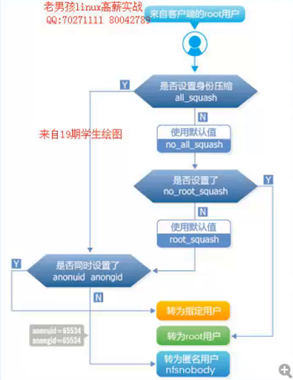
学习思想：

了解步骤的前提下，故意搞错一个地方，然后，观察提示。反推问题。

**NFS配置文件权限参数说明：**

NFS服务器端的权限设置，即/etc/exports文件配置格式中小括号里的参数。

|  |  |
| --- | --- |
| **参数名称** | **参数用途** |
| **rw** | Read-write，表示可读写权限 |
| **ro** | Read-only，表示只读权限 |
| **sync** | 请求或写入数据时，数据同步写入到NFS Server的硬盘后才返回。（数据安全不会丢，性能比不启用该参数要差） |
| **async** | 请求或写入数据时，先返回请求，再将数据写入到内存缓存和硬盘中，即异步写入数据。此参数可以提升NFS性能，但是会降低数据的安全。因此一般情况下建议不用，如果NFS处于瓶颈状态，并且允许数据丢失的话可以代开此参数提升NFS性能。  写入时数据会先写到内存缓冲区，等硬盘有空档再写入磁盘，这样可以提升写入效率！风险为若服务器宕机或不正常关机，会损失缓冲区中未写入磁盘的数据（解决方法：服务器主板电池或加UPS）  提示：同步和异步，电商秒杀是异步。 |
| **no\_root\_squash** | 访问NFS Server共享目录的用户如果是root的话，它对该共享目录具有root权限。这个配置原本为无盘客户端准备的。用户应避免使用！  Turn off root squashing. This option is mainly useful for diskless clients. |
| **root\_squash** | 对于访问NFS Server共享目录的用户如果是root的话，则它的权限将被压缩为匿名用户，同时它的UID和GID通常会变成nfsnobody账号身份  Map requests from uid/gid 0 to the anonymous uid/gid. |
| **all\_squash** | 不管访问NFS Server共享目录的用户身份如何，它的权限都将被压缩成匿名用户，同时它的UID和GID都会变成nfsnobody账号身份，在早期多个NFS客户端同时读写NFS Server数据时，这个参数很有用。  配NFS生产重要技巧：  1．确保所有客户端服务器对NFS共享目录具备相同的用户访问权限。  a. all\_squash把所有客户端都压缩成固定的匿名用户（UID相同）。  b. 就是anonuid，anongid指定的UID和GID的用户。  2.所有的客户端和服务端都需要有一个相同的UID和GID的用户，即nfsnobody（UID必须相同）。 |
| **anonuid** | 参数以anon\*开头即指anonymous匿名用户，这个用户的UID设置值通常为NFSnobody的UID值，当然也可自行设置UID值。但是，UID必须存在于/etc/passwd中，在多NFS Clients时，如多台web server共享一个NFS 目录时，通过这个参数可以使得不同的NFS Clients写入的数据对所有NFS Clients保持同样的用户权限，即为配置的匿名UID对应用户的权限，这个参数很有用。 |
| **anongid** | 通anonuid，区别就是把uid（用户id）换成gid（组id） |
| **secure** | 不允许Client使用大于1024的Client端port，也就是从server传递资料到client端的目的port要小于1024，此时client端一定要使用root账号才能mount远端NFS Server。通常会见识使用insecure。 |
| **insecure** | 允许client端自行决定自己机器使用的port，通常都会设这个，如此非使用root账号的client端才能mount NFS server |
| **nohide** | 当export出两个目录，而其中一个目录是另外一个目录的子目录，例如我们使用虚拟目录的例子，此时我们mount根目录时，会自动把所有子目录mount起来。 |



**NFS服务重点知识点梳理总结：**

当多个NFS客户端以NFS方式写入修改服务器的文件系统时，需要具有以下权限：

1. NFS服务器/etc/exports设置需要开放可写入的权限，即服务端的共享权限。
2. NFS服务器实际要共享的NFS目录权限具有可写入w的权限，即服务端本地目录的安全权限。
3. 每台机器都对应存在和nfs默认配置UID的相同UID 65534的nfsnobody用户（确保所有客户端的访问权限统一，否则每个机器需要同时建立相同UID的用户，并覆盖NFS的默认配置）。

当满足上述三个条件，多个NFS客户端才能具有互相写入，互相修改其他主机写入文件的权限，这在开篇讲解过的答规模集群环境，作为集群共享存储时尤为注意。

到这步为止，NFS服务端一个NFS共享目录/data共享给10.0.0.0/24内主机可读写就配置完了。下面还需要在客户端主机挂载服务端共享的/data，才可以在客户端真正读写到该共享目录。

**NFS服务重要文件说明**

|  |  |
| --- | --- |
| **NFS常用路径** | **说明** |
| **/etc/exports** | NFS服务主配置文件，配置NFS具体共享服务的地点，默认内容为空。以行为单位。  [root@NFS mnt]# cat /etc/exports  /data 10.0.0.\*(rw,sync) |
| **/user/sbin/exportfs** | NFS服务的管理命令，例如：可以加载NFS配置生效，还可以直接配置NFS共享目录，即无需配置/etc/exports实现共享。  [root@NFS mnt]# exportfs -rv  加载配置生效，等价优雅重启/etc/init.d/nfs reload  Exporting 10.0.0.0/24:/data  这里讲了一个服务平滑重启的概念，超市，银行到时间了如何提供服务，拒绝新来的，已有的顾客继续提供服务，网站平滑重启，提升用户体验是必须要考虑的。  Maintain table of exported NFS file systems  Exportfs 不但可以加载配置生效，也可以通过命令直接共享目录，越过/etc/exports，但是重启失效 |
| **/usr/sbin/showmount** | 常用来在客户端，查看NFS配置及挂载结果的命令。  Show mount information for an NFS server  配置nfs server，分别在服务端以及客户端查看挂载情况。 |
| **/var/lib/nfs/etab** | NFS配置文件的完整参数设定的文件（有很多没有配置但是默认就有的NFS参数） |
| **/var/lib/nfs/xtab** | 适合C5.x记录曾经挂载过的NFS客户端的信息，包括IP地址等，Centos6没有此文件了。 |
| **/proc/mounts** | 客户端挂载参数：  [root@lamp mnt]# grep mnt /proc/mounts  10.0.0.31:/data/ /mnt nfs4 rw,relatime,vers=4,rsize=32768,wsize=32768,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=10.0.0.7,minorversion=0,local\_lock=none,addr=10.0.0.31 0 0 |
| **/var/lib/ns/rmtab** | Table clients accessing server’s exports |

