服务器集群配置操作手册

## 服务器主机优化

### 1、关闭seLinux

sed -i 's#SELINUX=enforcing#SELINUX=disabled#gp' /etc/selinux/config

setenforce 0 临时生效

### 2、关闭防火墙

/etc/init.d/iptables stop

/etc/init.d/iptables stop

chkconfig iptables off

### 3、关闭不应该启动的服务

chkconfig --list|grep -Ev 'crond|sshd|network|rsyslog|sysstat'|awk '{print "chkconfig",$1,"off"}'|bash

### 4、为用户添加sudo权限

useradd lijinpeng

cp /etc/sudoers /etc/sudoers.ori

echo "lijinpeng ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers

tail -1 /etc/sudoers

visudo -c //检查语法

### 5、中文字符集

cp /etc/sysconfig/i18n /etc/sysconfig/i18n.ori

echo 'LANG="zh\_CN.UTF-8"' >/etc/sysconfig/i18n

source /etc/sysconfig/i18n

echo $LANG

### 6、时间同步

echo '#time sync by lijinpeng at 2017-05-01' >>/var/spool/cron>root

echo '\*/5 \* \* \* \* /usr/sbin/ntpdate time.nist.gov > /dev/null 2>&1' >>/var/spool/cron>root

crontab -l

### 7、命令行安全

echo 'export TMOUT=300' >> /etc/profile

echo 'export HISTSIZE=5' >>/etc/profile

echo 'export HISTFILESIZE=5'>>/etc/profile

tail -3 /etc/profile

### 8、加大文件描述

echo ‘\* - nofile 65535’>>/etc/security/limits.conf

tail -1 /etc/security/limits.conf

### 9、内核优化

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

net.ipv4.tcp\_fin\_timeout=2

net.ipv4.tcp\_tw\_reuse=1

net.ipv4.tcp\_tw\_recycle=1

net.ipv4.tcp\_syncookies=1

net.ipv4.tcp\_keepalive\_time=600

net.ipv4.ip\_local\_port\_range=4000 65000

net.ipv4.tcp\_max\_syn\_backlog=16384

net.ipv4.tcp\_max\_tw\_buckets=36000

net.ipv4.route.gc\_timeout=100

net.ipv4.tcp\_syn\_retires=1

net.ipv4.tcp\_synack\_retries=1

net.core.somaxconn=16384

net.core.netdev\_max\_backlog=16384

net.ipv4.tcp\_max\_orphans=16348

EOF

#以下参数是对iptables防火墙的优化 防火墙不开会提示 可 忽略 不理

net.nf\_conntrack\_max=25000000

net.netfilter.nf\_conntrack\_tcp\_timeout\_established=180

net.netfilter.nf\_conntrack\_tcp\_timeout\_time\_wait=120

net.netfilter.nf\_conntrack\_tcp\_timeout\_close\_wait=60

net.netfilter.nf\_conntrack\_tcp\_timeout\_fin\_wait=120

## 二、备份服务器（backup）

### 1、主机信息

内网ip:172.16.1.42

外网ip:10.0.0.42

主机名：backup02

### 2、配置备份服务器ip信息

#### 配置ip

使用模板机的IP登陆CRT。

Vim /etc/sysconfig/network-scripts/ifcfg-eth0 //编辑第一块网卡

TYPE=Ethernet

ONBOOT=yes

NM\_CONTROLLED=yes

BOOTPROTO=none

USERCTL=no

IPV6INIT=no

IPADDR=10.0.0.42 //修改ip

NETMASK=255.255.255.0

DNS2=223.5.5.5

GATEWAY=10.0.0.2

DNS1=10.0.0.2

Vim /etc/sysconfig/network-scripts/ifcfg-eth1 //编辑第二块网卡

DEVICE=eth1

TYPE=Ethernet

ONBOOT=yes

NM\_CONTROLLED=yes

BOOTPROTO=none

IPV6INIT=no

USERCTL=no

IPADDR=172.16.1.42 //修改内网ip

NETMASK=255.255.255.0

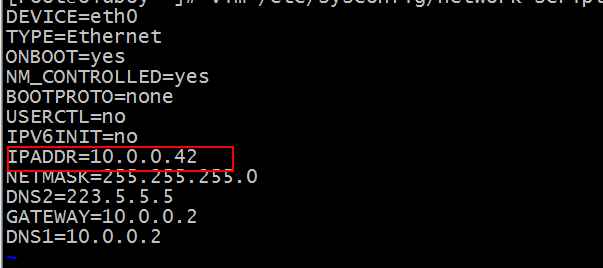
修改后

内网ip：172.16.1.42

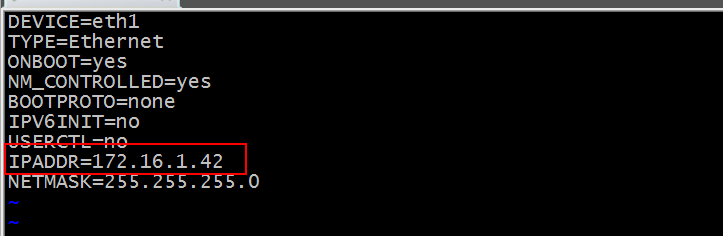
外网ip:10.0.0.42

真实环境操作：

vim /etc/sysconfig/network-scripts/ifcfg-eth0



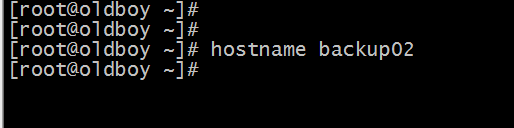
vim /etc/sysconfig/network-scripts/ifcfg-eth1



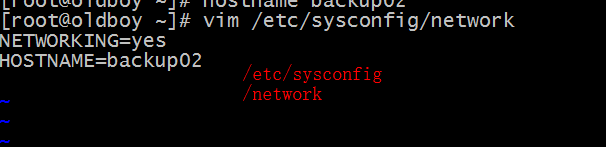
#### 配置主机名

配置主机名 需要修改两个 文件 三步操作

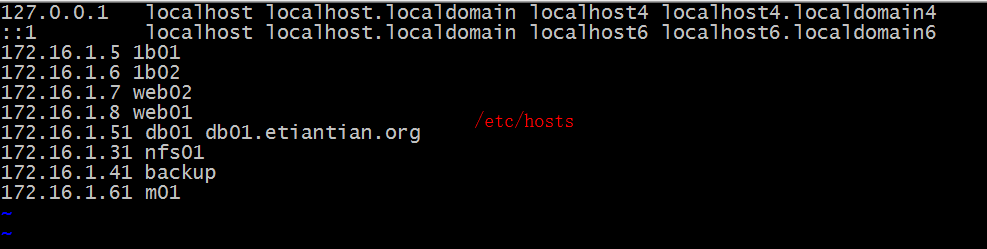
第一步：hostname backup02 //修改主机名 临时生效



第二步：vim /etc/sysconfig/network // hostname=backup02



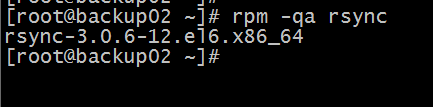
第三步：配置主机地址映射（这一步与本机主机名设置无关 作用让内部主机互相访问 速度更快!）



### 3、检查和安装rsync 备份服务器

**检查rsync是否安装**

rpm -qa rsync



一般linux 都预装了 rsync，如果没有安装，请使用命令在线安装

yum install rsync -y

### 编辑rsync配置文件

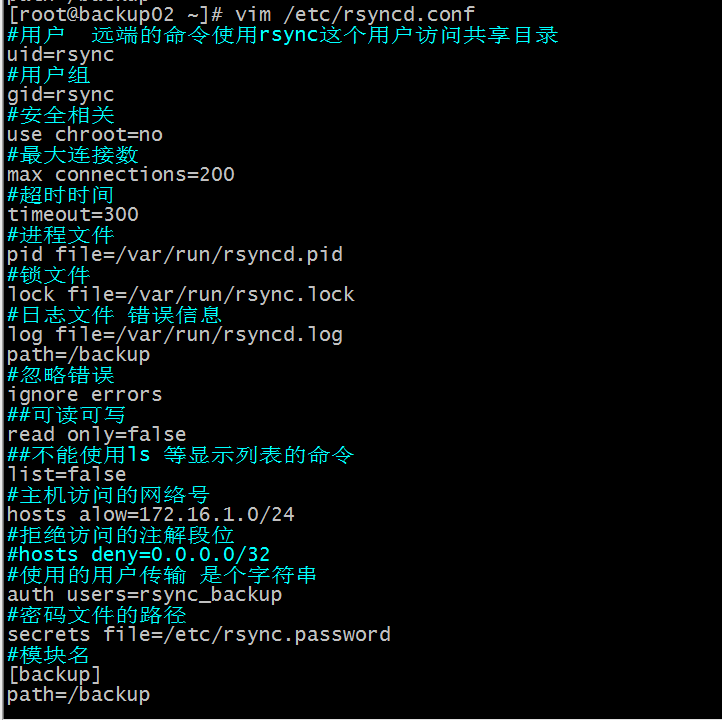
rsync的配置文件是 /etc/rsyncd.conf (默认不存在，需要手动创建)

创建 rsyncd.conf

touch /etc/rsyncd.conf

编辑rsyncd.d 配置文件

Vim /etc/rsyncd.conf



参数说明：

#用户 远端的命令使用rsync这个用户访问共享目录

uid=rsync

#用户组

gid=rsync

#安全相关

use chroot=no

#最大连接数

max connections=200

#超时时间

timeout=300

#进程文件（系统自动创建）

pid file=/var/run/rsyncd.pid

#锁文件（系统自动创建）

lock file=/var/run/rsync.lock

#日志文件 错误信息（系统自动创建）

log file=/var/run/rsyncd.log

#忽略错误

ignore errors

##可读可写（控制远程主机对备份目录 的操作权限）

read only=false

##不能使用ls 等显示列表的命令

list=false

#主机访问的网络号

hosts alow=172.16.1.0/24

#拒绝访问的注解段位（可不加）

#hosts deny=0.0.0.0/32

#使用的用户传输 是个字符串（备份过程中 数据传输安全 要靠它验证）

auth users=rsync\_backup

#密码文件的路径（远程主机备份时 需要携带安全用户 并且携带密码，服务端读取该文件进行判断）

secrets file=/etc/rsync.password

#模块名（远程主机备份时 必须接模块名，模块后可以加目录）

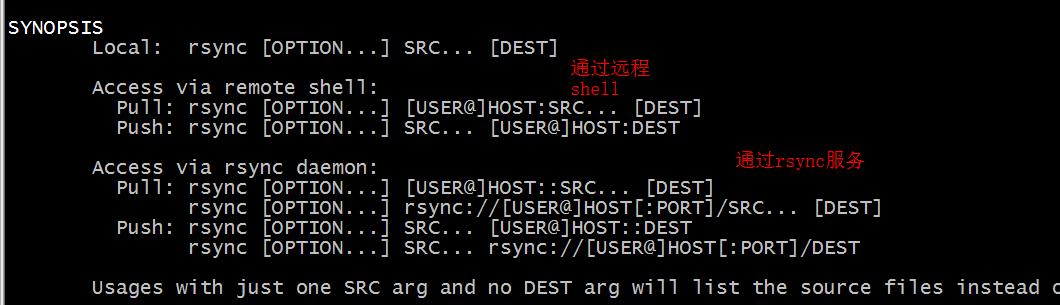
[backup]

path=/backup

**说明：该主机的目录/backup就是 一个备份目录，其他主机可以使用rsync命令将需要备份的数据，备份到该目录。**

### rsync命令说明

man rsync //查看帮助



rsync提供两种方式去远程 pull push 文件。

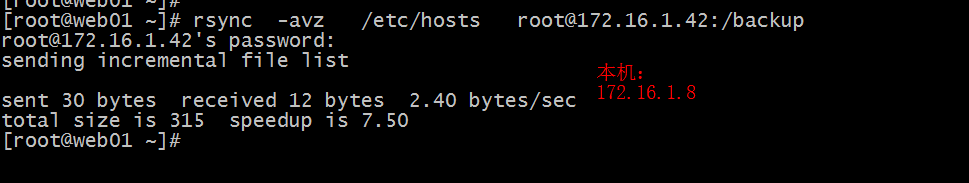
第一种：通过shell的方式去备份。（不常用）

1从本机把要备份的文件推送到备份服务器172.16.1.42的/backup目录下

Push(推)

rsync [OPTION...] SRC... [USER@]HOST:DEST

rsync -avz /etc/hosts [root@172.16.1.42:/backup](mailto:root@172.16.1.42:/backup)（过程需要密码验证）



这个命令的意思是：-avz备份文件的所有属性

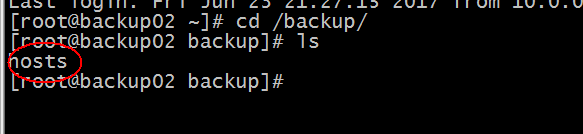
/etc/hosts 本机要备份的文件内容

root 备份文件时使用的备份服务器的用户名，

172.16.1.42 备份服务器主机名

/backup 备份文件到备份服务器的目的目录

查看备份服务器/backup文件下是否存在hosts 文件



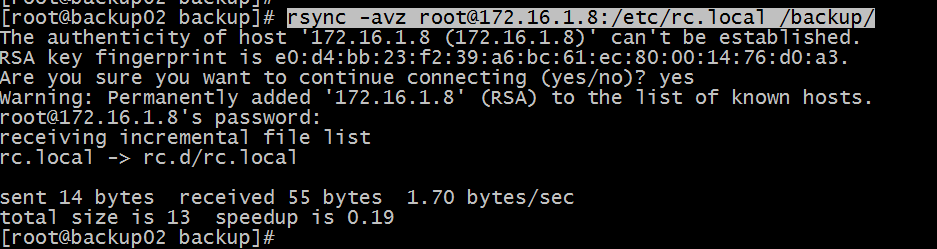
存在hosts文件证明文件推送成功！

2从备份服务器172.16.1.42拉去主机172.16.1.8的/etc/rc.local文件。

Pull(拉)

rsync [OPTION…] [USER@]:HOST:SRC… [DEST]

rsync -avz [root@172.16.1.8:/etc/rc.local /backup/](mailto:root@172.16.1.8:/etc/rc.local%20/backup/)(需要主机root用户的密码)



这个命令的意思是：-avz备份文件的所有属性

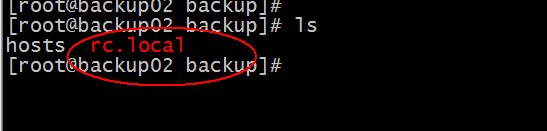
/etc/rc.local 要拉取主机的/etc/rc.local文件

root 备份服务器要拉去文件的主机用户，

172.16.1.42 被拉去主机的ip

/backup 备份文件到备份服务器的目的目录

查看备份服务器下是否存在rc.local 文件



存在rc.local 证明 拉去文件成功！

第二种：通过rsync服务去拉去，但是需要开启rsync（rsync --daemon）

下节为具体配置

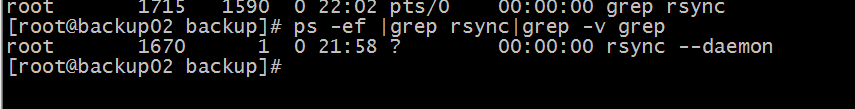
### 开启rsync服务

**开启服务**

rsync –daemon

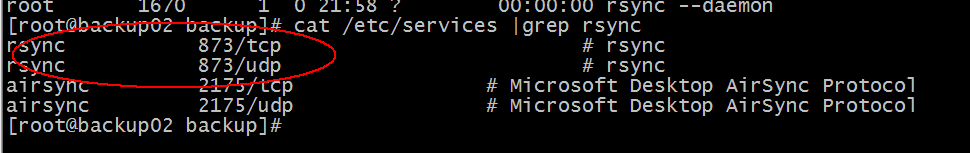
**查看服务是否启动**

ps -ef |grep rsync|grep -v grep



服务已启动

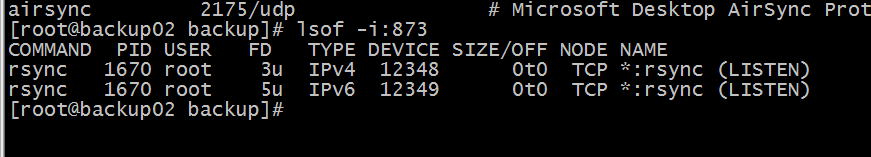
**查看rsync使用的端口号（扩展）**



rsync所使用的端口号为873

**通过端口号查看对应的服务（扩展）**

lsof -i:873



可知 873端口对应了rsync服务

**将rsync开启设置未开机启动**

将该操作加入/etc/rc.local文件即可。

echo ‘rsync –daemon’ >>/ect/rc.local

### 创建rsync用户

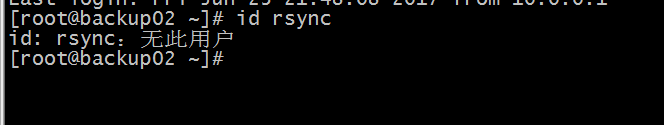
Rsync服务开启后，主机与服务器之间通讯，需要使用在备份服务器端建立一个相应的用户去处理 这个操作过程，如果把这个过程交给超级用户，这也太危险了。Rsync不会主动为我们创建这个用户，我们需要手动去创建，并且为这个用户赋予相应目录的操作权限，这样这个用户才可以完成备份文件的功能。

我们在rsyncd.conf里面的第一行uid指定了这个服务所用的用户为rsync,所以我们直接创建rsync

**创建rsync用户（无需创建家目录，仅仅完成文件传输而已）**

检查用户是否存在

id rsync



不存在

创建用户

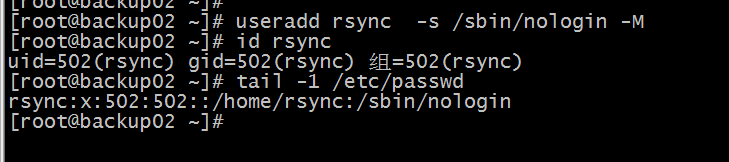
useradd rsync -s /sbin/nologin -M

命令说明： -s 指定用户的shell

-M 不创建家目录

检查用户

id rsync



用户创建成功！

### 分配文件权限给rsync

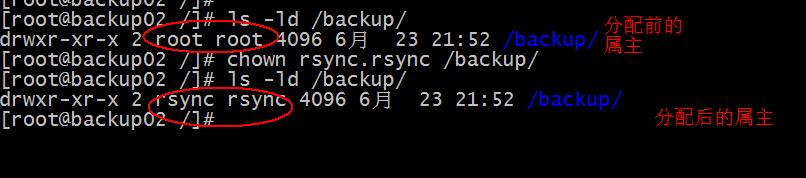
我们在rsyncd.conf文件中配置的模块名为：backup,这个模块对应的目录名为/backup. 所以我们应该先在备份服务器上创建该目录，然后将这个目录的读写权限分配给rsync用户。

mkdir -p /backup

将该目录的权限分给rsync，最好的办法就是 将该目录及子目录的属主分配给rsync

chown rsync.rsync /backup

查看文件的属性



### 设置密码文件

控制服务运行的用户有了，但是，只有用户，没有密码验证也是超级不安全的，我们在rsyncd.conf配置了密码文件secrets file=/etc/rsync.password

所以我们应该在备份服务器创建这个密码文件，因为是密码文件，所以应该设置这个文件权限 不能随随便便让人访问，所以必须为其设置600的权限。

**注意：这个密码是auth users=rsync\_backup这个用户的密码，相当于充当了认证的作用，可以理解成一个假的用户，一个字符串，但是必须有对应的密码，去作为连接时候的认证。**

远程用户连接的过程可以理解为：

rsync ………

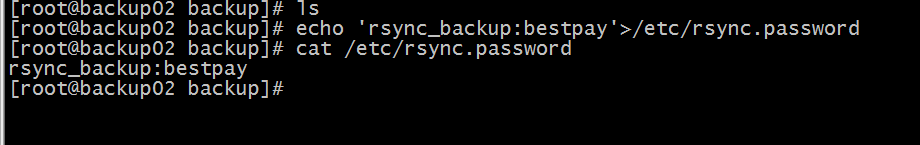
然后远程主机会去连接备份服务器，并且会在连接的时候，把密码传送过去，备份服务器读取密码文件secrets file=/etc/rsync.password，查找rsyncd.conf文件中配置的安全校验用户名（用户名）rsync\_backup，如果有，匹配密码，如果匹配上，就开始用rsync用户对备份目录做操作，否则就验证失败!

