上章回顾

- 如何使用yum工具安装、卸载一个软件包?
  - yum -y install 软件包　　安装一个包
  - yum -y remove 软件包　卸载一个包
- 在PXE远程装机服务器中，DHCP的next-server选项的作用是什么?
  - 指定TFTP服务的IP地址
- 如何生成安装应答文件?
  - yum安装syslinux程序
  - 文件在/usr/share/syslinux/pxelinux.0

本章内容

- 学会配置rsync备份源
- 学会rsync下行、上行异地备份
- 学会rsync+inotify实时备份

一、rsync介绍

1、rsync（Remote Sync，远程同步）是一款可以实现快速增量备份的工具

2、配合任务计划，rsync能实现定时或间隔同步，配合inotify，可以实现触发式的实时同步

3、软件特性

1）支持拷贝特殊的文件，比如：链接文件、设备文件等

2）可以排除指定文件或目录同步的功能，类似于tar包归档时排除某个目录

3）可以实现保持源文件或目录的权限、时间、链接、属组、属主等属性

4）可以完成增量备份，即只同步发生变化的数据，因此数据传输的效率很高

5）可以使用rcp、rsh、ssh等方式配合传输文件，rsync本身不具备数据加密功能

6）可以通过socket（进程方式）传输文件和数据

7）支持匿名和用户认证（**无需系统用户**）的进程模式传输，可实现方便安全的进程数据备份及镜像

> rcp
>
> - 用于在系统之间复制文件或目录的实用程序，可以本地也可以远程主机
> - 语法：rcp [-pr] [源文件或目录] [目标文件或目录]
> - rcp -pr root@218.6.132.5:/root/testfile /root/test
>
> rsh
>
> - 用于远程登录的Shell工具，与ssh不同是的，rsh采用明文传输，没有密钥机制
> - 语法是：rsh [-dn][-l <用户名称>][主机名称或IP地址][执行指令]
> - 命令：rsh -l hnlinux 192.168.1.88 /bin/ls
>
> socket
>
> - 是一种进程间通信（IPC）机制，它是应用层与TCP/IP协议通信的中间软件抽象层
> - 简单来说就是实现相同、不同、远程进程之间的通信