**NFS客户端挂载命令**

|  |  |  |  |
| --- | --- | --- | --- |
| 挂载命令 | 挂载的格式类型 | NFS服务器提供的共享目录 | NFS客户端要挂载的目录 |
| mount | -t nfs | 10.0.0.31:/data | /mnt（必须存在） |
| 完整挂载命令为：mount –t nfs 10.0.0.31:/data /mnt，此命令要在客户端执行。 | | | |

**NFS客户端mount深入挂载**

NFS客户端mount挂载参数说明：

在NFS服务端可以通过cat/var/lib/nfs/etab查看NFS服务端配置的参数细节。

在NFS客户端可以通过cat /proc/mounts查看mount的挂载参数细节。

1、mount挂载及fstab文件的参数表格

通过在NFS客户端测试挂载获取的默认挂载参数：

[root@lamp mnt]# grep mnt /proc/mounts

10.0.0.31:/data/ /mnt nfs4 rw,relatime,vers=4,rsize=32768,wsize=32768,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=10.0.0.7,minorversion=0,local\_lock=none,addr=10.0.0.31 0 0

**NFS client mount 挂载参数列表**

|  |  |  |
| --- | --- | --- |
| **参数** | **参数功能** | **默认参数** |
| fg  bg | 当在客户端执行挂载时，可选择是前台（fg）还是在后台（bg）执行。若在前台执行，则mount会持续尝试挂载，直到成功或挂载时间超时为止，若在后台执行，则mount会在后台持续多次进行mount，而不会影响到前台的其他的程序操作。如果网络联机不稳定，或是服务器常常需要开关机，建议使用bg比较妥当。 | fg |
| Soft | 当NFS client以soft挂载server后，若网络或server出现问题，造成client和server无法传输资料时，client会一直尝试到timeout后显示错误并且停止尝试。若使用soft mount的话，可能会在timeout出现时造成资料丢失，故一般不建议使用。  若用hard模式挂载硬盘时，刚好和soft相反，此时client会一直尝试连接到server，若server有回应就继续刚才的操作，若没有回应NFS client会一直尝试，此时无法umount或kill，所以常常会配合intr使用。这是默认值 | hard |
| Intr | 当使用hard挂载的自由timeout后，若有指定intr参数，可以在timeout后把它中断掉，这避免出问题时系统整个被NFS锁死，建议使用intr。 | 无 |
| rsize  wsize | 读出（rsize）与写入（wsize）的区块大小（block size），这个设置值可以影响客户端与服务器端传输数据的缓冲存储量，一般来说，如果在局域网内（LAN），并且客户端于服务器端都具有足够的内存，合格值可以设置大一点，比如说65535（bytes），提升缓冲区块将可提升NFS文件系统的传输能力。但设置的值也不要太大，最好是实现网络能够传输的最大值为限。 | C5：默认值  Rsize=1024  Wsize=1024  C6:默认值  R:131072  W:131072 |
| Proto=udp | 使用UDP谢顶来传输资料，在LAN中会有比较好的性能。若要跨越Internet的话，使用proto=tcp多传输的数据会有比较好的纠错能力。 | Proto=tcp |

**如果追求完美的话，可以使用如下参数挂载：**

**mount –t nfs –o bg,hard,intr,rsize=131072,wsize=131072 10.0.0.31:/data /mnt**

**但如果考虑简单、易用原则，可以使用如下参数挂载：**

**下面是mount –o 参数对应的选项列表**

|  |  |  |
| --- | --- | --- |
| **参数** | **参数意义** | **系统默认值** |
| Suid  Nosuid | 当挂载的文件系统上有任何SUID的程序时，只要使用nosuid就能够取消设置SUID的功能。 | SUID |
| Rw  ro | 可以指定文件系统是只读（ro）  或是可写（rw） | Rw |
| Dev  nodev | 是否可以保留装置文件的特殊功能？一般来说只要/dev才会有特殊的装置，因此可以选择nodev | Dev |
| Exec  noexec | 是否具有执行文件的权限？如果想要挂载的仅是普通资源数据区（例如：图片、附件），那么可以选择noexec | Exec |
| User  nouser | 是否允许用户进行文件的挂载与卸载功能？如果要保护文件系统，最好不要提供用户进行挂载与卸载 | Nouser |
| Auto  Noauto | 这个auto指的是“mount -a”时会不会被挂载的项目，如果不需要这个分区随时被挂载，可以设置为noauto | auto |

1、async所涉及到文件系统IO的操作都是异步处理，既不会同步写到磁盘，此参数会提高性能，但会降低数据安全，一般情况不适用。

2、sync同步处理

3、defaults这是个缺省值包括：rw,suid,exec,dev,auto,nouser,async

4、noatime不更新文件系统上的inode访问时间，高并发环境，推荐使用该选项，可提高IO性能。

5、nodiratime不更新文件系统上的directory iNode访问时间，高并发环境，推荐显示应用该选项，可以提高IO性能。

6、remount尝试重新挂载一个已经挂载了的文件系统，这通常被用来改变一个文件系统的挂载标志，从而使得一个只读的文件系统变为可写，这个动作不会改变设备或者挂载点，当系统故障进入单用户模式或者救援模式修复时，rw/将根文件系统重新挂载使得可写。

**NFS优化**

1、安全的挂载：

mount –t nfs –o nosuid,noexec,nodev,rw 10.0.0.31:/data /mnt

通mount –o指定挂载参数和在/etc/fstab里指定挂载参数效果是一样的。网络文件系统和本地的文件系统效果也是一样的。

2、mount挂载性能优化参数选项

企业生产环境nfs性能优化挂载的例子：

（1）禁止更新目录及文件时间戳挂载

mount –t nfs –o noatime,nodiratime 10.0.0.31:/data

（2）安全加优化的挂载方式

mount –t –o nosuid,noexec,nodev,noatime,nodiratime,intr,rsize=131072,wsize=131072 10.0.0.31:/data /mnt

（3）默认的挂载方式

mount –t nfs 10.0.0.31:/data /mnt

**NFS网络文件系统优化挂载的参数建议：**

1）centos5.8x86\_64服务器端和客户端环境

mount –t nfs –o noatime,nodiratime,nosuid,noexec,nodev,ew,bg,soft,rsize=32768,wsize=32768 10.0.0.31:/data /mnt