创建密码文件

echo ‘rsync:bestpay’ >/etc/rsync.password

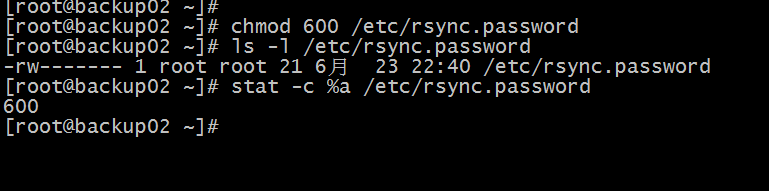
查看密码文件内容

cat /etc/rsync.password



设置密码文件的权限

Chmod 600 /etc/rsync.password



查看文件的权限

ls -l /etc/rsync.password

直接用二进制形式表示

stat -c %a /etc/rsync.password

至此，服务器端配置完毕。

### 配置完毕，测试

我们使用rsync提供的第二种连接方式，通过服务去推送或者拉取

Access via rsync daemon:

Pull: （拉取提供了两种方式）

rsync [OPTION...] [USER@]HOST::SRC... [DEST]

rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]

Push: （推送提供了两种方式）

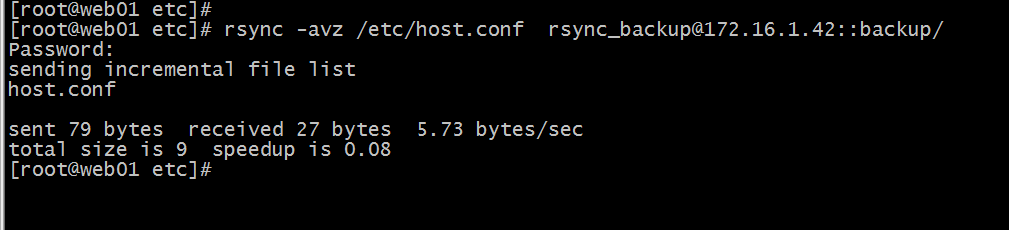
rsync [OPTION...] SRC... [USER@]HOST::DEST

rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST

1. 主机推送/etc/init.d/目录下的所有内容到备份服务器/backup

第一种命令

rsync -avz /etc/host.conf rsync\_backup@172.16.1.42::backup/



命令解析：

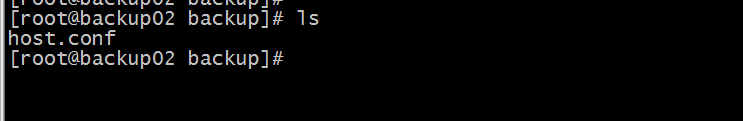
/etc/host.conf //要备份的主机文件

rsync\_backup //安全校验用户 输入的密码对应这个用户

172.16.1.42 //备份服务器ip

backup/ //备份服务器的模块，这个模块对应的目录为/backup。这是在配置文件里面配置的，文件名可以和模块名不一样。

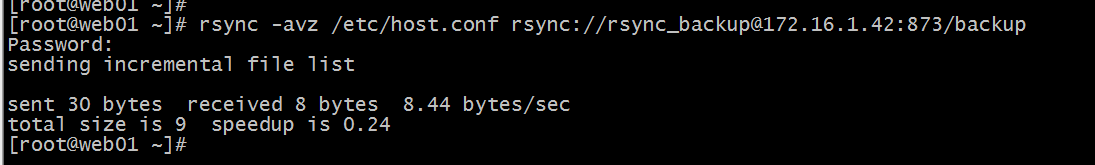
查看是否存在备份文件服务器 host.conf文件



存在该文件，证明备份成功！

第二种命令

rsync -avz /etc/host.conf rsync://rsync\_backup@172.16.1.42:873/backup



命令解释： 这是格式 没啥好解释的，不要有空格。

这种方式可以指定使用的端口好。

**自动输入密码**

我们每次输入密码很麻烦，所以我们可以设置自动读取密码，即在主机端创建密码文件，将密码输入文件，设置文件权限为600，然后在命令行后加 –password-file=path（密码文件路径）即可

**创建密码文件**

echo ‘bestpay’>>/etc/rsync.password（主机端无需加用户名）

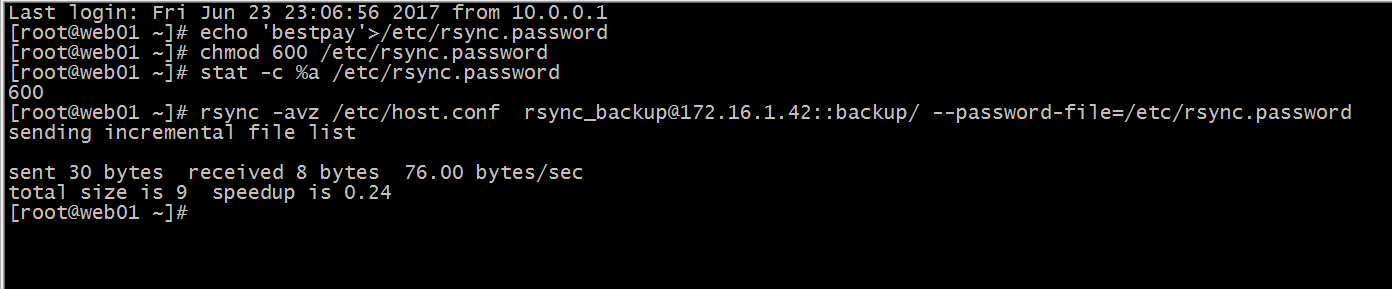
**设置密码文件权限**

chmod 600 /etc/rsync.password

**执行命令**

rsync -avz /etc/host.conf [rsync\_backup@172.16.1.42::backup/](mailto:rsync_backup@172.16.1.42::backup/) --password-file=/etc/rsync.password

**操作过程**



第二种方式 省略，配置一样。

### 案例

**备份全网服务器数据生产架构方案案例模型**

**企业案例：rsync上机实战考题**

某公司里有一台web服务器，里面数据很重要，但是如果硬盘坏了，数据就会丢失，现在领导要求你把数据在其他机器上做一个周期性定时备份，要求如下：

每天晚上00点整在web服务器A上打包备份网站程序目录，并通过rsync命令推送到服务器B上备份保留（备份思路可以是现在本地按日期打包，然后上传到备份服务器上、）

具体要求:

1. web服务器A和备份服务器B的备份目录必须为/backup
2. web服务器站点目录为/var/www/html/
3. web服务器本地保留7天内的数据
4. 检查备份 结果是否正确，并且发送邮件给管理员信箱
5. 备份服务器上周六数据都保存，其他数据只保留180天以内的数据。

思路：分三个步骤去处理

第一步：备份服务器搭建，确保主机可以推送备份文件

第二步：根据主机ip在本地生成备份文件，删除无需保留的文件

第三部：编写定时任务，推送到服务器，并检查文件和发送管理员邮箱

环境说明: 备份服务器的ip是172.16.1.42 测试主机为 172.16.1.8 模拟环境，在主机上新建一个/var/www/html文件夹，并在该文件夹中新建一些文件。

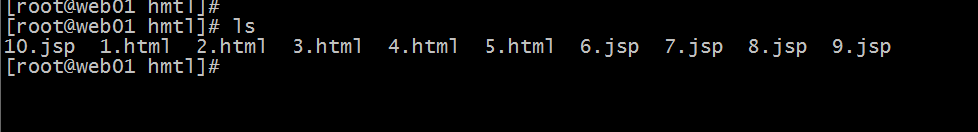
mkdir /var/www/hmtl -p

touch /var/www/hmtl/{1..5}.html

touch /var/www/hmtl/{6..10}.jsp

**查看/var/www/html/目录下的文件**

ls /var/www/html



环境已准备好，可以进行测试！

第一步：构建备份服务

编辑/etc/rsyncd.conf文件，其余配置不变，新增一个模块。

**新增模块（172.16.1.42）**

vim /etc/rsyncd.conf(在该文件中新增模块)

[www\_backup]

path=/backup

在服务端新建/backup目录

mkdir /backup

将/backup文件的属主改为rsync

chown -R rsync.rsync /backup

**客户端尝试是否可以正确推送到目的地（172.16.1.8）**

rsync -avz /etc/hosts rsync\_backup@172.16.1.42::www\_backup/ --password-file=/etc/rsync.password

检查服务端172.16.1.42的/backu的目录下是否存在hosts文件，如果存在则表示成功!

至此第一个步骤完成！

第二步：根据主机ip在本地生成备份文件，删除无需保留的文件

将/var/www/html按照日期打包,打包到/backup目录下

在主机端（172.16.1.8）新建/backup目录

mkdir /backup

按照日期打包

tar -avzf /backup/$(date +%F) \_backup\_web.tar.gz /var/www/hmtl/

检查是否存在

ls /backup

其实这样生成已经可以了，但是我们发送到备份服务器最好用ip区分下，到底是哪台机器的备份文件，因此需要新建一个文件，文件名为ip

根据ip新建文件夹

mkdir -p /backup/`ifconfig eth1|sed -n '2p'|awk -F '[ :]+' '{print $4}'`

打包的时候直接打包到IP目录下就可以了。

本地只保存7天以内的备份文件，所以删除七天前的文件

‘find /backup/172.16.1.8/ -type f -name '\*.tar.gz' -mtime +7|xargs rm -f

因为需要用到很多命令，所以写个脚本。更好。可以执行更多的操作

编写脚本/root/scripts/www.sh

#set IPS设置变量IP

IP=$(ifconfig eth1|sed -n '2p'|awk -F '[ :]+' '{print $4}')

#make a dir by ip 创建ip目录

mkdir -p /backup/$IP/www/

#创建flgs标志文件 用于记录文件传输工程中是否损坏

mkdir -p /backup/$IP/www/flags

#packing 打包

tar -cvzf /backup/$IP/www/$(date +%F)\_backup\_web.tar.gz /var/www/html/\*

#delete file 7 days ago

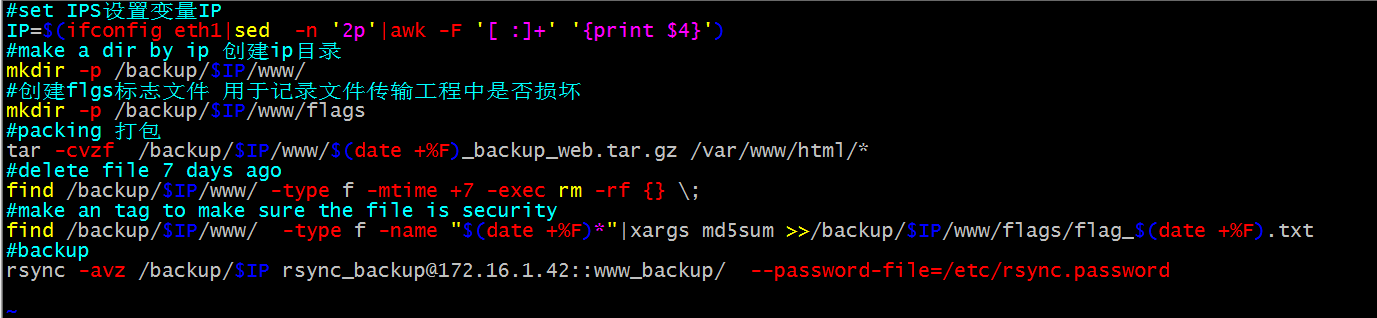
find /backup/$IP/www/ -type f -mtime +7 -exec rm -rf {} \;

#make an tag to make sure the file is security

find /backup/$IP/www/ -type f -name "$(date +%F)\*"|xargs md5sum >>/backup/$IP/www/flags/flag\_$(date +%F).txt

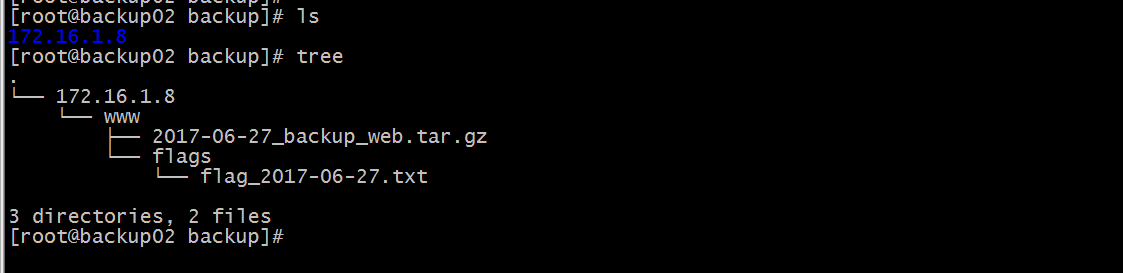
#backup

rsync -avz /backup/$IP rsync\_backup@172.16.1.42::www\_backup/ --password-file=/etc/rsync.password



测试：sh /root/scripts/www.sh

检查服务端是否存在



可以看到存在备份后的文件，并且目录结构是IP/www/备份内容 IP/www/fags/标识文件。

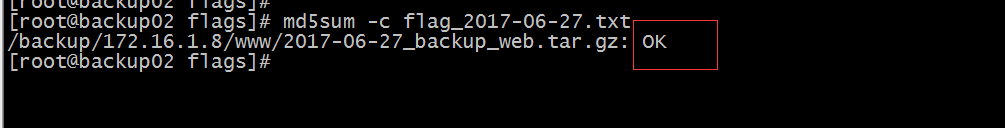
**md5sum**

Md5sum 可以对一个文件生成32字节的摘要，当文件改变时，就改变了文件的md5值，可以根据这个原理 判断文件在传输过程中是否有过变化

find /backup/$IP/www/ -type f -name "$(date +%F)\*"|xargs md5sum >>/backup/$IP/www/flags/flag\_$(date +%F).txt

查找备份的文件，并对备份的文件做摘要，在服务判断 md5um -c 文件即可。

服务端: md5sum -c /backup/172.16.1.8/www/flags/ flag\_2017-06-27.txt 得到如下结果证明没有被修改过！



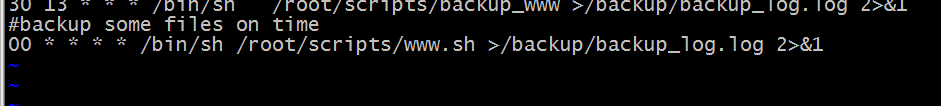
如果为OK则证明文件在传输过程并没有损坏。至此第二步已经完成！

**第三步: 编写定时任务**

我们 需要在客户端上编写定时任务，去执行这个脚本

crontab -e //对当前用户 编辑定时任务

00 \* \* \* \* /bin/sh /root/scripts/www.sh >/backup/backup\_log.log 2>&1



至此 配置完毕！

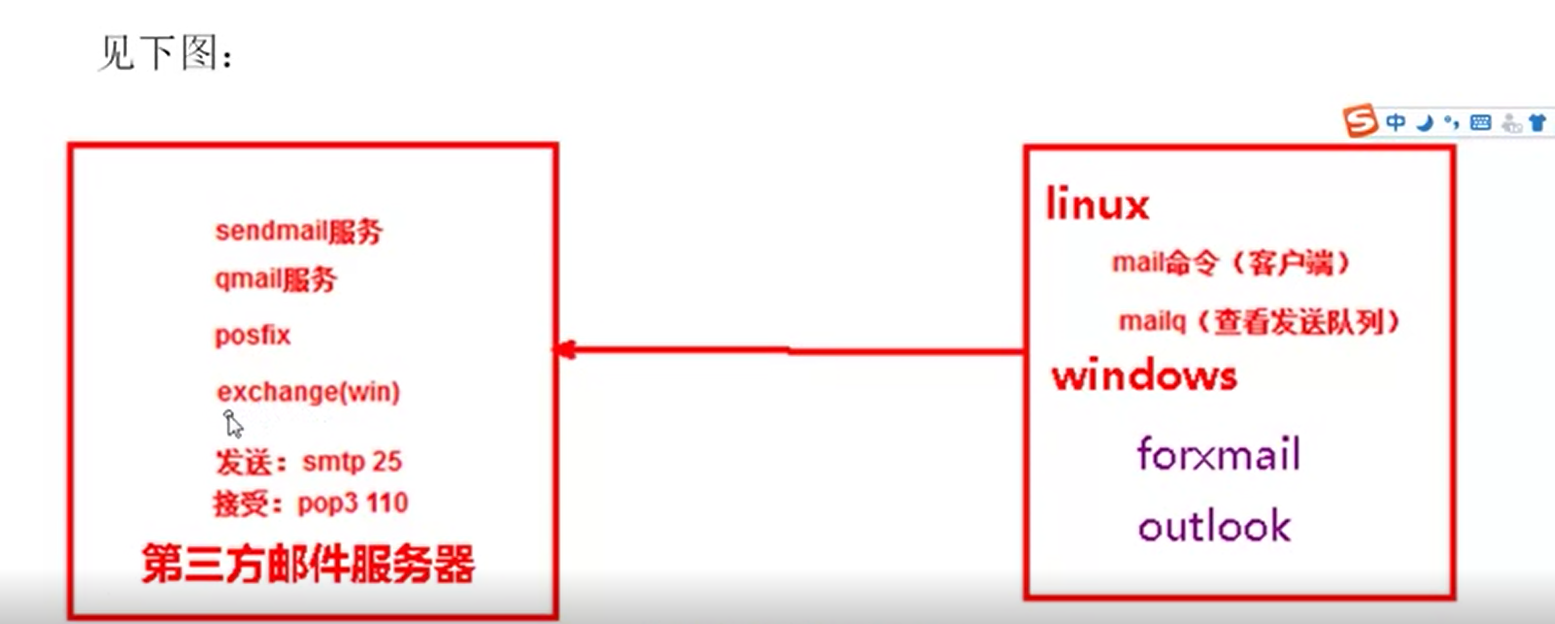
**发送邮件**

**发送邮件的语法（推荐）**

**mail -s ‘标题’ 邮件地址<文件**

**mail -s ‘oldboy’** [957143335@qq.com](mailto:957143335@qq.com)

**echo ‘正文’|mail -s “oldboy”** 957143335@qq.comm



## 网络文件共享主机(NFS)

### 1、主机信息

内网ip:172.16.1.32

外网ip:10.0.0.32

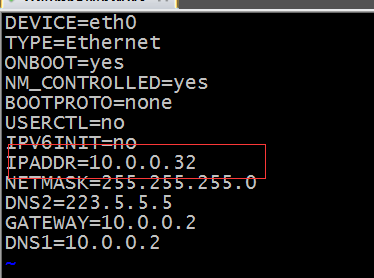
主机名：nfs02

### 2、配置nfs网络文件共享服务器IP

为了集群的需要，为nfs服务器配置两块网卡，用作内网ip和外网ip。

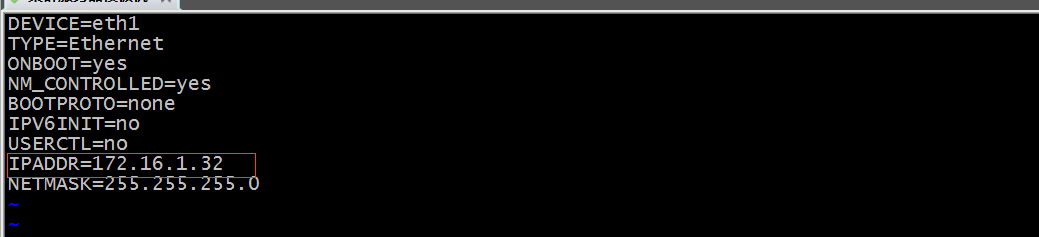
修改公网ip：10.0.0.32

vim /etc/sysconfig/network-scripts/ifcfg-eth0



修改内网ip：172.16.1.32

vim /etc/sysconfig/network-scripts/ifcfg-eth1



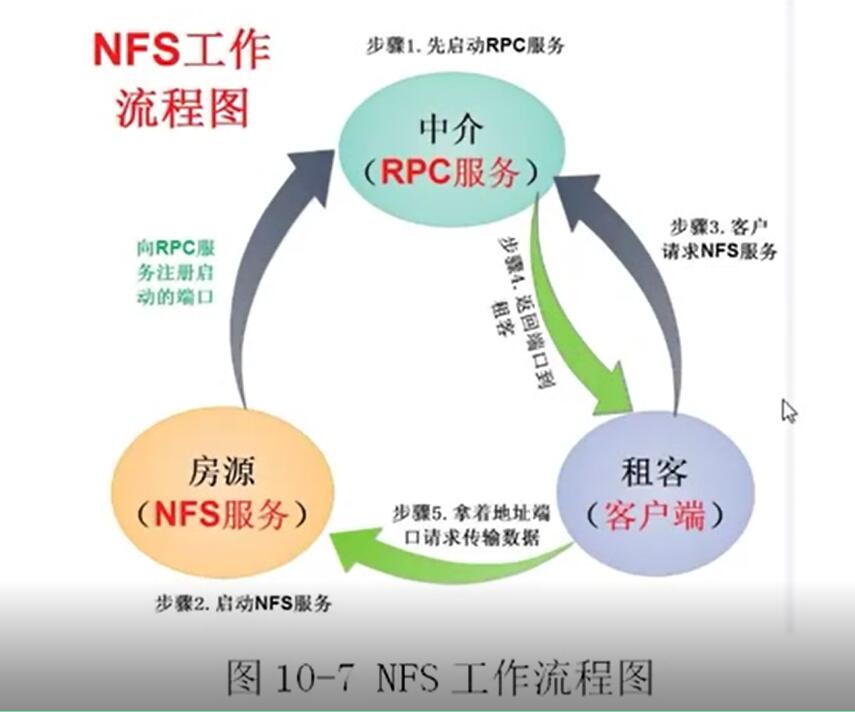
修改主机名：nfs02

hostname nfs02

vim /etc/sysconfig/network 

重启主机 即所有设置已生效

### 3、服务端：安装nfs-utils 和 rpcbind



检查nfs-utils和rpcbind是否安装

rpm -qa nfs-utils

rpm -qa rpcbind

如果没有安装，在线安装

yum install nfs-utils rpcbind -y

### 4、rpc服务

Rpc服务：rpc是一种远程调用的编程技术，在这里rpc通过TCP/IP协议 连接两台网络主机，使我们在编写程序时只需要按照本地的调用过程去实现，远程调用则交给rpc，无需考虑远程调用的数据传输过程。