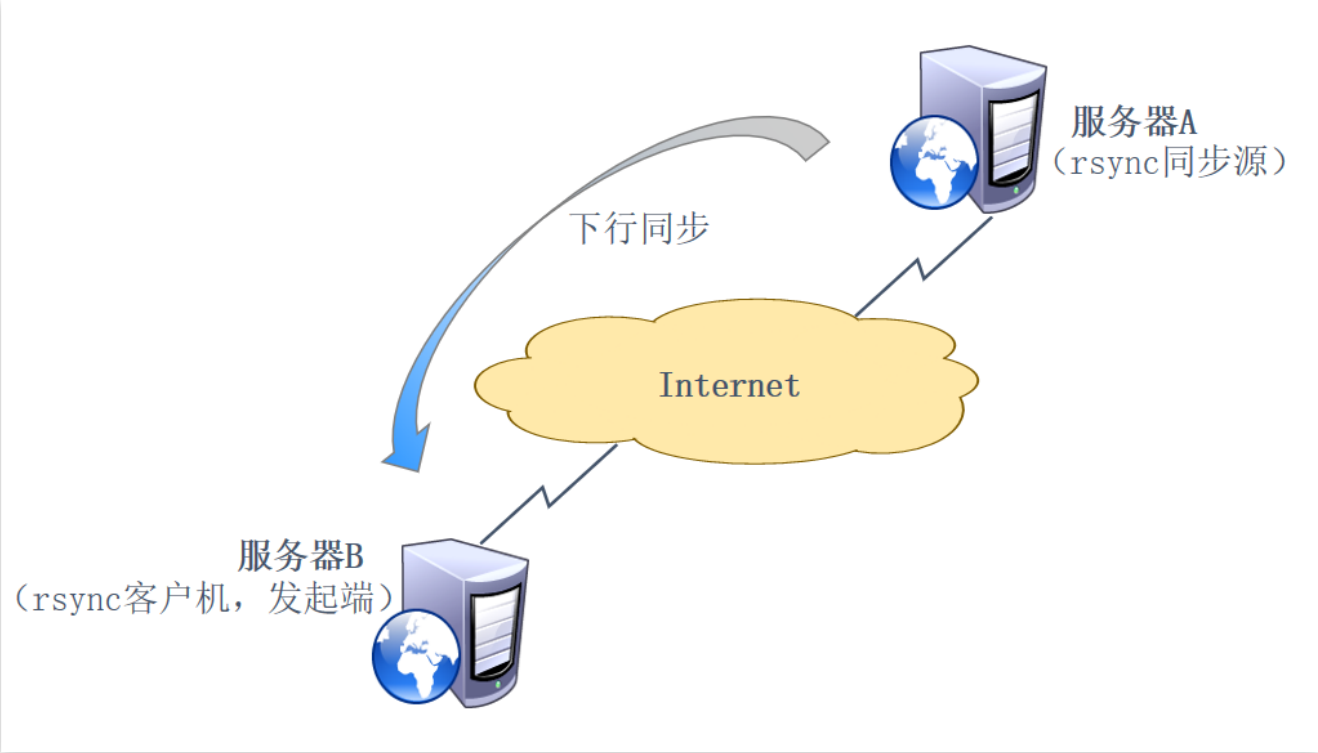
4、rsync的工作模式（主要分为三种）

1）单个主机本地之间的数据传输（类似于 cp 命令）

2）借助rcp、ssh等通道来传输数据（类似于 scp 命令）

3）以守护进程（socket）的方式传输数据（rsync自身的重要功能）

二、配置rsync源服务器（备份操作的远程服务器，一般称为源服务器）



1、基本思路

1）安装rsync，生成主配置文件/etc/rsyncd.conf

2）建立独立的rsync虚拟账号，用于登陆源服务器并且同步

3）启用rsync的 --daemon模式（后台监听模式）

2、实例要求：

1）qm公司为了保证web站点的数据目录的安全（目录在/var/www/html），现需要对用户上传的资料进行备份

3、实验环境

1）需要同步备份的目录为：/var/www/html

2）使用backuper用户，允许下行同步

3）服务器：192.168.1.1

4）客户机：192.168.1.254

4、安装rsync程序

```
[root@localhost ~] yum -y install rsync
```

三、配置文件rsync.conf

1、默认存储路径：/etc/rsyncd.conf，部分系统该文件不存要需要手动创建。语法类似于Samba配置。配置文件需要定义一个全局配置和多个rsync共享配置

2、通过man rsync.conf获取文件的帮助方式

3、rsync.conf 配置文件详解

```
[root@localhost ~] vim /etc/rsyncd.conf
# /etc/rsyncd: configuration file for rsync daemon mode

# See rsyncd.conf man page for more options.

# configuration example:

# uid = nobody
# gid = nobody
# use chroot = yes
# max connections = 4
# pid file = /var/run/rsyncd.pid
# exclude = lost+found/
# transfer logging = yes
# timeout = 900
# ignore nonreadable = yes
# dont compress   = *.gz *.tgz *.zip *.z *.Z *.rpm *.deb *.bz2

# [ftp]
#        path = /home/ftp
#        comment = ftp export area
```

1）其中已经定义了大量的全局配置项，但是被注释掉状态，管理员可以手动启动

2）[ftp] 就是rsync中的模块，ftp名称是模块的 id，在rsync中是唯一的

3）每个模块必须定义 path 选项，path 代表当前模块的路径

4、常用的配置项（也可以作为实例使用）

```
[root@localhost ~] vim /etc/rsyncd.conf
########## 全局配置参数 ##########
port = 873 # 指定rsync端口。默认873
uid = nobody # rsync服务的运行用户，默认是nobody，文件传输成功后属主将是这个uid
gid = nobody # rsync服务的运行组，默认是nobody，文件传输成功后属组将是这个gid
use chroot = no # rsync daemon在传输前是否切换到指定的path目录下，并锁定在该目录下
max connections = 200 # 指定最大连接数量，0表示没有限制
timeout = 300 # 链接超时时间，0表示永远不会超时
pid file = /var/run/rsyncd.pid # 指定rsync daemon的pid文件
lock file = /var/run/rsync.lock # 指定锁文件
log file = /var/log/rsyncd.log # 指定rsync的日志文件，而不把日志发送给syslog
```

```
dont compress = *.gz *.tgz *.zip *.z *.Z *.rpm *.deb *.bz2 # 指定哪些文件不用进行压缩传输
```