2）centos6.6x86\_64服务器端和客户端环境

mount –t nfs –o noatime,nodiratime,nosuid,noexec,nodev,ew,bg,hard,intr rsize=131072,wsize=131072 10.0.0.31:/data /mnt

mount –t nfs 10.0.0.31: /data /mnt 注意：非性能的参数越多，速度可能会越慢。

**NFS服务内核优化相关建议：**

**优化选项说明：**



cat >>/etc/sysctl.conf<<EOF  
net.core.wmem\_default = 8388608  
net.core.rmem\_default = 8388608  
net.core.rmem\_max = 16777216  
net.core.wmem\_max = 16777216  
EOF  
sysctl -p

**企业生产场景NFS共享存储优化小结：**

1、硬件：sas/ssd磁盘，买多块，raid0/raid10.网卡吞吐量要大，至少千兆（多块bond）。

2、NFS服务器端配置：/data

10.0.0.0/24（rw,sync,all\_squash,anonuid=65534,anongid=65534）

3、NFS客户端挂载：rsize,wsize,noatime,nodiratime,nosuid,noexec,soft（hard，intr）

mount –t nfs –o nosuid,noexec,nodev,noatime,nodiratime,rsize=131072,wsize=131072 10.0.0.31:/data /mnt

mount –t nfs –o noatime,nodiratime,rsize=131072,wsize=131072 10.0.0.31:/data /mnt

mount –t nfs –o noatime,nodiratime 10.0.0.31:/data /mnt

4、有关NFS服务的所有服务器内核优化：

cat >>/etc/sysctl.conf <<EOF

net.core.wmen\_default=8388608

net.core.rmem\_default=8388608

net.core.mem\_max=16777216

net.core.wmem\_max=16777216

EOF

执行sysctl –p 生效

5、如果卸载的时候提示：umount:/mnt:device is busy

需要退出挂载目录在进行卸载，或者是NFS server宕机了，需要强制卸载

mount –lf /mnt

6、大型网站NFS网络文件系统替代软件：分布式文件系统moosefs（mfs），glusterfs，fastDFS

**1.5 NFS系统应用优点缺点说明**

作用：

NFS服务可以让不同的客户端挂载使用同一个目录，作为共享存储使用，这样可以保证不同节点客户端数据的一致性，在集群架构环境中经常会用到。如果Windows+Linux可以用Samba。

**优点：**

1、简单🡪容易上手，容易掌握。

2、NFS文件系统内数据是在文件系统之上的，即数据能看得见的。

3、方便🡪部署快速，维护简单，可控且满足需求就是最好的。

4、可靠🡪从软件层面上看，数据可靠性高，经久耐用。数据是在文件系统之上的。

5、稳定🡪非常稳定。

**局限：**

1、局限性是存在单点故障，如果nfs server宕机了所有客户端都不能访问共享目录，这可通过负载均衡及高可用方案弥补。

2、在大数据高并发的场合，NFS效率\性能有限（一般几千万以下PV的网站不是瓶颈，除非网站架构太差，2千万PV/日）。

3、客户端认证是基于ip和主机名的，权限时根据ID识别，安全性一般（用于内网则问题不大）。

4、NFS数据是明文的，NFS本身对数据完整性不作验证。

5、多台客户机器挂载一个NFS服务器时，链接管理维护麻烦（耦合度高）。尤其NFS服务端出问题后，所有NFS客户端后，所有NFS客户端都挂掉状态（测试环境可使用autofs自动挂载解决）。

这里涉及到了同步（实时等待）和异步（解耦）概念知识 ，NFS服务端和客户端相对来说就是耦合度有些高。网站程序也是一样，尽量不要耦合度太高，系统及程序架构师的重要职责就是为程序及架构解耦，让网站的扩展性变得更好。

6、大中小型网站（2000万PV以下）线上应用，都有用武之地，门户站也会有应用。

**应用建议：**

大中小型网站（参考点2000万/日PV以下），线上应用，都有用武之地，门户网站也会有应用，生产场景应该多把数据的访问往前推，即尽量把静态存储里的资源通过CDN或缓存服务器提供服务，如果没有缓存服务或架构不好，存储服务器数量在多也是抗不住压力的，而且用户体验会很差。

**临时提供共享目录：**

exportfs -o rw,sync 172.16.1.0/24:/data相当于/etc/exports但是只是临时生效。

**NFS 章节重点**

1、NFS 服务的访问原理流程（会口述）\*\*\*\*\*

2、NFS 作为集群共享存储角色的搭建、部署

3、NFS 作为集群共享存储角色的排障，高级优化（会口述）\*\*\*\*\*

4、mount 命令的知识及参数。-o（noatime，nodirtime，noexec，nosuid，rsize，wsize）等

5、fstab 文件的知识

6、常用命令 showmount，exportfs，umount（-lf），rpcinfo

7、NFS 的优点、缺点，适合的应用场景。\*\*\*\*\*

8、替代产品 { Moosefs（mfs）※，glusterfs※，FastDF }\*\*\*\*\*

9、了解 autofs。

  defaults  
Use  default  options:  rw,  suid,  dev,  exec,  auto,nouser, async, and relatime.

# 2、Linux系统SSH服务

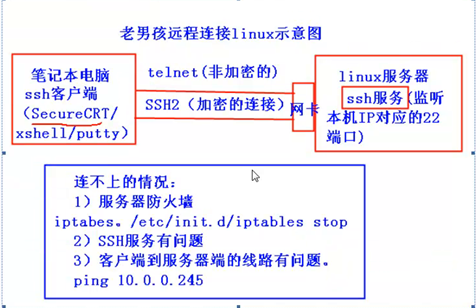
**2.1 SSH介绍**

SSH是secure shell protocol的简写，由IETF网络工作小组（network working group）指定：在进行数据传输之前，SSH先对联机数据包通过加密技术进行加密处理，加密后在进行数据传输。确保了传递的数据安全。

SSH是专为远程登录会话和其他网络服务提供的安全性协议。利用SSH协议可以有效的防止远程管理过程中的信息泄露问题，在当前的生产环境运维工作中，绝大多数企业普遍采用SSH协议服务来代替传统的不安全的远程联机服务软件，如Telnet（23端口，非加密的）等。

在默认状态下，SSH服务主要提供两个服务功能：一个是提供类似Telnet远程连接服务器的服务，即上面提到的SSH服务；另一个是类似FTP服务的sftp-server，借助SSH协议来传输数据的，提供更安全的SFTP服务（vsftp，proftp）。