比如在网络文件共享服务中，如果我们想在远程服务器上建立一个文件夹/data.我们只需要在本地使用 mkdir /data即可，具体调用远程服务器创建文件夹的过程交给rpc实现，主机的rpc服务会把这个请求交给远程服务器的rpc，远程服务器的rpc会调用 本地的文件系统命令，建立文件夹。

### 开启rpc服务

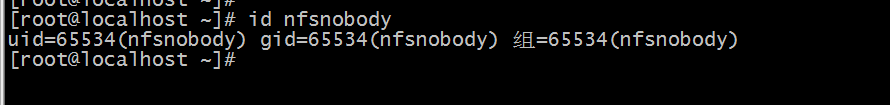
**NFS 工作原理：**

服务器开启rpc服务，此时rpc会处于服务状态，当nfs开启后，nfs会向rpc注册监听的端口号（端口号是随机的），当客户端nfs请求服务端的nfs时候，rpc会将本地nfs的端口号返回给客户端的nfs，客户端拿着服务端给的nfs端口号，去根据端口号，连接服务端的nfs，发送命令，由服务端做出响应。

注意：nfs服务与rsync服务一样，也需要一个用户（nfsnobody）去管理这个服务的调用的过程，但是由于nfs在安装的时候，会自动创建nfsnobody，所以我们无需再去手动创建

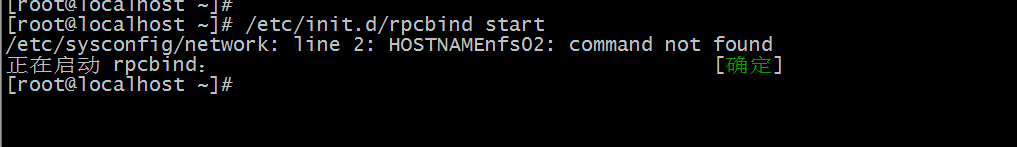
**查看nfsnobody用户**

id nfsbobody



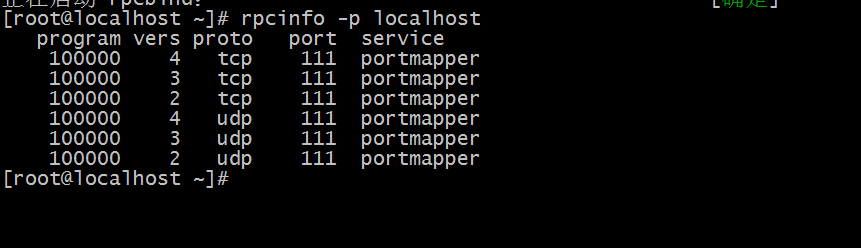
**开启rpc服务**

/etc/init.d/rpcbind start



查看rpc是否有nfs端口资源

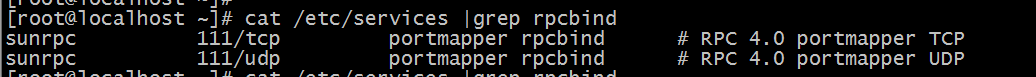
rpcinfo -p localhost



111是rpc使用的端口号，因为nfs服务器并未开启，所有没有nfs端口号注册进来。

**查看rpc使用的端口号**

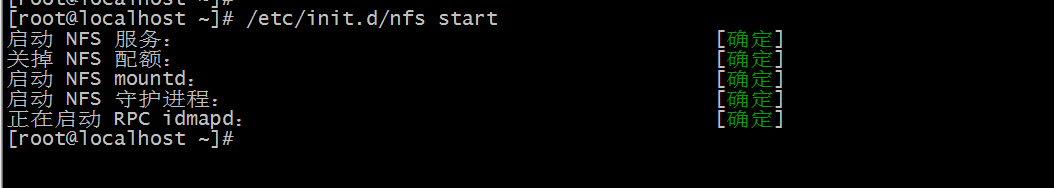
cat /etc/services |grep rpcbind



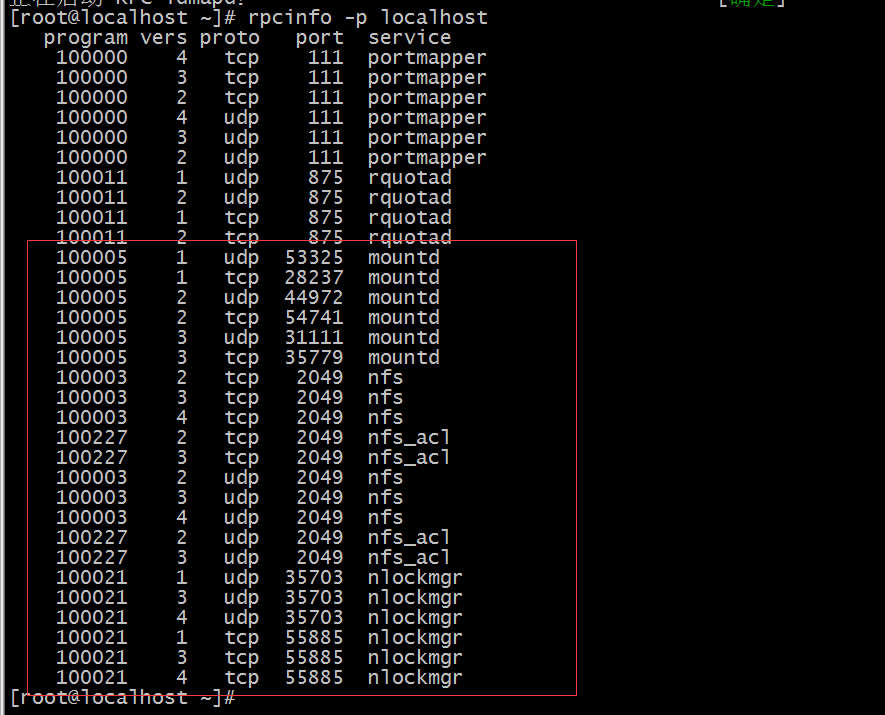
端口号为111

**开启nfs服务器**

**/etc/init.d/nfs start**

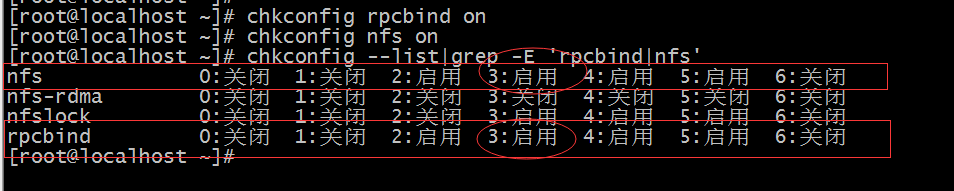


**查看rpc是否获取到了nfs端口**



证明 已经获取了本地nfs服务的端口号。

**设置rpcbind nfs为开机启动项（程序会自动指定开机顺序 无需操心）**



至此 所有服务已经启动了，下一步我们需要配置nfs的配置文件了。

### 7、配置nfs配置文件

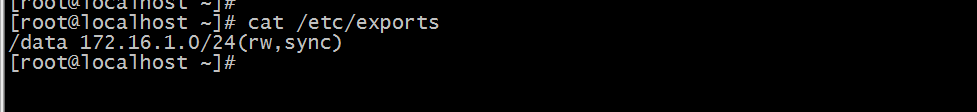
nfs的配置文件是/etc/exports ，默认已经创建，我们无需创建，只需要编辑即可。

vim /etc/exports

目录 网段 动作 要做的操作

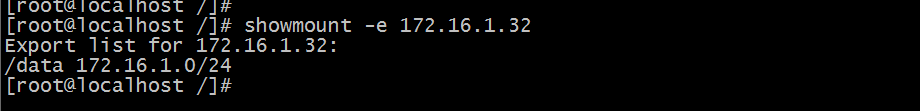
/data 172.16.1.0/24(rw,sync)

这个配置的意义是：共享本地的/data目录，允许子网掩码为172.16.1.0的主机可以访问，这些主机对该目录具有读写权限，以同步（sync）的方式处理。



**查看共享的信息**

showmount -e 172.16.1.31



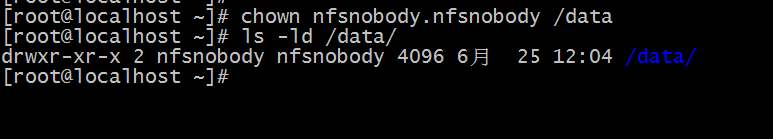
服务端出现这些信息，证明服务端已正确配置。(/data要存在哦！)

**为nfsnobody用户分配文件权限**

nfs与rysnc一样，也需要一个用户去管理服务端nfs服务的执行过程，只不过我们无需手动创建该用户，安装软件的后，程序已经为我们自动创建了，所以，我们只需要为其分配权限就可以了。

因为nfs是由nfsnobody管理的，所以所有的共享目录 nfsnobody 都应该具有读写操作权限，为了方便，我们直接将共享文件的属主 分配给nfsnobdy。

chown nfsnobody.nfsnobody /data



### 8、客户端连接测试

**检查rpcbind是否安装**

一般rpcbind不需要安装，但是centos好像不安装不行，具体没试过，为了不出错，还是安装一下吧。

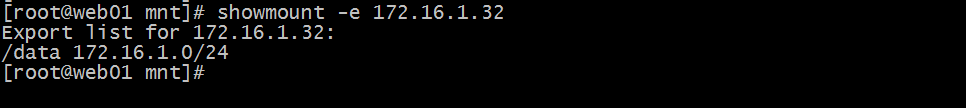
检查是否安装

rpm -qa rpcbind

如果没有安装，在线安装：yum install rpcbind -y

**检查远程文件共享文件主机的文件共享情况。**

Showmount -e 172.16.1.32



证明远程的确是存在的

**客户端挂载远程目录**

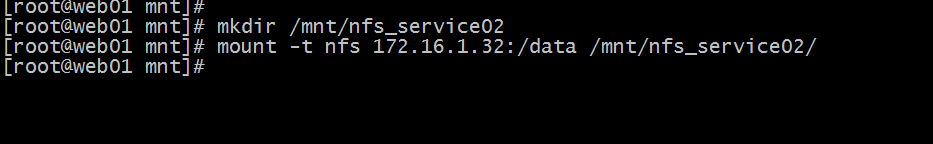
挂载目录可以任意制定，在这里，挂载到/mnt/nfs\_service02/目录下

创建文件夹

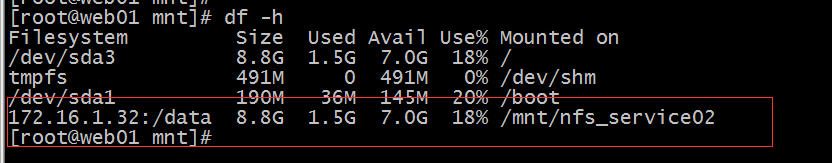
mkdir /mnt/nfs\_service02

挂载远程文件

mount -t nfs 172.16.1.32:/data /mnt/nfs\_service02/



查看挂载的文件系统



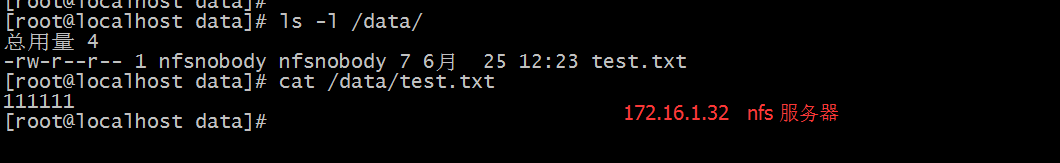
挂载成功！

**测试**

在本地的/mnt/nfs\_service02里面创建一个test.txt文件，里面内容 111111

echo ‘111111’>/mnt/nfs\_service02/test.txt

**服务端查看**



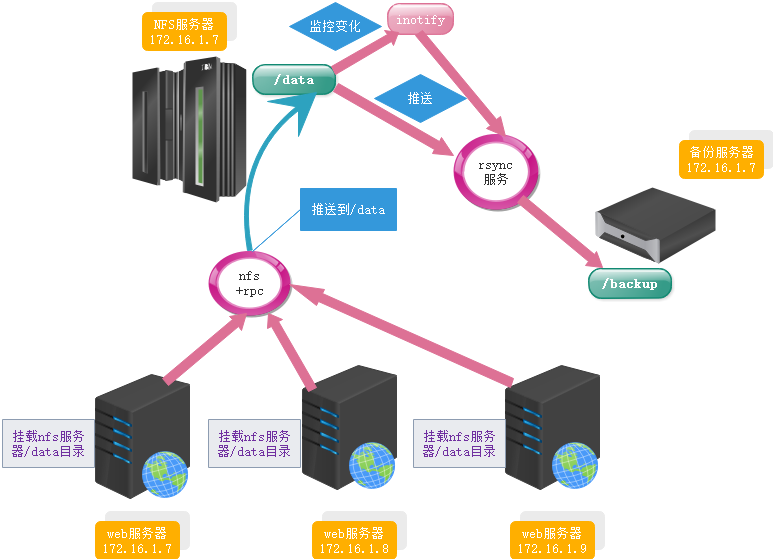
的确是存在的，证明简单的网络共享服务器已搭建成功！

## 四、全网实时同步(rsync+inotify rsync+sersync)

### 1、原理

在一些实际的生产场景中，为了数据安全和镜像同步，需要设置全网实时备份。我们可以通过rsync+inotify的方式实现，也可以通过rsync+sersync实现

其实现基础是在共享服务器的基础上，配置过rsync服务的条件下实现的。



在全网实时备份中，需要有三个里程碑操作

1. 配置rsync服务，使其可以正常推送文件到备份服务器
2. 监控NFS服务器中需要备份的目录变化
3. 当备份目录变化时，推送备份文件到备份服务器

### 配置rsync服务

配置rsync服务需首先配置rsync服务器，然后配置客户端。

操作步骤为:

服务端配置

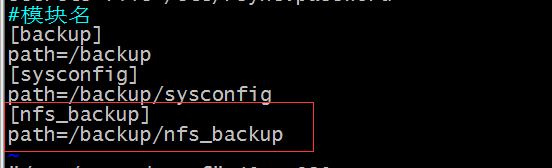
1. 编写/etc/rsyncd.conf配置文件
2. 创建rsync用户
3. 创建目录并更改目录的属主为rsync
4. 创建密码文件，并更改权限为600

客户端配置

1. 创建密码文件，并分配权限为600

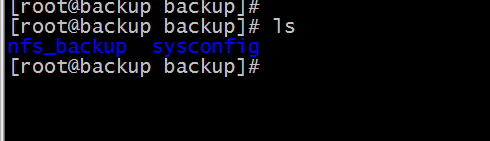
配置完成后可通过rsync命令推送备份文件到备份服务器。上面笔记里面已经配置过了rsync服务器，不再配置，只需新加一些配置即可。

在备份服务器中编辑/etc/rsyncd.config文件。新增模块nfs\_backup模块



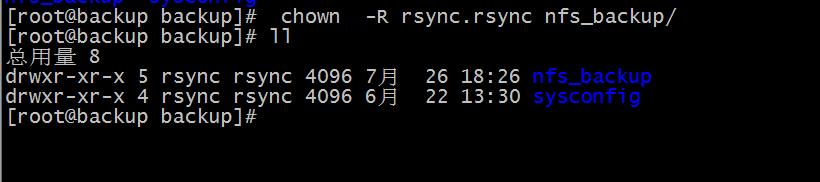
创建/backup/nfs\_backup目录

mkdir -p /backup/nfs\_backup



更改目录的属主

chown -R rsync.rsync nfs\_backup/



密码文件使用已存在的即可。

测试客户端推送

将172.16.1.31服务器上的/data目录推送到172.16.1.41服务器的/backup/nfs\_backup目录下。

rsync -avz /data/\* rsync\_backup@172.16.1.41::nfs\_backup/ --password-file=/etc/passwordService/rsync\_sysconf\_backup.passwd

检查备份服务器下是否存在相应的文件即可！

### 3、监控需要备份的nfs服务器的目录

当要实时备份的目录改变的时候，证明文件已经发生改变，这个时候需要进行远程推送到备份服务器达到备份的目的。因此我们需要监控目录的变化，我们可以通过使用inotyfy+rsync的方式，也可以通过使用sersync的方式。在这里，我们使用inotify,之后使用sersync的方式实现。

使用inotify 需要安装inotify-tools

检查inotify是否安装

Rpm -qa inotify-tools

如果没有安装，安装inotify-tools工具

第一步：wget -O /etc/yum.repos.d/epel.repo http:mirrors.aliyun.com/repo/epel-6.repo（下载安装包到本地）