> 在rsync中，锁文件是一种临时文件，主要用于防止两个不同的rsync进程同时更新同一个文件。当一个rsync进程正在对一个文件进行复制或删除等操作时，会创建一个锁文件，这个锁文件通常位于目标文件的同一目录下，并命名为该文件的名称加上“.lck”后缀。这样，如果另一个rsync进程也想要对这个文件进行相同的操作，它首先会检查是否存在这个锁文件。如果存在，那么这个进程就会知道该文件正在被使用，从而避免对同一个文件进行重复的操作，确保数据的一致性。

```
############下面指定模块，并设定模块配置参数，可以创建多个模块############
[webdoc] # 模块ID
    path = /var/www/html/ # 指定该模块的路径，该参数必须指定。启动rsync服务前该目录必须存在。rsync请求访问模块本质就是访问该路径。
    ignore errors # 忽略一些IO错误信息
    read only = false # 指定该模块是否可读写，即能否上传文件，false表示可读写，true表示可读不可写。所有模块默认不可上传
    write only = false # 指定该模式是否支持下载，设置为true表示客户端不能下载。所有模块默认可下载
    hosts allow = 10.0.0.0/24 # 指定允许连接到该模块的机器，多个ip用空格隔开或者设置区间
    hosts deny = 0.0.0.0/32 # 指定不允许连接到该模块的机器
    auth users = backuper # 指定连接到该模块的用户列表，只有列表里的用户才能连接到模块，用户名和对应密码保存在secrts file中，这里使用的不是系统用户，而是虚拟用户。不设置时，默认所有用户都
能连接，但使用的是匿名连接
    secrets file = /etc/rsyncd.passwd # 保存auth users用户列表的用户名和密码，每行包含一个username:passwd
```

1）deny | allow规则说明

- **两者都没有设置默认允许访问**
- **仅出现hosts allow：定义白名单，如果没有匹配规则默认允许**
- **仅出现hosts deny：定义黑名单，添加的条目被拒绝**
- **二者同时出现：先允许、后拒绝，默认允许**

2）auth user、secrets file说明

- **认证配置auth users、secrets file，不加则为匿名**

3）rsync认证文件

- **采用“用户名:密码”的格式记录**
- **独立的账号文件，不依赖系统账户**

5、创建账号文件

```
[root@localhost ~] vim /etc/rsyncd_users.db
backuper:123.com
```

```
zhangsan:123.com
[root@localhost ~] chmod 600 /etc/rsyncd_users.db
```

6、实例配置文件

```
[root@localhost ~] vim /etc/rsyncd.conf
# /etc/rsyncd: configuration file for rsync daemon mode
# See rsyncd.conf man page for more options.
# configuration example:
 uid = nobody
 gid = nobody
 use chroot = yes
# max connections = 4
 pid file = /var/run/rsyncd.pid
 lock file = /var/run/rsyncd.lock
 port 873
 log file = /var/log/rsyncd.log
 hosts allow = 192.168.1.0/24
# exclude = lost+found/
# transfer logging = yes
# timeout = 900
# ignore nonreadable = yes
# dont compress   = *.gz *.tgz *.zip *.z *.Z *.rpm *.deb *.bz2

[wwwroot]
        path = /var/www/html
        comment = sui bian xie ,mei guan xi !!!
        read only = yes
        dont comperss = *.gz *.bz2 *.tgz *.zip *.rar *.z
        auth users = backuper
        secrets file = /etc/rsyncd_users.db
```