特别提醒：SSH客户端（ssh命令）还包含一个很有用的远程安全拷贝命令scp，也是通过ssh协议工作的。



知识小结：

1、SSH是安全的加密协议，用于远程连接Linux服务器。

2、SSH默认端口是22，安全协议版本SSH2，除了2之外还有SSH1（漏洞）。

3、SSH服务端主要包含了两个服务功能SSH远程连接，SFTP服务。

4、Linux SSH客户端包含ssh远程连接命令，以及远程拷贝scp命令等。

**2.2 SSH结构**

SSH服务由服务端软件OpenSSH（openssl）和客户端（常见的有SSH（Linux），secureCRT，putty，xshell）组成，SSH服务默认使用22端口提供服务，它有两个不兼容的SSH协议版本，分别是1.x和2.x。

查看服务端上的ssh相关软件

[root@web02 10.0.0.7]# rpm -qa openssh

openssh-5.3p1-111.el6.x86\_64

[root@web02 10.0.0.7]# rpm -qa openssh openssl

openssh-5.3p1-111.el6.x86\_64

openssl-1.0.1e-42.el6.x86\_64

OpenSSH同时支持SSHSSH 1.x和2.x。用SSH2.x的客户端程序不能连接到SSH1.x的服务程序上。

SSH服务端是一个守护进程（demon），他在后台运行并相应来自客户端的连接请求。SSH服务端的进程名为sshd，负责实时监听远程SSH客户端的连接请求，并进行处理，一般报货公共秘钥认证、秘钥交换、对称秘钥加密卡和非安全连接等。这个SSH服务就是我们前面基础系统优化中保留开机自启动的服务之一。

SSH客户端包含ssh以及像scp（远程拷贝）、slogin（远程登录）、sftp（安全FTP文件传输）等应用程序。

SSH的工作机制大致是本地的ssh客户端先发送一个连接请求到远程的ssh服务端，服务端检查链接的客户端发送的数据包和IP地址，如果确认合法，就会发送秘钥给SSH的客户端，此时，客户端本地再将秘钥发回给服务端，自此连接建立。SSH1.x和SSH2.x在链接协议上有一些安全方面的差异。

**2.3 SSH加密技术**

简单的说，SSH加密技术就是将人类可以看得懂的数据，通过一些特殊的程序算法，把这些数据变成杂乱的无意义的信息，然后，通过网络进行传输，而当到了目的地后，在通过对应的解密算法，把传过来的加密的数据信息解密成加密前的可读的正常数据。因此，当数据在互联网上传输是即使被有心的黑客监听窃取了，也很难获取到真正需要的数据。

当前，网络上的数据包加密技术一般是通过所谓的一对公钥与私钥（public key and pivate key）组合成的秘钥对进行加密与解密操作。如下图1-1所示，A-server要给B\_client传数据，首先会通过本地的公钥加密后再发到网络上传输。而加密的数据到达B\_client端后，在经由B\_client本地的私钥将加密的数据解密出来。由于Internet上传输过程中的数据是加密过多的，所以，传输的数据内容一般来说是比较安全的。



openSSH是SSH服务端的软件之一，可同时支持SSH1和SSH2协议，可以在配置文件中使用protocol指令指定只支持其中一种或两种都自此。默认情况centos5.X系统默认配置的是仅支持SSH2协议，如下例所示：

[root@web02 10.0.0.7]# less /etc/ssh/sshd\_config

# activation of protocol 1

Protocol 2

另外，SSH2同时支持RSA和DSA密钥，但是SSH1仅支持RSA密钥。

**1、SSH 1.x**

每台SSH服务器主机都可以使用RSA加密方式来产生一个1024-bit的RSA Key，这个RSA的加密方式就是用来产生公钥与私钥的算法之一。SSH 1.x的整个联机加密步骤如下：

a．当SSH服务启动时，就会产生一个768-bit的临时公钥（sshd\_config配置文件中serverKeyBit 768）存放在server中。

[root@web02 10.0.0.7]# grep ServerKey /etc/ssh/sshd\_config

#ServerKeyBits 1024

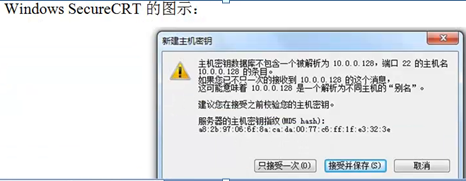
当client端SSH联机请求传送过来时，server就会将这个1024-bit的公钥传给client端，此时client会将此公钥与先前存储的公钥进行对比，看是否一致，判断标准是client端联机用户目录下~/.ssh/known\_hosts文件的内容（Linux客户端）

[root@web02 10.0.0.7]# ssh -p52113 oldboy@10.0.0.7

The authenticity of host '[10.0.0.7]:52113 ([10.0.0.7]:52113)' can't be established.

RSA key fingerprint is 3e:26:68:c2:cc:20:83:5e:22:80:f7:30:85:b9:a3:39.

Are you sure you want to continue connecting (yes/no)?



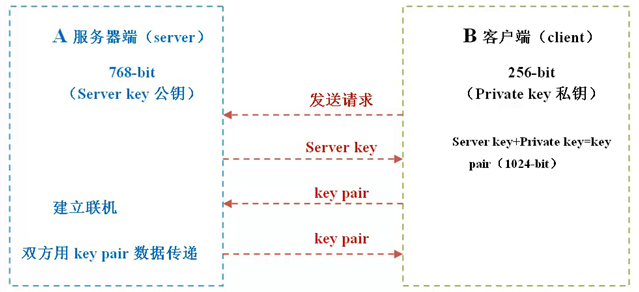
在client端接受到这个768-bit的server key后，client本地也会随机产生一个256-bit的私钥，并且以加密的方式（具体的加密算法由客户端在服务器提供的所有可用算法中选择，默认为3DES算法）将server key与host key整合成一对完整的key pair，并将这对key pair再传给server。

当客户端发送完成以后，server与client端在这次的联机中，就以一对1024-bit的key pair来进行数的传递。

也就是说，public key是放在server上的，而client端的软件需要能接受public key、计算出private key，并把二者组合成一把独一无二的key pair，因为client每次的256-bit的host key是随机产生的，所以这次与此次联机的host key可能就会不一样。此外，在client端的用户默认目录下的~/.ssh/know\_hosts会记录曾经联机过的主机的public key，用以确认每次来自该主机的联机是否正确。

当回话结束时，服务器将向客户端发送会话终止信号，双方退出。

整个连接的图如下所示：



**2、SSH 2.x**

在SSH 1.x的联机过程中，当server接受client端的private key后，就不在针对该次联机的key pair进行检验，此时若有恶意黑客针对该联机的key pair对插入恶意程序代码时，由于服务端不会在检验联机的正确性，因此可能会接收该程序代码，从而造成系统被黑掉的问题。

为了改正这个缺点，SSH version 2 多家了一个确认联机正确性的diffie-hellman机制，在每次数据传输中，server都会以该机制检查数据的来源是否正确，这样，可以避免联机过程中被插入恶意程序代码的问题。SSH version 2 是比较安全的。

由于SSH1协议本身存在较大的安全问题，因此，建议使用SSH2的联机模式。而联机版本的设置需要在SSH主机端与客户端均设置好才行。

**2.4 SSH服务认证类型**

从SSH客户端来看，SSH服务主要提供两种级别的安全验证，具体级别如下：

基于口令的安全验证：

基于口令的安全验证的方式就是一直在用的，只要知道服务器的SSH链接账号和口令（当然也需知道对应服务器的IP及开发的SSH端口，默认为22），就可以通过SSH客户端登录到远程主机。此时，联机过程中所有传输数据都是加密的。

[root@web02 10.0.0.7]# ssh -p52113 root@10.0.0.31

The authenticity of host '[10.0.0.31]:52113 ([10.0.0.31]:52113)' can't be established.