第二步：yum install inotify-tools -y（安装）

安装完inotify，实际上也就是安装了两个命令

Inotifywait和inotifywatch

#### Inotifywait –help 查看帮助

语法：  
inotifywait [-hcmrq] [-e ] [-t ] [--format ] [--timefmt ] [ ... ]  
参数：  
**-h,–help**  
输出帮助信息  
**@**  
排除不需要监视的文件，可以是相对路径，也可以是绝对路径。  
**–fromfile**  
从文件读取需要监视的文件或排除的文件，一个文件一行，排除的文件以@开头。  
**-m, –monitor**  
接收到一个事情而不退出，无限期地执行。默认的行为是接收到一个事情后立即退出。  
**-d, –daemon**  
跟–monitor一样，除了是在后台运行，需要指定–outfile把事情输出到一个文件。也意味着使用了–syslog。  
**-o, –outfile**  
输出事情到一个文件而不是标准输出。  
**-s, –syslog**  
输出错误信息到系统日志  
**-r, –recursive**  
监视一个目录下的所有子目录。  
**-q, –quiet**  
指定一次，不会输出详细信息，指定二次，除了致命错误，不会输出任何信息。  
**–exclude**  
正则匹配需要排除的文件，大小写敏感。  
**–excludei**  
正则匹配需要排除的文件，忽略大小写。  
**-t , –timeout**  
设置超时时间，如果为0，则无限期地执行下去。  
**-e , –event**  
指定监视的事件。  
**-c, –csv**  
输出csv格式。  
**–timefmt**  
指定时间格式，用于–format选项中的%T格式。  
**–format**  
指定输出格式。  
%w 表示发生事件的目录  
%f 表示发生事件的文件  
%e 表示发生的事件  
%Xe 事件以“X”分隔  
%T 使用由–timefmt定义的时间格式

inotifywatch

语法：  
inotifywatch [-hvzrqf] [-e ] [-t ] [-a ] [-d ] [ ... ]  
参数：  
**-h, –help**  
输出帮助信息  
**-v, –verbose**  
输出详细信息  
**@**  
排除不需要监视的文件，可以是相对路径，也可以是绝对路径。  
**–fromfile**  
从文件读取需要监视的文件或排除的文件，一个文件一行，排除的文件以@开头。  
**-z, –zero**  
输出表格的行和列，即使元素为空  
**–exclude**  
正则匹配需要排除的文件，大小写敏感。  
**–excludei**  
正则匹配需要排除的文件，忽略大小写。  
**-r, –recursive**  
监视一个目录下的所有子目录。  
**-t , –timeout**  
设置超时时间  
**-e , –event**  
只监听指定的事件。  
**-a , –ascending**  
以指定事件升序排列。  
**-d , –descending**  
以指定事件降序排列。

可监听事件

|  |  |
| --- | --- |
| access | 文件读取 |
| modify | 文件更改。 |
| attrib | 文件属性更改，如权限，时间戳等。 |
| close\_write | 以可写模式打开的文件被关闭，不代表此文件一定已经写入数据。 |
| close\_nowrite | 以只读模式打开的文件被关闭。 |
| close | 文件被关闭，不管它是如何打开的。 |
| open | 文件打开。 |
| moved\_to | 一个文件或目录移动到监听的目录，即使是在同一目录内移动，此事件也触发。 |
| moved\_from | 一个文件或目录移出监听的目录，即使是在同一目录内移动，此事件也触发。 |
| move | 包括moved\_to和 moved\_from |
| move\_self | 文件或目录被移除，之后不再监听此文件或目录。 |
| create | 文件或目录创建 |
| delete | 文件或目录删除 |
| delete\_self | 文件或目录移除，之后不再监听此文件或目录 |
| unmount | 文件系统取消挂载，之后不再监听此文件系统。 |

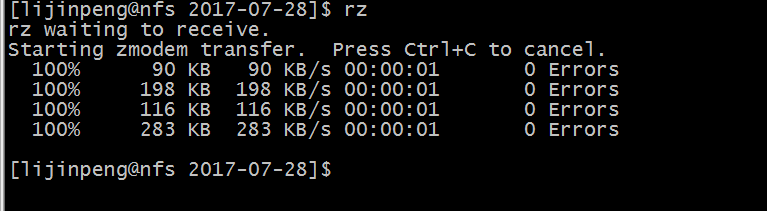
#### 测试

监控nfs服务器中的 /data目录的创建与删除

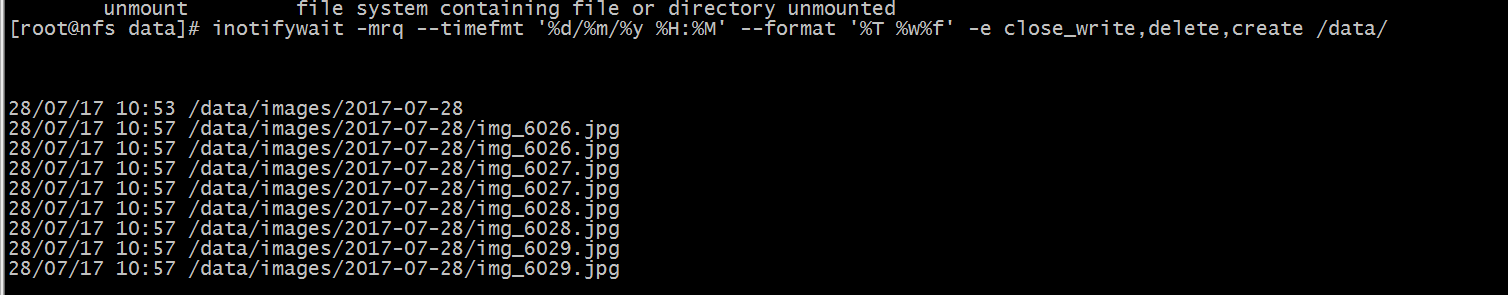
inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%T %w%f' -e close\_write,delete,create

在/data目录下 上传图片

上传 rz



查看变化



证明监控成功！

我们也可以通过watch ls 查看目录的变化

### 4、推送文件的变化到备份服务。

一般使用一个脚本做一个死循环去监控 inotify.sh

脚本内容如下

Path=/data

Ip=172.16.1.41

#watch /data dir change

inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%T %w%f' -e close\_write,delete,create $Path \

|while read file

do

cd $Path && \

rsync -avz /data/\* --delete rsync\_backup@$Ip::nfs\_backup/ --password-file=/root/backup/backup.password

done

执行脚本bash inotify.sh 对目录去做监控。查看变化即可

### 5、Sersync全网实时备份

#### 原理

##### 01、为什么要用rsync+sersync架构？

1、sersync是基于inotify开发的，类似于inotify-tools的工具

2、sersync可以记录下被监听目录中发生变化的（包括增加、删除、修改）具体某一个文件或者某一个目录的名字，然后使用rsync同步的时候，只同步发生变化的文件或者目录

##### 02、rsync+inotify-tools与rsync+sersync架构的区别？

1、rsync+inotify-tools

a、inotify只能记录下被监听的目录发生了变化（增，删，改）并没有把具体是哪个文件或者哪个目录发生了变化记录下来；

b、rsync在同步的时候，并不知道具体是哪个文件或目录发生了变化，每次都是对整个目录进行同步，当数据量很大时，整个目录同步非常耗时（rsync要对整个目录遍历查找对比文件），因此效率很低

2、rsync+sersync

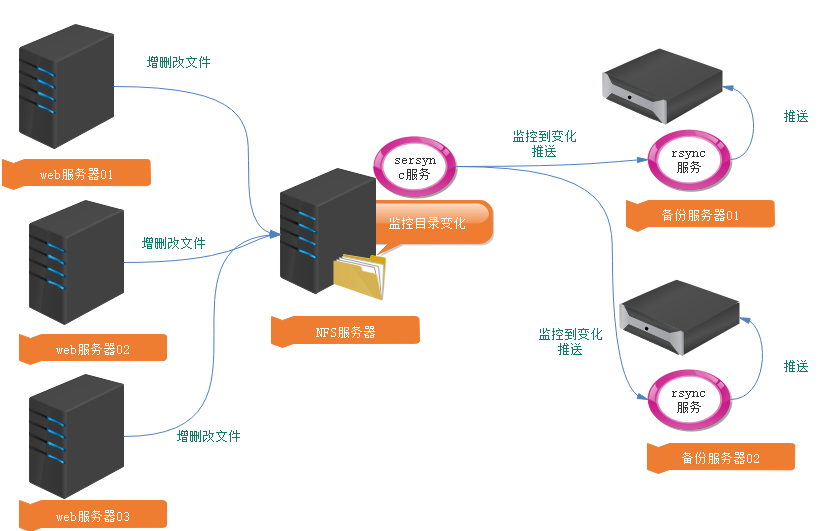
a、sersync可以记录被监听目录中发生变化的（增，删，改）具体某个文件或目录的名字；

b、rsync在同步时，只同步发生变化的文件或目录（每次发生变化的数据相对整个同步目录数据来说很小，rsync在遍历查找对比文件时，速度很快），因此效率很高。

总结： 当同步的目录数据量不大时，建议使用rsync+inotify

当同步的目录数据量很大时（几百G甚至1T以上）文件很多时，建议使用rsync+sersync

##### 03、Sersync同步逻辑



**原理步骤：**

1. 在NFS服务器上开启sersync服务，sersync负载监控配置路径中的文件系统事件变化；

2. 调用rsync命令把更新的文件同步到备份服务器（01 和 02）；

3. 需要在NFS服务器配置sersync，在备份服务器配置rsync server（注意：是rsync服务）

同步原理：

1. web服务器实时的往sersync服务器上写入更新文件数据；

2. 此时需要在NFS服务器上配置sersync服务；

3. 在备份服务器（01和02）上开启rsync守护进程服务，以同步接受NFS服务器推送过来的备份文件

通过rsync的守护进程服务后可以发现，实际上sersync就是监控本地的数据写入或更新事件；然后，在调用rsync客户端的命令，将写入或更新事件对应的文件通过rsync推送到备份服务器（01 和02），如此简单；

#### 实操

###### 1、配置rysnc服务（在备份服务器上操作！）

使用sersync同样也许要先提供rsync服务 简单配置如下

**服务端配置**

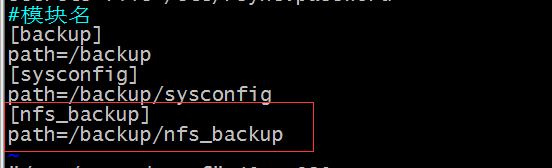
1. 编写/etc/rsyncd.conf配置文件
2. 创建rsync用户
3. 创建目录并更改目录的属主为rsync
4. 创建密码文件，并更改权限为600

**客户端配置**

1. 创建密码文件，并分配权限为600

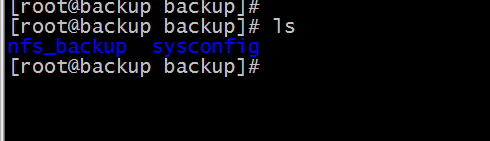
配置完成后可通过rsync命令推送备份文件到备份服务器。上面笔记里面已经配置过了rsync服务器，不再配置，只需新加一些配置即可。

在备份服务器中编辑/etc/rsyncd.config文件。新增模块nfs\_backup模块



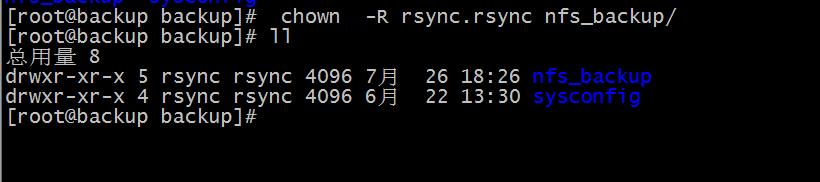
创建/backup/nfs\_backup目录

mkdir -p /backup/nfs\_backup



更改目录的属主

chown -R rsync.rsync nfs\_backup/



密码文件使用已存在的即可。

测试客户端推送

将172.16.1.31服务器上的/data目录推送到172.16.1.41服务器的/backup/nfs\_backup目录下。

rsync -avz /data/\* rsync\_backup@172.16.1.41::nfs\_backup/ --password-file=/etc/passwordService/rsync\_sysconf\_backup.passwd

检查备份服务器下是否存在相应的文件即可！

配置rsync服务有几个要点要注意 sersync服务需要用到的

模块名：nfs\_backup （备份服务器配置文件中）

校验用户名：rsync\_backup（备份服务器配置文件中）

密码文件路径：/etc/passwordService/rsync\_sysconf\_backup.passwd（共享服务器的密码路径 截图未展示！）

###### 2、安装sersync服务

下载sersync的安装包。获取路径

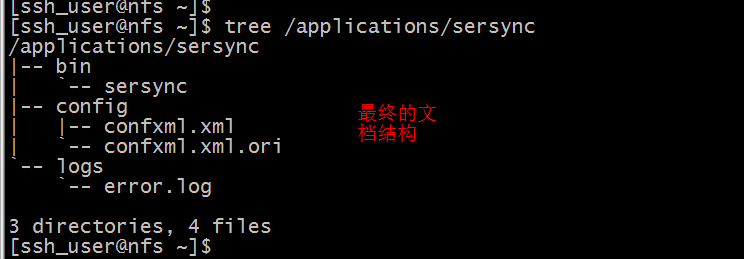
###### 3、配置sersync

解压安装包后，新建文件夹

mkdir -p /applications/sersync

在/applications/sersync/下新建config bin logs文件夹

将安装包中的sersync 拷贝到 bin 中，将config.xml 拷贝到config下。



编辑config.xml配置文件

<?xml version="1.0" encoding="ISO-8859-1"?>

<head version="2.5">

<host hostip="localhost" port="8008"></host>

<debug start="false"/>

<fileSystem xfs="false"/>

<filter start="false">

<exclude expression="(.\*)\.svn"></exclude>

<exclude expression="(.\*)\.gz"></exclude>

<exclude expression="^info/\*"></exclude>

<exclude expression="^static/\*"></exclude>

</filter>

<inotify>

<delete start="true"/>

<createFolder start="true"/>

<createFile start="false"/>

<closeWrite start="true"/>

<moveFrom start="true"/>

<moveTo start="true"/>

<attrib start="false"/>

<modify start="false"/>

</inotify>

<sersync>

<localpath watch="/data">

<remote ip="172.16.1.41" name="nfs\_backup"/>

<?xml version="1.0" encoding="ISO-8859-1"?>

<head version="2.5">

<host hostip="localhost" port="8008"></host>

<debug start="false"/>

<fileSystem xfs="false"/>

<filter start="false">

<exclude expression="(.\*)\.svn"></exclude>

<exclude expression="(.\*)\.gz"></exclude>

<exclude expression="^info/\*"></exclude>

<exclude expression="^static/\*"></exclude>

</filter>

<inotify>

<delete start="true"/>

<createFolder start="true"/>

<createFile start="false"/>

<closeWrite start="true"/>

<moveFrom start="true"/>

<moveTo start="true"/>

<attrib start="false"/>

<modify start="false"/>

</inotify>

<sersync>

<localpath watch="/data">

<remote ip="172.16.1.41" name="nfs\_backup"/>

<!--<remote ip="192.168.8.39" name="tongbu"/>-->

<!--<remote ip="192.168.8.40" name="tongbu"/>-->

</localpath>

<rsync>

<commonParams params="-avz"/>

<auth start="true" users="rsync\_backup" passwordfile="/etc/passwordService/rsync\_sysconf\_backup.passwd"/>

<userDefinedPort start="false" port="874"/><!-- port=874 -->

<timeout start="true" time="100"/><!-- timeout=100 -->

<ssh start="false"/>

</rsync>

<failLog path="/applications/sersync/logs/error.log" timeToExecute="60"/><!--default every 60mins execute once-->

<crontab start="false" schedule="600"><!--600mins-->

<crontabfilter start="false">

<exclude expression="\*.php"></exclude>

<exclude expression="info/\*"></exclude>

</crontabfilter>

</crontab>

<plugin start="false" name="command"/>

</sersync>

<plugin name="command">

<param prefix="/bin/sh" suffix="" ignoreError="true"/> <!--prefix /opt/tongbu/mmm.sh suffix-->

<filter start="false">

<include expression="(.\*)\.php"/>

<include expression="(.\*)\.sh"/>

</filter>

</plugin>

<plugin name="socket">

<localpath watch="/opt/tongbu">

<deshost ip="192.168.138.20" port="8009"/>

</localpath>

</plugin>

<plugin name="refreshCDN">

<localpath watch="/data0/htdocs/cms.xoyo.com/site/">

<cdninfo domainname="ccms.chinacache.com" port="80" username="xxxx" passwd="xxxx"/>

<sendurl base="http://pic.xoyo.com/cms"/>

<regexurl regex="false" match="cms.xoyo.com/site([/a-zA-Z0-9]\*).xoyo.com/images"/>

</localpath>

</plugin>

</head>

参数说明：

<localpath watch="/data">：#源服务器同步目录(共享服务器的目录 要监控的目录)

remote ip="172.16.1.41":  #备份服务器的地址

name="nfs\_backup”:rsync同步目录模块名称

users="rsync\_backup”：要同步的模块名（同步服务器配置文件中）

passwordfile="/etc/passwordService/rsync\_sysconf\_backup.passwd "： #目标服务器rsync同步用户的密码在源服务器的存放路径（共享服务器上的密码文件）

failLog path="/tmp/rsync\_fail\_log.sh"  #脚本运行失败日志记录

start="true"  #设置为true，每隔600分钟执行一次全盘同步

###### 4、参数说明

|  |  |
| --- | --- |
| **Sersync参数** | **说明** |
| **./sersync -r** | -r参数作用是：开启实时监控的之前对主服务器目录与远程目标机器的目录进行一次整体同步；如果需要将sersync运行前，主服务器目录下已经存在的所有文件或目录全部同步到远端，则要以 -r参数运行sersync，将本地与远程整体同步一次；  提别说明：如果设置了过滤器，即在xml文件中，filter为true，则暂时不能使用-r参数进行整体同步； |
| **./sersync -o xx.xml** | 不指定 -o参数： sersync使用sersync可执行文件目录下的默认配置文件confxml.xml  指定 -o 参数：可以指定多个不同的配置文件，从而实现sersync多进程多实例的数据同步 |
| **./sersync -n num** | -n参数为：指定默认的线程池的线程总数；  例如： ./sersync -n 5 则指定线程总数为5，如果不指定，默认启动线程池数量是10，如果cpu使用过高，可以通过该参数调低，如果机器配置较高，可以调高默认的线程总数，提升同步效率； |
| **./sersync -d** | -d参数为：后台服务，通常情况下使用 -r参数对本地到远端整体同步一遍后，在后台运行此参数启动守护进程实时同步；在第一次整体同步时，-d 和 -r参数经常会联合使用； |
| **./sersync -m**  **pluginName** | -m参数：不进行同步，只运行插件 ./sersync -m pluginName  例如：./sersync -m command，则在监控到事件后，不对远程目标服务器进行同步，而是直接运行command插件 |
| 组合命令使用说明： | |
| **-n 8 -o liubl.xml -r -d** | 多个参数可以配合使用，例如：./sersync -n 16 -o config.xml -r -d 表示设置线程池工作线程为16个，指定liubl.xml作为配置文件，在实时监控前 做一次整体同步，以守护进程方式在后台运行； |
| **./sersync --help** | 很遗憾，它没有查看帮助（需要的话2条路，要么看源代码，要么自测求验证） |

###### 5、测试

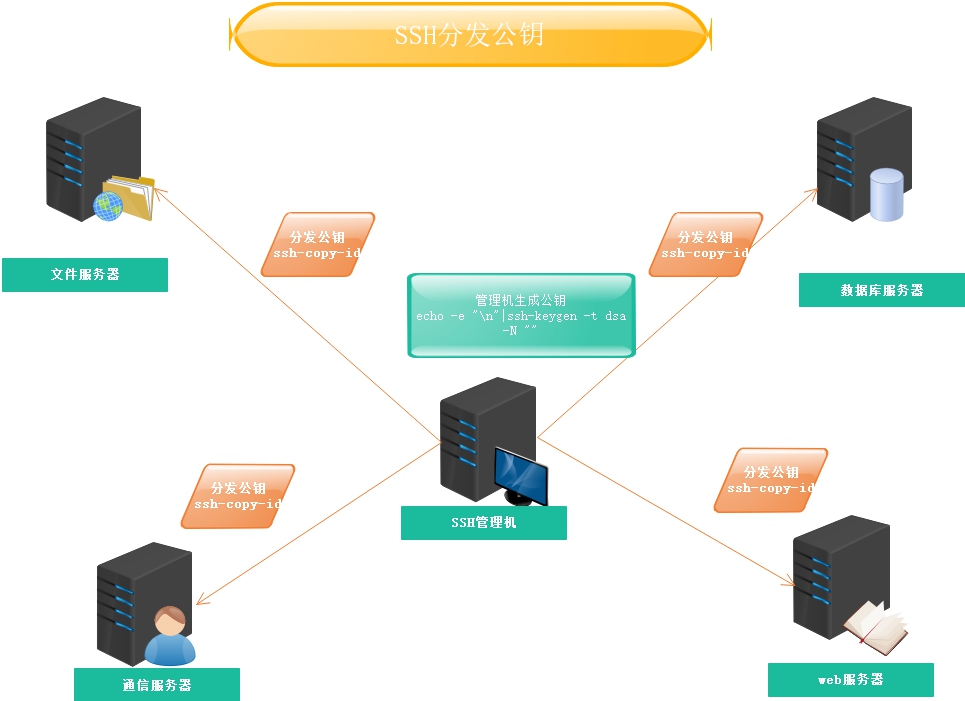
sersync -r -d -o /applications/sersync/config/confxml.xml