7、启动rsync服务

1）rsync运行时通过 --daemon选项独自提供服务，rsync daemon是事先在server端上运行rsync后台进程，用来监听套接字等待client的连接，连接完成后所有通信方式都是由套接字完成的。

> Linux的套接字是一种通信机制，可以实现多个客户和服务器之间的网络通信，支持流套接字和数据报套接字。 它是一种全双工通信方式，分别有读写缓冲区。 套接字也是一种文件格式，和管道文件一样，它是一种伪文件，存

> 在于内核的缓冲区中，大小不变，一直是0。 套接字一定是成对出现的，有server套接字，就一定有client套接字。

```
[root@localhost run] rsync --daemon
[root@localhost run] ps aux | grep rsync
root      15624  0.0  0.0 114844   392 ?        Ss   16:36   0:00 rsync --daemon
root      15635  0.0  0.0 112824   980 pts/1    R+   16:37   0:00 grep --color=auto
rsync
[root@localhost run] netstat -anpt | grep rsync
tcp        0      0 0.0.0.0:873             0.0.0.0:*               LISTEN
15644/rsync
tcp6       0      0 :::873                  :::*                    LISTEN
15644/rsync
[root@localhost run] pkill -9 rsync
             &&
[root@localhost run] kill $(cat /var/run/rsyncd.pid)
```

> 如果出现failed to create pid file /var/run/rsyncd.pid: File exists字样，删除pid文件重新启动即可

```
[root@localhost run] rsync --daemon
[root@localhost run] failed to create pid file /var/run/rsyncd.pid: File exists

[root@localhost run] rm -rf /var/run/rsyncd.pid
[root@localhost run] rsync --daemon
[root@localhost run] netstat -anpt | grep rsync
tcp        0      0 0.0.0.0:873             0.0.0.0:*               LISTEN
15652/rsync
tcp6       0      0 :::873                  :::*                    LISTEN
15652/rsync
```

四、使用rsync工具

1、语法：rsync 选项 原始位置 目标位置

rsync -avz --delete [本地路径] zhangsan@192.168.1.1::qm

2、常用选项

- **-a：归档模式，递归并保留对象属性，等同于：-rlptgoD**
  - **-r：递归模式，包含子目录中所有的文件**
  - **-l：保留链接文件属性**
  - **-p：保留权限属性**
  - **-t：保留时间标记**
  - **-g：保留属组标记（仅root使用）**

        - **-o：保留属主标记（仅root使用）**
        - **-D：保留设备文件以及其他特殊文件**
    - **-v：显示同步过程的详细信息**
    - **-z：传输时进行数据压缩**
    - **--delete：删除目标位置有而原始位置没有的文件**
    - **--checksum：根据对象的校验确认是否跳过文件**

3、同步源服务器表示方法

1）rsync 选项 用户名@主机地址::共享模块名称 本地路径 #下行同步

2）rsync://用户名@主机地址/共享模块名称

3）例子：

- **rsync -avz backuper@192.168.1.1::wwwroot**
- **rsync//backuper@192.168.1.1/wwwroot**

4、同步操作

1）下行同步：（提取-拿）

- **客户端从服务器上获取数据同步到本地**

> 如果当前主机比目标目录的数据多，可以使用--delete选项删除无用数据

```
[root@Client ~] yum -y install rsync
[root@Client ~] rsync -avz backuper@192.168.1.1::wwwroot /myweb
[root@Client ~] rsync -avz --delete rsync://backuper@192.168.1.1/wwwroot /myweb/
```

2）上行同步：（推送-给）

- **客户端将数据推送给服务器**

```
[root@localhost html] vim /etc/rsyncd.conf
uid = 0
gid = 0
[wwwroot]
    read only = false
[root@localhost html] kill $(cat /var/run/rsyncd.pid)
[root@localhost html] rsync --daemon
[root@Client myweb] rsync -avz /myweb/ backuper@192.168.1.1::www
```

5、免交互处理

1）使用选项--password-file=密码文件

- 创建密码文件

```
[root@Client myweb] vim /etc/server.pass
123.com

[root@Client myweb] chmod 600 /etc/server.pass
```