RSA key fingerprint is 3e:26:68:c2:cc:20:83:5e:22:80:f7:30:85:b9:a3:39.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[10.0.0.31]:52113' (RSA) to the list of known hosts.

root@10.0.0.31's password:

Last login: Wed Nov 11 18:55:47 2015

Windows 10.0 (final)

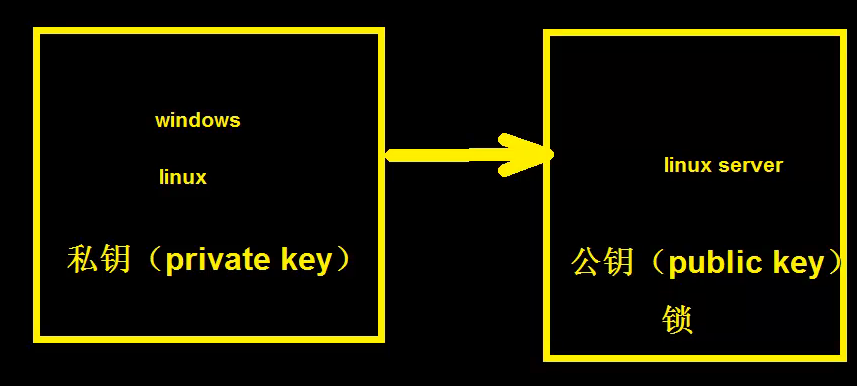
welcome to oldboy linux training.

[root@NFS ~]#

基于密钥的安全验证：

一对密钥对：包含一把钥匙和一个锁，锁（公用密钥（public key）和钥匙（私有密钥（private key））。

基于密钥的安全验证方式是指，需要依靠密钥，也就是必须事先建立一对密钥对，然后把公用密钥放在需要访问的目标服务器上，另外，还需要把私有密钥放到SSH的客户端或对应的客户端服务器上。



企业面试题：给你一个端口，如何在命令行查出对应的服务是什么？

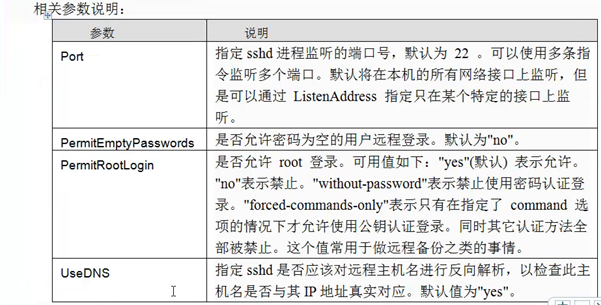
lsof –I tcp:52113

netstat –lntup|grep 52113

企业面试题：如果给你一个进程名，如何查看对应的端口是什么？

netstat –lntup|grep sshd





企业场景运维案例：利用sed实战修改多行配置技巧

<http://oldboy.blog.51cto.com/2561410/1610998>

SSH入侵网友案例：

<http://phenixikki.blog.51cto.com/7572938/1546669>

如何防止SSH登录入侵：

1、用密钥登录，不用密码登录。

2、牤牛阵法：解决SSH安全问题。

3、开启SSH只监听本地内网IP（listenaddress10.0.0.8），尽量不给服务器外网IP

远程连接：

1、连接到对端的服务器上。

2、 ssh –p 52113 [root@10.0.0.41](mailto:root@10.0.0.41)

**2.5 远程链接SSH服务**

ssh 基本语法使用：

ssh –p22 [oldboy@10.0.0.41](mailto:oldboy@10.0.0.41)

#--SSH连接远程主机命令基本语法：

#-- -p （小写）阶端口，默认22端口时可以省略-p22；

#--“@”前面为用户名，如果用当前用户连接，可以不指定用户。

#-- “@”后面为要连接的服务器的IP，更多用法，请man ssh；

#-- /root/.ssh/known\_hosts 为远程连接密钥存放处。

#-- 另外一种用法：ssh –p22 [oldboy@10.0.0.41](mailto:oldboy@10.0.0.41) ifconfig eth0

**直接登录远程主机的方法：**

在未禁止root远程登录及更改SSH端口前的登录方法为：

[root@web02 ~]# ssh -p22 root@10.0.0.31

如果端口已修改为特殊端口，那么用上面的命令连接就会发生问题：

[root@web02 ~]# ssh -p22 root@10.0.0.31

ssh: connect to host 10.0.0.31 port 22: Connection refused #提示拒绝连接

报错字符串对应的可能问题：

1、no route to host 可能为防火墙

2、connection refused 可能为防火墙：connection refused还可能是连接的对端服务没开。

**SSH小结：**

1、切换到别的机器上 ssh –p52113 [root@10.0.0.31](mailto:root@10.0.0.31)

2、到其他机器执行命令（不会切刀机器上） ssh –p52113@ip 命令 （命令要加全路径）

3、当第一次SSH连接的时候，本地会产生一个密钥文件~/.ssh/known\_hosts （多个密钥）

**2.6 ssh客户端附带的远程拷贝scp命令**

scp的基本语法使用：scp –secure copy （remote file copy program）每次都是全量拷贝

推：PUSH

scp –P22 –r –p /tmp/oldboy [oldboy@10.0.0.41:/tmp](mailto:oldboy@10.0.0.41:/tmp)

拉：PULL

scp –P22 –rp [root@10.0.0.7:/tmp/oldboy /opt/](mailto:root@10.0.0.7:/tmp/oldboy%20/opt/)