此时已开启监控！ 做备份测试即可！

## 五、SSH管理机的搭建

### 1、为什么要使用管理机

在生产环境中，当我们需要对N台服务器进行统一的管理，比如目录的创建、软件的安装等，或者根据需求登陆某一台服务器的时候，我们不可能一台一台的去登陆服务器，因此我们会使用一台机器作为堡垒机，去连接众多的机器，进行统一的管理。传统的ftp 方式登陆，用户名密码会以明文的方式在网络上传输，很不安全，因此我们现在一般使用SSH 的方式去连接 。 SSH有两种登陆方式，一种是用户名/密码的方式，另一种是密钥（公钥的方式）。SSH 是基于RSA非对称加密，因此数据传递过程 会比对公钥，如果比对上 才会接受处理，因此很安全！



### 2、SSH用户名密码登陆

#### 01、环境准备

管理机：172.16.1.61

被管理服务器：172.16.1.7

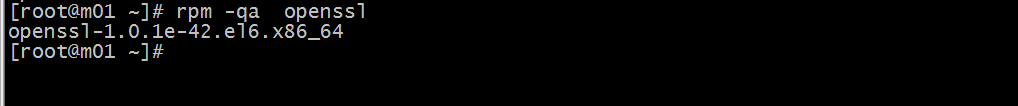
172.16.1.8

172.16.1.31

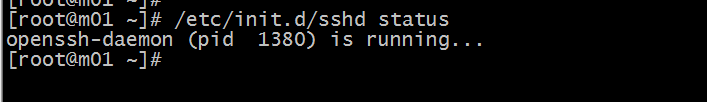
172.16.1.41

目前采用的比较多的都是ssh登陆，无论我们使用CRT 还是Xsheel等都是采用SSH的方式进行登录的，SSH登陆默认端口是22，需要启用的服务是sshd服务器。因此我们需要检查机器是否安装了sshd 并且服务是否启用。

检查是否安装：rpm -qa openssl



检查服务是否开启：/etc/init.d/sshd status



以正确安装ssl 并且服务已正确开启！

#### 02、配置ssh

如果需要修改SSH的一些初始化设置，比如修改端口，指定机器可以访问，我们修改/etc/ssh/sshd\_config.conf文件就可以了。（在被管理的机器上修改！）

编辑/etc/ssh/ssh\_config

Port=52113

ListenAdress 172.16.1.61:52113 暂时不配置

Protocol 2

PermitRootLogin no

PermitEmptypasswords no 不允许空密码登陆

#GSSAPIsTRUCAutjentication no 影响连接速度

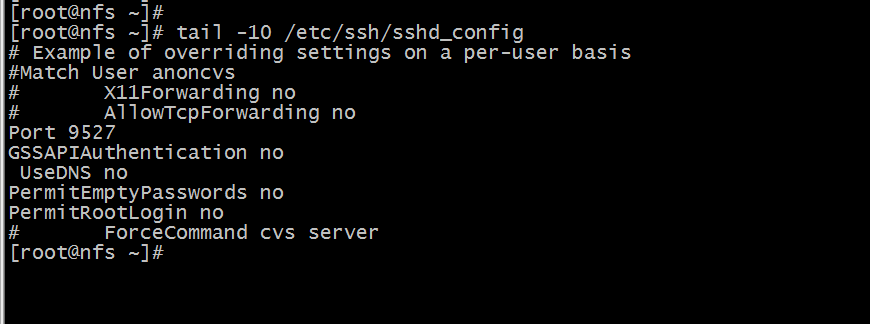
UseDNS no 影响连接速度

我们可以使用sed 命令 批量修改！

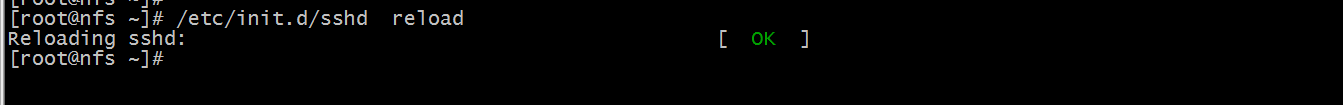
sed -i '138i Port 9527 \nGSSAPIAuthentication no\n UseDNS no\nPermitEmptyPasswords no\nPermitRootLogin no' /etc/ssh/sshd\_config

至此sshd基本已经修改完成，我们的端口号换成了9527，并且不允许root用户登陆！

检查是否生效 tail -10 /etc/ssh/sshd\_config



平滑重启sshd服务器 /etc/init.d/reload



服务已经准备完毕！

#### 03创建用户

在ssh配置中，我们已经禁用了根用户的登陆，因此我们需要创建一个用户，用于管理机登陆。

创建ssh\_user用户：useradd ssh\_user

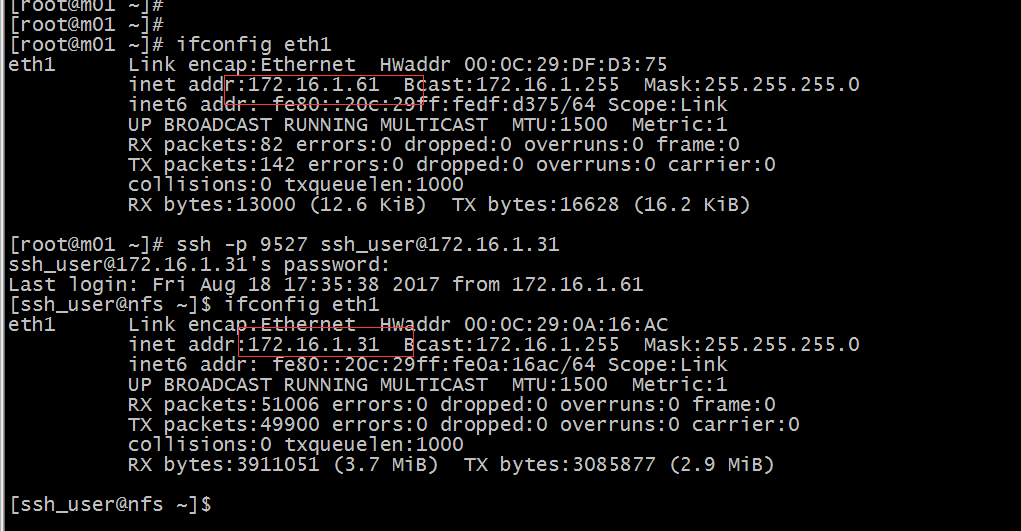
设置密码:echo ‘123456’|passwd –stdin ssh\_user

#### 04、管理机远程登陆服务器

登陆命令: ssh -p port username@ip

我们用172.16.1.61（管理机） 登陆 172.16.1.31(nfs服务器)

ssh -p 9527 [ssh\_user@172.16.1.31](mailto:ssh_user@172.16.1.31)



这就证明 登陆成功了！我们可以在管理机上对 NFS服务器进行相关操作了！

05、ssh相关命令s

scp可以实现文件的远程拷贝

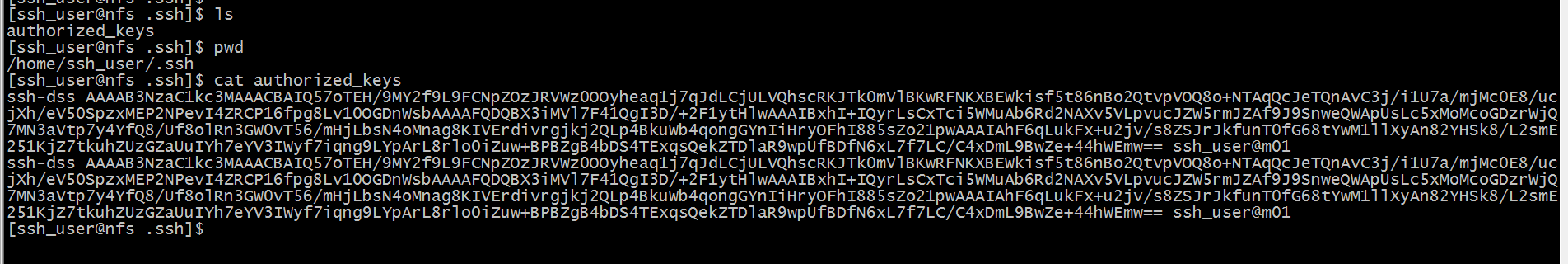
scp -pPort 本地目录 用户@ip 远程目录

sftp -oPort=端口 用户@ip 连接远程ftp

### 3、SSH密钥登陆

由于密码登陆的时候，我们需要执行一些批量任务的时候，每次都输入密码，这样就给我们管理带来了一定的不便捷性，而使用公钥的方式进行登陆，登录过程会验证本地是否存在该机器的公钥 如果存在则连接成功，否则连接失败。

公钥保存的位置在服务器用户目录下的.ssh目录下 .ssh/ authorized\_keys

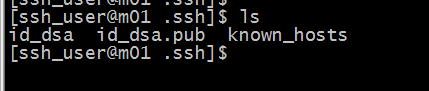


#### 01、生成公钥

使用ssh 的相关命令：ssh-keygen

ssh-keygen -t dsa 一路回车即可

查看生成的密钥: ls .ssh/



Id\_dsa.pub就是我们要分发到被管理机器上的公钥

上面生成公钥的方式 需要一直回车，当我们使用脚本安装的时候，很麻烦，所以我们使用非交互是的命令。

1、ssh-keygen -t dsa -P '' -f ~/.ssh/id\_dsa>/dev/null 2>&!（暂时有点问题）

2、 echo -e "\n"|ssh-keygen -t dsa -N ""

#### 02、分发公钥

经过步骤01后，在管理机上已经创建了密钥对，接下来我们需要将公钥发送其他机器上，这样连接的时候，才可以根据公钥去匹配身份，通过验证！

使用ssh命令: ssh-copy-id

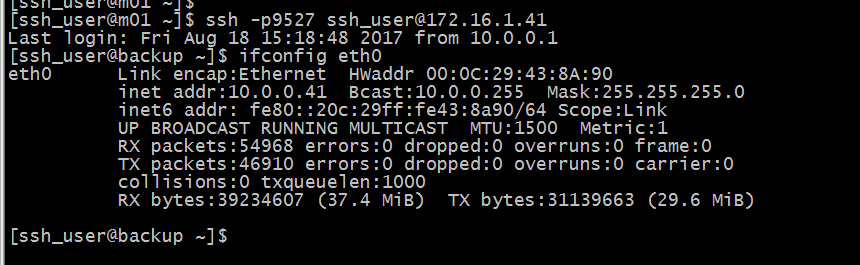
**不指定端口**: ssh-copy-id -i .ssh/id\_dsa.pub [oldgril@172.16.1.41](mailto:oldgril@172.16.1.41)

**指定端口**: ssh-copy-id -i .ssh/id\_dsa.pub "-p9527 ssh\_user@172.16.1.41"

去其他机器的 ssh\_user用户目录下检查 是否存在 id\_dsa.pub 公钥！

#### 03、测试连接

经过上面两个步骤，我们在各个机器上的环境已经搭建完成，接下来 我们可以直接测试连接，如果不需要密码，则证明已正确配置！



如截图所示，密钥登陆已经正确配置完成！

### 4、企业实现SSH方案

#### 01、SSH连接问题

在企业生产案例中，无论是使用用户名/密码的登陆方式，还是使用密钥的登陆方式，都有一个权限的问题需要去考虑，就是我们在连接的时候，使用什么用户登陆，这个用户具有何种权限，如果分配权限太大，某些人的不当操作就对生产环境会造成潜在的巨大灾难，给的权限太小，又发挥不了其作用，所以我们需要一些解决方法。

#### 02、SSH连接的权限分配方案

针对这些问题，大概有以下三种解决方案：

##### （1）root登陆

直接使用root进行ssh登陆，s这种操作对一些小的系统，不给除运维以外的其他人员使用的系统，还是非常简单的，但是安全性是最低的，一般没人使用！

##### （2）分配sid

如果我们想让某个命令以root的权限去执行，我们可以给这个命令加sid去执行。比如，我们想通过ssh\_user用户，去挂载共享服务器nfs的共享目录，当我们使用的时候，肯定没有这个权限的。但当我们为这个命令添加sid后，其他用户执行该命令后，便可以以root用户的身份去执行该命令!

##### （3）sudo提权

我们可以使用root用户编辑/etc/sudoer文件，sudo进行提权。使某个用户具有执行或者查看某项文件的权限。比如我们可以对ssh\_user用户进行sudo提权，使其可以使用rsync命令，并且在使用过程中不需要使用密码。

例如：

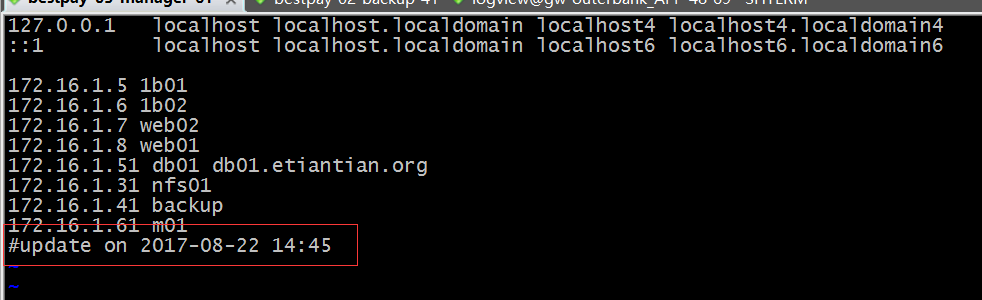
当我们没有为ssh\_user用户sudo提权的时候，

1. 我们先将管理机的hosts文件拷贝到当前用户目录下

cp /etc/hosts ~/hosts

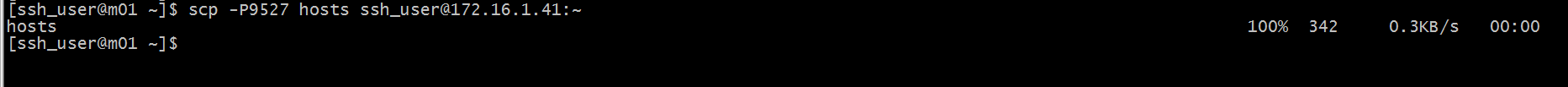
编辑hosts,对文件做一个更新日期的标识

vim ~/hosts

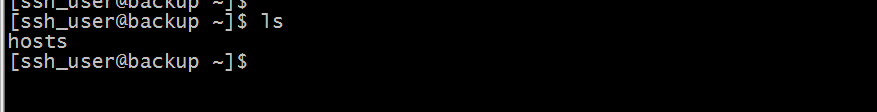


1. 然后将当前用户目录下的hosts文件通过scp推送到备份服务器上（172.16.1.41 backup）

scp -P9527 hosts [ssh\_user@172.16.1.41:~](mailto:ssh_user@172.16.1.41:~)

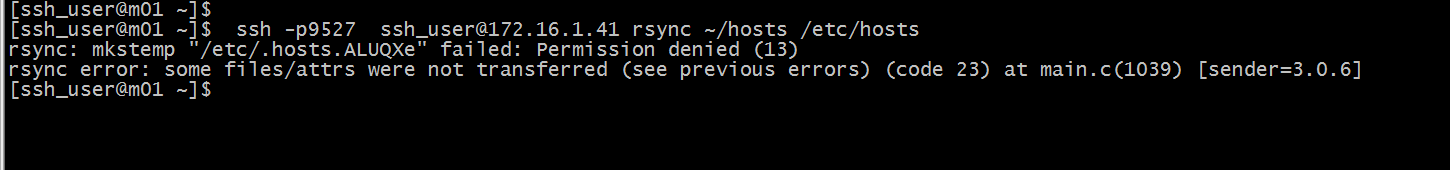


检查backup服务器的ssh\_user目录下是否存在



1. 在ssh模式下，使用rsync命令将备份服务器ssh\_user用户目录下的hosts推送到/ect/hosts

ssh -p9527 ssh\_user@172.16.1.41 rsync ~/hosts /etc/hosts



此时ssh\_user用户是没有rsync的权限的，我们需要添加sudo权限！

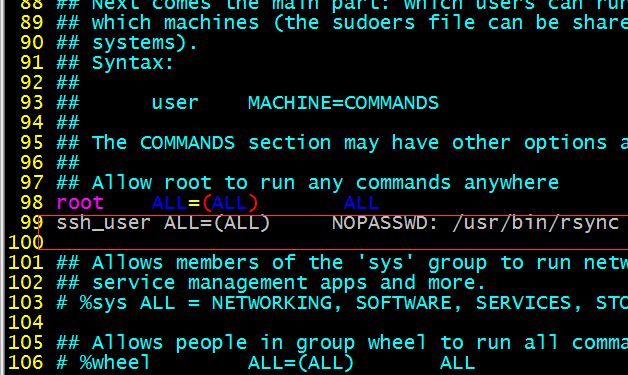
4、 编辑/etc/sudoer文件，添加ssh\_user用户，为其分配rsync的权限.。

编辑/etc/sudoer文件一定要使用root用户，否则没有权限执行。

切换用户：su – root

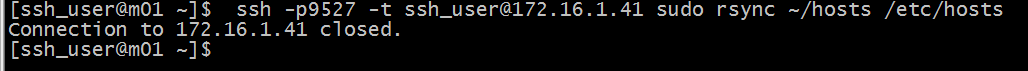
输入密码即可。

vim /etc/sudoers

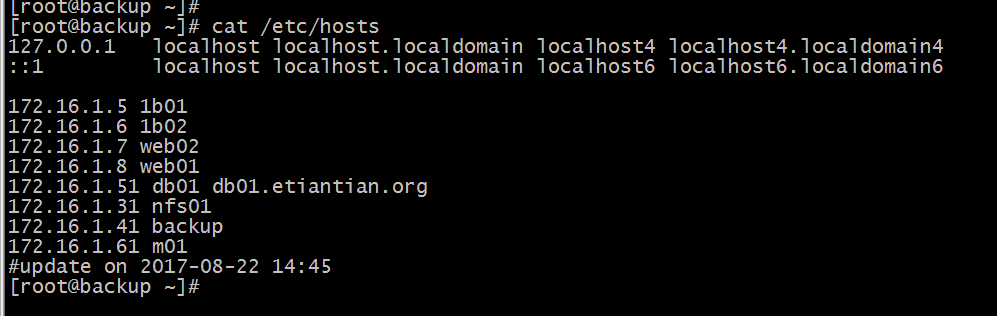


wq!并保存即可！

5、再次在ssh模式下通过rsync推送hosts文件到/etc/下

ssh -p9527 -t ssh\_user@172.16.1.41 sudo rsync ~/hosts /etc/hosts

在备份服务器上检查是否正确的他推送



通过sudo提权可以在正常的解决问题！

#### 03、总结

在生产环境中，，我们一般不会使用第有一种方式，因为极度不安全。。

通常采用分用户的管理方式，针对一些常用而且相对安全的命令，我们一般可以直接对该命令+sid的方式。

针对一些复杂的生产环境，每个部门都有其特定的智能，这样就需要为他们分配不同的用户，并且拥有不同的权限，所以sudo提权的方式就很实用，而关于如何规划命令，这是有一个应该认真考虑的问题。

### 5、export分发密钥

由于我们在真实的环境中，有成百上千台服务器需要统一管理，所以我们不可能一台台的去登陆，每次分发的时候都手动输入密码，因此需要一个脚本去管理，而我们可以通过export去分发。export可以根据用户提示自动填充输入美容，如果要求输入密码的时候，我们把密码放入到脚本中，直接放到输入域，就就可以自动的完成，我们也就不需要一次次的输入密码了。