- 下行同步

```
[root@Client myweb] rsync -avz --delete --password-file=/etc/server.pass
backuper@192.168.1.1::www /myweb/
receiving incremental file list
deleting ceshi

sent 20 bytes  received 84 bytes  208.00 bytes/sec
total size is 0  speedup is 0.00
```

- 上行同步

```
[root@Client myweb] rsync -avz --password-file=/etc/server.pass /myweb/
backuper@192.168.1.1::www
sending incremental file list
./
ceshi

sent 145 bytes  received 38 bytes  366.00 bytes/sec
total size is 0  speedup is 0.00
```

6、结合计划任务自动完成

```
[root@Client myweb] crontab -e
*/1 * * * * /usr/bin/rsync -az --delete --password-file=/etc/server.pass
backuper@192.168.1.1::www /myweb
```

五、配置inotify实时同步

1、计划任务执行同步命令

1）执行备份的时间比较固定，不够灵活，实时性差

2）如果同步源长期没有数据变动，周期性进行计划任务占用系统资源

2、实时同步

1）触发式更新，当同步源出现变动后实时完成同步，否则将不执行备份，减轻服务器压力

3、配置inotify实时同步

1）inotify 是一个 Linux 内核特性由inotify-tools包提供，它监控文件系统，并且及时向专门的应用程序发出相关的事件警告，比如删除、读、写和卸载操作等。

2）安装inotify-tools

```
[root@Client ~] tar zxvf inotify-tools-3.14.tar.gz
[root@Client ~] cd inotify-tools-3.14
[root@Client inotify-tools-3.14] ./configure && make && make install
[root@Client ~] inotifywa
inotifywait   inotifywatch
[root@Client ~] ll /proc/sys/fs/inotify/
总用量 0
-rw-r--r--. 1 root root 0 1月  16 22:34 max_queued_events
-rw-r--r--. 1 root root 0 1月  16 22:34 max_user_instances
-rw-r--r--. 1 root root 0 1月  16 22:34 max_user_watches

```

3）inotity命令

- **inotifywait：用于持续监控，实时输出结果**
- **inotifywatch：用于短期监控，任务完成后再出结果（少数使用）**

4）优化配置文件

- max_queued_events文件：该文件包含一个整数值，表示rsync进程可以同时处理的最大事件数。当达到这个限制时rsync进程将不再接受新的事件，直到队列中的事件被处理完毕。
- max_user_instances文件：用于设置每个用户可以同时连接到文件系统的最大实例数。例如，如果将该值设置为1024，则每个用户最多只能同时连接到1024个文件系统实例。
- max_user_watches文件：设置inotifywait或inotifywatch命令可以监视的文件数量。例如，如果将该值设置为2048，则每个用户最多只能同时监视2048个文件。

4、inotifywait命令应用

1）常用选项

- **-m：表示始终监控，否则监控一次后就退出**
- **-r：递归监控，可能超出max_user_watches的值导致错误，需要适当进行修改**
- **-q：--quiet静默输出，简化输出信息**

- **-e：指定监控的事件，如：delete、create、attrib、modify、close_write**

2）了解以下的选项

- --exclude ： 通过模式匹配来指定不被监控的文件，区分大小写
- --excludei ： 通过模式匹配来指定不被监控的文件，不区分大小写
- --timefmt：监控到事件触发后，输出的时间格式，可指定可不指定该选项，一般设置为[--timefmt '%Y/%m/%d %H:%M:%S']
- --format：用户自定义的输出格式，如[--format '%w%f %e%T']
  - %w：产生事件的监控路径，不一定就是发生事件的具体文件，例如递归监控一个目录，该目录下的某文件产生事件，将输出该目录而非其内具体的文件
  - %f：如果监控的是一个目录，则输出产生事件的具体文件名。其他所有情况都输出空字符串
  - %e：产生的事件名称
  - %T：以"--timefmt"定义的时间格式输出当前时间，要求同时定义"--timefmt"