#--🡪scp为远程拷贝文件或目录的命令；

#--🡪-P（大写，与ssh目录不同）接端口，默认22端口时可省略-P22；

#--🡪-r 递归，表示拷贝目录。

#--🡪-p表示在拷贝前后保持文件或目录属性；

#--🡪-l limit限制速度。

#--🡪/tmp/oldboy 为本地目录，“@”前为用户名，“@”后为要连接的服务器的IP。IP后的:/tmp目录，为远端的目标目录。

#--🡪还可以把远端目录抓到本地，具体命令为： scp –P22 –r [oldboy@10.0.0.\*:/tmp](mailto:oldboy@10.0.0.*:/tmp) /tmp/oldboy 对调原目录和目标目录即可。

**SCP知识小结：**

1、scp是加密的远程拷贝，而cp仅为本地拷贝。

2、可以把数据从一台机器推送到另一台机器，也可以从其他服务器拉回到本地执行命令的服务器。

3、每次都是全量完整拷贝，因此，效率不高，适合第一次拷贝用，如果需要增量拷贝，用rsync。

**2.7 ssh服务附带的sftp功能服务**

**sftp小结：**

**1、Linux下链接命令：sftp –oPort=22** [**root@10.0.0.10**](mailto:root@10.0.0.10)

**2、上传put加载客户端本地路径：upload：（上传）**

put /etc/hosts 上传到远端的目录

put /etc/hosts /tmp 上传到远端的tmp目录（安全风险，需要控制，作业）

**3、下载get服务端内容：download：（下载）**

get hosts 下载到本地当前目录

get /etc/hosts /tmp 下载到本地的/tmp目录

**4、链接到的远端家目录限制到指定目录。**

[root@web02 ~]# sftp -oPort=52113 root@10.0.0.31

Connecting to 10.0.0.31...

root@10.0.0.31's password:

sftp> put /data /tmp

skipping non-regular file /data

sftp> put /etc/hosts /tmp

Uploading /etc/hosts to /tmp/hosts

/etc/hosts 100% 180 0.2KB/s 00:00

sftp> sftp> put /etc/hosts

Uploading /etc/hosts to /root/hosts

/etc/hosts 100% 180 0.2KB/s 00:00

sftp> touch ll.txt

Invalid command.

sftp> put /etc/hosts

Uploading /etc/hosts to /root/hosts

/etc/hosts 100% 180 0.2KB/s 00:00

sftp> ls

anaconda-ks.cfg data ett.txt hosts

install.log install.log.syslog log.txt oldboy

oldboy.txt oldboy.txtr

sftp>

sftp> get oldboy.txt /opt

Fetching /root/oldboy.txt to /opt/oldboy.txt

/root/oldboy.txt 100% 42 0.0KB/s 00:00

sftp>

**章节重点小结：**

1、ssh为加密的远程连接协议，相关软件有openssh、openssl。

2、默认端口22

3、协议版本1.x和2.x，2.x更安全。了解SSH协议原理。

4、服务端ssh远程连接服务，sftp服务，sshd守护进程，开机要自启动。

5、ssh客户端包含ssh、scp、sftp等。

6、ssh安全验证方式：口令和密钥，这两种都是基于口令的，SSH密钥登录的原理。

7、ssh服务安全优化，修改默认端口22，禁止root远程连接，禁止dns，SSH只监听内网IP。

8、ssh密钥对，公钥（public key）在服务器端，比喻就是锁头，私钥（private key）在客户端，比喻就是钥匙。

**2.8 ssh的企业生产应用场景**

0.1批量分发文件或数据

IT公司企业级批量分发\管理方案：

1、中小企业最基本使用的SSHKEY密钥的方案。\*\*\*\*\*

2、门户网站puppet（复制，太重）。sina cfengine

3、赶集，小米，saltstack批量管理（轻重）

**运维原则：简单、易用、高效。**

批量：

1、批量分发

2、批量部署，执行命令。

3、批量配置管理。

批量分发数据（一把钥匙开多把锁）

**2.8.1操作系统环境**

1.部署环境

操作系统：

[root@NFS ~]# cat /etc/redhat-release

CentOS release 6.7 (Final)

[root@NFS ~]# uname -r

2.6.32-573.el6.x86\_64

内核版本:

[root@NFS ~]# uname -m

x86\_64

**集体创建用户**

在很多企业的工作环境中，直接用root用户分发和管理操作，这样很不安全，很不规范，另外，如果做安全优化时禁止了root远程连接，那么使用root分发管理方法就无用了。（密钥认证也可以是不同用户）

在部署密钥前，首先要分别在A、B、C服务器上添加好oldgirl用户并设置密码，然后，通过oldgirl用户来实现多个服务器之间免密码登录，以A服务器为例，具体步骤如下：

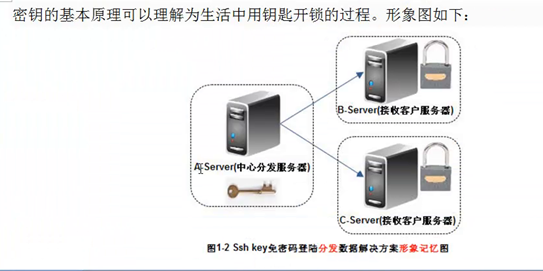
[root@nfs ~]# useradd oldgirl

[root@nfs ~]# echo 123456|passwd --stdin oldgirl

更改用户 oldgirl 的密码 。

passwd： 所有的身份验证令牌已经成功更新。

因为A服务器为中心分发服务器，所以选择在A端建立public key（公钥比作锁）与private key （私钥比作钥匙），实际上只需要有一对密钥就可以，在哪个机器上建立都是一样的。



特别提示：在整个方案实现中，钥匙（private key）和锁（public key）仅需要建立一次即可，可以在A、B、C任意机器上来执行。

[root@backup ~]# grep authorized\_keys /etc/ssh/sshd\_config  
#AuthorizedKeysFile     .ssh/authorized\_keys

SSH批量管理步骤：

1、SSH优化和HOSTS解析

2、创建用户 oldboy

3、生产密钥对

su – oldboy

ssh-keygen –t dsa 一路回车

4、分发公钥到所有服务器

默认端口

5、测试

远程连接：ssh 10.0.0.10

远程执行命令：ssh 10.0.0.1 /sbin/ifcofig eth0

远程拷贝文件：scp,

[root@NFS ~]# su - oldgirl

[oldgirl@NFS ~]$ ssh-keygen -t dsa

#--🡪ssh-keygen是生成密钥的工具，-t参数指建立密钥的类型，这里是建立dsa类型密钥。

#--🡪也可以执行ssh-keygen –t rsa来建立rsa类型密钥。

#--🡪RSA与DSA加密算法的区别：

#--🡪RSA，是一种加密算法（PS：RSA也可以进行数字签名的），它的简写的由来是Ron Rivest、Adi Shamir和Leonard Adleman

#--🡪这三个人姓氏的第一个字母连接起来就是RSA。

#--🡪DSA就是数字签名算法的英文全称的简写，即 digital signature algorithm，简写就是DSA

#--🡪RSA既可以进行加密，也可以进行数字签名实现认证，而DSA只能用于数字签名从而实现认证。

Generating public/private dsa key pair.