##### 检查 安装export

检查是否安装export

rpm -qa export

安装export

yum install export -y

##### 编写自动分发脚本

vim ssh\_export.sh



先判断输入的参数个数是否为2，

第一个参数为要拷贝的文件名，第二个参数是 拷贝的路径

#！/usr/bin/expect

if { $argc !=2 } {

send\_user "usage:in the command ssh-expect ,you must user two paramter!\n"

exit

}

#define var

set file [lindex $argv 0]

set host [lindex $argv 1]

set password "123456"

spawn ssh-copy-id -i $file "-p 9527 ssh\_user@$host"

expect {

"yes/no" {send "yes\r";exp\_continue}

"\*password" {send "$password\r"}

}

expect eof

exit -onexit {

send\_user "this command is end! ssh\_expect\_user say byte to you \n"

}

#script usage

#expect oldboy-6.exp file host dir

#example

#expect fenfa.exp ~/hosts 10.0.0.41:~

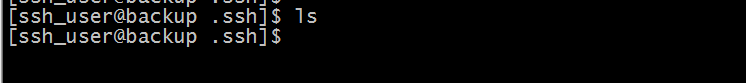
**使用方式：expect fenfa.exp ~/hosts 10.0.0.41:~**

##### 03、测试 自动分发密钥（不需要手动输入密码）

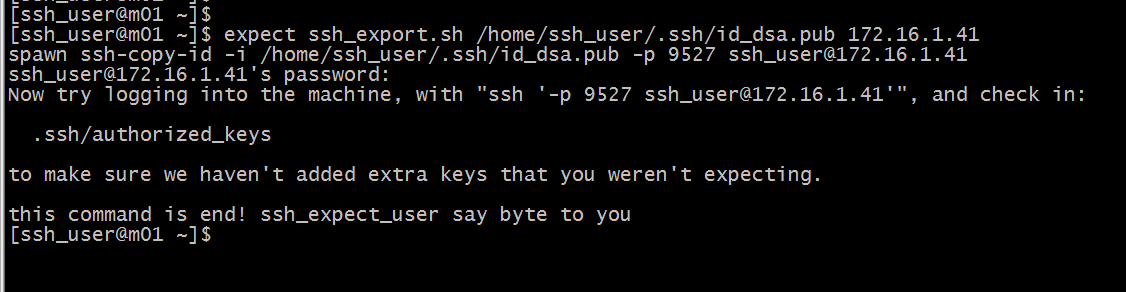
测试推送公钥文件

将管理机器上当前用户目录下的.ssh/id\_dsa.pub文件推送到172.16.1.41（backup）备份服务器的当前用户目录的.ssh木目录下

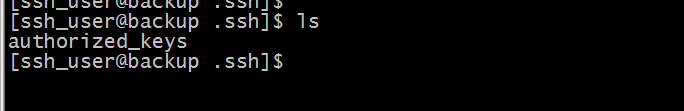
检查备份服务器.ssh目录



管理机分发公钥到备份服务器

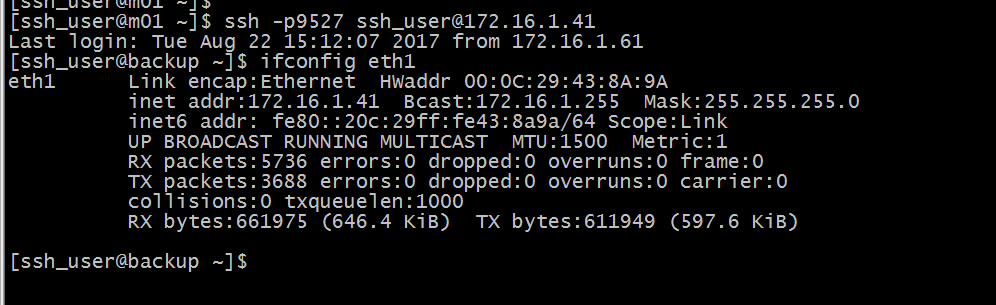


再次检查备份服务器.ssh目录



公钥已存在！

SSH登陆测试是否还需要密钥

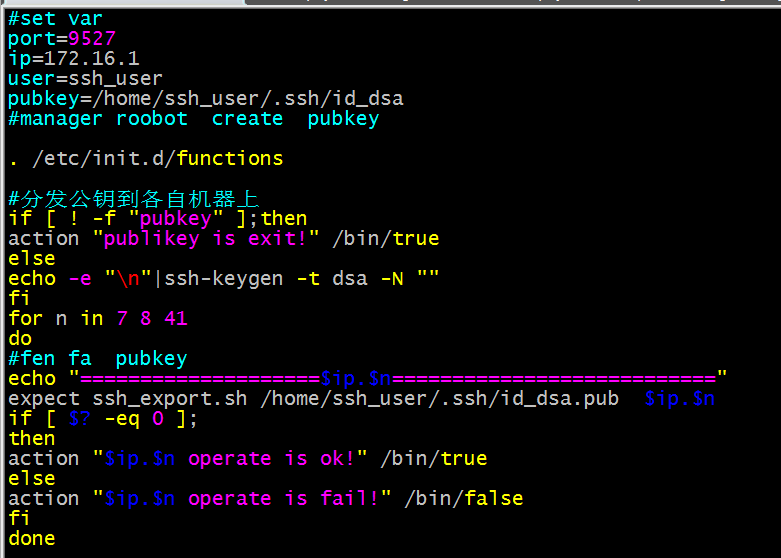


分发已经成功！

##### 04、脚本实现批量分发

先清除所有机器上当前用户.ssh下的公钥记录！ rm -f .ssh/\*

批量分发公钥（ssh04.sh），我们需要编写一个脚本，脚本的内容可分为两个部分：



1、在管理机器上生成公钥（非交互式）

2、使用export实现非交互式分发公钥（非交互式）

脚本:

#set var

port=9527

ip=172.16.1

user=ssh\_user

pubkey=/home/ssh\_user/.ssh/id\_dsa

#manager roobot create pubkey

. /etc/init.d/functions

#分发公钥到各自机器上

if [ ! -f "pubkey" ];then

action "publikey is exit!" /bin/true

else

echo -e "\n"|ssh-keygen -t dsa -N "" #生成公钥

fi

for n in 7 8 41

do

#fen fa pubkey

echo "=================$ip.$n====================="

expect ssh\_export.sh /home/ssh\_user/.ssh/id\_dsa.pub $ip.$n #执行分发脚本

if [ $? -eq 0 ];

then

action "$ip.$n operate is ok!" /bin/true

else

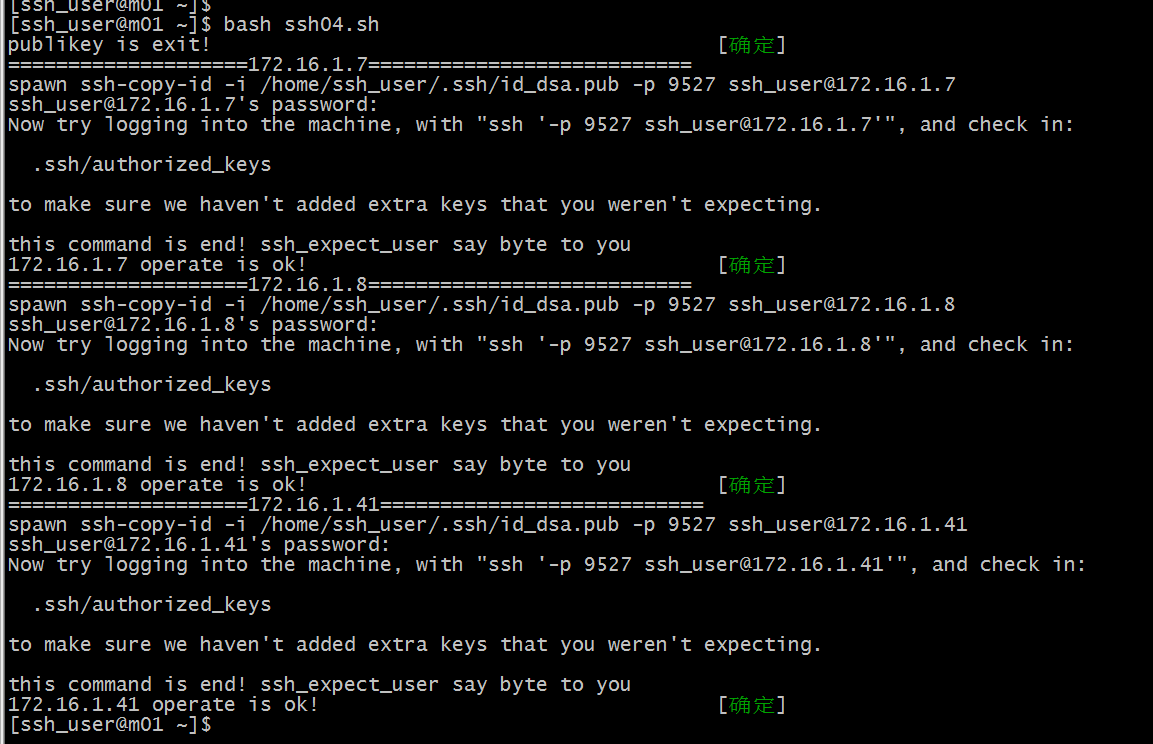
action "$ip.$n operate is fail!" /bin/false

fi

done

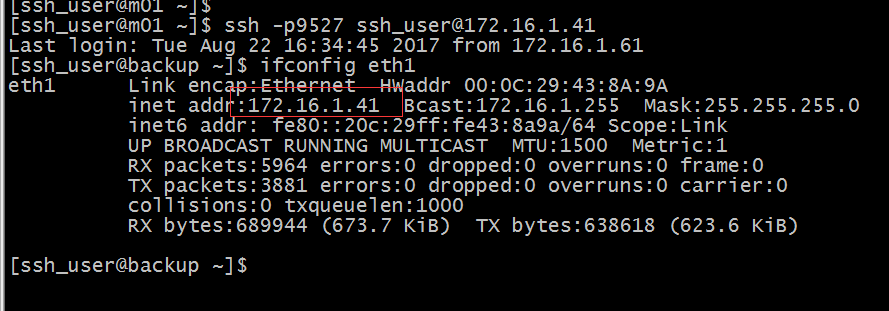
**测试脚本**

bash ssh04.sh



脚本执行成功！

**测试连接**



由测试看来，并不需要密码，证明分发脚本是成功的，至此通过export实现批量分发，已经成功!

### 6、saltStack自动化运维工具

##### 01、saltStack简介

Saltstack是一个新的基础设施管理工具。目前处于快速发展阶段，可以看做是pssh+弱化的Puppet的组合。间接的反映出了saltstack的两大功能：远程执行和配置管理。

Saltstack使用Python开发，是一个非常简单易用和轻量级的管理工具。由Master和Minion构成，通过ZeroMQ进行通信。

Saltstack的master端监听4505与4506端口，4505为salt的消息发布系统，4506为salt客户端与服务端通信的端口；salt客户端程序不监听端口，客户端启动后，会主动连接master端注册，然后一直保持该TCP连接，master通过这条TCP连接对客户端控制，如果连接断开，master对客户端就无能为力了。当然，客户端若检查到断开后会定期的一直连接master端的。

##### 02、saltStack的安装（使用root用户）

由01可知，我们需要在服务端安装master，在客户端安装minion .

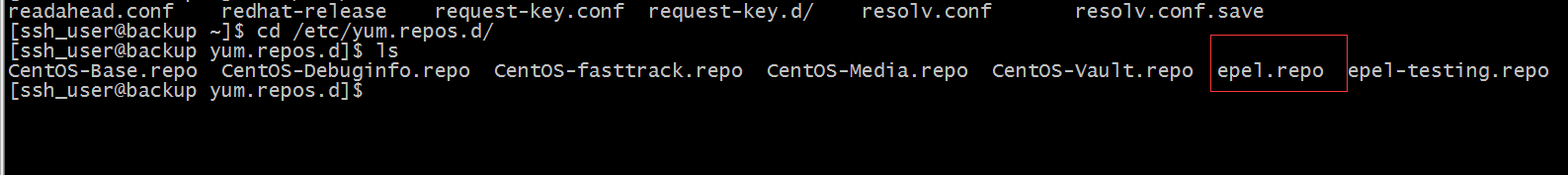
客户端安装完minion后，会自动连接到已开启的服务端master。

**SaltStack的安装需要epel源**

EPEL (Extra Packages for Enterprise Linux)是基于Fedora的一个项目，为“红帽系”的操作系统提供额外的软件包，适用于RHEL、CentOS和Scientific Linux.

查看是否安装yum源(客户端和服务端都需要安装)

cd /etc/yum.repos.d/



如果未安装，则安装: yum install epel-release -y

服务端安装master

yum install salt-master -y

客户端安装minion

yum install salt-minion – y

**将master服务添加为开机自启动（服务端）**

chkconfig salt-master on

客户端应该会自动加，

检查是否在开机自启动是否在3 这个

chkconfig –list |grep ‘salt- minion’

如果没有，则添加

chkconfig salt- minion on

##### 配置master文件（服务器端）

**服务端配置文件修改**

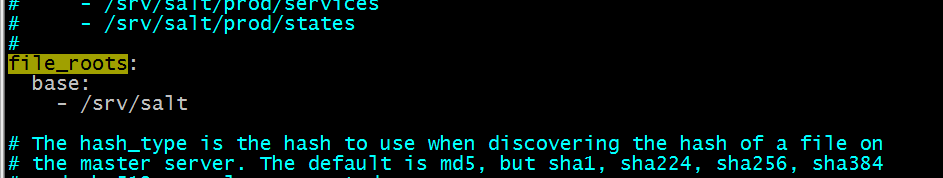
编辑 /etc/salt/master

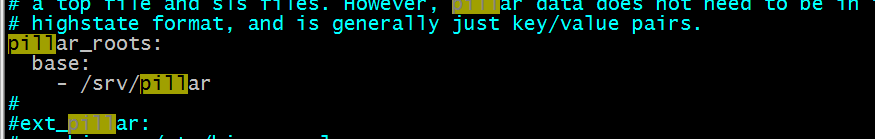
vim /etc/salt/master

搜索/file\_roots

将 注释打开 一定要保留 有空格 2 第一行 两个空格 第二行 四个空格

搜索pillar\_roots: 注释打开 要求一样





**开启master服务**

/etc/init.d/salt-master start

**客户端配置文件修改**

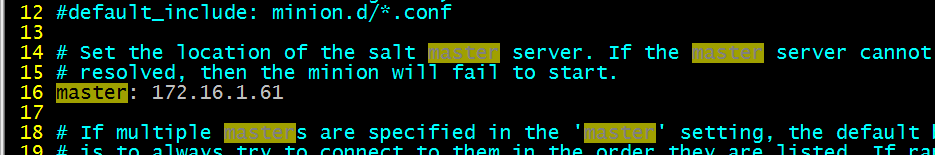
**编辑/etc/salt/minion**

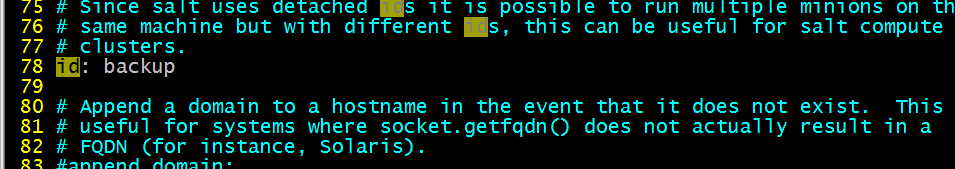
**vim /etc/salt/minion**

**修改**

**master：172.16.1.61(改成管理机的ip)**

**id:backup(改成本机的用户名)**

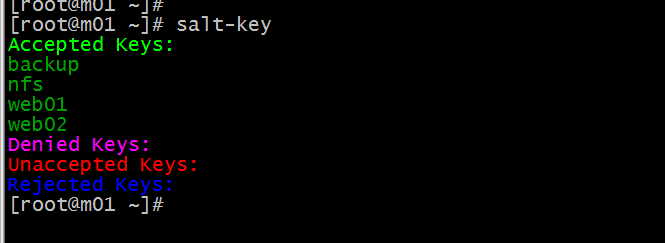




**平滑重启服务：/etc/init.d/salt-minion reload**

1. **测试**

**使用salt-key查看连接的情况。**



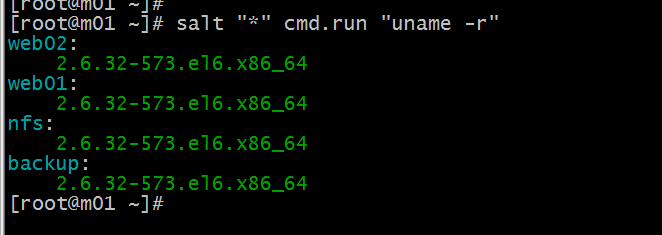
Accepted Keys：代表可以管理的机器。

Unaccepted Keys:代表可以管理，但是并未添加，我们可以通过

salt-key -a id名添加进去

**查看所有机器的内核:**

salt "\*" cmd.run "uname -r"



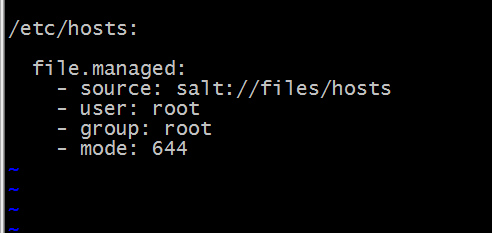
因为我们在环境里面配置的salt路径为/srv/salt, /srv/pillar,与salt相关的脚本需要放在这个路径下。

新建目录: mkdir /srv/{salt,pillar}

##### 04、命令实例

实现文件拷贝的功能，将当前目录下的files目录下的hosts文件，拷贝到目的主机的/ect/hosts下

新建脚本 vim file\_host.sls(要在/srv/salt/目录下建立)



格式为：顶格

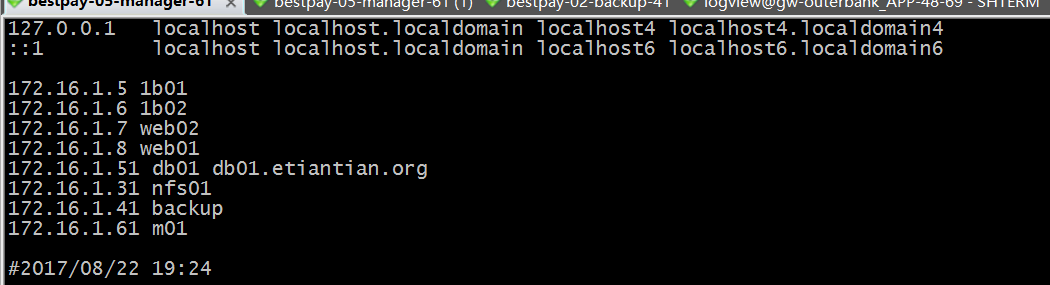
空两格

空四格

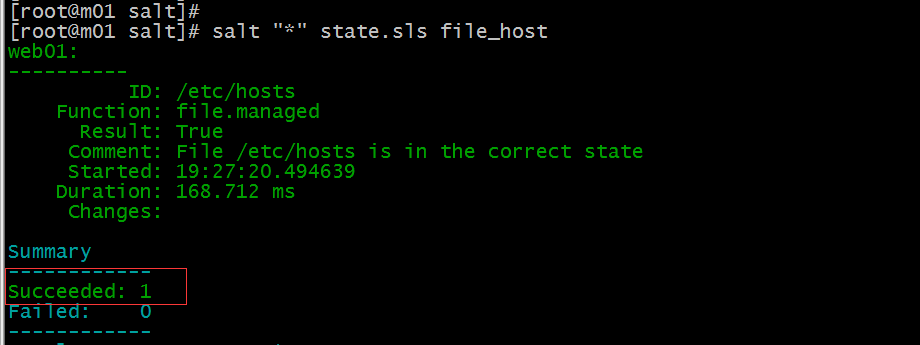
新建文件夹:mkdir /srv/salt/files

拷贝主机文件到 files文件夹 cp /etc/hosts files/hosts

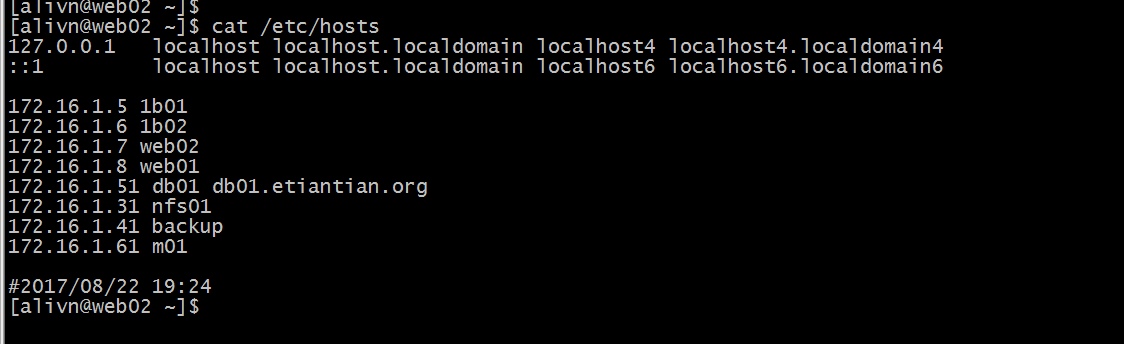
修改hosts备注时间:



执行: salt ‘\*’ state.sls host\_file



查看web01的 /etc/hosts



成功！

## 六、Nginx Web 服务器

### 1、编译安装nginx

目前还没有linux镜像自带的nginx安装，或者还未出现通过rpm yum方式安装的rpm包的情况，而且如果使用yum的方式安装，还需要下载epel yum源去官方下载，因此我们使用本地编译安装的方式进行安装。

#### 1、安装nginx所需要的依赖

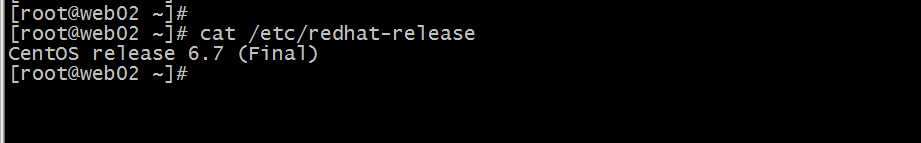
pcre中文意思为”perl兼容正则表达式”,安装pcre的目的是为了让nginx具有url重写(rewrite)的功能，URL 重写几乎是所有网站都必要的功能，如果不安装则无法使用rewrite模块的内容，与pcre相关的模块依赖: pcre pcre-devel

Nginx在使用https作为请求方式时候，需要安装openssl openssl-devel。

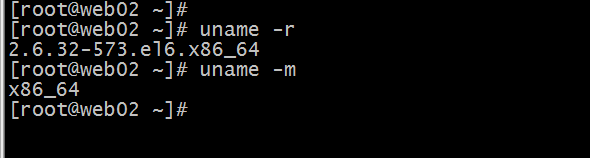
在这些依赖中，-devel是必选项。

安装pcre过程如下:

查看当前linux 版本



uname -r 查看所有信息（包括内核 版本 位数）

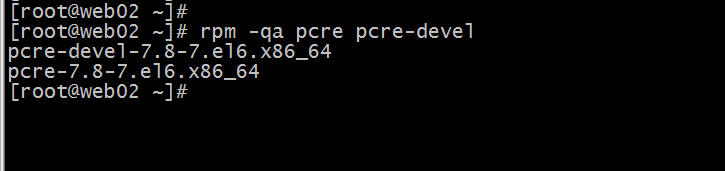


使用yum 安装pcre

yum install pcre pcre-devel -y

检查是否安装成功

rpm -qa pcre pcre-devel



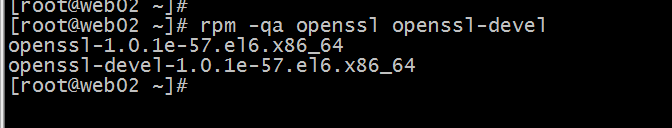
已成功安装！

安装open openssl-devel

yum install openssl openssl-devel -y

检查是否安装

rpm -qa openssl openssl-devel



#### 2、下载Nginx安装包

使用编译安装的方式安装Nginx,先从官网下载安装包。<http://nginx.org>;

获取nginx安装包也可以通过wget的方式 直接在线下载资源。

使用wget获取安装包

创建tools目录

mkdir tools

进入tools 目录

cd tools

获取安装包

wget -q <http://nginx.org/download/nginx-1.6.3.tar.gz>

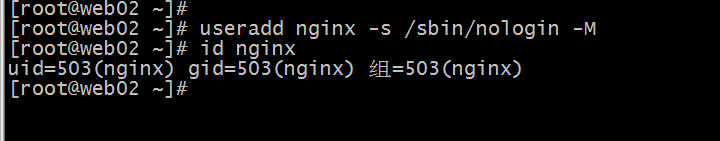


安装包已下载完毕！

#### 3、解压 编译 安装

创建nginx用户

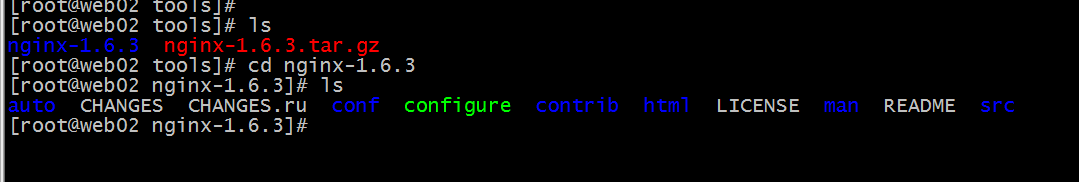
useradd nginx -s /sbin/nologin -M



解压

tar -vxf nginx-1.6.3.tar.gz

进入nginx 目录 查看目录结构



安装前初始化设置

./configure --user=nginx --group=nginx --with-http\_ssl\_module --with-http\_stub\_status\_module --prefix=/application/nginx-1.6.3/

参数解释：

--user：进程用户权限

--group：进程用户组权限

--with-http\_ssl\_module enable --with-http\_ssl\_module：激活 禁用ssl

--with-http\_stub\_status\_module enable --with-http\_stub\_status\_module：激活状态信息

--prefix：设置安装路径

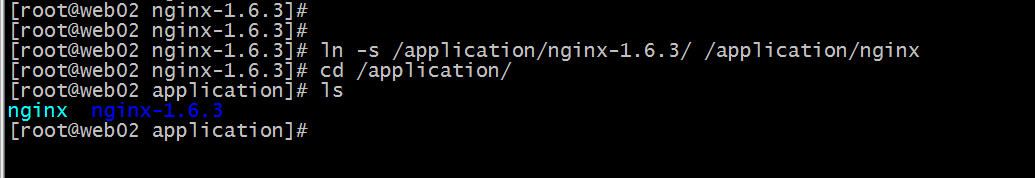
编译 安装

make

make install

创建软链接（以后直接操作软连接就行）

ln -s /application/nginx-1.6.3/ /application/nginx



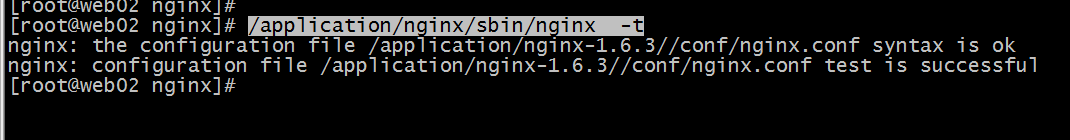
#### 4、启动并检查安装结果

进入软链接目录

cd /application/nginx

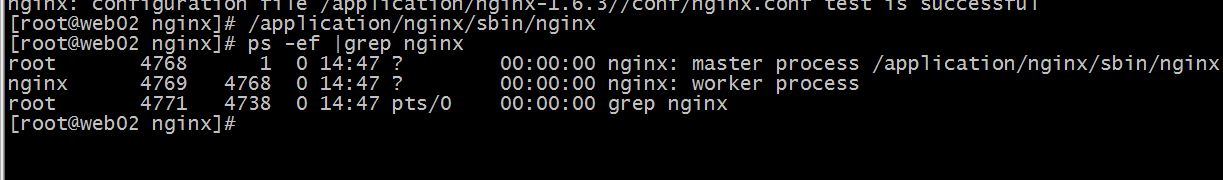
启动前语法检查

/application/nginx/sbin/nginx -t



启动

/application/nginx/sbin/nginx

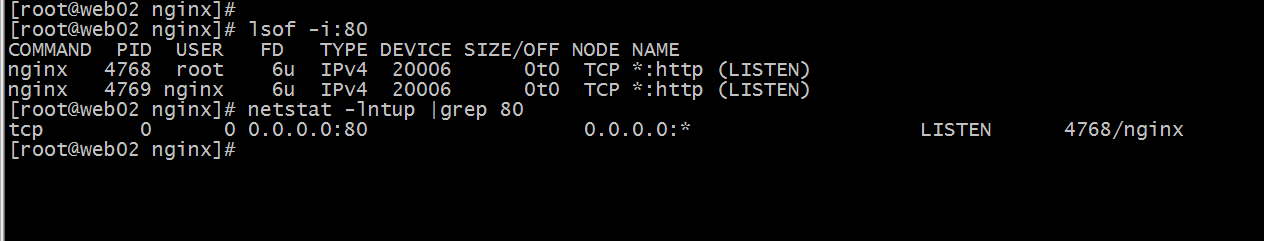


检查端口 是否启动成功

lsof -i:80

或者

netstat -lntup |grep 80



在本机上检测 是否启动成功

在浏览器上输入ip 回车 即可！



出现这个界面，证明安装成功！

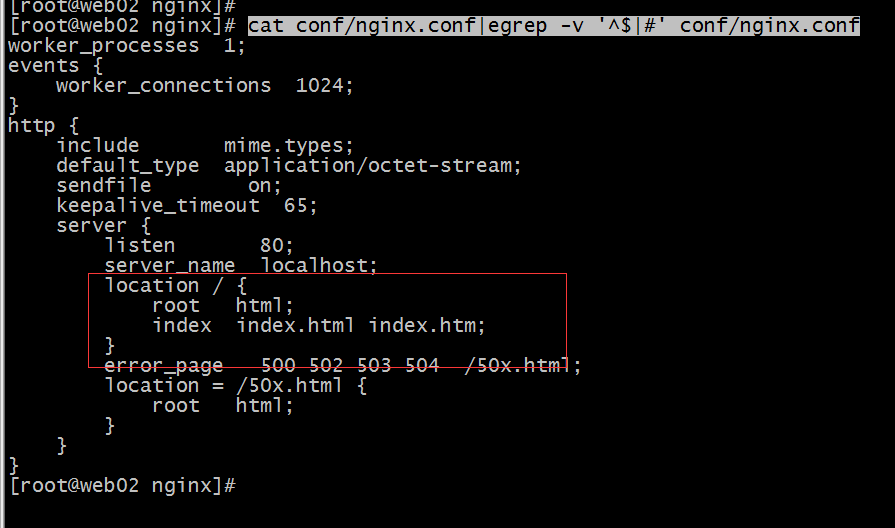
### 2、Nginx简单部署

Nginx的配置文件为安装目录下的conf/nginx.conf 文件。

#### 1、查看nginx.conf 文件

我们可以不看注释和空行，直接查看内容

cat conf/nginx.conf|egrep -v '^$|#' conf/nginx.conf



在配置文件中： location 的内容

root html; //网站的根目录是html

index index.html index.htm;//显示默认首页

我们可以定义自己的访问首页。

进入html 目录，删除index.html 界面

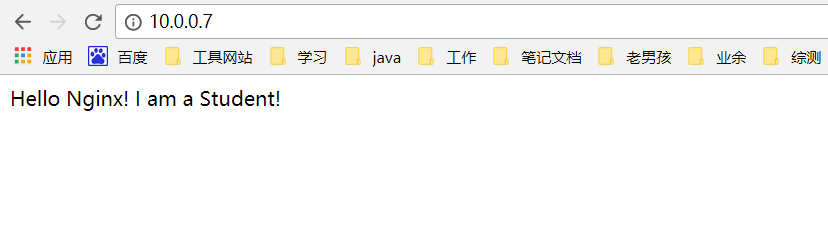
cd /application/nginx/html/

rm -f index.html

建立自己的 html 欢迎界面！

echo 'Hello Nginx! I am a Student!'>index.html

#### 2、查看自定义的欢迎界面



同理，我们可以部署复杂的web应用！

### 3、Nginx 虚拟主机

#### 1、虚拟主机的概念

所谓虚拟主机就是一个独立的网站站点，这个站点对应独立的域名（也可能是端口或者ip），具有独立的程序资源目录等。 可以独立的提供服务让外部用户访问。配置文档可从官网链接查看：http://nginx.org/en/docs/

这个独立的站点的配置是由一定格式的标签段组成的，对于Apache，通常包含在<VirtualHost></VirtualHost>标签体内，而nginx是使用一个server{} 标签标识的。 一个web服务里面可以有多个虚拟主机标签对，可以同时支持多个虚拟主机站点。

虚拟主机可分为三类：

1. 基于域名的虚拟主机 通过域名来区分虚拟主机
2. 基于端口的虚拟主机，通过端口来区分虚拟主机
3. 基于IP的虚拟主机，几乎不用，不支持iconfig别名，配置文件可以。

#### 2、基于域名的虚拟主机

基于域名的虚拟主机用的比较多，主要用于外部大中型web应用。

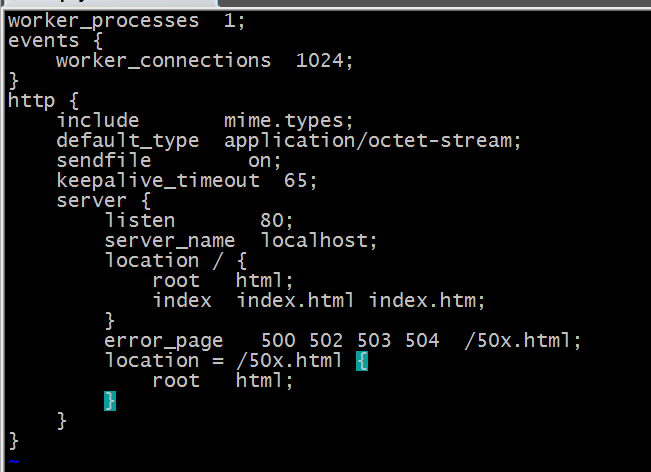
我们可以事先定义两个域名服务 [www.cctv.com](http://www.cctv.com) [www.bestpay.com](http://www.bestpay.com)

当输入[www.cctv.com](http://www.cctv.com) 时出来的欢迎界面是：hello,I am cctv!

当输入[www.bestpay.com](http://www.bestpay.com)时欢迎界面是：hello,I am bestpay!

我们可以通过配置nginx.conf 来完成这个操作。 为了使配置更加清晰，可以删除nginx.cong 中的空行和注释。

egrep -v '^$|#' /application/nginx/conf/nginx.conf.default >nginx.conf



配置server标签段

一个server标签段可以对应一个主机，我们可以新增两个server标签段

Listen:端口 server\_name 域名 location： 资源文件路径

因为是基于域名的，所以我们只需要修改 server\_name 和location即可。

域名 [www.cctv.com](http://www.cctv.com)的配置

server {

listen 80;

server\_name www.cctv.com;

location / {

root html/cctv;

index index.html index.htm;

}

}

域名[www.bestpay.com](http://www.bestpay.com)的配置

server {

listen 80;

server\_name www.bestpay.com;

location / {

root html/bestpay;

index index.html index.htm;

}

}

创建站点资源

在html目录下，创建cctv和bestpay 目录，并创建index.html欢迎界面

mkdir html/{cctv,bestpay} -p

echo 'hello,I am cctv!'>cctv/index.html

echo 'hello,I am bestpay!'>bestpay/index.html

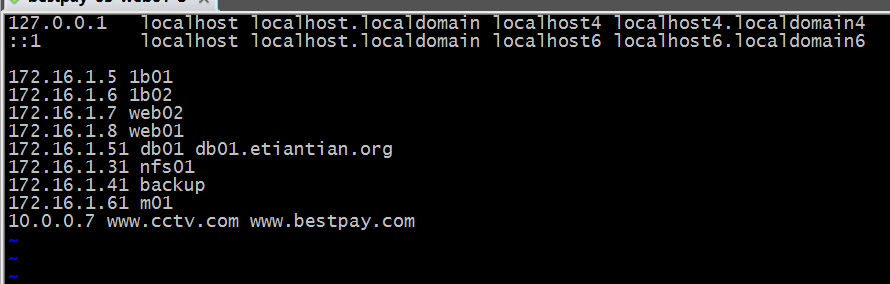
平滑重启

/application/nginx/sbin/nginx -s reload

开启另一台机器测试（10.0.0.8）

在10.0.0.8的/etc/hosts文件中做域名解析

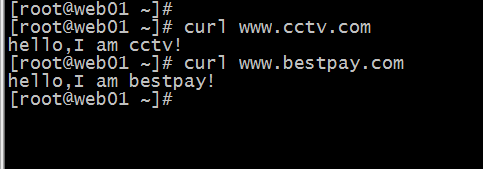
vim /etc/hosts



测试域名

curl [www.cctv.com](http://www.cctv.com)

curl [www.bestpay.com](http://www.bestpay.com)



这样证明配置成功啦！

#### 3、基于端口的虚拟主机

基于端口和基于域名的修改方式类似，不一样的就是 域名修改server\_name 和location 而 端口需要修改listen和location就行。

假设我们访问[www.nginx.com](http://www.nginx.com) [www.nginx.com:443](http://www.nginx.com:443) 来区分端口

新增server标签

配置[www.nginx.com](http://www.nginx.com)

server {

listen 80;

server\_name www.nginx.com;

location / {

root html/nginx;

index index.html;

}

}

配置 [www.nginx.com:443](http://www.nginx.com:443)

server {

listen 443;

server\_name www.nginx.com;

location / {

root html/nginx;

index main.html;

}

创建资源文件

在html目录下新建nginx文件。并且新增index.html 和main.html界面。 mkdir nginx

echo 'hello , I am port 80!'>nginx/index.html

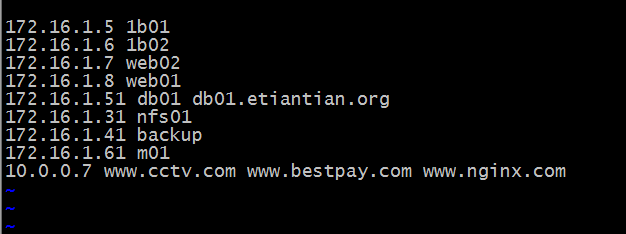
echo 'hello , I am port 443!'>nginx/main.html

平滑重启

/application/nginx/sbin/nginx -s reload

测试（在10.0.0.8）

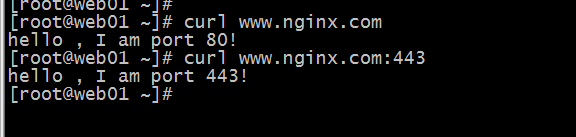
在10.0.0.8的/etc/hosts中配置域名解析



测试

curl [www.nginx.com](http://www.nginx.com)

curl [www.nginx.com:443](http://www.nginx.com:443)