3）inotifywait -e可监控的事件

- move：是moved_to和moved_from的结合
  - moved_to：用来表示文件或目录移动到的位置
  - moved_from：则表示文件或目录从哪个位置被移动
- modify：文件被写入
- create：在被监控的目录中创建了文件或目录
- delete：删除了被监控目录中的某文件或目录

4）了解下列选项

- access：文件被访问
- attrib：元数据被修改。包括权限、时间戳、扩展属性等等
- close_write：打开的文件被关闭，是为了写文件而打开文件，之后被关闭的事件
- close_nowrite：read only模式下文件被关闭，即只能是为了读取而打开文件，读取结束后关闭文件的事件
- close：是close_write和close_nowrite的结合，无论是何种方式打开文件，只要关闭都属于该事件
- open：文件被打开
- moved_self：被监控的文件或目录发生了移动，移动结束后将不再监控此文件或目录
- delete_self：被监控的文件或目录被删除，删除之后不再监控此文件或目录
- umount：挂载在被监控目录上的文件系统被umount，umount后不再监控此目录
- isdir ： 监控目录相关操作

5、实例

```
[root@Client ~] inotifywait -mrq -e modify,create,move,delete /myweb/
/myweb/ CREATE 4
/myweb/ CREATE,ISDIR dir
/myweb/ MODIFY 1
/myweb/ MOVED_FROM 4
/myweb/ MOVED_TO aaa
```

```
/myweb/ DELETE aaa
```

六、通过inotifywait触发rsync实时同步

1、使用while、read持续获得监控结果

2、实例（下行同步）

```
#!/bin/bash
INOTIFY_CMD="inotifywait -mrq -e modify,create,attrib,move,delete /myweb"
RSYNC_CMD="rsync -rzH --delete --password-file=/etc/server.pass zhangsan@192.168.237.129::qm /myweb"
if [ ! -e /etc/server.pass ];then
    echo "file not exist"
    exit 1
fi
$INOTIFY_CMD | while read DIRECTORY EVENT FILE
do

$RSYNC_CMD

done
```

3、实例（上行同步）

```
#!/bin/bash
INOTIFY_CMD="inotifywait -mrq -e modify,create,attrib,move,delete /myweb"
RSYNC_CMD="rsync -rzH --delete --password-file=/etc/server.pass /myweb backuper@192.168.112.128::www"
$INOTIFY_CMD | while read DIRECTORY EVENT FILE
do

$RSYNC_CMD

done
```

4、脚本详解：

1）用于监控/myweb目录的变化并执行相应的操作。

2）它使用了inotifywait命令来监视目录的变化，当该目录下的文件或目录发生变化时，就会触发一个事件。

3）在脚本中，首先定义了两个变量：INOTIFY_CMD和RSYNC_CMD

- INOTIFY_CMD存储了inotifywait命令的路径和参数，用于监控目录的变化
- RSYNC_CMD存储了rsync命令的路径和参数，用于同步文件。

4）接下来，通过管道符将inotifywait命令的输出传递给一个while循环。

5）并将其赋值给三个变量：DIRECTORY、EVENT和FILE。

- DIRECTORY：发生变化的目录路径
- EVENT：发生的事件类型
- FILE：发生变化的文件名。

6）然后，使用if语句检查当前是否有正在运行的rsync进程。

7）如果没有正在运行的rsync进程，则执行rsync命令进行同步操作。

8）总结起来，这段代码的作用是：当指定的目录（/myweb）发生变化时，如果当前没有正在运行的rsync进程，则执行rsync命令将本地备份文件同步到远程服务器上。

小结

- 如何配置可匿名访问的rsync服务器?
- rsync同步源的资源表示方式有哪两种?

本章作业

一、练习

1、rsync实现上行、下行同步

2、使用--delete选项完再次完成上行同步

3、使用--password-file选项完成下行同步

二、案例

- 使用rsync构建镜像网站
- 作业要求
  - 服务器A（北京）作为rsync发起端
  - 服务器B（深圳）作为远程rsync服务器
  - 结合inotify机制实现触发式的下行同步（需要使用--delete选项）