Enter file in which to save the key (/home/oldgirl/.ssh/id\_dsa): #--🡪默认回车一路到底即可

Created directory '/home/oldgirl/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/oldgirl/.ssh/id\_dsa.

Your public key has been saved in /home/oldgirl/.ssh/id\_dsa.pub.

The key fingerprint is:

75:20:37:9a:45:a2:3c:79:a9:69:c2:0f:4e:d0:da:f9 oldgirl@NFS

The key's randomart image is:

+--[ DSA 1024]----+

| o.\* |

| . . o O o |

| . . = = . . |

| = . = . . |

| . B + S |

| o \* |

| . E |

| |

| |

+-----------------+

[oldgirl@NFS ~]$

[oldgirl@NFS ~]$ ls -l .ssh/

总用量 8

-rw------- 1 oldgirl oldgirl 668 11月 12 21:51 id\_dsa #--🡪私钥（钥匙）

-rw-r--r-- 1 oldgirl oldgirl 601 11月 12 21:51 id\_dsa.pub #--🡪公钥（锁）

开始实践：

默认端口（22）的命令格式：

[oldgirl@NFS ~]$ ssh-copy-id -i .ssh/id\_dsa.pub oldgirl@10.0.0.8

修改过的端口命令格式：

[oldgirl@NFS ~]$ ssh-copy-id -i .ssh/id\_dsa.pub "-p 52113 oldgirl@163.16.1.7"

此时将公钥（锁）传送到.8服务器的普通用户家目录下，但是由于第一次连接，需要输入验证和密码，以后均无需再次输入。

The authenticity of host '[163.16.1.7]:52113 ([163.16.1.7]:52113)' can't be established.

RSA key fingerprint is 3e:26:68:c2:cc:20:83:5e:22:80:f7:30:85:b9:a3:39.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[163.16.1.7]:52113' (RSA) to the list of known hosts.

oldboy@163.16.1.7's password:

Now try logging into the machine, with "ssh '-p 52113 oldboy@163.16.1.7'", and check in:

.ssh/authorized\_keys

to make sure we haven't added extra keys that you weren't expecting.

[root@web01 ~]# ll /home/oldgirl/.ssh/

总用量 4

-rw------- 1 oldgirl oldgirl 601 11月 12 22:04 authorized\_keys

此时再次使用ssh远程连接命令，不用输入密码即可进行相关操作。

[oldgirl@NFS ~]$ ssh -p52113 oldgirl@10.0.0.8 /sbin/ifconfig

eth1 Link encap:Ethernet HWaddr 00:0C:29:67:C4:8D

inet addr:10.0.0.8 Bcast:10.0.0.255 Mask:255.255.255.0

inet6 addr: fe80::20c:29ff:fe67:c48d/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:2233 errors:0 dropped:0 overruns:0 frame:0

TX packets:1019 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:159218 (155.4 KiB) TX bytes:124303 (121.3 KiB)

#######开始编写关于批量部署的脚本：#######

[oldgirl@NFS ~]$ vim 1.sh

ssh -p52113 oldgirl@10.0.0.7 /sbin/ifconfig eth1

echo =========================================

ssh -p52113 oldgirl@10.0.0.8 /sbin/ifconfig eth1

echo =========================================

ssh -p52113 oldgirl@10.0.0.41 /sbin/ifconfig eth1

[oldgirl@NFS ~]$ sh 1.sh

eth1 Link encap:Ethernet HWaddr 00:0C:29:20:BA:D0

inet addr:10.0.0.7 Bcast:10.0.0.255 Mask:255.255.255.0

inet6 addr: fe80::20c:29ff:fe20:bad0/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:2531 errors:0 dropped:0 overruns:0 frame:0

TX packets:1111 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:176420 (172.2 KiB) TX bytes:135455 (132.2 KiB)

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

eth1 Link encap:Ethernet HWaddr 00:0C:29:67:C4:8D

inet addr:10.0.0.8 Bcast:10.0.0.255 Mask:255.255.255.0

inet6 addr: fe80::20c:29ff:fe67:c48d/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:2699 errors:0 dropped:0 overruns:0 frame:0

TX packets:1261 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:201362 (196.6 KiB) TX bytes:163001 (159.1 KiB)

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

eth1 Link encap:Ethernet HWaddr 00:0C:29:69:0E:8E

inet addr:10.0.0.41 Bcast:10.0.0.255 Mask:255.255.255.0

inet6 addr: fe80::20c:29ff:fe69:e8e/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:3728 errors:0 dropped:0 overruns:0 frame:0

TX packets:2476 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:367047 (358.4 KiB) TX bytes:197910 (193.2 KiB)

#######继续编写关于分发文件的脚本：#######

[oldgirl@NFS ~]$ vim fenfa.sh

scp -P52113 hosts oldgirl@10.0.0.7:~

scp -P52113 hosts oldgirl@10.0.0.8:~

scp -P52113 hosts [oldgirl@10.0.0.41:~](mailto:oldgirl@10.0.0.41:~)

"fenfa.sh" [新] 3L, 112C 已写入

[oldgirl@NFS ~]$ sh fenfa.sh

hosts 100% 300 0.3KB/s 00:00

hosts 100% 300 0.3KB/s 00:00

hosts 100% 300 0.3KB/s 00:00

将脚本优化一下之后的内容为：

[oldgirl@NFS ~]$ vim fenfa.sh

#!/bin/sh

for n in 7 8 41

do

scp -P52113 life.txt oldboy@163.16.1.$n:~

done

[oldgirl@NFS ~]$ 此时已将本地文件分发给另外三台设备的家目录中。

将脚本优化，显得更专业化模式：

[oldgirl@nfs ~]$ vim fenfa.sh

#!/bin/sh

. /etc/init.d/functions

for n in 7 8 41

do

scp -pr -P52113 $1 oldgirl@163.16.1.$n:~ &>/dev/null

if [ $? -eq 0 ]

then

action "fenfa $1 ok" /bin/true

else

action "fenfa $1 ok" /bin/false

fi

done

"fenfa.sh" 12L, 237C 已写入

[oldgirl@nfs ~]$ sh fenfa.sh hosts

fenfa hosts ok [确定]

fenfa hosts ok [确定]

fenfa hosts ok [确定]