测试成功！

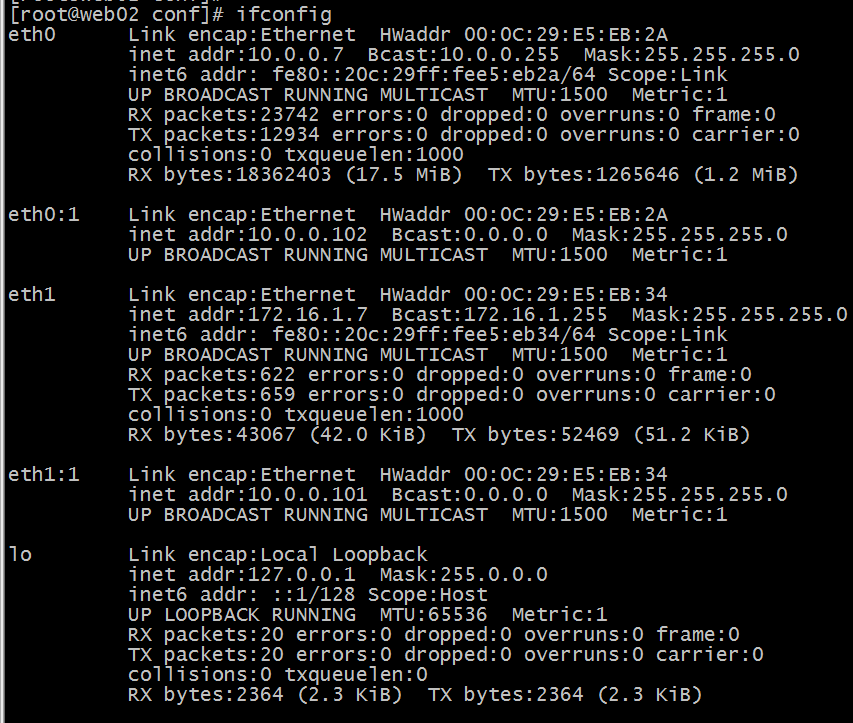
#### 4、基于IP的虚拟主机

基于ip的虚拟主机，也就是将server\_name修改成ip即可，我们可以在为一台机器分配多个ip。然后配置nginx.conf 即可。

为web服务器新增ip

ip addr add 10.0.0.101/24 dev eth1 label eth1:1

ip addr add 10.0.0.102/24 dev eth0 label eth0:1



编辑nginx.conf 文件

server {

listen 80;

server\_name 10.0.0.101;

location / {

root html/IP;

index index.html;

}

}

server {

listen 80;

server\_name 10.0.0.102;

location / {

root html/IP;

index main.html;

}

}

新增资源文件

mkdir html/IP

echo ‘I am 10.0.0.101’>html/IP/index.html

echo ‘I am 10.0.0.102’>html/IP/main.html

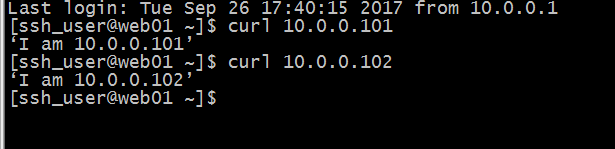
平滑重启

/application/nginx/sbin/nginx -s reload

测试

curl 10.0.0.101

curl 10.0.0.102



测试成功！

#### 5、规范优化Nginx配置文件

在nginx配置文件中，当我们存在多个虚拟主机的时候，我们 可以直接写在住配置文件nginx.conf中，但如果存在大量的独立虚拟主机的话，在住配置文件里面写入大量的server标签段，很不直观， 因此我们可以使用include标签，将配置的独立主机文件包含进来就可以了。

以本次配置的 虚拟主机为例，我们可以将基于域名 基于端口 基于IP的 三类虚拟主机放在三个独立的配置文件中，分别名为domain.conf ports.conf ips.conf,然后将相关配置添加进去即可。并将这些配置文件放入extra目录下

在nginx安装目录下的conf目录下创建extra目录

mkdir extra

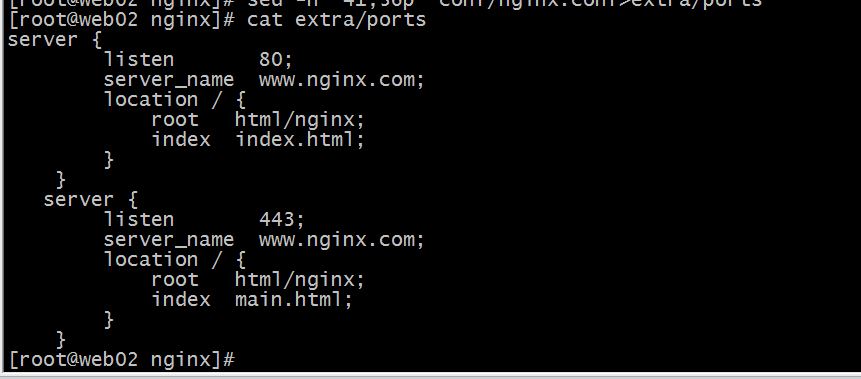
将基于域名的配置放入domain.conf 文件中

sed -n '24,40p' conf/nginx.conf>extra/domain.conf



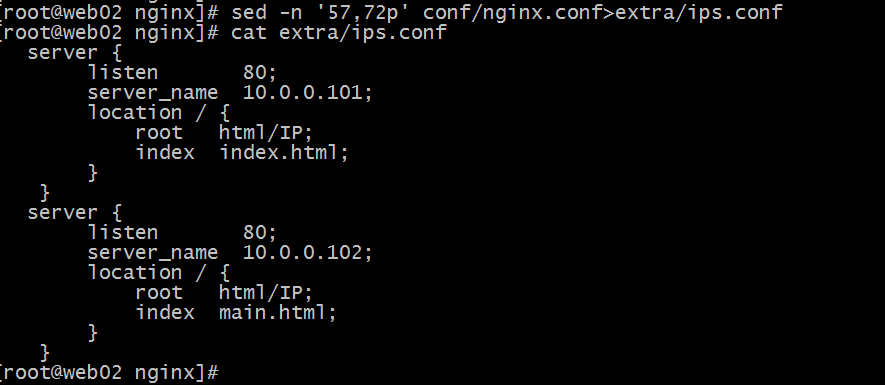
将基于端口的配置放入ports.conf 文件中

sed -n '41,56p' conf/nginx.conf>extra/ports



将基于IP的配置放入ips.conf 文件中

sed -n '57,72p' conf/nginx.conf>extra/ips.conf

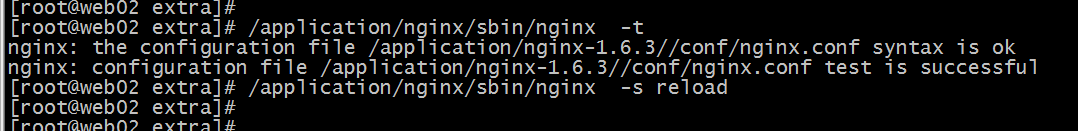


配置已经转移完毕!删除原配置文件的虚拟主机配置！



平滑重启

/application/nginx/sbin/nginx -s reload



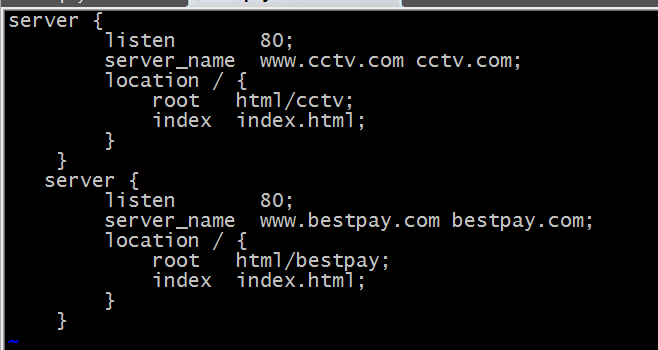
测试（在10.0.0.8机器上）



测试成功！

#### 6、配置别名

有时候我们需要为一个虚拟机配置多个域名，比如我们访问 [www.baidu.com](http://www.baidu.com) ,baidu.com都可以得到百度的响应。这样就需要为虚拟主机配置别名了。 配置别名，直接在server\_name 后面 用空格分割 各个域名就行。



平滑重启

/application/nginx/sbin/nginx -s reload

在10.0.0.8机器的/etc/hosts文件里面新增域名映射

10.0.0.8 cctv.com bestpay.com

测试

curl [www.cctv.com](http://www.cctv.com)

curl cctv.com

curl [www.bestpay.com](http://www.bestpay.com)

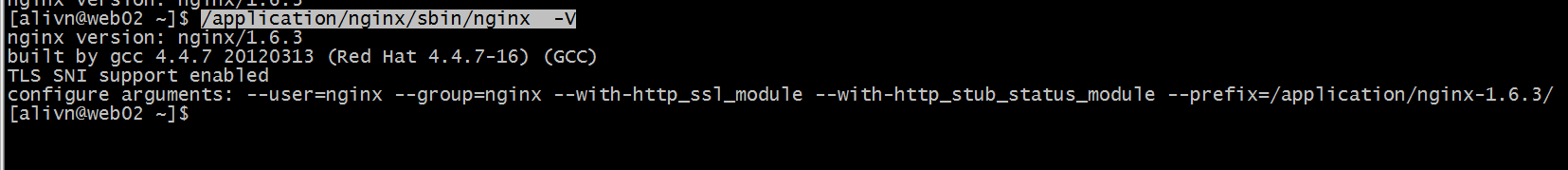
curl bestpay.com



#### 7、Nginx状态功能信息

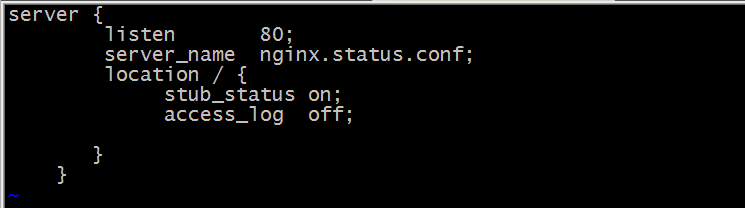
Nginx有一个模块--with-http\_stub\_status\_module专门记录nginx访问状态的基本信息，可以让使用者了解nginx的基本访问状态，例如连接数等信息。这个模块需要在编译的时候设置，我们可以通过 /application/nginx/sbin/nginx -V 来检查编译时是否指定了该参数。

/application/nginx/sbin/nginx -V

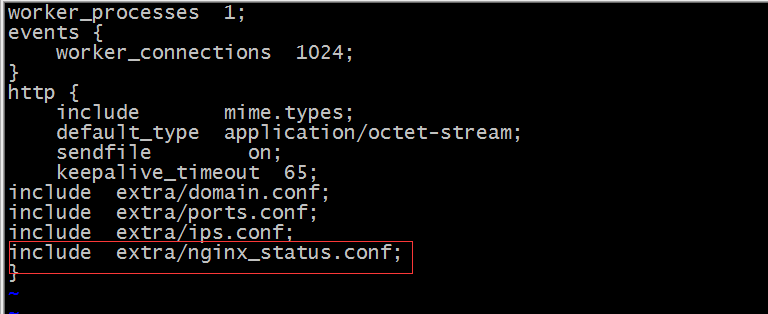


配置nginx状态参数

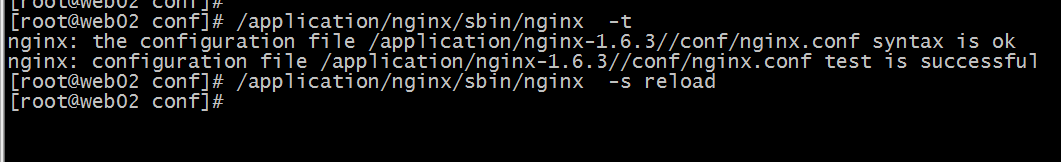
新建一个虚拟主机server



将状态虚拟主机文件包含进主配置文件。



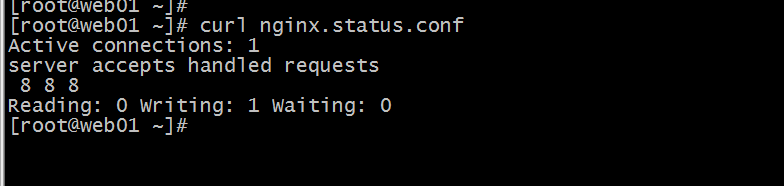
检查语法 平滑重启



在10.0.0.8的/etc/hosts配置域名解析



测试



参数解释：

Active connections：1 //表示nginx正在处理的连接数为1

server accepts handled requests

8 8 8

server表示启动到现在共处理的8次连接

accepts表示nginx启动到现在共创建了8次握手

handled requests表示从启动到现在共处理了8次请求。

请求丢失数=握手数-连接数 （本次并没有丢失的情况发生）

Reading: 0 Writing: 1 Waiting: 0

Reading:表示读取到连接的Header数

Writing：表示返回给客户端的Header数

Waiting:Nginx已经处理完正在等待下一次请求指令的驻留连接。在开启keep-alive的情况下，这个值=active-（reading+writing）;

#### 8、为nginx增加错误日志配置

我们可以记录nginx运行过程中可能出现的错误，我们可以设置输入日志的位置，也可以设置日志输出的级别， 属于核心模块ngx-core-module。这个参数是error\_log,这个参数可以放在Main全局配置域中，也可以单独配置虚拟主机中。

error\_log参数及格式

error\_log file level

关键字不可改变 日志文件 记录日志的等级

其中error\_log是关键字不可改那边，file是日志文件，可以制定日志文件存储位置。level是日志的级别。 日志的级别“stderr”, “emerg”, “alert”, “crit”, “error”,”warn”, “notice”, “info”, “debug” (级别由高到低)，级别越高记录的越少。但是不能使用最低级别，会极大的消耗磁盘IO。

error\_log默认值

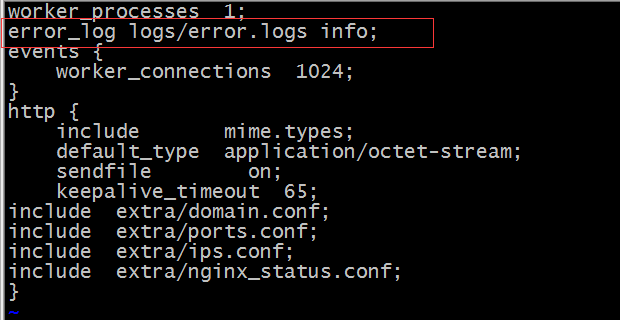
#default error\_log logs/error.log error

可以放置的标签段为

#context: main http server location

配置日志

在Main主配置区域



这样就会将info级别的日志 全部记录的 logs/error.log这个文件中。

#### 9、nginx访问日志

* Nginx会将每个用户的访问记录记录下来，供网站运营分析用户的浏览行为，此功能那个由[ngx\_http\_log\_module](http://nginx.org/en/docs/http/ngx_http_log_module.html)负责。

Nginx访问日志主要由两个参数指定:

log\_format 用于记录日志的格式。

access\_log 用来指定日志文件的路径及使用何种日志格式记录日志。

Nginx默认配置：

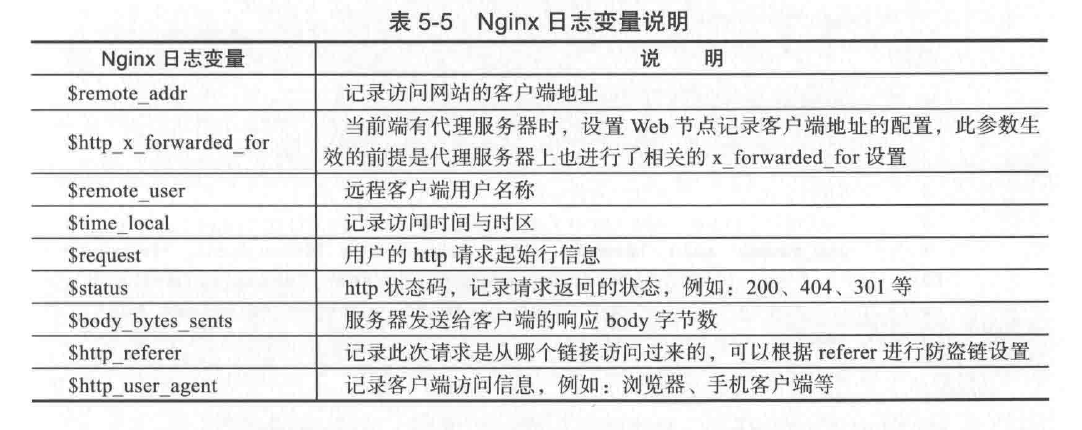
log\_format main'$remote\_addr - $remote\_user [$time\_local] '

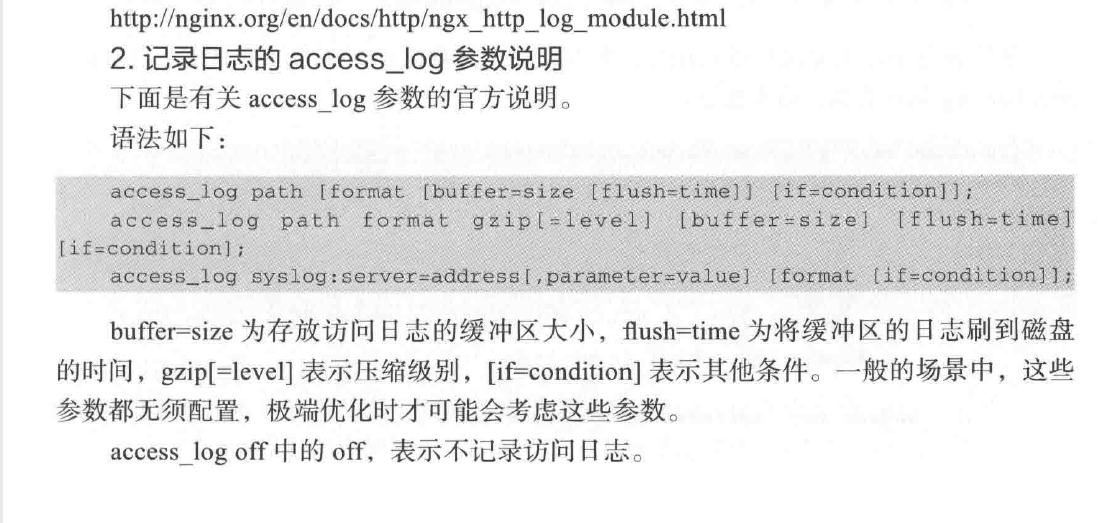
'"$request" $status $bytes\_sent '

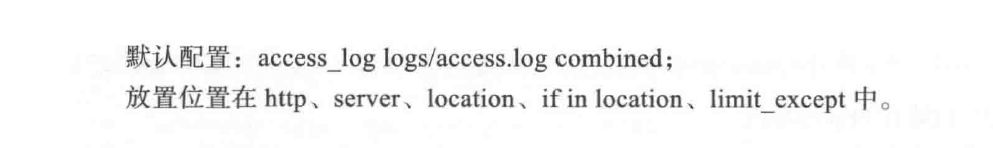
'"$http\_referer" "$http\_user\_agent" "$gzip\_ratio"';

access\_log logs/access.log

日志参数说明



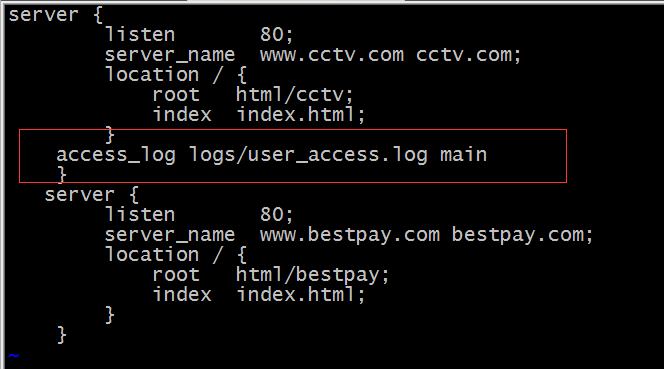




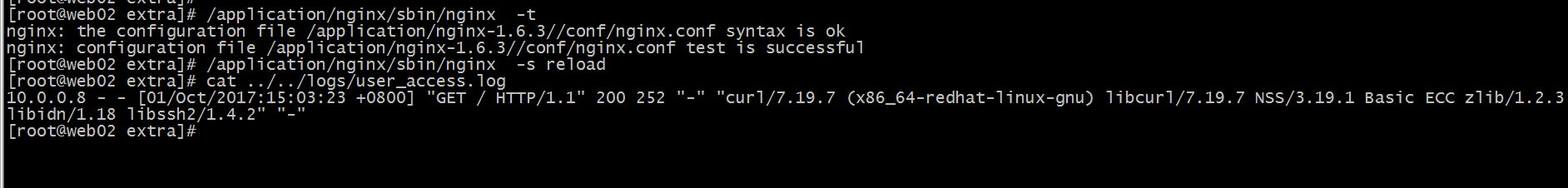
在主配置文件中配置日志格式



在其他虚拟主机中配置日志格式 和日志文件



测试。



七、mysql安装

Mysql的机器是：172.16.1.51 10.0.0.51

1. 下载mysql

wget http://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.30-linux-glibc2.5-x86\_64.tar.gz