[oldgirl@nfs ~]$

################最具专业的脚本##############批量分发#########

[oldboy@nfs ~]$ vim fenfa.sh

#!/bin/sh

. /etc/init.d/functions

if [ $# -ne 1 ];then

echo "USAGE:$0 {FILENAME|DIRNAME}"

exit 1

fi

for n in 7 8 41

do

scp -rp -P52113 $1 oldboy@163.16.1.$n:~ &>/dev/null

if [ $? -eq 0 ]

then

action "fenfa $1 ok" /bin/true

else

action "fenfa $1 ok" /bin/false

fi

done

~

~

~

"fenfa.sh" 16L, 288C 已写入

[oldboy@nfs ~]$ sh fenfa.sh

USAGE:fenfa.sh {FILENAME|DIRNAME}

#############################继续脚本########################批量管理########

[oldboy@nfs ~]$ vim ip.sh

#!/bin/sh

if [ $# -ne 1 ];then

echo "USAGE:$0 COMMAND"

exit 1

fi

for n in 7 8 41

do

ssh -p52113 oldboy@163.16.1.$n $1

done

"ip.sh" 9L, 126C 已写入

[oldboy@nfs ~]$ sh ip.sh

USAGE:ip.sh COMMAND

[oldboy@nfs ~]$ sh ip.sh "/sbin/ifconfig eth1"

eth1 Link encap:Ethernet HWaddr 00:0C:29:F5:0E:23

inet addr:10.0.0.7 Bcast:10.0.0.255 Mask:255.255.255.0

inet6 addr: fe80::20c:29ff:fef5:e23/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:1848 errors:0 dropped:0 overruns:0 frame:0

TX packets:212 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:132257 (129.1 KiB) TX bytes:26529 (25.9 KiB)

注意：在分发的时候需查看客户端公钥文件的权限与目录的权限，是否为600,700，若不是的话讲无法传送文件。

**SSH批量分发与管理方案小结：**

1、利用root做ssh key 验证。

优点：简单，易用。

缺点：安全差，同时无法禁止root远程连接这个功能。

企业应用：80%的中小企业。

2、利用普通用户如oldboy来做

思路：先把分发的文件拷贝到服务器用户家目录，然后sudo提权拷贝分发的文件放到对应权限目录。

优点：安全，无需停止root远程连接这个功能。

缺点：配置比较复杂。

3、使用suid对固定的命令进行修改（如rsync，意思为任何人在执行rsync时都拥有该命令属主的权限（root）这样既可随意存放文件）

优点：相对安全

缺点：复杂，安全性较差，任何人都可以处理带有suid权限的命令。

导师对于批量分发提权的建议：

a.最求简单，选择1.

b.追求安全建议，选择2

有能力可以选择：puppet，saltstack。

suid：普通用户运行程序，没权限时也可以suid，然后运行，必须是编译好的程序命令。

SSH 批量分发提权  
**1、sudo方案**

将所有客户端的visudo配置文件修改，允许/bin/cp命令或/usr/bin/rsync命令可执行的权限。（###远程sudo需要加-t###）  
scp -P52113 hosts oldboy@172.16.1.8:~

ssh -t -p 52113 oldboy@172.16.1.8 sudo /bin/cp hosts /etc/

sudo配置

echo "oldboy ALL=(ALL) NOPASSWD: /bin/cp" >>/etc/sudoers

scp -P52113 -rp abc oldboy@172.16.1.8:~

ssh -t -p 52113 oldboy@172.16.1.8 sudo /bin/cp -rp /home/oldboy/abc /etc/

[oldboy@nfs01 ~]$ cat fenfa.sh

#!/bin/sh

. /etc/init.d/functions

if [ $# -ne 2 ];then

echo "USAGE:/bin/sh $0 FILENAME PATH"

exit 1

fi

for n in 41 8 9 10

do

scp -P52113 -rp $1 oldboy@172.16.1.$n:~ &>/dev/null &&\

ssh -t -p 52113 oldboy@172.16.1.$n sudo /bin/cp -rp ~/$1 $2 &>/dev/null

if [ $? -eq 0 ];then

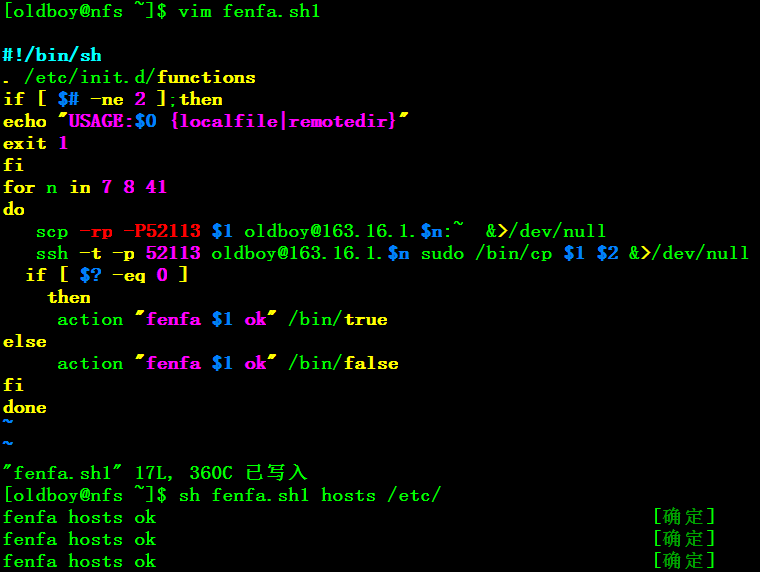
action "dis $1 to 172.16.1.$n" /bin/true

else

action "dis $1 to 172.16.1.$n" /bin/false

fi

done

  
**2、suid特性完成任务（不要用）**  
chmod 4755 `which rsync`  
  
[oldboy@nfs01 ~]$ cat fenfa.sh   
#!/bin/sh  
. /etc/init.d/functions  
  
if [ $# -ne 2 ];then  
  echo "USAGE:/bin/sh $0 FILENAME PATH"  
  exit 1  
fi  
  
for n in 41 8 9 10  
do  
   scp -P52113 -rp $1 oldboy@172.16.1.$n:~ &>/dev/null &&\  
   ssh -p 52113 oldboy@172.16.1.$n /usr/bin/rsync  ~/$1 $2 &>/dev/null  
  if [ $? -eq 0 ];then  
    action "dis $1 to 172.16.1.$n" /bin/true  
  else  
    action "dis $1 to 172.16.1.$n" /bin/false  
  fi  
done  
**3、root用户（\*\*\*\*\*）**  
取消禁止远程登录的限制  
简单 易用